

IFIP AICT 679

Norbert Meyer
Anna Grocholewska-Czuryło
(Eds.)



ICT Systems Security and Privacy Protection

38th IFIP TC 11 International Conference, SEC 2023
Poznan, Poland, June 14–16, 2023
Revised Selected Papers

 Springer

Editor-in-Chief

Kai Rannenberg, Goethe University Frankfurt, Germany

Editorial Board Members

TC 1 – Foundations of Computer Science

Luis Soares Barbosa , University of Minho, Braga, Portugal

TC 2 – Software: Theory and Practice

Jacques Carette, Department of Computer Science, McMaster University, Hamilton, ON, Canada

TC 3 – Education

Arthur Tatnall , Victoria University, Melbourne, Australia

TC 5 – Information Technology Applications

Erich J. Neuhold, University of Vienna, Austria

TC 6 – Communication Systems

Burkhard Stiller, University of Zurich, Zürich, Switzerland


TC 7 – System Modeling and Optimization

Lukasz Stettner, Institute of Mathematics, Polish Academy of Sciences, Warsaw, Poland

TC 8 – Information Systems

Jan Pries-Heje, Roskilde University, Denmark


TC 9 – ICT and Society

David Kreps , National University of Ireland, Galway, Ireland

TC 10 – Computer Systems Technology

Achim Rettberg, Hamm-Lippstadt University of Applied Sciences, Hamm, Germany


TC 11 – Security and Privacy Protection in Information Processing Systems

Steven Furnell , Plymouth University, UK

TC 12 – Artificial Intelligence

Eunika Mercier-Laurent , University of Reims Champagne-Ardenne, Reims, France

TC 13 – Human-Computer Interaction

Marco Winckler , University of Nice Sophia Antipolis, France

TC 14 – Entertainment Computing

Rainer Malaka, University of Bremen, Germany

IFIP Advances in Information and Communication Technology

The IFIP AICT series publishes state-of-the-art results in the sciences and technologies of information and communication. The scope of the series includes: foundations of computer science; software theory and practice; education; computer applications in technology; communication systems; systems modeling and optimization; information systems; ICT and society; computer systems technology; security and protection in information processing systems; artificial intelligence; and human-computer interaction.

Edited volumes and proceedings of refereed international conferences in computer science and interdisciplinary fields are featured. These results often precede journal publication and represent the most current research.

The principal aim of the IFIP AICT series is to encourage education and the dissemination and exchange of information about all aspects of computing.

More information about this series at <https://link.springer.com/bookseries/6102>


Norbert Meyer · Anna Grochowska-Czuryło
Editors

ICT Systems Security and Privacy Protection

38th IFIP TC 11 International Conference, SEC 2023
Poznan, Poland, June 14–16, 2023
Revised Selected Papers

Editors

Norbert Meyer 
Poznan Supercomputing
and Networking Center
Poznań, Poland

Anna Grocholewska-Czuryło 
Poznań University of Technology
Poznań, Poland

ISSN 1868-4238 ISSN 1868-422X (electronic)
IFIP Advances in Information and Communication Technology
ISBN 978-3-031-56325-6 ISBN 978-3-031-56326-3 (eBook)
<https://doi.org/10.1007/978-3-031-56326-3>

© IFIP International Federation for Information Processing 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Preface

The 38th International Conference on ICT Systems Security and Privacy Protection (IFIP SEC 2023) was held in Poznań on June 14–16, 2023. The conference focused on current and future IT Security and Privacy Challenges and was organized by the Poznan Supercomputing and Networking Center (PSNC) in Poznan, Poland. The conference was a part of a series of well-established international conferences on Security and Privacy. SEC is the flagship event of the International Federation for Information Processing (IFIP) Technical Committee 11 (TC-11).

From a total of 80 papers that were submitted from all over the world, 27 were selected for publication in the conference proceedings by the Program Committee (PC). The Program Chair was supported by a PC consisting of 80 leading experts in the field of IT security and privacy.

The selection of papers was made through a double-blind peer review process. Each submitted paper was assigned to at least 3 reviewers. Following the review phase, the discussions phase took place. The conflict-of-interest rules were imposed to ensure that papers are not handled by PC members with a close working relationship with the authors. Authors of the rejected papers received feedback on their work to help them improve future submissions.

This volume of the conference proceedings contains the revised versions of the 26 papers that were selected. The final revised versions of papers were not reviewed again, and the authors are responsible for their contents.

The IFIP SEC 2023 program featured two excellent invited talks. On the first day of the conference Josef Pieprzyk gave a talk on “Joint Compression and Encryption”. Then, on the second day, Sebastiano Batiati presented his research on “DeepFake Technology: Current Trends and Perspectives”.

Many people contributed to the success of IFIP SEC 2023 conference. First and foremost, we would like to thank the authors for submitting their innovative research results to the conference. We are very grateful to all the PC members for their hard work during the papers’ review process determining the program of the conference. We sincerely thank Norbert Meyer and Andrzej Jaszkiwicz, the general chairs of the conference, Paul Haskell-Dowland, IFIP TC11 Chair, and the members of the local Organizing Committee of the Poznan Supercomputing and Networking Center for handling all the organizational work of the conference. We also thank the Springer team for managing the publication of these conference proceedings. Last but not least, we thank the invited speakers, session chairs, and all participants for coming to Poznań and contributing to IFIP SEC 2023.

December 2023

Norbert Meyer
Anna Grochowska-Czuryło

Yvo Desmedt	University of Texas at Dallas, USA
Nicola Dragoni	Technical University of Denmark, Denmark
Givemore Dube	CSZ, Zimbabwe
Isao Echizen	National Institute of Informatics, Japan
Simone Fischer-Hübner	Karlstad University, Sweden
Gerard Frankowski	Poznan Supercomputing and Networking Center, Poland
Steven Furnell	University of Nottingham, UK
Mariusz Głabowski	Poznan University of Technology, Poland
Hélder Gomes	Universidade de Aveiro, Portugal
Nils Gruschka	University of Oslo, Norway
Rene Rydhof Hansen	Aalborg University, Denmark
Paul Haskell-Dowland	Edith Cowan University, Australia
Dominik Herrmann	University of Bamberg, Germany
Xinyi Huang	Fujian Normal University, China
Dieter Hutter	DFKI GmbH, Germany
Pedro Inácio	Universidade da Beira Interior, Portugal
Martin Gilje Jaatun	SINTEF Digital, Norway
Jan Jürjens	University of Koblenz-Landau, Germany
Georgios Kambourakis	University of the Aegean, Greece
Farzaneh Karegar	Karlstad University, Sweden
Stefan Katzenbeisser	University of Passau, Germany
Dogan Kesdogan	Universität Regensburg, Germany
Zbigniew Kotulski	Warsaw University of Technology, Poland
Stephan Krenn	AIT Austrian Institute of Technology, Austria
Łukasz Krzywiecki	Wroclaw University of Technology, Poland
Peeter Laud	Cybernetica AS, Estonia
Wenjuan Li	Hong Kong Polytechnic University, China
Maciej Liskiewicz	University of Luebeck, Germany
Luigi Logrippo	Université du Québec en Outaouais, Canada
Javier Lopez	University of Malaga, Spain
Suryadipta Majumdar	Concordia University, Canada
Salman Manzoor	TU Darmstadt, Germany
Marian Margraf	Hochschule Darmstadt, Germany
Vashek Matyas	Masaryk University, Czechia
Weizhi Meng	Technical University of Denmark, Denmark
Zlatogor Minchev	Bulgarian Academy of Sciences, Bulgaria
Yuko Murayama	Tsuda College, Japan
Maurizio Naldi	LUMSA University, Italy
Lili Nemec Zlatolas	University of Maribor, Slovenia
Brajendra Panda	University of Arkansas, USA
Miguel Pardal	Universidade de Lisboa, Portugal
António Pinto	Politécnico do Porto, Portugal
Kai Rannenber	Goethe University Frankfurt, Germany
Carlos Rieder	ISEC AG, Switzerland
Juha Röning	University of Oulu, Norway

Reihaneh Safavi-Naini	University of Calgary, Canada
Christophe Rosenberger	Normandy University, France
Pierangela Samarati	Università degli Studi di Milano, Italy
Jürgen Schönwälder	Jacobs University Bremen, Germany
Jun Shao	Zhejiang Gongshang University, China
Nicolas Sklavos	University of Patras, Greece
Daniel Slamanig	AIT Austrian Institute of Technology, Austria
Agusti Solanas	Rovira i Virgili University, Spain
Shamik Sural	IIT, Kharagpur, India
Neeraj Suri	Lancaster University, UK
Vicenc Torra	University of Skovde, Sweden
Lingyu Wang	Concordia University, Canada
Edgar Weippl	University of Vienna, Austria
Roman Wyrzykowski	Czestochowa University of Technology, Poland
Øyvind Ytrehus	University of Bergen, Norway
André Zúquete	University of Aveiro, Portugal

Contents

Web Content Integrity: Tamper-Proof Websites Beyond HTTPS	1
<i>Sven Zemanek, Sebastian Tauchert, Max Jens Ufer, and Lilli Bruckschen</i>	
Privacy-Preserving Clustering for Multi-dimensional Data Randomization Under LDP	15
<i>Hiroaki Kikuchi</i>	
Hierarchical Model-Based Cybersecurity Risk Assessment During System Design	30
<i>Tino Jungebloud, Nhung H. Nguyen, Dong Seong Kim, and Armin Zimmermann</i>	
The Influence of Privacy Concerns on Cryptocurrency Acceptance	45
<i>Peter Hamm, Sebastian Pape, and Kai Rannenber</i>	
Automated Enrichment of Logical Attack Graphs via Formal Ontologies	59
<i>Kéren Saint-Hilaire, Frédéric Cuppens, Nora Cuppens, and Joaquin Garcia-Alfaro</i>	
Detecting Web Bots via Mouse Dynamics and Communication Metadata.	73
<i>August See, Tatjana Wingarz, Matz Radloff, and Mathias Fischer</i>	
Practical Single-Round Secure Wildcard Pattern Matching	87
<i>Jun Xu, Shengnan Zhao, Chuan Zhao, Zhenxiang Chen, Zhe Liu, and Liming Fang</i>	
Efficient Non-interactive Anonymous Communication	102
<i>Sigurd Eskeland and Svetlana Boudko</i>	
PointPuff: An Ed25519 Optimization Implementation	117
<i>Mengqing Yang, Chunxiao Ye, Yuanmu Liu, Yan Jin, and Chunming Ye</i>	
Detecting Web Tracking at the Network Layer	131
<i>Maximilian Wittig and Doğan Kesdoğan</i>	
What’s Inside a Node? Malicious IPFS Nodes Under the Magnifying Glass . . .	149
<i>Christos Karapapas, George C. Polyzos, and Constantinos Patsakis</i>	

Quantum-Secure Communication for Trusted Edge Computing with IoT Devices 163
George Kornaros, Georgia Berki, and Miltos Grammatikakis

Evaluation of a Red Team Automation Tool in Live Cyber Defence Exercises 177
Hannes Holm and Jenni Reuben

Automated and Improved Detection of Cyber Attacks via an Industrial IDS Probe. 191
Almamy Touré, Youcef Imine, Thierry Delot, Antoine Gallais, Alexis Semmont, and Robin Giraud

An Accurate and Real-Time Detection Method for Concealed Slow HTTP DoS in Backbone Network. 207
Jinfeng Chen, Hua Wu, Suyue Wang, Guang Cheng, and Xiaoyan Hu

Towards an Information Privacy Competency Model for the Usage of Mobile Applications 222
Aikaterini Soumelidou and Aggeliki Tsohou

SecPassInput: Towards Secure Memory and Password Handling in Web Applications 236
Pascal Wichmann, August See, and Hannes Federrath

Bl0ck: Paralyzing 802.11 Connections Through Block Ack Frames. 250
Efstratios Chatzoglou, Vyron Kampourakis, and Georgios Kambourakis

Enhancing the ACME Protocol to Automate the Management of All X.509 Web Certificates 265
David A. Cordova Morales, Ahmad Samer Wazan, David W. Chadwick, Romain Laborde, April Rains Reyes Maramara, and Kalil Cabral

MADONNA: Browser-Based MALicious Domain Detection Through Optimized Neural Network with Feature Analysis 279
Janaka Senanayake, Sampath Rajapaksha, Naoto Yanai, Chika Komiya, and Harsha Kalutarage

Cyber Key Terrain Identification Using Adjusted PageRank Centrality 293
Lukáš Sadlek and Pavel Čeleda

Machine Learning Metrics for Network Datasets Evaluation 307
Dominik Soukup, Daniel Uhříček, Daniel Vašata, and Tomáš Čejka

Factors of Intention to Use a Photo Tool: Comparison Between
 Privacy-Enhancing and Non-privacy-enhancing Tools 321
Vanessa Bracamonte, Sebastian Pape, and Sascha Löbner

Real-Time Platform Identification of VPN Video Streaming Based on
 Side-Channel Attack 335
Anting Lu, Hua Wu, Hao Luo, Guang Cheng, and Xiaoyan Hu

Toward the Establishment of Evaluating URL Embedding Methods Using
 Intrinsic Evaluator via Malicious URLs Detection 350
Qisheng Chen and Kazumasa Omote

Key Management Based on Ownership of Multiple Authenticators
 in Public Key Authentication 361
Kodai Hatakeyama, Daisuke Kotani, and Yasuo Okabe

Author Index 377



Web Content Integrity: Tamper-Proof Websites Beyond HTTPS

Sven Zemanek^(✉), Sebastian Tauchert, Max Jens Ufer,
and Lilli Bruckschen

Fraunhofer FKIE, Bonn, Germany

{sven.zemanek, sebastian.tauchert, max.jens.ufer,
lilli.bruckschen}@fkie.fraunhofer.de

Abstract. We propose Web Content Integrity, a framework that allows a service provider to guarantee the integrity of their static website, even in the face of a compromised web server. Such integrity assurances can then be used to implement a secure end-to-end encryption application built in the form of a website. Our framework encompasses developers, the Domain Name System, and web browsers. To accomplish the integrity guarantees, our framework makes use of an index of queryable URLs and allowed redirects for the website, and publishes the cryptographic hash value of the index in the DNS. Web browsers can then use the information from the DNS to verify that the resources they retrieve from the web server have not been tampered with. The required data structures can be generated automatically, and the framework introduces an initial delay of about 4 ms and a recurring delay for each request of about 2 ms for a sample website.

Keywords: Web Security · Integrity · DNS

1 Introduction

Current web security measures such as HTTPS protect the integrity of data transmission between web server and client. However, the web server serving the website is not technically restricted from answering requests with arbitrary response content. An attacker who compromised the web server can leverage this to provide false or misleading information to website visitors. In addition, such an attacker can possibly extract secret information from clients or distribute malware. Some service providers have extra strong integrity requirements, like a voting authority that publishes election results, or a distributor of software used in classified contexts. Frequently, service providers do not host their websites themselves, on their own infrastructure, but use the services of dedicated web hosting providers. On the one hand, these hosting providers have all the options of an attacker who compromised a web server, and on the other hand, they could be breached themselves, giving an attacker control over a website's content. This means that even if one has trust in their legal framework to guarantee that

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 1–14, 2024.

https://doi.org/10.1007/978-3-031-56326-3_1

hosting providers behave properly, a technical solution to guarantee the integrity of one’s website is still needed.

We propose *Web Content Integrity* (WCI), a framework that allows service providers to cryptographically ensure that visitors of their website receive only content that has not been tampered with. This is accomplished by first compiling an index of URLs and corresponding cryptographic hashes of the content available under the respective URL as well as allowed redirects. The cryptographic hash of this index is then published in the Domain Name System, next to the entries for the domain name that connects the domain name with IP addresses. WCI effectively eliminates the possibility for a web server to serve modified or entirely different content for a domain name. This leaves malicious actors only with the option to completely disable access to certain content. While our framework is tailored towards static websites, we also sketch a migration path for dynamic websites towards using WCI.

In summary, our contributions are as follows:

1. We propose a framework for service providers to make the integrity of their website verifiable using cryptographic hashes, and for web browsers to obtain those cryptographic hashes via DNS
2. We demonstrate that the creation of metadata required for the integrity verification can be automated, by implementing plugins for multiple static site generators
3. We determine properties of websites that can reasonably apply the framework
4. We provide upper bounds for the delays introduced when WCI is implemented by web browsers and deployed for a website

The remainder of this paper is structured as follows: After presenting related work in Sect. 2, we define the WCI framework in Sect. 3. We discuss the applicability of the framework, its compatibility with existing web infrastructure, possible attacks, and introduced overheads in Sect. 4, and summarize our findings in Sect. 5.

2 Related Work

There are already existing concepts for making sure the content of remote resources matches an expectation.

Subresource Integrity [11] allows resources from “script” or “link” tags that are included in a website, i.e., JavaScript or CSS files, to be accompanied by cryptographic hash values. The resource is checked against the hash values when it is retrieved. If no hash matches, the script or style is not applied, and it is treated as a network error where the resources could not be loaded. While this prevents tampering with the contents of specific types of resources, it does not cover all types of content (like images), and the HTML content which loads the resources is itself not protected from tampering. Nevertheless, the principle of checking loaded resources against cryptographic hashes before using them is

also used in WCI, and the widespread availability of Subresource Integrity in web browsers demonstrates the technical feasibility.

Mylar [6] ensures the integrity of static web applications by cryptographically signing the root HTML content and serving subresources from a second domain in order to leverage the web browser’s same-origin policy. Included subresources are accompanied by a cryptographic hash that is checked similarly to Subresource Integrity. The key for verifying the signature of the root HTML page is stored in the TLS certificate. This may be problematic since the TLS certificate is being provided by the web server and can therefore be manipulated or replaced by an attacker with access to a Certificate Authority that is trusted by the web browser.

There are also approaches for integrity verification that have been developed before the widespread deployment of HTTPS.

Already in 1998, Peacock and Powell [7] described an algorithm to calculate a MD5 hash value over the contents of a website and selected subresources. This hash value was to be included in a metadata set for the BIBLINK project. The hash value could then be used to verify that the parts of a website covered by the hash value did not change since the metadata set had been created. The concept represents an early framework for expressing expectations about website contents, albeit partly outside the web browsing context.

Bayardo and Sorensen [1] suggested the construction of a Merkle tree over the contents of a website to provide integrity guarantees. The root hash value of that Merkle tree has to be transferred to clients over a secure channel that cannot be tampered with by web servers, for which the authors suggest among others to use the DNS. Besides guaranteeing the integrity of web content, the framework can also express that a specific resource does not exist on the website. The Merkle tree is constructed in a way that does leak the number of resources available on the website, but does not leak the actual URLs to those resources. Due to the properties of that construction, the use of wildcards or regular expressions in path specifiers is not possible. Also, since clients always need to know the current root hash of the Merkle tree in order to validate the website contents, updates of the website contents require immediate redistribution of the root hash value over the secure channel. When a resource is requested, parts of the Merkle tree below the root hash value are transferred in a special response header. This requires modifications to the web servers’ behavior.

Sedaghat et al. [9] proposed to run additional software on the web server that validates the web server’s responses against cryptographic hash values. This is not a suitable mechanism to protect against tampering from an attacker-controlled web server, since the integrity mechanism on the web server can be tampered with as well.

Singh et al. [10] advocated for HTTPi, a protocol that extends HTTP with integrity guarantees. Reis et al. [8] proposed what they called “Web Tripwires”: JavaScript snippets that are included in websites, check client-side whether the site has been modified in transit, and report back to the web server if a modification is detected. Both approaches focus on the manipulation of content in transit

between a web server and a web browser. They do not protect against a compromised web server and are also made obsolete by the widespread deployment of HTTPS.

3 Web Content Integrity

We propose *Web Content Integrity* (WCI), a framework to cryptographically verify the integrity of content served by a web server. The framework involves three parties: Developers who generate the content that is served by web servers, the Domain Name System (DNS), and web browsers. Section 3.1 defines the framework, Sect. 3.2 describes scaled-down variants of the framework with reduced scope, Sect. 3.3 lays out considered alternatives for some design decisions, and Sect. 3.4 details how we automated the creation of the necessary data structures.

3.1 Framework

Web Content Integrity ensures the integrity of websites in the following way:

1. When a website is created, a WCI index for the website contents is created alongside it
2. The website and WCI index are deployed to a web server
3. A WCI DNS record is configured
4. Web browsers use the WCI DNS record to validate the WCI index, and use the WCI index to validate the integrity of resources they retrieve from the web server

The WCI index file represents a cryptographically verifiable expectation about the responses that a web server gives to requests. A website for which WCI is configured must provide a WCI index file under the path `/wci.txt`.

This WCI index file consists of lines of text and is structured as follows:

1. The file starts with a `REDIRECTS` section, mapping all paths for which the web server may send a “Location” header to redirect to another URL to a list of the allowed values of the “Location” header.
2. This is followed by a `PATHS` section, which maps all queryable paths the website is supposed to provide to the SHA-256 hash value of the expected response content for that path.

Figure 1 depicts a sample WCI index for a website with one redirect rule and three URLs with hashes in the `PATHS` section.

Paths in the `PATHS` and the `REDIRECTS` section are treated as regular expressions. The `PATHS` and the `REDIRECTS` section may be empty, with only their respective section header being present. Location values in the `REDIRECTS` section may use capture groups from the regular expression in their respective path. This flexible definition of paths enables developers to define hashes for pages that are available under many URLs, without having to enumerate all of those URLs.

```
#REDIRECTS
"/goto/(.+)" : "https://example.org/$1"
#PATHS
"/" : 858d80f786e52cdece36d141de383...c9ce
"/about" : 309d702383574787c1925ae13703...5bb2
"/download.zip" : 15cfd081fc216ea06d74b31ff343...7837
```

Fig. 1. Sample WCI index with a REDIRECTS section with one entry and a PATHS section with three entries. Three dots indicate shortened lines for display purposes.

This is especially useful for covering Single-Page-Applications. It also makes it possible to specify hashes for default pages, like a 404 error page that is displayed for URLs that are not available on the website. This reduces false-positive alerts on websites that deploy WCI.

The cryptographic hash value is calculated over the contents of the WCI index file. Note that this is purely an operation on a string. No whitespace or newlines are manipulated. This makes the construction robust against differences in serialisation/deserialisation of the various clients working with the file.

A web browser must know that WCI is configured for a domain in order to act accordingly. Also, the web server that serves the website under that domain must not be able to interfere with the web browser learning this information. Therefore, DNS is a suitable vehicle to convey this information and the cryptographic public key. Similar to DANE [3], which uses DNS records to pin TLS certificates, we announce the cryptographic hash value that can be used to verify the WCI index via DNS records.

We define a new WCI DNS record to consist of three components:

1. specification version indicator
2. identifier of a cryptographic hash function (sha256)
3. cryptographic hash value (base64 encoded)

Figure 2 shows an annotated example of such a DNS record. When a DNS server receives an A (IPv4) or AAAA (IPv6) query, and one or more WCI records exists for the queried domain, all WCI records must be sent in the additional records section of the response, alongside the answer to the original request. As “the additional records section contains RRs which relate to the query, but are not strictly answers for the question” [4], it is a suitable place to transmit this information, and existing clients that follow RFC 1035 should not malfunction due to the presence of data in the additional records section of a DNS response.

Web browsers query DNS information without having to interact with a web server first.¹ If a WCI DNS record is present, WCI is configured for the corresponding domain, and the absence of corresponding features from the responses of the web server must be considered a severe error, possibly indicating an attack.

¹ Chromium and Firefox implement DNS clients that issue their own queries and process the responses. They do not require DNS support from the underlying operating system.

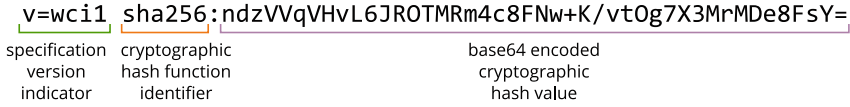


Fig. 2. Structure of a WCI DNS record

For the WCI framework to function, web browsers must implement verification and application of the WCI index. When a web browser queries a domain for which WCI is configured, it must first fetch the WCI index under the path `/wci.txt` and verify that the cryptographic hash value of the WCI index content is announced by at least one of the WCI DNS records.

If this verification fails, this must be considered a severe error, possibly indicating an attack.

When the verification of the WCI index was successful, for each queried path, the response from the web server must be checked against the given hash values in the `PATHS` section. If no entry for the queried path is present, or if there is no such entry with a matching hash value, which means the queried URL is not covered by the WCI index, this must again be considered a severe error, possibly indicating an attack.

If the response from the web server contains a `Location` header, its value must be checked against the values given for the path in the `REDIRECTS` section. If no entry for the queried path is present or none of the allowed values match the given value, this must once again be considered a severe error, possibly indicating an attack.

To prevent false positives, e.g. during a site update, we additionally mandate the following retry policy:

1. A validation mismatch of a resource triggers a reload of the WCI index. If the reloaded WCI index is equal to the previous one, the error is confirmed and access to the Website is blocked. Otherwise, the WCI index is validated against the known WCI DNS records.
2. A validation mismatch of the WCI index triggers a reload of the WCI DNS records. If the reloaded WCI DNS records are equal to the previous ones, the error is confirmed and access to the Website is blocked. Otherwise, the WCI index is validated against the new WCI DNS records.

For requests during a site update, there may be multiple rounds of reloading. However, a malicious web server cannot maintain an endless chain of reloads because an error is confirmed once the same WCI DNS records are observed twice, which the web server cannot influence.

Website operators usually wish to update the contents of their website from time to time. To update a website on a domain for which WCI is configured, the following steps should be executed in order:

1. Create the new version of the website, create the WCI index and calculate its hash value
2. Add a WCI DNS record with the new hash value
3. Set the TTL of the WCI DNS records with the old hash value to 0
4. Wait for the original TTL of the WCI DNS records with the old hash value
5. Deploy the new version of the website on the web server
6. Remove WCI DNS records with old hash values

This sequence of events ensures that the old hash value is no longer cached in intermediate DNS resolvers when the new version of the website is deployed on the web server. Clients that contact the website at that point will either already have fetched the new WCI DNS record, or be prompted to do so by a validation failure.

When the procedures outlined above are properly executed, our presented framework guarantees that a web browser only processes resources that have not been tampered with, and prevents access to the modified website in all other cases. It does so in a failsafe way that takes into account the effects of caching.

3.2 Scaled-Down Variants

The full WCI framework requires modifications to web browser and DNS server behavior. For evaluation purposes and local applications, less invasive scaled-down variants of the framework can be used. Instead of a native implementation, the web browser logic can be implemented by an add-on, which either performs its own DNS queries or uses hardcoded WCI DNS records. Such a browser add-on can intercept responses and perform the WCI validation before either handing off the unaltered response to the web browser or blocking access to the resource when a validation error occurs.

Current DNS servers can return the necessary information in TXT DNS records instead of dedicated WCI DNS records. Explicitly querying WCI data in the DNS effectively doubles the DNS traffic. For a full-scale implementation of WCI, the behavior as described in Sect. 3.1 is therefore preferable.

3.3 Considered Alternatives

The WCI index is represented in a custom line based text format. We considered using commonly found structured data formats like JSON or YAML to store the WCI index, as tools for parsing and representing data in those formats are usually broadly available. These data formats provide a lot more features than required, like nested properties. To keep the processing of the WCI index as simple as possible, we chose to use the custom line based format instead.

It would be possible to alternatively specify a binary format and representation for the WCI index. However, the web ecosystem generally tends to use plain text files. Since files can be compressed for transit, we expect the impact

of the decision to choose a plain text format for the WCI index to be negligible with regard to the actual transmission size.

Due to how it is constructed, the WCI index leaks all queryable paths of the website. An alternative would have been to use hash values of the paths instead of the path specifiers, like [1] have chosen to. This would however prevent the use of regular expressions as path specifiers, since correctly guessing the used regular expression from a given path is infeasible for nontrivial cases. We prefer the flexibility that the use of regular expressions enables over the concealment of all valid paths, and have therefore decided against this alternative.

The WCI framework requires the selection of a cryptographic hash algorithm. Output size is the main point to consider besides general security considerations. SHA-256 has an output size of 256 bits, high general availability, and is currently considered cryptographically secure [2, 5]. It is therefore a suitable choice compared to alternatives with bigger output size, like SHA-512 or SHA3-512.

We considered distributing a cryptographic public key instead of the cryptographic hash value in the WCI DNS record. The WCI DNS record would then also include a version identification number that is incremented with each new version of the website to prevent rollback attacks, and the WCI index would be extended by a signatures section that contains signatures over the version identification number and the redirects and paths sections. While such an approach could eliminate retries in some scenarios during website updates, it introduces additional challenges with regard to cryptographic key management, key rotation, key loss and compromise, requires the calculation of cryptographic signatures over selected parts of a file as well as the application of public key cryptography in the first place, and finally makes calculations considerably slower compared to the calculation of cryptographic hash values. We therefore opted for the current framework specification without public key cryptography.

3.4 Automation

To ease adoption of a new concept, it is desirable to make its implementation as easy as possible. Most of the steps required for configuring WCI for a domain can be automated. We have developed tooling to generate the WCI index and the content of the WCI DNS record. To demonstrate that this functionality can be integrated into the build processes of static websites, we have created plug-ins for the popular static site generators Gatsby, which is written in JavaScript, and MkDocs, which is written in Python. Additionally, we have developed a generic npm package that provides the same functionality when applied to a directory structure of files, and can be integrated in arbitrary build pipelines. We use the developed tools to generate valid WCI indexes and DNS record data in our performance measurement experiments. Our implementations are available under <https://github.com/fkie-cad/Web-Content-Integrity>.

4 Discussion

4.1 Applicability

Currently, we expect the majority of websites to employ server-side processing. As WCI only works on static websites, such websites must be adapted before configuring WCI for them. Websites can be functionally separated into a static data component, a dynamic data component, and a static application component that works on the static or dynamic data. The integrity of the static data and the static application can be protected with WCI. Dynamic data can then be handled via another domain or subdomain for which WCI is not configured. Such data must be considered untrustworthy by the application component.

Investing the effort of separating dynamic data from the rest of the website in order to be able to configure WCI can be especially worthwhile for websites which provide authoritative information, like law texts or election results, or which provide a web application like a text or graphics editor for sensitive contents.

4.2 Compatibility with Existing Web Infrastructure

We want WCI to require only sparse modifications to existing web infrastructure. In this section we argue that many phenomena of the existing web infrastructure are not impaired by WCI.

DNS Caching. Resource records in DNS responses carry a time-to-live (TTL) field, indicating how long the contained information may be considered valid until it should be fetched again [4]. After a website update (cf. Section 3.1), clients will not yet accept the updated WCI index because its hash value has changed. The validation failure for the WCI index triggers a forced update of the hash value from the DNS, which resolves issue.

Websites Under Multiple Domains. Sometimes, websites are available under multiple domains or subdomains. A common example would be a website that is available under both `example.org` and `www.example.org`. As the WCI index does not encode the domain name in any way, it is possible to configure WCI for all of those (sub-) domains by setting the WCI DNS record, and serving the exact same files for all of them.

Specification Updates. The version identifier at the start of the WCI DNS record facilitates updates to the WCI framework. The involved entities can use the value of this field to adapt their behavior or content generation, or to determine that they do not recognize the specific version. As it is only present in the WCI DNS record, a compromised web server has no way to feign an older version of the specification. This prevents downgrade attacks based on the version of the WCI specification.

4.3 Defense Against Attacks

In order to connect to a website, web browsers query the DNS for the IP address(es) connected to a domain name and receive a response from a DNS server. Then, a connection is opened to an IP address from the response. While WCI does not protect against attackers who can control the content of responses to DNS queries, we argue that WCI does also not enable such an attacker to launch a new kind of attack. An attacker who controls the content of DNS responses has the following options with regard to WCI:

1. Remove the WCI record from a response if WCI is configured for the domain
2. Modify the WCI record so it no longer matches the content on the genuine web server
3. Add a WCI record for a domain for which WCI is not configured

The first case enables the web server to serve modified content. However, an attacker who can control the content of DNS responses can also replace the genuine IP address with one of a server under their own control, and serve modified content from there.

The second and third case lead to a denial of service, as the web server's responses no longer match the expectation that has been given by the DNS record. However, an attacker who can control the content of DNS responses can also suppress the genuine IP address from the response, or respond with an invalid IP address. This equally leads to a denial of service.

We conclude that control over the content of responses to DNS queries gives an attacker the ability to sidestep WCI, but WCI does not enable novel attacks. The following considerations are therefore all under the assumption that DNS queries and responses are not tampered with.

WCI protects against arbitrary modifications of the contents of a website. Even a compromise of the web server does not permit an attacker to circumvent the protection that WCI provides. Another attack method to consider are rollback attacks, where a compromised web server serves a previous version of the website. Such an attack can only be successful against clients which still have the WCI DNS record with the corresponding hash value cached. This only works for the duration of the TTL for the WCI DNS record. Since a rollback attack can only be conducted with a previous version of the website, the success of such an attack depends heavily on the goal of the attacker and whether that previous version of the website contains vulnerabilities or required content that enables the attack in the first place.

4.4 Overheads

We identify impacts on the network load caused by DNS requests and responses and on the processing of requests by DNS servers. We also look at the performance impact on the processing of responses by web browsers, and estimate a margin of improvement by comparison with SRI. We argue that the overhead needed for deployment of WCI in terms of development time and effort can be alleviated by use of automated tools.

DNS Load. In order to be able to grant the security guarantees that WCI provides, web browsers must query the WCI DNS record for each domain name they resolve. A scaled-down implementation (cf. Sect. 3.2) would send out two queries instead of one, effectively doubling the amount of DNS traffic caused by web browsers, which is undesirable. The alternative is to mandate that each response for an A or AAAA query include the WCI DNS record if one exists in the additional records section. Similar lookup behavior is already happening for A or AAAA queries for which a CNAME record is defined. This alternative does not increase the number of DNS requests. In this scenario, DNS servers do have to look up whether a WCI DNS record exists for each A and AAAA request they receive and have an answer for. This likely increases processing times of A and AAAA requests on dns servers, similar to how CNAME chains are resolved. Compared to a scenario where most clients send two DNS requests that have to be processed independently by the DNS servers, the version where WCI DNS records are automatically included in A and AAAA responses seems favorable both in terms of network usage and used processing power of DNS servers.

Performance Considerations and Comparison with SRI. Introducing WCI has a negative impact on performance. We have set up a local test setup that minimizes latency introduced by network communications and uses the minimal variant (see Sect. 3.2) with a browser extension that uses a hardcoded DNS WCI record. Using this test setup, we have measured two kinds of delay: Initial delay, which occurs once for each domain that is being connected to, and recurring delay, which occurs for every retrieval of a resource. Our performance measurements show about 4 ms initial delay and about 2 ms recurring delay for single resources, and about 3 ms initial delay and 13 ms recurring delay in total for a test of a full sample website with multiple subresources. Initial delay depends on the size of the WCI index, and recurring delay depends on the size of the fetched content.

These delays are composed as shown in Fig. 3. Depicted are three browser setups for testing to separate out the interesting kinds of delay (S1 is a baseline setup with a dummy extension for comparison):

- S2 The WCI index is already cached in the web browser extension, so no initial delay occurs
- S3 The WCI index is not cached in the web browser extension, so initial delay occurs every time
- S4 An outdated WCI index is cached initially in the web browser extension, so the WCI index has to be reloaded once when a request is processed

The “WCI setup” step is where most of the initial delay occurs (~ 4 ms). Fetching and verifying the response, on the other hand, require around 4 ms and 2 ms respectively. Note that the concrete load time only refers to internal network traffic. These times are expected to increase in an actual deployment.

A special case is when a retry occurs. In this case, the time required to fetch and validate the WCI file is shifted to the “Verify content” step. In addition,

two hash validations are performed. A failing one, which triggers the retry, and a successful one after updating the cached WCI file. Thus, the time required in this case is about the same as if the WCI file had to be loaded directly from the server.

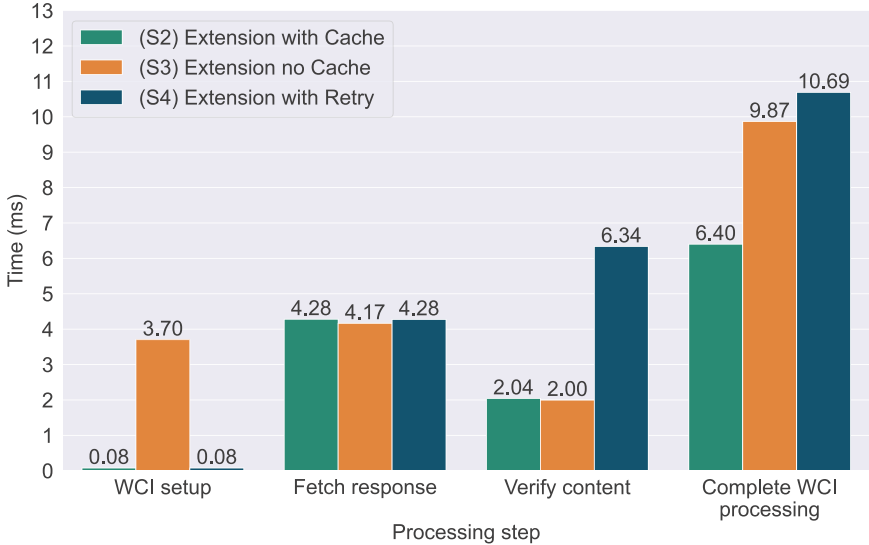


Fig. 3. Durations for sub-steps during the WCI validation.

We have conducted similar experiments with Subresource Integrity. The performance results from our tests with SRI are shown in Fig. 4.

SRI and WCI appear similar in terms of load time for files below 100 KiB. Once file sizes surpass this limit however, the native implementation of SRI outclasses our sample WCI implementation, taking only one third of the time of our WCI implementation at a file size of 10,000 KiB. The results spark hope that an efficient implementation of WCI can further reduce the processing time of larger resources by a factor of 2 to 3 or more. This transferability of results is justified by the similar nature of the processes in SRI and WCI.

Automation. The adoption of a new technology like WCI by developers can be facilitated by appropriate tooling support. We demonstrate that the creation of the WCI index file can effectively be fully automated by implementing plug-ins for MkDocs and Gatsby, two popular static site generators, as well as a generic tool that can be applied to arbitrary collections of files. The availability of easy-to-use and easy-to-integrate tools for WCI index creation keeps overheads for adopting this technology for existing static websites low.



Fig. 4. Comparison of average processing times for a minimal HTML page with inline (WCI) or linked (SRI) JavaScript of different sizes

5 Conclusion

Web Content Integrity focuses on an advanced understanding of the relationship between service providers and website visitors that goes beyond what HTTPS covers. It provides integrity guarantees for static websites that are resistant to tampering by a compromised web server. WCI is also resilient with regard to common maintenance tasks. We have demonstrated that the creation of the data structures required for WCI can be automated, hence facilitating adoption. With our sample implementation of the web browser logic in the form of a web browser extension, we have determined that the overhead in terms of additional delay introduced by WCI is expected to be around 9 ms initially, and about 2 ms for every resource. Initial delay increases with the size of the WCI index, and recurring delay increases with the size of the fetched content. Experiments with Subresource Integrity make it seem probable that the performance of our extension for bigger files can significantly be undercut by a native implementation. The evaluation of processing delays introduced at DNS servers by the deployment of WCI is considered future work, as is the development of methods to efficiently handle websites with many distinct queryable URLs.


References

1. Bayardo, R., Sorensen, J.: Merkle tree authentication of http responses, pp. 1182–1183 (2005). <https://doi.org/10.1145/1062745.1062929>
2. Bundesamt für Sicherheit in der Informationstechnik: Kryptographische Verfahren: Empfehlungen und Schlüssellängen (BSI TR-02102-1), Version: 2023-01 (2023)

3. Hoffman, P.E., Schlyter, J.: The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA. RFC 6698 (2012). <https://doi.org/10.17487/RFC6698>, <https://www.rfc-editor.org/info/rfc6698>
4. Mockapetris, P.V.: Domain names - implementation and specification. RFC 1035 (1987). <https://doi.org/10.17487/RFC1035>, <https://www.rfc-editor.org/info/rfc1035>
5. National Institute of Standards and Technology: Fips pub 180-4 – secure hash standard (shs) (2015). <https://doi.org/10.6028/NIST.FIPS.180-4>
6. Popa, R.A., Stark, E., Valdez, S., Helfer, J., Zeldovich, N., Balakrishnan, H.: Building web applications on top of encrypted data using mylar. In: 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14) (2014)
7. Powell, A., Peacock, I.: Metadata: Biblink.checksum. Ariadne (17) (1998). <http://www.ariadne.ac.uk/issue/17/biblink/>
8. Reis, C., Gribble, S.D., Kohno, T., Weaver, N.C.: Detecting in-flight page changes with web tripwires. In: NSDI, vol. 8 (2008)
9. Sedaghat, S., Pieprzyk, J., Vossough, E.: On-the-fly web content integrity check boosts users' confidence. *Commun. ACM* **45**(11), 33–37 (2002)
10. Singh, K., Wang, H.J., Moshchuk, A., Jackson, C., Lee, W.: Practical end-to-end web content integrity. In: Proceedings of the 21st International Conference on World Wide Web (2012)
11. Weinberger, J., Braun, F., Akhawe, D., Marier, F.: Subresource integrity. W3C recommendation, W3C (2016)



Privacy-Preserving Clustering for Multi-dimensional Data Randomization Under LDP

Hiroaki Kikuchi^{1,2} 

¹ Graduate School of Advanced Mathematical Science, Meiji University,
4-21-1 Nakano, Tokyo 164-8522, Japan

kikn@meiji.ac.jp

² Universitat Rovira i Virgili, Tarragona, Catalonia, Spain

Abstract. Randomization of multi-dimensional data under local differential privacy is a significant and practical application of big data. Because of the dimensionality issues, most existing works suffer from low accuracy when estimating joint probability distributions. In this paper, a set of attributes is divided into smaller clusters where the attributes are associated in terms of their dependencies. A privacy-preserving algorithm is proposed to estimate the dependencies of an attribute without disclosing the private values in the multi-dimensional data. Local differential privacy is guaranteed in the scheme. Using the clusters of attributes, the joint probabilities for multi-dimensional data can be estimated efficiently using two building blocks, called RR-independent and RR-Ind-Joint schemes. The experiments using some open datasets demonstrate that the dependencies of attributes can be estimated accurately and that the proposed algorithm outperforms existing state-of-the-art schemes in cases where the dimensionality is high.

1 Introduction

Local differential privacy (LDP) [6] enables the calculations of statistics about private data without having access to the actual private data. LDP techniques have been deployed by Google [7], and Microsoft [4].

Unfortunately, LDP cannot always be applied to randomization of big-data in general because of the multi-dimensional nature of such data. In many practical applications, there are number of attributes. A straightforward approach to randomizing the multi-dimensional data is to apply a conventional LDP scheme to the full domain for all attributes. However, this does not work because of the *curse of dimensionality*, which causes the following.

- *Exponential domain growth.* The number of values of the Cartesian product of multiple domains grows exponentially. Any analysis of the aggregated randomization matrix entails a high computational and communication costs.

- *Domain sparsity.* Domain sparsity is an additional undesirable consequence of the exponential growth of attribute combinations. Here, the combination of values increases exponentially, while the number of respondents remains constant. The number of individuals associated with any specific combination therefore becomes very small.

Many studies on high-dimensional data with LDP have been done including [2, 13, 21, 22], and [20]. Ren et al. [15] studied an LDP scheme called LoPub, which estimates multi-dimensional joint probability distributions. Jiang et al. [9] introduced DP-FED-WAE, which combines a generative Wasserstein autoencoder (WAE) [17] with federated learning. CALM, proposed by Zhang et al. [22], partitions the set of individuals into some groups aggregates to obtain a noisy marginal table, and then performs a reconstruction.

Domingo-Ferrer and Soria-Comas [5] proposed RR-Clusters. This approach splits the set of attributes into clusters according to mutual dependencies and estimates the joint probabilities (RR-Joint) for each cluster. The inter-cluster joint probabilities (RR-Independent) are given by the product of intra-cluster estimations under an assumption that the dependencies among attributes are neglectable. However, finding the optimal clusters is not trivial because the dependencies among attributes cannot be evaluated without using actual private values.

To address this issue of clustering, we propose a new algorithm for estimating the dependencies among attribute from independently perturbed multi-dimensional data under LDP. Our proposed method combines the RR-independent scheme [5], which estimates marginal probabilities of attributes randomized for each attribute, with the RR-Ind-Joint scheme [10], which performs the inverse of the aggregated randomization matrices. Based on the estimated correlation between pairs of attributes, we construct an algorithm for estimating the joint probability distribution for multi-dimensional data. We conduct an experiment using several open-source datasets and show that this estimation accuracy improves as the number of individuals increases.

Our contributions of this paper are as follows.

- We propose a new algorithm for the privacy-preserving clustering of multi-dimensional data, based on the estimated dependencies between multiple attributes. The privacy of the computation is guaranteed under LDP.
- We also present an algorithm that estimates the w -way joint probability estimation of a subset of w attributes. The accuracy of the estimation with regard to the size of dataset (number of individuals) is derived.
- We have conducted experiments with several open-source datasets to show the performance of the proposed algorithm with regard to the dimensionality, the number of users, and the privacy budget used for randomization.

The remainder of the paper is organized as follows. Section 2 gives some fundamental definitions and discusses some representative existing works. We show our problem statement and the proposed algorithms in Sect. 3. Our proposed algorithm has two primitive schemes, a privacy-preserving clustering and an

estimation of joint probability distributions. Section 4 provides the specification of open-source datasets used for the experiment and the results obtained. The experimental results demonstrates that the proposed algorithm outperforms the conventional approaches in terms of estimation accuracy when the data dimensionality is high. We conclude our work in Sect. 5.

2 Fundamental Definitions

2.1 Randomized Response (RR)

An RR [19] is a mechanism whereby each individual masks their own data before forwarding them to an aggregator. Each response item is randomly replaced by a new item with probabilities determined by a randomization matrix.

Let X be a set of d elements, labeled $1, \dots, d$ without loss of generality. A $d \times d$ matrix of probabilities $P = \begin{pmatrix} p_{11} & \cdots & p_{1d} \\ \vdots & \ddots & \vdots \\ p_{d1} & \cdots & p_{dd} \end{pmatrix}$ is a *randomization matrix* of X if and only if $p_{i1} + \cdots + p_{id} = 1$ for $i = 1, \dots, d$ and p_{uv} is the conditional probability of a randomized element being v , given that the true element is u , (i.e., $p_{uv} = Pr(Y = v|X = u)$ for all $u, v \in \{1, \dots, d\}$).

An RR is a randomized mechanism whereby input X of d possible values a_1, \dots, a_d is randomized to give the response $Y = (y_1, \dots, y_n)$ according to P , i.e.,

$$y_i = \begin{cases} x_i & \text{with } p = p_{ii} = \frac{e^\epsilon}{e^\epsilon + d - 1}, \\ v \in \Omega - \{x_i\} & \text{with } q = p_{ij} = \frac{1}{e^\epsilon + d - 1}. \end{cases}$$

More specifically, if we let π_1, \dots, π_d be the proportions of respondents whose true values fall in each of the d values in X and let λ_a be the empirical probabilities of the observed values, we can write $(\lambda_1, \dots, \lambda_d)^T = P^T(\pi_1, \dots, \pi_d)^T$. According to Warner [19], an unbiased estimator π can be computed as $\hat{\Pi} = (P^T)^{-1}\hat{\lambda}$, where $\hat{\lambda} = (\hat{\lambda}_1, \dots, \hat{\lambda}_d)^T$ is the vector of observed empirical probabilities for Y .

2.2 Generalized Randomized Response (GRR)

This protocol applies RR to the generalized domain of multi-dimensional data with m domains, $\Omega_1, \dots, \Omega_m$. The full domain is $\Omega = \Omega_1 \times \cdots \times \Omega_m$ with d^m possible elements. A private value $x_i \in \Omega$ is perturbed to give

$$y_i = \begin{cases} x_i & \text{with } p = \frac{e^\epsilon}{e^\epsilon + d^m - 1}, \\ v \in \Omega - \{x_i\} & \text{with } q = \frac{1}{e^\epsilon + d^m - 1}. \end{cases}$$

This satisfies ϵ -LDP because $p/q = e^\epsilon$. Given the perturbed Y , the aggregator estimates the joint probability $\hat{\pi}(x)$ as $\hat{\pi}(x) = \frac{c(x) - q^n}{p - q}$, where $c(x)$ is the count of x in Y .

The Optimized Unary Encoding (OUE) [18] perturbs the input with the optimal probability 0.5 to minimize the variance of the estimation. The perturbation and the estimation are given by $p = 1/2$ in the GRR.

2.3 RR-Clusters

Domingo-Ferrer and Soria-Comas [5] proposed RR-Clusters. This splits the set of attributes into clusters according to mutual dependencies and estimates the joint probabilities for each cluster (RR-Joint). The inter-clusters joint probabilities are given by the product of intra-cluster estimations (RR-Independent) because the dependencies among attributes are assumed to be neglectable.

RR-Joint. This is a natural way to apply RRs to multiple attributes. Given attributes (A_1, \dots, A_m) , we consider the Cartesian product $A_1 \times \dots \times A_m$ as a single attribute and perform RR on it. The distribution of the true data is estimated as

$$\hat{\Pi}_{\text{RR-Joint}}^{(X_1, \dots, X_m)} = (P^T)^{-1} \hat{\lambda}^{X_1, \dots, X_m}. \quad (1)$$

RR-Joint is severely affected by the curse of dimensionality, because the number of value combinations for $A_1 \times \dots \times A_m$ grows exponentially with the number of attributes m .

RR-Independent. Each party applies RR independently for each of the m attributes X^1, \dots, X^m in a dataset to give $Y = (Y^1, \dots, Y^m)$, where $Y^j = \text{RR}_{P^j}(X^j)$. After estimating the marginal probabilities for the j -th attribute as $\hat{\pi}^j = (P^j)^T)^{-1} \hat{\lambda}^j$, the joint probability distribution for X^1, \dots, X^m is estimated by the product of the marginal distributions as

$$\Pi_{\text{RR-Ind}}^{(X_1, \dots, X_m)}(a_1, \dots, a_m) = \hat{\pi}^1(a_1) \cdots \hat{\pi}^m(a_m). \quad (2)$$

The issue with RR-Independent is that it only yields an accurate estimate when the independence assumption among attributes is (approximately) true.

2.4 RR-Ind-Joint

A multi-dimensional LDP scheme [10] randomizes all attributes independently, and then aggregates these randomization matrices into a single aggregated matrix. The multi-dimensional joint probability distributions are then estimated via matrix inversion. Here, the inverse of the aggregated randomization matrix can be computed efficiently at a lightweight computation cost, linear with regard to dimensionality, and with manageable storage requirements.

With $\epsilon = \log(3)$, $p = \frac{e^{\log(3)}}{e^{\log(3)} + d - 1} = 3/4$, and $q = 1 - p$, the randomization matrix for $\Omega = \{a_1, a_2\}$ $P = \begin{pmatrix} p & q \\ q & p \end{pmatrix} = \begin{pmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{pmatrix}$ satisfies ϵ -LDP. RR-Ind-Joint treats the two independent randomization matrices $P^A = P^B = P$ as a single aggregated matrix

$$P^A \otimes P^B = \begin{pmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{pmatrix} \otimes \begin{pmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 9 & 3 & 3 & 1 \\ 3 & 9 & 1 & 3 \\ 3 & 1 & 9 & 3 \\ 1 & 3 & 3 & 9 \end{pmatrix}.$$

With the same privacy budget $\epsilon = \log(3)$, GRR [19] (RR-Joint [5]) perturbs the full domain $\Omega_A \times \Omega_B$ with $p = \frac{3}{3+2^2-1} = 1/2$, represented as the randomization matrix

$$P_{GRR} = \frac{1}{6} \begin{pmatrix} 3 & 1 & 1 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 3 \end{pmatrix}.$$

2.5 Independence of Attributes

Cramer's V statistics [3] gives a value between 0 and 1, with 0 indicating the complete independence between two attributes. Cramer's V_{ij} is defined as

$$V_{ij} = \sqrt{\frac{\chi_{ij}^2/n}{\min(d_i - 1, d_j - 1)}}, \quad (3)$$

where d_i is the number of values in attribute A^i and χ_{ij}^2 is the chi-squared independence statistic defined as

$$\chi_{ij}^2 = \sum_{a=1}^{d_i} \sum_{b=1}^{d_j} \frac{(o^{ij}(a, b) - e^{ij}(a, b))^2}{e_{ab}^{ij}}, \quad (4)$$

for which $o^{ij}(a, b)$ and $e^{ij}(a, b)$ are the observed and the expected frequencies of the combinations a and b , respectively.

3 Proposed Scheme

3.1 Problem Statement

Our goal is to perturb multi-dimensional data to obtain an LDP privacy guarantee, while being able to use the perturbed data to estimate the joint probability distributions for the true data.

Consider n individuals, each with a record of m attributes (their respective true answers). Each attribute has a domain Ω_i of possible values. The full domain for the m attributes is $\Omega = \Omega_1 \times \dots \times \Omega_m$. Each respondent uses RR to perturb their private answer x_i^1, \dots, x_i^m to give y_i^1, \dots, y_i^m and submits the latter to a central server. Given this perturbed data Y^1, \dots, Y^m , where $Y^i = (y_1^i, \dots, y_n^i)$, and a randomization mechanism (dependent on the privacy budget ϵ of LDP), the central server aims to estimate the w -way joint probability distribution $\hat{\Pi}^S$ of a subset S of $w \leq m$ attributes, without having access to the respondents' true data X^1, \dots, X^m .

We wish to obtain a solution that estimated probabilities should be close to the true probabilities. That is, $\hat{\Pi}^S \approx \Pi^S$ for any S . See Table 1 for the list of symbols.

Table 1. List of symbols

Symbol	Description
n	The number of responders
S	A subset of attributes
w	Dimensionality of S ($w \leq m$)
A_i	i -th attribute
Ω_i	Domain of i -th attribute A_i
d_i	The number of values in Ω_i
Π^S	The joint probability distribution for S

Table 2. Example of the full contingency table

C	c_1		c_2	
	b_1	b_2	b_1	b_2
a_1	20	0	20	0
a_2	10	20	10	20

3.2 Ideas

Generally, attributes in multi-dimensional data will have some dependence. If we can identify clusters of attributes such that the various clusters are regarded as nearly independent of each other, we can estimate the joint probability distributions efficiently. By neglecting the dependencies among attributes in different clusters, we reduce the dimensionality of the data, according to the size of the clusters, and improve the accuracy of the estimation.

Example 1. Consider a three-dimensional dataset X for $n = 100$ individuals with three attributes A , B , and C , where the values are from domains $\Omega_A = \{a_1, a_2\}$, $\Omega_B = \{b_1, b_2\}$, and $\Omega_C = \{c_1, c_2\}$. Suppose that the full contingency table is given as in Table 2.

Attributes A and B are well associated, but C is independent of A and B . Cramer's V between A and B is 0.6, while that between C and both A and B is 0 (completely independent). Therefore, the optimal clusters are $\{A, B\}, \{C\}$. To estimate the 3-way joint probability function, there are several possible approaches: (1) estimating attributes A, B, C independently and taking their products, (2) estimating 2-way marginals for (A, B) first and multiplying it with that of C , (3) estimating 2-way marginals for (A, C) first and multiplying it with that of B , and (4) estimating 3-way marginals for (A, B, C) directly. The results (shown only for $C = c_1$) are as follows.

$$\begin{aligned} \hat{\Pi}^A \hat{\Pi}^B \hat{\Pi}^C &= \begin{pmatrix} 12 & 8 \\ 18 & 12 \end{pmatrix} \quad (\text{MAE}^{A,B,C} = 0.08), \\ \hat{\Pi}^{AB} \hat{\Pi}^C &= \begin{pmatrix} 20.5 & 9.5 \\ 0.5 & 12 \end{pmatrix} \quad (\text{MAE}^{AB,C} = 0.0025), \\ \hat{\Pi}^{AC} \hat{\Pi}^B &= \begin{pmatrix} 12.4 & 7.6 \\ 18.6 & 11.4 \end{pmatrix} \quad (\text{MAE}^{AC,B} = 0.081), \\ \hat{\Pi}^{ABC} &= \begin{pmatrix} 20 & 0 \\ 10 & 20 \end{pmatrix} \quad (\text{MAE}^{ABC} = 4 \cdot 10^{-17}). \end{aligned}$$

Using the proper cluster $\{A, B\}$ reduces the computational cost for estimating 3-way joint probabilities with minimum accuracy loss. Note that using the

inappropriate cluster $\{A, C\}$ would produce an estimation failure just as bad as ignoring all dependencies.

Finding the optimal clusters, however, is not trivial because the dependencies of attributes cannot be estimated without disclosing private values. To address this issue, Domingo-Ferrer and Soria-Comas [5] proposed several possible approaches, as follows. (1) Using a trusted party who holds all private records (Sect. 4). (2) Using an approximation algorithm for the covariance from the perturbed data (Sect. 4.1). (3) Using a secure sum protocol using with secret sharing (Sect. 4.2) or a public-key algorithm with homomorphic properties. However, the approximation the covariance is not entirely accurate, and the secure protocols are costly with regard to both computation and communication. Therefore, the secure and accurate approach is preferred.

To address this issue, we propose a new algorithm for estimating V for any two attributes. Having users perturb their records of m columns independently under LDP, an aggregator estimates χ^2 statistics for any two combinations of m attributes. The estimated χ^2 statistics follows Cramer's V , which gives the total independency among attributes. Based on the estimated dependencies being used as the similarities between pairs of attributes, an arbitrary clustering algorithm can be performed. We propose a hierarchical clustering scheme that allows adaptive control over the size of clusters.

The proposed algorithm is based on two building blocks, namely RR-Independent and RR-Ind-Joint, and runs in $\mathcal{O}(d)$ computation time, where d is the maximum of the domain sizes. It can be executed without needing expensive secure sum protocols. The accuracy of the estimation is improved as the number of users increases.

Attribute-independent perturbation works not only for clustering but also for the privacy budget. A single private value is used for both the clustering of attributes and the estimation of the joint probabilities for clusters. However, the straightforward use of a conventional LDP scheme such as GRR [19] or OUE [18] over the full domain of attributes in a cluster spends the addition of privacy budgets, according to the sequential composition theorem for the differential privacy [11]. In Example 1, a user needs to perturb a_1 with ϵ for clustering A and perturb again with ϵ for cluster $\{A, B\}$, resulting in 2ϵ of the budget being spent. By contrast, RR-Ind-Joint allows estimation of joint probabilities from the original independently perturbed values without spending additional privacy budget.

3.3 Privacy-Preserving Clustering

Given independently perturbed data $Y = (Y^1, \dots, Y^m)$, the central server estimates the joint probabilities of any pair of attributes A_i and A_j via either of two schemes, RR-Independent and RR-Ind-Joint, to obtain $\hat{\Pi}_{Ind}^{A_i, A_j}$ and $\hat{\Pi}_{IndJoint}^{A_i, A_j}$, respectively. The former provides the mean joint probability between A^i and A^j under the assumption that A^i and A^j are independent. The latter estimates the 2-way joint probability between A^i and A^j . Hence, the central server can obtain the estimation of χ_{ij}^2 statistics as

$$\hat{\chi}_{ij}^2 = \sum_{a \in \Omega_i} \sum_{b \in \Omega_j} \frac{n(\hat{\Pi}_{IndJoint}^{A_i A_j}(a, b) - \hat{\Pi}_{Ind}^{A_i A_j}(a, b))^2}{\hat{\Pi}_{Ind}^{A_i A_j}(a, b)}. \quad (5)$$

Theorem 1. *The estimation $\hat{\chi}_{ij}^2$ tends to χ_{ij}^2 statistics as the number of responders n increases.*

Proof. First, we note that $\hat{\Pi}_{Ind}(a, b) = Pr(a)Pr(b)$ and $\hat{\Pi}_{IndJoint}(a, b) = Pr(a, b)$ for any $a \in A_i$ and $b \in A_j$ from Eq. (2). Second, noticing that the n perturbations are identically and independently distributed, we can regard the frequencies of the perturbed value y_i as following a Binomial distribution with probability $\hat{\Pi}_{Ind}(a, b)$. Therefore, the mean of the Binomial distribution gives the expected frequency as $n\hat{\Pi}_{Ind}(a, b) \approx e^{ij}(a_i, a_j)$. Similarly, $n\hat{\Pi}_{IndJoint}(a, b) \approx o^{ij}(a_i, a_j)$, as the observed frequency for (a, b) . Replacing $e^{ij}(a_i, a_j)$ and $o^{ij}(a_i, a_j)$ in Eq. (4) by $n\hat{\Pi}_{Ind}(a, b)$ and $n\hat{\Pi}_{IndJoint}(a, b)$, respectively, we have Eq. (5). Finally, we consider the estimation error. [10] derived (in Theorem 6) an upper bound to the estimation error of RR-Ind-Joint. By specifying $w = 2$ for the pair of attributes and d for the domain size, the upper bound of the estimation error in the joint probability is $(1 + \frac{d-1}{e^\epsilon})^2 \frac{d^2}{n}$, which tends to zero as n increases. Wang et al. [18] showed that the variance in the estimation of frequencies for RR is $Var[\Pi_{RR}] = \frac{d-2+e^\epsilon}{(e^\epsilon-1)^2 n}$, which also tends to zero as n increases. Therefore, in Eq. (4), both estimations $\hat{\Pi}_{Ind}$ and $\hat{\Pi}_{IndJoint}$ approximate $e^{ij}(a, b)$ and $o^{ij}(a, b)$ accurately when n is sufficiently large.

It is straightforward to obtain the clusters based on the independence between pairs of attributes. The central server uses the estimated χ^2 statistics to compute Cramer's V statistics given in Eq. (3). With the similarity of attributes A_i and A_j being defined as $1 - V_{ij}$, the server can execute an arbitrary clustering algorithm to find the clusters.

Algorithm 1 shows the overall procedure for the privacy-preserving clustering of attributes.

3.4 Joint Probability Estimation Based on Clusters

Given the clusters of attributes, we aim to estimate the w -way joint probability of S .

First, we define an equivalence relation between attributes. Attributes A_i and A_j are V -equivalent if and only if $V_{ij} \leq 1 - v^*$, where v^* is the threshold above which two attributes are considered independent. Using V -equivalent, we can easily obtain the clusters of attributes such that two attributes are V -equivalent. Consider a set of clusters $C_{v^*} = \{C_1, \dots, C_\ell\}$. Note that $C_i \cap C_j = \emptyset$ for any $i \neq j$ and $C_1 \cup \dots \cup C_\ell = \{A_1, \dots, A_m\}$.

Next, we partition the subset S of attributes belonging to the V -equivalent class. The V -equivalent class of S with a set of clusters C_{v^*} is defined as

$$S/C_{v^*} = \{S' \subset S \mid \exists C_i \in C_{v^*}, S' \subset C_i\}.$$

Algorithm 1. Privacy-preserving clustering

- 1: The i -th responder perturbs the private input x_i with ϵ and sends the perturbed data y_i to a central server.
 - 2: The central server collects all y_1, \dots, y_n .
 - 3: **for all** $i \neq j \in \{1, \dots, m\}$ **do**
 - 4: Estimates Π_{Ind}^{ij} and Π_{Ind}^{ij} .
 - 5: Estimates $\hat{\chi}_{ij}^2$ in Eq. (5).
 - 6: Estimates \hat{V}_{ij} in Eq. (3).
 - 7: $h_{ij} \leftarrow 1 - \hat{V}_{ij}$.
 - 8: **end for**
 - 9: Perform a clustering algorithm with $h_{i,j}$.
 - 10: **return** clusters C_1, \dots ,
-

Algorithm 2. Joint probability estimation with clusters

Require: $Y_1, \dots, Y_m \leftarrow$ independently perturbed data.

Require: $C_1, \dots, C_\ell \leftarrow$ clusters of attributes.

- 1: $S/C_{v^*} \leftarrow \{S' \subset S \mid \exists C_i \in C_{v^*}, S' \subset C_i\}$.
 - 2: $\hat{\Pi}_{C_{v^*}}^S \leftarrow \Pi_{s \in S/C_{v^*}} \hat{\Pi}_{IndJoint}^s$.
 - 3: **return** the estimated joint probabilities $\hat{\Pi}_{C_{v^*}}^S$.
-

Note that S is divided into several clusters where attributes depend each other with degrees of $1 - v^*$.

Finally, we show the joint probability estimation based on the clusters of attributes. For the V -equivalent class of S , the w -way joint probability estimation is performed as

$$\hat{\Pi}_{C_{v^*}}^S = \Pi_{s \in S/C_{v^*}} \hat{\Pi}_{IndJoint}^s \quad (6)$$

Note that the notation Π represents the product of all joint estimations $\hat{\Pi}_{IndJoint}$.

Example 2. Consider a set of clusters $C_{v^*} = \{C_1, C_2, C_3, C_4\}$ such that $C_1 = \{A_{workclass}, A_{occupation}\}$, $C_2 = \{A_{education}, A_{income}\}$, $C_3 = \{A_{marital}, A_{relation}, A_{sex}\}$, and $C_4 = \{A_{race}\}$. (This is the example from the Adult dataset for $v^* = 0.3$.) A subset $S = \{A_{sex}, A_{income}, A_{relation}\}$ is divided into several clusters according to their V -equivalent class as $S/C_{v^*} = \{\{A_{income}\}, \{A_{relation}, A_{sex}\}\}$

The ($w = 3$)-way joint probability distribution is given in Eq. (6) as, $\hat{\Pi}_{C_{v^*}}^S = \hat{\Pi}_{IndJoint}^{A_{income}} \cdot \hat{\Pi}_{IndJoint}^{A_{relation}A_{sex}}$. The size of the cluster increases as the threshold v^* decreases (stricter independence). For two thresholds of independence $v^* = 0.015$ and 0.3 , the estimations are $\hat{\Pi}_{C_{0.015}}^S = \hat{\Pi}_{IndJoint}^{A_{income}, A_{relation}, A_{sex}}$, $\hat{\Pi}_{C_{0.3}}^S = \hat{\Pi}_{IndJoint}^{A_{income}} \hat{\Pi}_{IndJoint}^{A_{relation}} \hat{\Pi}_{IndJoint}^{A_{sex}}$, respectively.

Algorithm 2 shows the steps in estimating the joint probability distribution for subset S given an optimal clustering of attributes.

Table 3. Dataset specifications

dataset	description	# records n	# att. w	domain $ \Omega_i $
Adult	UCI Census Income Data [1]	32,561	8	1.8e+06
Census	US Census Data (1990) [12]	2,458,285	68	1.7e+44
Credit	German Credit Data (2000) [8]	1,000	13	3.5e+07
Nursery	Enrollment Data in 1980's Nursery school [14]	12,960	9	6.5e+04

3.5 Performance

The most expensive part of the estimation algorithm is matrix inversion, which has $\mathcal{O}(d^{2.807})$ complexity, with d being the domain size for the attribute. With w -element subset S , the Castell matrix inversion [10] runs in time proportional to w .

Theorem 2. *Let w and d be the number of attributes of S and the maximum domain size of ω attributes of S , respectively. Algorithm 2 runs in $\mathcal{O}(wd^{2.807})$ complexity time.*

Proof. Strassen's algorithm [16], known as the best-performing algorithm, performs the inversion of a d^2 matrix in $\mathcal{O}(d^{2.807})$ time. Algorithm 2 requires a matrix inversion for each of the w attributes. Therefore, the total time complexity is $\mathcal{O}(wd^{2.807})$.

4 Evaluation

4.1 Data

To investigate the estimation accuracy with regard to independence, we conducted some experiments using open-source (publicly available) datasets.

Table 3 shows the specifications for the open-source datasets used in evaluating the performance of the proposed schemes.

4.2 Methodology

We evaluated the estimation accuracy in terms of the average variation distance (AVD) between the real joint probability distributions Π^C and the estimated distributions $\hat{\Pi}^C$. The AVD is defined as $AVD = \frac{1}{|A|} \sum_{C \in A} \sup_{c \in C} |\Pi^c - \hat{\Pi}^c|$, where A is a power set of attributes such that C has w distinct attributes.

4.3 Results

Figure 1 shows a scatter plot for the real and estimated Cramer's V statistics for all combinations of eight attributes in the Adult dataset (only categorical attributes). It shows the estimation varying with privacy budgets $\epsilon = 1$ and 2.

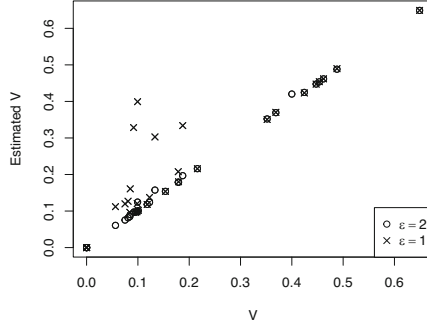


Fig. 1. AVDs with regard to the full domain size $|\Omega|$

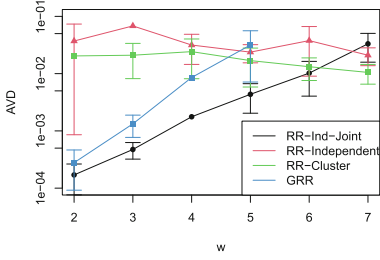
Note that Eq. (5) estimates the V statistics correctly from perturbed data with $\epsilon = 2$, but it is less accurate when V is small for $\epsilon = 1$. Pearson’s correlations between the real and estimated V are 0.999 and 0.867 for $\epsilon = 2$ and 1, respectively.

Figure 2a shows the AVDs for the Adult dataset, as estimated in four LDP schemes: GRR [19], RR-Independent [5], RR-Ind-Joint [10] and the proposed RR-clusters. We varied the dimensionality $w = 2, \dots, 7$ to show the accuracy reduction with regard to increasing w . Using a logarithmic scale, we plotted the AVDs at a 68% confidence interval, estimated as $\mu \pm \sigma$ for 20 trials. Figure 2b shows the dendrogram of the clusters created via a hierarchical clustering algorithm. Using a threshold $v^* = 0.3$ ($h = 1 - v^* = 0.7$), we obtained four clusters of attributes.

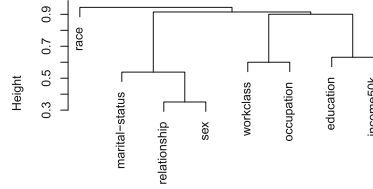
Note that GRR for $w > 5$ is infeasible because of its full domain size. RR-clusters had a the minimum estimation error at $w = 7$. The AVDs were 0.0105, 0.0201, and 0.0332 for RR-Clusters, RR-Independent and RR-Ind-Joint, respectively. For $w < 7$, the results for RR-Clusters are between those for RR-independent and RR-Ind-Joint. Its AVD becomes closer to the AVD for RR-Ind-Joint as the threshold v^* decreases.

Figure 2c shows the distribution of AVD with regard to the number of users n . Here, we randomly sampled n users from the dataset. The results show that the AVD of the clusters slightly decreases as n increases. The estimation based on clusters has the least AVD for $n \leq 1000$ in the experiment, but the difference is not statistically significant. We therefore consider that RR-independent is more robust against a paucity of users and contributes the estimation accuracy based on clusters.

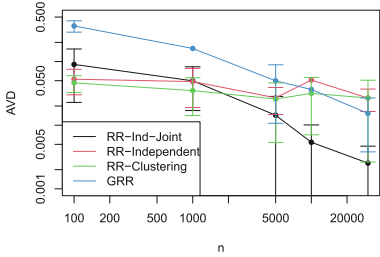
In Fig. 2d, we show the AVDs for four algorithms with regard to the privacy budget ϵ at $w = 4$. The AVDs based on clusters are mostly between those for RR-Ind-Joint and RR-independent. Note that the estimation accuracy is relatively stable across the range of ϵ .



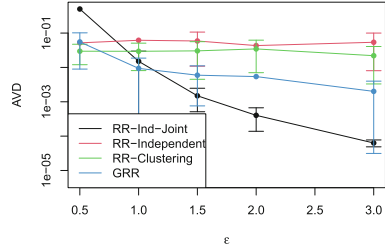
(a) AVDs for the four algorithms with regard to the dimensionality w



(b) Clusters



(c) AVDs for the four algorithms with regard to the number of users n



(d) AVDs for the four algorithms with regard to privacy budget ϵ

Fig. 2. Results for the Adult dataset

The accuracy improvement with clusters depends on the overall independence of the attributes within datasets. In Fig. 3, we show the AVDs for the four datasets with regard to the dimensionality w . If there are sufficient users in a dataset and most attributes are significantly associated (3a US Census), RR-Ind-Joint provides the most accurate estimation. By contrast, when the number of users is limited, the cluster-based approach works efficiently for higher dimensional data, as shown for the Credit and Nursery datasets ($w > 4$ in Fig. 3b, $w > 5$ in Fig. 3c).

Figure 4 shows the dendrograms of attributes for the four datasets with hierarchical clustering. It shows that most of the attributes in Fig. 4a (US Census) are associated, as plotted in the lower area of the dendrogram (Vs are greater). Therefore, RR-Ind-Joint had the lowest AVDs.

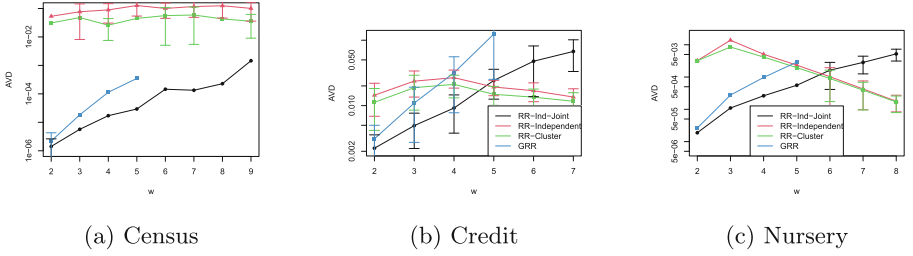


Fig. 3. AVDs between real and estimated values for w -way joint probabilities with regard to dimensionality w

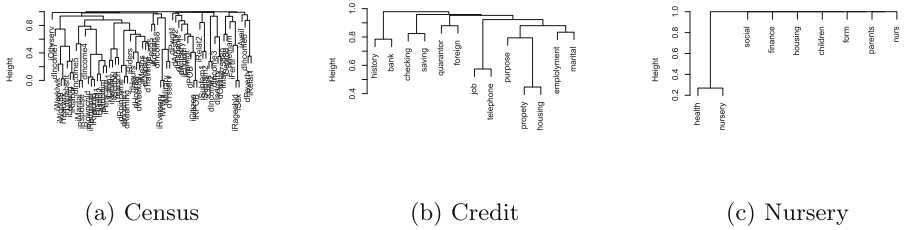


Fig. 4. Hierarchical clustering

5 Conclusion

In this paper, we have studied the randomization of multi-dimensional data under conditions of LDP. We proposed two algorithms for privacy-preserving clustering (based on χ^2 statistics estimated from the independently randomized data) and for estimating w -way joint probability distributions (based on optimal clustering of attributes).

Our experiments with open-source datasets showed that Cramer’s V statistics of attributes can be estimated accurately (with 0.999 Pearson’s correlations between the real and estimated values). We found that the proposed algorithm can outperform the baseline GRR, RR-Independent and RR-Ind-Joint schemes in terms of estimation accuracy for high-dimensional data with sufficient numbers of users. The improvements in accuracy were most apparent for high dimensionality: $w > 6$, $w > 4$ and $w > 5$ for the Adult, Credit, and Nursery datasets, respectively. However, it was less effective when used with a large dataset (Census).

Acknowledgment. Part of this work was supported by JSPS KAKENHI Grant Number JP18H04099, 23K11110 and JST, CREST Grant Number JPMJCR21M1, Japan. Author thanks prof. Josep Domingo-Ferrer and Dr. Jordi Soria-Comas for discussion and suggestions.





References

1. Bache, K., Lichman, M.: Adult data set (2013). <https://archive.ics.uci.edu/ml/datasets/adult>
2. Chen, R., Xiao, Q., Zhang, Y., Xu, J.: Differentially private high-dimensional data publication via sampling-based inference (2015)
3. Cramér, H.: *Mathematical Methods of Statistics*. Princeton University Press, Princeton (1946)
4. Ding, B., Kulkarni, J., Yekhanin, S.: Collecting telemetry data privately. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, pp. 3574–3583. Curran Associates, Inc (2017)
5. Domingo-Ferrer, J., Soria-Comas, J.: Multi-dimensional randomized response. *IEEE Trans. Knowl. Data Eng.* **34**(10), 4933–4946 (2022)
6. Dwork, C., Roth, A.: *The Algorithmic Foundations of Differential Privacy*, vol. 9. Now Publishers Inc., Hanover (2014)
7. Erlingsson, U., Pihur, V., Korolova, A.: *Rappor: Randomized Aggregatable Privacy-preserving Ordinal Response*. Association for Computing Machinery, New York (2014)
8. Hofmann, H.: Statlog (german credit data) data set. <https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29>
9. Jiang, X., Zhou, X., Grossklags, J.: Privacy-preserving high-dimensional data collection with federated generative autoencoder. *Proc. Priv. Enhancing Technol* **2022**(1), 481–500 (2022)
10. Kikuchi, H.: Castell: scalable joint probability estimation of multi-dimensional data randomized with local differential privacy (2022). <https://doi.org/10.48550/arXiv.2212.01627>, [arXiv:2212.01627](https://arxiv.org/abs/2212.01627)
11. McSherry, F.D.: *Privacy Integrated Queries: An Extensible Platform for Privacy-preserving Data Analysis*. Association for Computing Machinery, New York (2009)
12. Meek, C., Thiesson, B., Heckerman, D.: The learning curve method applied to clustering. In: Richardson, T.S., Jaakkola, T.S. (eds.) *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research*, vol. R3, pp. 196–202. PMLR (2001)
13. Qardaji, W., Yang, W., Li, N.: *Privid: Practical Differentially Private Release of Marginal Contingency Tables*. Association for Computing Machinery, New York (2014)
14. Rajkovic, V., et al.: Nursery data set. <https://archive.ics.uci.edu/ml/datasets/Nursery>
15. Ren, X., et al.: LoPub: high-dimensional crowdsourced data publication with local differential privacy. *IEEE Trans. Inf. Forensics Secur.* **13**(9), 2151–2166 (2018)
16. Strassen, V.: Gaussian elimination is not optimal. *Numer. Math.* **4**, 354–356 (1969)
17. Tolstikhin, I., Bousquet, O., Gelly, S., Schoelkopf, B.: Wasserstein auto-encoders. In: *International Conference on Learning Representations* (2018)
18. Wang, T., Blocki, J., Li, N., Jha, S.: Locally differentially private protocols for frequency estimation. In: *26th USENIX Security Symposium (USENIX Security 17)*, pp. 729–745. Vancouver, BC (2017)
19. Warner, S.L.: Randomized response: a survey technique for eliminating evasive answer bias. *J. Am. Stat. Assoc.* **60**(309), 63–69 (1965)
20. Xu, C., Ren, J., Zhang, Y., Qin, Z., Ren, K.: DPPro: differentially private high-dimensional data release via random projection. *IEEE Trans. Inf. Forensics Secur.* **12**(12), 3081–3093 (2017)

21. Zhang, J., Cormode, G., Procopiuc, C.M., Srivastava, D., Xiao, X.: Privbayes: private data release via Bayesian networks. *ACM Trans. Database Syst. (TODS)* **42**(4), 1–41 (2017)
22. Zhang, Z., Wang, T., Li, N., He, S., Chen, J.: CALM: Consistent Adaptive Local Marginal for Marginal Release Under Local Differential Privacy. Association for Computing Machinery, New York (2018)



Hierarchical Model-Based Cybersecurity Risk Assessment During System Design

Tino Jungebloud¹ , Nhung H. Nguyen² , Dong Seong Kim² ,
and Armin Zimmermann¹ 

¹ SSE Group, Technische Universität Ilmenau, Ilmenau, Germany
{tino.jungebloud, armin.zimmermann}@tu-ilmenau.de

² School of ITEE, The University of Queensland, St Lucia, Australia
{hongnhung.nguyen, dan.kim}@uq.edu.au

Abstract. Cybersecurity risk assessment has become a critical priority in systems development and the operation of complex networked systems. However, current state-of-the-art approaches for detecting vulnerabilities, such as automated security testing or penetration testing, often result in late detections. Thus, there is a growing need for security by design, which involves conducting security-related analyses as early as possible in the system development life cycle. This paper proposes a novel hierarchical model-based security risk assessment approach that enables the early assessment of security risks during the system design process. The approach uses different OMG UML-based models, supplemented by a lightweight extension using profiles and stereotypes. Various security attributes, including vulnerability information and asset values, are then used by algorithms to compute relevant properties including threat space, possible attack paths, and selected network-based security metrics. A real-life industrial example is then used to demonstrate the approach.

Keywords: Cybersecurity · Model-Based Systems Engineering

1 Introduction

Cybersecurity has become an increasingly important topic in the development and operation of all kinds of computing systems. Protecting confidential data, guaranteeing data integrity and availability, and maintaining system functionality are essential goals of cyber security. One common approach to analyzing cybersecurity-related properties of systems is information security risk assessment. All activities and efforts targeting risk assessment aim to answer the following questions: (i) *what attack scenarios can happen in the given context* (ii) *what impact can a particular attack trigger, and how severe are attacks*, (iii) *how to respond to the identified risks*, and finally (iv) *how should risks be treated with respect to monitoring*. The current state of the art in systems and software security includes automated security testing, such as penetration testing (pentest)

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 30–44, 2024.

https://doi.org/10.1007/978-3-031-56326-3_3

and dynamic application security testing (DAST). However, these tests are often conducted only after development activities are completed or when the system is already in a productive environment. Unfortunately, this approach can lead to severe undetected vulnerabilities and the opportunity for remote exploitability. This is especially problematic in critical systems that human health or life may depend on, such as vehicles. Addressing these issues later can be challenging and expensive, and may damage an organization’s reputation and finances. Existing works demonstrated the potential impact of these issues. For example, Nie et al. [15] showed in their study how an attack chain could compromise a Tesla Model S network by injecting malicious CAN messages. Another example is given by Cai et al. [1], they demonstrated how they exploited vulnerabilities in BMW cars to gain full control of them. The vulnerabilities that led to unauthorized control of the car in these examples were due to many reasons, but one of them was the lack of security concerns in the system design process. However, assessing security risks during the system design and development process is a challenging task. When the system is not yet fully implemented, it is hard to identify all potential vulnerabilities and attack surfaces. To address this problem, we propose a novel approach to start basic security analyses as early as possible in the system development process. Our approach utilizes a UML-based model-driven engineering process, where existing or newly created system models are annotated using a lightweight extension by stereotypes. As far as we know, there hasn’t been any previous research that integrates Hierarchical Attack models with UML for comprehensive security analysis.

The main contributions of this paper are summarized as follows: (1) Introducing a seamless hierarchical model for assessing security risk that enables early security evaluation in system development. (2) Supporting the OMG UML standard for maximum interoperability with existing system models and/or other UML-based development tool-chains. (3) Developing an automated procedure to transform the UML model into graphs and using various quantitative metrics to construct a threat space and score builder. This allows for the automated calculation of the overall security posture of the system. (4) Evaluating the effectiveness of the proposed model in a real-world in-vehicle network, as demonstrated in a case study conducted by Keen Security Lab at Tencent in 2019 [1].

The rest of this paper is organized as follows: Sect. 2 presents a review of related literature. The proposed approach is described in Sect. 3. In Sect. 4, we apply our approach to an automotive in-vehicle network as a case study. Finally, we conclude this paper in Sect. 5.

2 Related Work

In this section, we briefly discuss the existing related works in two areas: (1) Cybersecurity risk assessment using OMG UML/SysML, and (2) other security models and metrics.

Security Models Using UML/SysML: SysML Version 1 [18] is an extension of UML [17] which is used for specifying and analyzing complex systems. Aprville

and Roudier [21] introduced the SysML extension called *SysML-Sec* with concepts and constructs to support secure software and hardware components. For modeling and analyzing security of systems, SysML-Sec adds security-specific constructs such as threat models, security goals, security requirements, security constraints, and security mechanisms. Pedroza et al. [19,20] employed SysML to conduct security risk analysis. The method relies on the principles of safety and security co-engineering and is aligned with the EUROCAE method specifications, including ED-203A [5] and ED-202A [6]. Specifically, SysML block diagrams are used for architecture modeling, and profile extensions enrich models with security information. This information can be used to compute an attack path from an attack vector to a safety-critical component. Furthermore, the approach supports the identification of attack scenarios by corporations with MITRE CAPEC [12] for attack pattern and MITRE CWE [13] for software weaknesses.

Security Models and Metrics: Shaked et al. [22] proposed a model-based approach for designing cybersecurity-related systems, which considers both cybersecurity threats and system composition. The methodology introduces a domain ontology and a high-level modeling concept to aggregate components into hierarchies, which is similar to the modeling approach used in SysML. The system structure and the allocation of threats to single components are represented using a box-like notation. The methodology helps incorporate cybersecurity into systems design by identifying desirable security controls and associating them as functional capabilities at different levels, based on threats and risks. Another popular method for modeling system and system security is using an attack graph (AG) and an attack tree (AT). Hong and Kim et al. [10] proposed two layers of Hierarchical Attack Representation Models (HARM) to evaluate the cybersecurity of enterprise networks. Their work combined both AGs and ATs to construct a comprehensive model to calculate the attack risk, and attack cost of an update phase of a network. There are numerous works extending HARM to meet the characteristics of different networks [2]. For example, HARM can be used to assess the effectiveness of the system when applying several different Moving target defense (MTD) techniques [11] to model the security of different types of networks, such as IoT networks [7,8] or maritime vessel networks [4]. Another approach to modeling system security is to use qualitative methods. Monteuis et al. [14] presented a SARA framework for systematically analyzing threats and assessing risks of autonomous vehicles. This framework takes into account various factors that can impact the network security of the system, such as human factors, infrastructure sign recognition, the ability to control the car, and the severity of attacks. The security risk of the system was defined by a risk matrix function that considers factors including attack likelihood, controllability of the car, and attack severity.

Summary: The literature review presents various approaches for security modeling and risk assessment. However, some methods do not consider the system design or provide a means of modeling the system during the development process. Furthermore, some methods require modifications to the standard UML notation or necessitate starting over to conduct security analysis. Hence, we pro-

Table 1. Symbols, Definitions, and Functions

SYMBOL	DEFINITION
<i>Graph & Tree</i>	
\mathbb{N}	Set of all components/nodes in the system, $\mathbb{N} = \{n_1, n_2, n_3, \dots, n_i\}$
\mathbb{V}	Set of all vulnerabilities in the system
v_j	Vulnerability j of the system, such that $v_j \in V$
α_{v_j}	Impact of vulnerability v_j
<i>Paths</i>	Set of <i>Attack Paths</i> in the system. Paths between an <i>attack entry point</i> and an <i>attack target</i>
<i>Surface</i>	Attack Surface (set of <i>attack entry points</i>), $\Delta = \{\delta_1, \delta_2, \dots, \delta_n\}$, where $\Delta \subseteq \mathbb{N}$, $\delta_i \in \mathbb{N}$
<i>Assets</i>	Set of all <i>Assets</i> in the system (attack target), $\eta_i \in \mathbb{N}$
<i>T-Space</i>	Threat Space: Set of possible <i>attack entry point to attack target</i> combinations \rightarrow <i>Surface</i> \times <i>Assets</i> = $\{(x, y) \mid (x \in \text{Surface}) \wedge (y \in \text{Assets})\}$ (Cartesian product)
<i>UML Model Views (Diagrams)</i>	
$\mathbb{D}_{csd}^{H_i}$	Composite Structure Diagram at model hierarchy level i . Shows the internal structure of a classifier, classifier interactions with the environment through ports, or behavior of a collaboration
\mathbb{D}_{class}	Class Diagram. Shows the static structure of classifiers and their relationships in a system, structural features (attributes), behavioral features (operations)
<i>UML Model</i>	
\mathbb{M}	Set of all model elements of type <code>uml::Element</code>
\mathbb{C}	Set of all Classifiers in model \mathbb{M} used as Components in one of the Composite Structure Diagram $\mathbb{C} = \{c_1, c_2, \dots, c_i\}$
\mathbb{P}	Set of Encapsulated Classifier attributes in the UML model used as ports of Components and parts in the Composite Structure Diagram $\mathbb{P} = \{p_1, p_2, \dots, p_j\}$
\mathbb{E}	Set of all UML Connectors in model \mathbb{M} . Connections between Component ports $\mathbb{E} \subseteq \{\mathbb{P} \times \mathbb{P}\}$
c_i	One component in the system or sub-system, $c_i \in \mathbb{C}$
p_j	One port classifier attribute of a component c_i . $p_j \in \mathbb{P}$

pose the use of standard UML with profile extensions for system modeling and risk assessment using information read from the model. Our proposed approach is presented in the next section.

3 Proposed Approach

This section outlines the workflow of our hierarchical model-based security risk assessment, along with the security metrics used in the process. We introduce the general modeling approach using UML [17], the program interfaces, and tools used to build the model and conduct the security assessment at each phase of the workflow. Subsequently, we present the intuition and semantics of used paradigms, applied algorithms, and transformations, as well as the selected

metrics for conducting security assessment at each phase. To ensure a clear and unambiguous technical description, we introduce the definitions, symbols, and functions in Table 1. Our workflow consists of four phases, as shown in Fig. 1: (1) Graphical system modeling using a UML modeling tool, parameterization, and transformation into executable C++ source code; (2) Model analysis for collecting key properties for further steps; (3) Calculation of the *Threat Space* and the determination of *Attack Paths* using intermediate tree/graph data structures and graph algorithms; and (4) Calculation of selected metrics (e.g., *n Attack Path* level as well as *Network* level) to quantify cyber-security risk. Next, we describe each phase in detail.

Phase 1: Graphical System Modeling, Parameterization, and Model-Transformation. In this phase, we use the UML Designer [16, 24], which is a free and open-source UML modeling tool based on Eclipse Sirius to create graphical models. Our approach relies on UML meta-model and Composite Structure Diagrams as a graphical representation. To ensure that we can generate source code, apply security analysis, and facilitate the framework MDE4CPP [9, 23], we define a precise way to model security components as follows: (1) Composites with `uml::Part` attributes are used for functional decomposition. (2) The concept of *Port on Part* utilizes strongly typed `uml::Connector` to model logical and physical communication channels between entities of type `uml::Encapsulated Classifier`. (3) The communication channels are modeled using Class Diagrams, including *Interface Classes*, *Required Interface*, and *Provided Interface*.

The hierarchical decomposition of our approach is defined as follows: (1) At the highest hierarchy level (H0), system modeling begins by defining the *Security Scope*. This is done by adding an entity of type `uml::Class` to represent the *System Environment*, along with other entities of the *System Environment*. For example, H0 may include two UML entities: `uml::Class` representing the *Attacker*, and `uml::Class` representing the *System* itself. (2) Every sub-system is then modeled as the next hierarchy level ($H1, H2, H3, \dots, Hn$), which includes the entities that require analysis later on. For example, a higher level functional block represented as a `uml::Class`, can be added to the model repository,

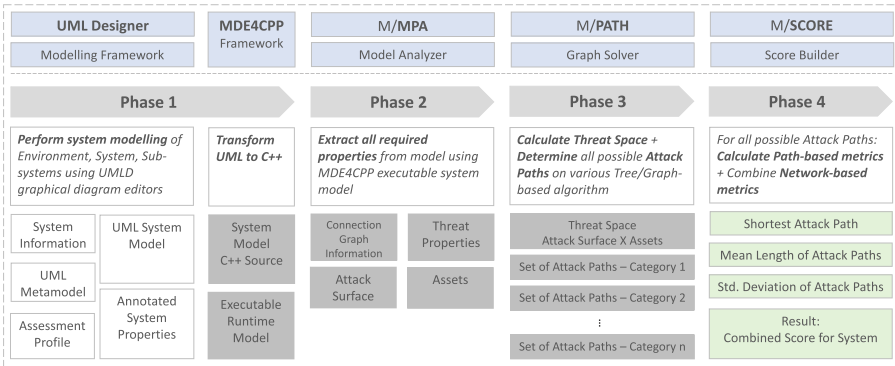


Fig. 1. A Workflow for a hierarchical model-based cyber-security risk assessment.

followed by adding `Class::ownedAttribute` to the relevant component. In this way, during the system development process, higher hierarchy levels can be added to provide a full model of the system. The way to break down the functions into higher levels depends on the available information at the time of modeling and the analysis to be performed later. Figure 2 illustrates the connection between different hierarchy levels as well as the diagrams used for the case study in Sect. 4. Once the system modeling is complete, a profile extension is applied to the model. New stereotypes are defined along with Metaclass relationship, as shown in Fig. 3. These stereotypes enable adding semantics to the model with numeric and text values and named enumeration constants. Each stereotype property has a descriptive name for improved understandability. Specifically, components in the system are annotated with stereotypes `CSElement`, `CSAsset` and `CSMeasure`, while ports can have stereotype `CSPort`. All stereotype values added to the model are later used in algorithms and/or security metrics. Next, the UML model is transformed into executable C++ source code using the MDE4CPP [23] framework.

Phase 2: Model Analysis and Intermediate Data Structures. In this phase, the UML model from the previous phase is used to create a graph data structure for later analysis. To achieve this, we use the MDE4CPP:uml4cpp generator, which outputs C++ interface headers and implementation sources for every model element linked into shared libraries. This includes the complete UML meta-model, enabling versatile model analysis on both meta-model and model layers. The model analysis algorithm, referred as *M/MPA*, is developed

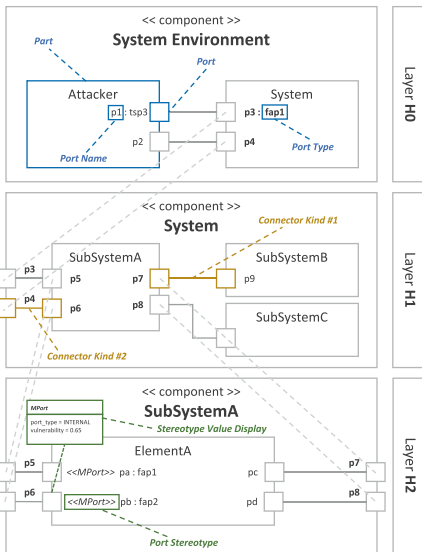


Fig. 2. Hierarchical decomposition with UML Composite Structure Diagrams and Stereotype Visualization.

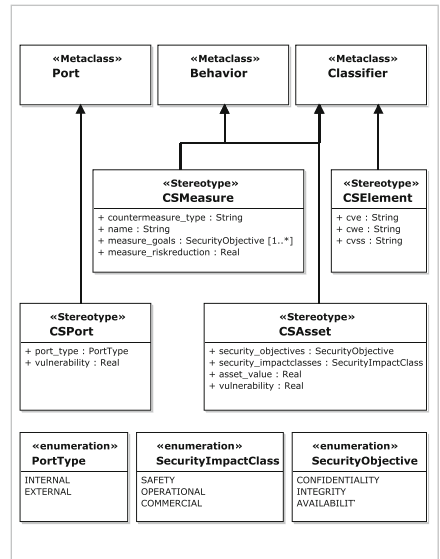


Fig. 3. Extract of the Risk Assessment UML profile including Meta-Class relationships.

Algorithm 1: Algorithm for Phase 2: UML Model Analyzer (M/MPA).

```

input : UML meta-model based system model  $\mathbb{M}$ .
result : Data structures representing the network topology of the system as
          graph  $G = (V, E)$ ;  $Surface \subseteq V$ ;  $Assets \subseteq V$ .

Initialize:  $V, E, Surface, Assets = \emptyset$ ;
parts  $\leftarrow$  FindAllUMLParts();
foreach  $v_i \in$  parts do
     $V \leftarrow v_i$ ;
     $p_i \leftarrow$  FindPortsOnPart( $v_i$ );
    foreach  $v_j \in$  parts \  $v_i$  do
         $p_j \leftarrow$  FindPortsOnPart( $v_j$ )
        if CompareInterface( $p_i[k], p_j[k]$ ) then
             $E \leftarrow e(v_i, v_j)$ ;
        end
    end
end

```

in C++ to enable access to the object model in Phase 1. The details and principles of this algorithm are outlined in Algorithm 1. The required input is the UML model itself, denoted by \mathbb{M} . The algorithm initializes internal variables for handling graph elements **Node** and **Edge**. It then creates an instance of the system by calling the MDE4CPP model factory. On this instance level, method `::FindAllUMLParts()` is executed, resulting in applying a set of `uml::Class` elements with a `StructuredClassifier::part:Property` role. In the next step, the interface matching loop is run on all *Part* pairs in the system. If method `::CompareInterface()` detects a communication pair, the corresponding edge $E \leftarrow e(v_i, v_j)$ is added. Finally, the graph data structure G as well as the lists of *Surface* and *Assets* elements are written to files in JSON format.

Phase 3: Calculation of Threat Space and Attack Paths. In this phase, we conduct two tasks as outlined in Algorithm 2. Firstly, we calculate the $ThreatSpace = Surface \times Assets$, which is the Cartesian product of the two element lists: **Surface**, **Assets**, obtained from the result of previous phase. The *Threat Space* can be interpreted as a list of possible end-to-end relations, that start at components representing an entry point and end at components that require protection against the loss of one or more security goals (\rightarrow *confidentiality*, *integrity*, *availability*). Essentially, it helps to identify potential security threats and vulnerabilities in the system by representing all possible combinations of the *Surface* and *Assets* elements. Secondly, the program then proceeds to determine *Attack Paths* by reading the graph data structure from phase 2 and executing graph algorithms. We use two popular graph algorithms, Breadth First Search (BFS) and Depth First Search (DFS), to traverse and search the graph data structure to find the path(s) from the entry point to the target node.

Algorithm 2: Algorithm of Phase 3: Graph Solver (M/PATH).

input : Network topology as graph $G = (V, E)$; $Surface \subseteq V$; $Assets \subseteq V$.
result : $T\text{-Space}$: $\{(x, y) \mid (x \in Surface) \wedge (y \in Assets)\}$;
 $Paths$: $\{P_i\}_{i=1..n} \mid P_i = (v_1, \dots, v_N) : (v_i, v_{i+1}) \in E \text{ for } 1 \leq i \leq N - 1$.

Initialize: $t\text{space}, \text{paths} \leftarrow \emptyset$;
Initialize: $\text{graph} \leftarrow G$;

Step 1: Calculation of Threat Space;
 $t\text{space} \leftarrow \text{surface} \times \text{assets}$;

Step 2: Calculation of Paths;
forall the $\text{pair} \in t\text{space}$ **do**
 $\text{start} \leftarrow \text{pair.first}$;
 $\text{target} \leftarrow \text{pair.second}$;
 $\text{paths} \leftarrow \text{paths} + \text{doPathSearch}(\text{graph}, \text{start}, \text{target})$;
end

Phase 4: Calculation of Security Metrics and Overall Network Security Risk. This is the final phase in our approach. Here, security metrics are used to calculate risk scores on both the *Attack Path* and *Network* level. We obtain a selection of path-based metrics (cf. Enoch et al. [3, p. 12]) to analyse the attack paths. These security metrics are summarized in Table 2. This includes the Number of Attack Paths (NAP), which is the number of routes that the attacker can take to compromise the target in the system. The higher the number of attack paths, the less secure the system.

The Shortest Attack Path (SAP) denotes the length of the shortest way for the attacker to reach the target. As we use the unweighted graph for an attack graph, the number of nodes in the shortest path is the minimum number of components that the attacker needs to pass to compromise the target. The Mean of Attack Path Lengths (MAPL) equals the average value of all the attack path lengths. This metric shows the average effort that the attacker may have to apply to be able to reach the target. Finally, the Standard Deviation of Attack Path Lengths (SDPL) describes the standard deviation value of all the attack path lengths to understand if the MAPL value is representative or not.

4 Case Study: BMW In-Vehicle Infotainment System

In this section, we describe the use of our proposed method and metrics using an example of an automotive in-vehicle network. Specifically, we (1) briefly present the case study with network model, attack model, and defense model, (2) describe the application of our proposed approach to assess the cybersecurity risk of this in-vehicle networks.

4.1 Case Study Description

System Model: To demonstrate the practical applicability of the proposed model, we use the real in-vehicle architecture and attack scenarios that were

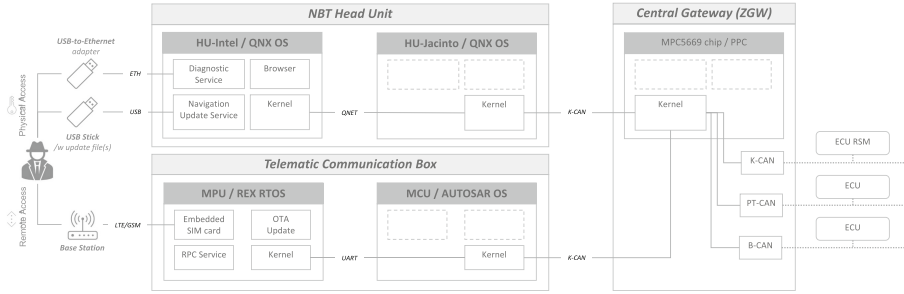


Fig. 4. An example automotive in-vehicle network with attack vectors/paths

previously introduced [1]. Their work analyzed the architecture of a BMW car and provided a comprehensive security analysis with different attack vectors to the car. Figure 4 shows an example in-vehicle network. Attackers can achieve their attack goals through multiple entry points including (1) USB-to-Ethernet adapter enabling Ethernet-based access; (2) USB Stick containing automatically loaded update file(s); and (3) LTE/GSM wireless access. The main focus is on two ECUs of the car, the Head Unit (HU) and Telematic Unit (TU), which connect to the outside network of the car. These units serve as the main attack vectors that allow the attackers to potentially access the inside network of the car remotely or physically.

Attacker Model: Our assumption is that the attacker’s goal is to gain physical control over at least one of the car’s ECUs, which could allow them to manipulate various aspects of the car’s body (such as door, light, horn control) or more seriously, active or deactivate the car, or manipulate the airbag system. The attacker can use one of two approaches to achieve this goal: physical or remote attack. *Physical attack:* An attacker can exploit vulnerabilities in the diagnostic or navigation services by using USB (USB-to-Ethernet adapter; USB Stick) to gain root shell access to HU-Intel. Once this is compromised, the attacker can log-in to HU-Jacinto and send CAN messages to ECUs on K-CAN Bus. Furthermore, the attacker can send diagnostic messages to different CAN BUS from the QNX hu-intel x86 to control different ECUs. *Remote attack:* The attacker can

Table 2. Selected Attack Path metrics

Metric Name	Abbrev.	Formula
Number of Attack Paths	NAP	$ Paths $
Shortest Attack Path	SAP	$\min_{\forall Path \in Paths} Path $
Mean of Attack Path Lengths	MAPL	$\frac{\sum_{Path \in Paths} Path }{NAP}$
Standard Deviation of Attack Path Lengths	SDPL	$\sqrt{\frac{\sum_{Path \in Paths} (Path - MPM^2)}{NAP}}$

Table 3. Vulnerabilities information in the in-vehicle network with Sequential Number, CVE ID, CVSS3 Base Score, Impact, Likelihood and CVSS3 Base Vector String.

No.	CVE	CVSS	I.	LH.	CVSS3 Base Vector String
v_1	CVE-2012-3748	7.7	5.3	1.8	AV:N/AC:H/PR:L/UI:N/S:C/C:L/I:H/A:L
v_2	CVE-2018-9322	7.8	5.9	1.8	AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
v_3	CVE-2018-9314	6.8	5.9	0.9	AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
v_3	CVE-2018-9312	7.8	5.9	1.8	AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
v_4	-	6.7	4.7	1.5	AV:L/AC:L/PR:H/UI:N/S:C/N/I:H/A:L
v_5	CVE-2018-9311	9.8	5.9	3.9	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

set up a rogue base station to force the car to connect to their network, using it as a gateway to access the HU and TU of the vehicle. The attacker can then compromise the Browser and escalate privilege to get the root access to the Kernel of the Intel x86 chip. Using a rogue base station, the attacker can also remotely compromise the kernel of the Qualcomm Kernel of the Telematic Unit. With the root privilege to the Head Unit and Telematic Unit, the attacker is able to send diagnostic messages to different CAN BUS to control different ECUs. Table 3 shows the details of the vulnerabilities described above. We use the Common Vulnerabilities and Exposures (CVE)/National Vulnerability Database (NVD) to collect the information about the vulnerabilities. We use the metric of the base score from the Common Vulnerability Scoring System (CVSS) version 3.0 to acquire the value of impact and likelihood of the vulnerability. For those vulnerabilities that do not have version 3.0 score yet or do not have CVE-ID, we read the description of the vulnerability to set the parameters in the CVSS score system to find out the estimated value for it.

Defense Model: The aim of the defense strategy is to achieve the highest level of security. To achieve this, we consider using two defense methods: (a) *Patch Management* to address vulnerabilities and weaknesses with software patches. However, closing some vulnerabilities may be complex and time-consuming, so we assume that only a few vulnerabilities can be patched. (b) *Intrusion Detection and Prevention (IDPS)* to monitor occurring events in the systems communication network and detection of signs of possible incidents. However, the computational performance of in-vehicle components is limited. Therefore, only a reduced set of functions can be applied to specific types of attacks. We assume that an IDPS can only be applied at the Central Gateway to monitor CAN Buses to the connected ECUs, or to prevent the HU and TU from sending malicious UDS messages.

4.2 Applying Our Approach to Assess Security Risk

System Modeling, Parameterization, and Transformation: We use system model and attack model information to conduct UML modeling. To emulate the entire system design process, we divide the modeling into three levels that

Table 4. Analysis Results with AP Length (L), Asset Value (V), Score (S1), Score when applying Patching defense (S2), Score when applying IDS (S3), and AP Components.

L.	V.	S1	S2	S3	Attack Path Components	
1	5	3.5	2.92	1.36	1.56	ATK_USB ▷ USB ▷ HU_C1 ▷ HU_C2 ▷ ECU_RSM
2	5	7.5	2.92	1.36	1.56	ATK_USB ▷ USB ▷ HU_C1 ▷ HU_C2 ▷ ECU_RSM.FNC
3	4	9.5	3.90	1.95	3.90	ATK_USB ▷ USB ▷ NUS ▷ KHUC1
4	5	3.5	2.92	1.36	1.56	ATK_USBETH ▷ USBETH ▷ HU_C1 ▷ HU_C2 ▷ ECU_RSM
5	5	7.5	2.92	1.36	1.56	ATK_USBETH ▷ USBETH ▷ HU_C1 ▷ HU_C2 ▷ ECU_RSM.FNC
6	4	9.5	1.95	0.00	1.95	ATK_USBETH ▷ USBETH ▷ DS ▷ KHUC1

to generate the corresponding *Threat Space*, and identify all *Attack Paths*. The result data is stored as JSON file and the set of *Attack Paths* are given in Table 4.

Security Metrics and Overall Risk: In this step, the path-based metrics are calculated. We use the vulnerabilities information in Table 3 as an input for the metrics in Table 2. Specifically, the input values (vulnerabilities) for the network was as follows: *BROWSER* = 7.7 (CVE-2012-3748), *HU* = 7.8 (CVE-2018-9322), *HU_C1* = 7.8 (CVE-2018-9322), *KHUC1* = 7.8 (CVE-2018-9322), *HU_C2* = 6.8 (CVE-2018-9314), *TU* = 9.8 (CVE-2018-9311), *NUS* = 9.8 (CVE-2018-9312). Then, the security metrics are calculated as follows: *NAP* = 6; *SAP* = 4; *MPL* = 4.67; *SDPL* = 0.47; Finally, an overall risk score is calculated. Considering the six possible attack paths, the overall risk score is 13.63.

Application of Defense Models: To enhance the security of our system, we have implemented two defense strategies: patch management and the use of IPDS. We are measuring the impact of these strategies on the overall security score of the system through calculation and comparison. To analyze the impact of *Patching management*, we changed input vulnerability value of components *HU*, *HU_C1* and *KHUC1* to 0.00. This can be interpreted as solving software issues related to CVE-2018-9322. The risk scores of affected Attack Paths are given in Table 4 column *S2*. The overall risk score of this defense model is 5.83. When using patch management, it would be ideal to patch all known and possible vulnerabilities. However, this is not practical in the real world, as patching can be costly and time-consuming. Therefore, we choose one CVE in the head unit with the highest CVSS score to see how the overall risk changes. In practice, our proposed method can help choose the most suitable patch management approach. As for other defense measure, we implement an *IDPS* to enhance system's security posture. Specifically, we assume that the IDPS is designed to detect and prevent malicious software updating behavior that may be initiated by an attacker who has physical access to the system. In our case study, this behavior is associated with the CVE-2018-9314 vulnerability. To model this, we change the vulnerability score of component *HU_C2* to 0.00. With recalculation of all metrics, the new risk scores are given in Table 4 column *S3*. Overall risk score of this defense model is 9.10. Through this example, our proposed framework enables the identification of potential attack paths and calculation

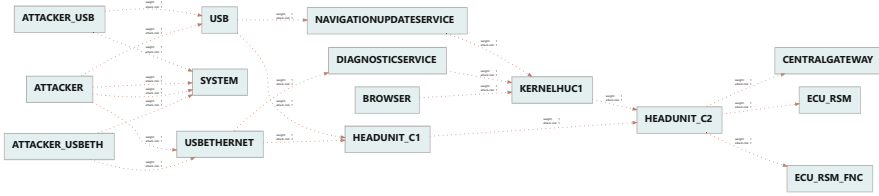


Fig. 6. System Connection Graph.

of the overall risk of the system. Additionally, by using the framework, one can assess and compare the effectiveness of different security strategies, helping to ensure the most effective security strategies are implemented. The analysis of our example demonstrates a significant reduction in the overall security risk. Using *Patching management*, the risk score is lowered from $13.63 \rightarrow 5.83$. For comparison, the *IDPS* approach merely lowers the risk score from $13.63 \rightarrow 9.10$.

This comprehensive approach provides insights into the security of the system design as well as improves its overall security posture.

Limitations and Future Work: There are some limitations to consider. Firstly, in the case study hierarchical layers are used to emulate three abstract levels at different stages of the system design process. Therefore, it may not fully represent the complexity of real-world industry system design processes. The model needs validation with a test-bed that represents the system in the early stages of development. Additionally, there has been no comprehensive comparison of the proposed approach with other models in terms of effectiveness. Secondly, the proposed approach has not yet considered modeling the dynamic evolution of the system’s security state. Future work is needed to validate its effectiveness with different types of networks and to address system behavior by modeling the transition state of the system. In future work, we plan to extend this study by implementing UML Composite Structure transformation to Petri Nets, obtaining stochastic metrics to analyze the change in the system state, and then calculating some quantitative security values such as attack probability and mean time to compromise at different levels of the system. The model introduced in this paper will be the basis for this model-to-model transformation.

5 Conclusions

This paper has presented a novel security assessment approach for the system development process using a hierarchical attack modeling approach based on standard UML models and domain-specific UML profile extensions. The proposed model includes an automated procedure to transform the UML model into graphs and calculate the overall security of the system. A real-world in-vehicle example demonstrates the applicability of our approach in actual systems.

Acknowledgements. This work was made possible by RTAPHM (Real-Time Analytic, Prognostics and Health Management) and MISU (Model-based Development of Secure Digital Infrastructures for Service-Driven UAV Systems), reference numbers 20X1720A and 20X1736E. Partially funded by the Federal Ministry for Economic Affairs and Climate Action (BMWK). The statements made herein are solely the responsibility of the authors.

References

1. Cai, Z., Wang, A., Zhang, W.: 0-days & Mitigations: Roadways to Exploit and Secure Connected BMW Cars (2019)
2. Enoch, S.Y., Ge, M., Hong, J.B., Kim, D.S.: Model-based cybersecurity analysis: past work and future directions. In: 2021 Annual Reliability and Maintainability Symposium (RAMS) (2021)
3. Enoch, S.Y., Hong, J.B., Ge, M., Kim, D.S.: Composite metrics for network security analysis (2020)
4. Enoch, S.Y., Lee, J.S., Kim, D.S.: Novel security models, metrics and security assessment for maritime vessel networks. *Comput. Netw.* **189**, 107934 (2021)
5. European Organisation for Civil Aviation Equipment: ED-202A - Airworthiness Security Process Specification (2014)
6. European Organisation for Civil Aviation Equipment: ED-203A - Airworthiness Security Methods and Considerations (2018)
7. Ge, M., Cho, J.H., Kim, D., Dixit, G., Chen, I.R.: Proactive defense for internet-of-things: moving target defense with cyberdeception. *ACM Trans. Internet Technol.* **22**, 1–31 (2021)
8. Ge, M., Hong, J.B., Guttman, W., Kim, D.S.: A framework for automating security analysis of the Internet of Things. *J. Netw. Comput. Appl.* **83**, 12–27 (2017)
9. Hammer, M., Maschotta, R., Wichmann, A., Jungebloud, T., Bedini, F., Zimmermann, A.: A model-driven implementation of PSCs specification for C++. In: Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development (2022)
10. Hong, J.B., Kim, D.S.: HARMs: hierarchical attack representation models for network security analysis. In: 10th Australian Information Security Management Conference (2012)
11. Hong, J.B., Kim, D.S.: Assessing the effectiveness of moving target defenses using security models. *IEEE Trans. Dependable Secure Comput.* **13**, 163–177 (2016)
12. MITRE: CAPEC - Common Attack Pattern Enumeration and Classification (2023). <https://capec.mitre.org>
13. MITRE: CWE - Common Weakness Enumeration (2023). <https://cwe.mitre.org>
14. Monteuis, J.P., Boudguiga, A., Zhang, J., Labiod, H., Serval, A., Urien, P.: SARA: security automotive risk analysis method. In: Proceedings of the 4th ACM Workshop on Cyber-Physical System Security, pp. 3–14 (2018)
15. Nie, S., Liu, L., Du, Y.: Hacking Tesla From Wireless to CAN BUS (2017)
16. OBEO: UML Designer (2023). <https://www.uml designer.org>
17. Object Management Group: Unified Modeling Language, Version 2.5.1 (2017)
18. Object Management Group: Systems Modeling Language, Version 1.6 (2019)
19. Pedroza, G.: Towards safety and security co-engineering: challenging aspects for a consistent intertwining. In: Hamid, B., Gallina, B., Shabtai, A., Elovici, Y., Garcia-Alfaro, J. (eds.) CSITS ISSA 2018. LNCS, vol. 11552, pp. 3–16. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16874-2_1

20. Pedroza, G., Mockly, G.: Method and framework for security risks analysis guided by safety criteria. In: 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (2020)
21. Roudier, Y., Apvrille, L.: SysML-Sec - a model driven approach for designing safe and secure systems. In: 2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD) (2015)
22. Shaked, A., Reich, Y.: Model-based threat and risk assessment for systems design. In: Proceedings of the 7th International Conference on Information Systems Security and Privacy (2021)
23. SSE: Model-driven Software Engineering for C++ (2023). <https://github.com/MDE4CPP>
24. SSE: UML Designer - TUI.SSE branch (2023). <https://github.com/MDE4CPP>



The Influence of Privacy Concerns on Cryptocurrency Acceptance

Peter Hamm^(✉) , Sebastian Pape , and Kai Rannenberg

Goethe University Frankfurt, Frankfurt, Germany
peter.hamm@m-chair.de

Abstract. Despite the hype, cryptocurrencies have to far failed to establish themselves as a means of payment for everyday transactions, spawning a wealth of research into acceptance factors and obstacles for cryptocurrency adoption. Our paper adds to this literature by investigating the role of organizational privacy concerns and risk perceptions on cryptocurrency acceptance. Employing a representative survey of German e-commerce users with 257 respondents we find that while risk perceptions and concerns about data collection do affect adoption willingness for cryptocurrencies, neither are useful for predicting actual adoption behavior. This is especially notable since the lack of central counterparties that may steal funds or personal data was one of the original motivations for the creation of the first cryptocurrencies. Our results provide insight into the nature of cryptocurrency adoption and highlights a discrepancy between intention and behavior.

Keywords: Cryptocurrencies · Technology adoption · Concern for information privacy

1 Introduction

When first appearing on the scene, cryptocurrencies such as Bitcoin were hailed as a revolutionary technology that could transform the financial industry by obviating the need for trusted central counterparties [46]. However, in spite of the hype, Bitcoin and other cryptocurrencies have so far failed to establish themselves as a tool for everyday payments [26], with most use limited to investment purposes [17], as well as some for illicit activities [16]. The extant literature identified perceived risk, trust, as well as lack of self-efficacy as core reasons for this [1, 33, 48, 49]. This is notable, since an important part of the original justification for Bitcoin and other cryptocurrencies was the lack of central counterparties that one would need to trust [36]. Against this backdrop, we want to investigate how privacy concerns specifically towards online entities affect cryptocurrency adoption as a payment system, due to these concerns being a *raison d'être* for cryptocurrencies in the first place. These concerns have received little attention in the cryptocurrency adoption literature, even though they have been shown to be of high importance for similar technologies such as electronic [35, 37]

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 45–58, 2024.

https://doi.org/10.1007/978-3-031-56326-3_4

and mobile payments [34,54], as well as for the burgeoning area of central bank digital currencies [44]. We aim help closing this research gap by answering the following research questions:

- RQ1** How do risk perception and organizational privacy concerns affect the adoption willingness for cryptocurrencies?
RQ2 How do risk perception and organizational privacy concerns affect the adoption behavior for cryptocurrencies?

2 Background

2.1 Privacy Concerns

Privacy has long been a topic of discussion and research, with a myriad of definitions given. As far back as 1890, Brandeis and Warren [6] defined privacy as an inherent “right to be left alone”. Later, Westin [51] defined privacy more narrowly as the right to prevent the disclosure of personal information, and identified the issue of privacy concerns. Due to privacy itself being a multifaceted variable that may be impossible to measure directly, privacy concerns offer a useful proxy for privacy issues, and have emerged as the central construct in most empirical research work on the topic [41], although other terms such as “privacy beliefs” or “privacy attitudes” are sometimes used as well [53]. The importance of privacy concerns in e-commerce and internet services is well established [11, 28, 45, 50].

Probably the first very popular instrument to measure privacy concerns in an e-commerce setting was the *Concern for Information Privacy (CFIP)* construct [42], which focuses on organizational information privacy practices and has proven itself as a reliable measure of individual privacy concerns in past studies [43]. The instrument employs four sub-scales: *collection*, which covers concerns about the scale of data collected and stored, *unauthorized secondary use*, which covers use of the data for purposes not originally intended and agreed for, *errors* covering concerns about intentional or unintentional errors in the stored data, and *improper access*, which encapsulates the concern that individuals not authorized to view the data may still access it [42]. A further development is the *Internet Users’ Information Privacy Concerns (IUIPC)* [30], which is a more complex second-order construct that has found use for modelling human factors in privacy enhancing technologies [18, 22]. More recently, the *Antecedents, Privacy Concerns, Outcomes* or *APCO* model [41] introduced a full causal model incorporating a number of antecedents as well as outcome variables such as trust and behavioral reactions.

Due to the organizational focus of our research question, we will employ the *CFIP* as we consider it to be the most appropriate measure for our study.

2.2 Acceptance of Cryptocurrencies

The adoption of technologies has long been a lively area of research [10, 15, 47], and found successful application in areas such as e-commerce and mobile payments [7, 8]. In the area of cryptocurrencies, prior work applied the Technology

Acceptance Model (TAM) [10] on Bitcoin, adapting the model by including perceived risks and perceived benefits, finding that the effect of perceived risks on Bitcoin use behavior had both a higher effect size as well as higher statistical significance compared to perceived benefits [1]. Another approach using the TAM but adding perceived trust also found a significant effect of perceived risk on the intention to use cryptocurrencies for C2C e-commerce applications, albeit with a smaller effect size than perceived trust or perceived usefulness [33]. This is in contrast to another study by Arias-Oliva et al. [2], who integrate perceived risk into the UTAUT model [47], but do not find it to be a statistically significant factor predicting cryptocurrency adoption intention [2]. However, it should be noted that for both aforementioned studies, the operationalization of perceived risk did not consider privacy risk directly. Looking specifically at payment transactions with cryptocurrencies Mashatan et al. found perceived information privacy risk, anonymity, and traceability to significantly affect trust, which in turn had a significant effect on the intention to use crypto-payments, while no significant evidence was found for a role of perceived information security fraud risk [32].

On the influence of privacy perceptions, previous research has found that existing users of Bitcoin tend to rate their concerns as either low or medium [14].

A related topic that has more recently attracted significant attention are central bank digital currencies (CBDC). These currencies, that would enable households to hold central bank money directly without participation of the private financial sector, were first seriously discussed in the form of cryptocurrencies issued by central banks [4], although researchers quickly argued that these currencies will not be true decentralized cryptocurrencies [5]. One large empirical study with more than 1000 respondents concerning a potential digital Euro found that privacy concerns exhibit a negative effect on the willingness to use this currency [44]. The study found a strong effect of soft trust factors (i.e. credibility, image, and security) on both privacy concerns as well as the willingness to use the currency; other significant antecedents of privacy concerns in the digital Euro were perceived vulnerability from the currency, self-efficacy, and general information privacy concerns. Another paper employing the privacy-calculus found evidence that privacy concerns do have a negative influence on the willingness of customers to use a CBDC and thus disclose personal information, although they may still be willing to do so if the offered benefits of this technology outweigh these concerns [24].

3 Methodology

In this section, we briefly cover the development of the questionnaire, the data collection and the research model. We estimate how concerns for information privacy, as defined by the CFIP construct, as well as risk perceptions, affect the willingness to use cryptocurrencies as a means of payment.

3.1 Questionnaire, Data Collection and Ethical Considerations

The data was collected with the support of a panel provider in Germany (certified following the ISO 20252 norm). The survey was implemented with the software LimeSurvey [39] and hosted on a university server. We sampled the participants in a way to achieve a sample representative of German e-commerce users. For that purpose, we set quotas to end up with approximately 50% females and 50% males in the sample as well as a distribution of age following the EUROSTAT2018 census [13]. The Questionnaire was in German. The English translation can be found in the appendix.

The German translation of the concerns for information privacy (CFIP) [42] construct was taken from the work by Harborth and Pape [21], who employed two independent verified translators and verified the validity and reliability of the translation. To measure risk perception, we consider risk to be composed of four items each representing a form of risk identified in previous literature: legal risk [12, 49], market risk [12, 49], counterparty risk [32], and operational risk [29]. We aggregate these by taking the average of the responses. Finally, to gauge adoption willingness, we asked participants if they have made a purchase with Bitcoin or another cryptocurrency in the past, to which they could reply that they have done so, that they have not done so but that they have considered it, or that they did not even consider it.

The participants were informed about the purpose of the study, the storage location and that they stay anonymous unless they reveal their identity within the free texts. Minors were not allowed to participate. This was ensured by our panel provider and an additional information text before our survey. Participants agreed that their data is used for research and consequent publications. The user study was evaluated by the university’s ethics board and has been classified as “ethically acceptable”.

3.2 Research Model and Hypotheses

To distinguish between *adoption willingness* and *adoption behavior*, we consider the former to include individuals that at least considered using cryptocurrencies in the past, while the latter only includes individuals who have actually done so.

$$\text{adoption willingness} = \begin{cases} 1, & \text{if respondent says they have used} \\ & \text{cryptocurrencies in the past} \\ 1, & \text{if respondent says they have considered} \\ & \text{using cryptocurrencies in the past} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{adoption behavior} = \begin{cases} 1, & \text{if respondent says they have used} \\ & \text{cryptocurrencies in the past} \\ 0, & \text{otherwise} \end{cases}$$

The independent variables are *risk perception (RP)*, and the four CFIP subscales *collection (CO)*, *errors (ER)*, *unauthorized secondary use (US)*, and *improper access (IA)*. As the dependent variables in our dataset are binary, we employ a logistic regression. This type of regression estimates the odds of a random variable being equal to one given a set of independent predictor variables [52, pp. 584–595]. Our general research model is as follows:

$$\begin{aligned} AW_i &= \beta_0 + \beta_1 RP_i + \beta_2 CO_i + \beta_3 ER_i + \beta_4 US_i + \beta_5 IA_i \\ AB_i &= \beta_0 + \beta_1 RP_i + \beta_2 CO_i + \beta_3 ER_i + \beta_4 US_i + \beta_5 IA_i \end{aligned}$$

where AW_i are the log-odds, i.e. the logarithm of the odds ratio, for the respondent i to be willing to use cryptocurrencies, and AB_i the corresponding log-odds for them already using the currencies. We further investigate models where we only look at one CFIP-subscale to gauge the importance of the other items on the result.

Risk perception measures the perceived risk of using cryptocurrencies, consisting of legal, market, counterparty and operational risk. The established literature finds strong evidence that perceived risk is a significant obstacle to cryptocurrency adoption [1, 33], thus we hypothesize:

H1a: Risk perception (RP) has a negative effect on the likelihood of adoption willingness concerning cryptocurrencies.

H1b: Risk perception (RP) has a negative effect on the likelihood of adoption behavior concerning cryptocurrencies.

The privacy concerns are not directly worded in reference to cryptocurrencies in themselves. That is because decentralized cryptocurrencies are operated algorithmically and at least theoretically do not depend on any specific player who has access to personal data. Thus, we are considering how the perception of privacy risk with online entities influences cryptocurrency adoption. These entities may include players in the cryptocurrency ecosystem such as exchanges, as well as merchants or other entities. As the declared original goal of cryptocurrencies was to obviate the need for trusted third parties [36], privacy concerns should make cryptocurrencies more attractive by removing the need to engage with central counterparties and disclose personal information to them. However, some papers have found positive associations between trust in entities like banks and trust in cryptocurrencies [3], indicating that concerns towards the behavior of involved companies may have a different effect on cryptocurrencies than assumed. Still, as the original justification is theoretically sound and has not consistently been disproven, we hypothesize that:

H2a: Concerns about data collection (CO) have a positive effect on the likelihood of adoption willingness concerning cryptocurrencies.

H2b: Concerns about data collection (CO) have a positive effect on the likelihood of adoption behavior concerning cryptocurrencies.

H3a: Concerns about data errors (ER) have a positive effect on the likelihood of adoption willingness concerning cryptocurrencies.

H3b: Concerns about *data errors (ER)* have a positive effect on the likelihood of *adoption behavior* concerning cryptocurrencies.

H4a: Concerns about *unauthorized secondary use (US)* of the data have a positive effect on the likelihood of *adoption willingness* concerning cryptocurrencies.

H4b: Concerns about *unauthorized secondary use (US)* of the data have a positive effect on the likelihood of *adoption behavior* concerning cryptocurrencies.

H5a: Concerns about *improper access (IA)* have a positive effect on the likelihood of *adoption willingness* concerning cryptocurrencies.

H5b: Concerns about *improper access (IA)* have a positive effect on the likelihood of *adoption behavior* concerning cryptocurrencies.

4 Results

This section will describe the statistical results of our analysis. Overall, 257 individuals completed the survey, of which 7 had already used cryptocurrencies, while a further 26 stated that they had at least considered it in the past, leaving 224 individuals who have not even considered doing so. The sample demographics are given in Table 1. The variables *Risk Perception*, *Collection*, *Errors*, *Unauthorized Secondary Use*, and *Improper Access* were computed by taking the average of their corresponding items. None of the resulting variables are normally distributed according to the Shapiro-Wilk-Test [40], with each variable achieving a significance level below 0.0001. Internal consistency was evaluated by employing Cronbach’s alpha [9], as shown in Table 2. All values are higher than the lower limit of 0.7, thus indicating that the individual items measure the same construct, and below the upper limit of 0.95, indicating that none of the items are redundant [19].

Table 1. Demographics of our Sample

Education	N	Percent	Age	N	Percent
Lower secondary education (Hauptschulabschluss)	10	3.8%	18–29	60	22.3%
			30–41	71	27.6%
Secondary school (Realschulabschluss)	70	27.2%	42–53	69	26.8%
University entrance qualification (Abitur)	79	30.7%	54–65	52	20.2%
Bachelors’ degree	38	14.8%	66 and older	5	1.9%
Masters’ degree and equivalent	51	19.8%	Gender	N	Percent
PhD and higher	9	3.5%	Male	132	51.3%
			Female	125	48.6%

The first step of our analysis was to consider whether risk perception or any of the CFIP items were valued significantly different between demographic groups. For age, we divided the sample according to the median age, which

Table 2. Internal consistency and Mann-Whitney U tests for population differences

Variable	Cronbach's alpha	Mann-Whitney U tests		
		Age	Gender	Education
Risk Perception	0.8505	0.0020	0.2713	0.3564
Collection	0.8738	0.0019	0.5126	0.6527
Errors	0.8969	0.0380	0.8674	0.0888
Unauthorized Secondary Use	0.9382	0.0004	0.1635	0.0478
Improper Access	0.9236	0.0045	0.0065	0.0580

p - values, bold for $p < 0.05$.

was 41 in our sample. Gender distinguished between male and female respondents, while for education we drew a line between respondents holding at least a Bachelor's degree and those that did not. We evaluated this by employing the Mann-Whitney U test, a nonparametric test that evaluates whether the distribution of one random variable is larger than another [31]. The results of this evaluation are given in Table 2.

The results show that respondents younger than the median age of 41 were significantly more concerned about cryptocurrencies, but at the same time significantly less concerned about each of the CFIP subscales. In the case of gender, female respondents exhibited more concern about *Improper Access*, while we see no statistically significant differences otherwise. As for education, respondents without university education were significantly more worried about *Unauthorized Secondary Use*, albeit with a significance level barely below 5%, with *Improper Access* barely missing this threshold, and again with respondents without university education scoring higher on the measure on average.

Finally, we want to consider whether Risk Perception or any of the CFIP scales influence the willingness or the actual decision to adopt cryptocurrencies as payment systems. We first use each concern variable separately before employing a model with all variables together.

The results for the willingness to use cryptocurrencies are given in Table 3. We find a negative effect of *Risk Perception* across all models with a stable coefficient, indicating support for hypothesis *H1a*, i.e. higher risk perceptions lower the odds that a respondent did consider using cryptocurrencies for payments. Among the items indicating privacy concerns, only data *Collection* exhibited statistically significant values in the corresponding simple as well as the full model with the expected positive sign, so we can confirm hypothesis *H2a*, indicating that individuals worried about undue collection of their data by online companies are more likely to consider using cryptocurrencies for payment. We did not find support for an effect for data *Errors*, *Unauthorized Secondary Use*, or *Improper Access* of the data, thus we cannot confirm hypotheses *H3a*, *H4a*, or *H5a*.

The regression results for actual use behavior are given in Table 4. Here, only *Unauthorized Secondary Use* is statistically significant in the full model. However, it is not significant in the partial models, i.e. those without the other CFIP factors, and the sign is negative, meaning that high concerns about unautho-

Table 3. Adoption Willingness on CFIP Subscales and Risk Perception

variable	simple models					full model
const	-0.384 (0.551)	-2.679* (0.025)	-1.982 (0.07)	-0.259 (0.843)	0.357 (0.779)	-2.101 (0.172)
RP	-0.316* (0.017)	-0.361** (0.007)	-0.337* (0.011)	-0.315* (0.018)	-0.311* (0.02)	-0.390** (0.005)
CO		0.473* (0.017)				0.591* (0.012)
ER			0.343 (0.069)			0.37 (0.073)
US				-0.02 (0.913)		-0.047 (0.889)
IA					-0.123 (0.502)	-0.422 (0.186)

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$, p-values in parentheses.

alized secondary use by online companies make it less likely that individuals use cryptocurrencies. Thus, even though the coefficient may be statistically significant, we cannot confirm hypothesis $H4b$. Furthermore, *Risk Perception*, which played a significant role for willingness, does not in turn predict actual use of cryptocurrencies. We thus find no support for hypotheses $H1b$, $H2b$, $H3b$ or $H5b$.

5 Discussion

Concerning our first research question, we find strong evidence that risk perception towards cryptocurrencies has a significant influence on adoption willingness across all models, indicating that perceived risk stops individuals from even considering to use cryptocurrencies for payments. As for privacy concerns, only worries about data *collection* had a significant influence on adoption willingness, showing that individuals who think that online companies collect too much data are more interested in using cryptocurrencies. Among the other factors, only concerns about data *errors* had the expected positive effect on willingness on average, but the significance level barely missed the cutoff value of 5%. We found no evidence for an effect of *unauthorized secondary use* or *improper access*.

We do not find support for any of our hypotheses concerning the second research question. Risk perceptions seems to play no role for the actual adoption of cryptocurrencies, with the effect on adoption behavior actually being positive on average (albeit statistically insignificant). This finding adds to the picture developed in prior research that found no effect of risk perception on actual behavior for cryptocurrencies [48], even though an effect on intention was found in a number of earlier studies [1, 33].

Concerning privacy risk, the only variable exhibiting a significant effect on use behavior is *unauthorized secondary use*. However, it is only significant if

Table 4. Adoption Behavior on CFIP Subscales and Risk Perception

variable	simple models					full model
const	-4.197* (0.012)	-6.206* (0.021)	-7.030** (0.005)	-2.551 (0.281)	-5.274 (0.1)	-8.168* (0.022)
RP	0.121 (0.696)	0.063 (0.832)	0.053 (0.855)	0.184 (0.585)	0.101 (0.740)	0.066 (0.838)
CO		0.428 (0.31)				0.725 (0.183)
ER			0.614 (0.126)			0.681 (0.168)
US				-0.316 (0.354)		-1.881* (0.012)
IA					0.185 (0.689)	1.335 (0.098)

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$, p-values in parentheses.

we include every other CFIP variable, and even then the results indicate that individuals worried about unauthorized secondary use of their data by online-companies are in fact less likely to use cryptocurrencies. This may be explained by existing users identifying cryptocurrency exchanges or other market players under the umbrella term “online companies”, where non-users may apply the term to merchants due to their lack of familiarity with the cryptocurrency ecosystem. Furthermore, the sample of actual users with only seven responses is very small. Thus, further research is needed to investigate to which degree the results can be generalized.

The observed discrepancy between stated intentions and actual behavior is a well known phenomenon in privacy-related areas, where it is generally referred to as the privacy paradox [27].

6 Limitations and Future Research

An obvious limitation of our study is that we used privacy concerns towards e-commerce companies to gauge privacy concerns, which may not be equivalent to concerns about cryptocurrencies in themselves. Previous research found that trust in cryptocurrencies is associated with trust in other entities, notably the government [3] and interpersonal trust in general [25], and more research could shed light on the connection between privacy concerns towards entities in the e-commerce and the cryptocurrency ecosystems and the currencies themselves. We further focused on privacy concerns and risk perception, leaving out factors such as perceived ease of use and perceived usefulness [10], cost-effectiveness [20], or self-efficacy [48]. We did this to afford ourselves the possibility to look at facets of privacy concerns without the risk of overfitting the model, but future research

should still consider these factors and how they interact with privacy concerns. Us employing a representative sample of German users of e-commerce meant that even though we reached 257 respondents, only seven of these had already used cryptocurrencies, which means a low representation for actual users as in individuals who have used these for payment purposes. While we believe using this type of sample was necessary to ensure that the results are applicable to the German population overall, future research may repeat this type of study with a larger focus on existing users. Finally, our study only asks German respondents, which may limit our studies applicability to other countries, as significant differences in privacy concerns and perceptions are well-founded in the literature [23,38]. Future research may replicate our study for different countries or cultures.

7 Conclusion

In this study, we contribute to the literature by investigating the effect of risk perceptions and privacy concerns towards online companies on whether individuals are willing to use cryptocurrencies, as well as their actual behavior. We find that perceived risk as well as worries about the collection of personal data exert a significant influence on whether individuals would consider using cryptocurrencies. However, neither variable was useful for explaining actual use behavior. This is consistent with the extant literature on cryptocurrency adoption [48]. Our results add to that picture, and open avenues to further research as to why perceived risk seems to play a major role for intention, but not behavior when it comes to cryptocurrency usage.

Questionnaire

Demographics. We asked for the following demographics, answer options are listed in brackets: age ($> 18, 18, \dots, 65, > 65$), gender (female, male) and education (cf. Table 1).

Adoption Willingness/Behavior¹

Have you made a purchase with bitcoin or another cryptocurrency in the past?

¹ Yes; No, but I have considered it before; No, I haven't considered it yet either.

Risk Perception²

RP1 When I use cryptocurrency, I worry about the risk of fraud because of the lack of legal regulations.

RP2 When I pay with cryptocurrencies, I worry about the value of my money because of the volatility of these currencies.

RP3 When I pay with cryptocurrencies I do not feel absolutely protected from illegal attacks and activities.

RP4 When I pay with cryptocurrencies, I worry about my electronic devices not working well due to cryptographic errors and the payment not being recorded correctly.

Collection (see Footnote 2)

CO1 It usually bothers me when companies ask me for personal information.

CO2 When companies ask me for personal information, I sometimes think twice before providing it.

CO3 It bothers me to give personal information to so many companies.

CO4. I am concerned that companies are collecting too much personal information about me.

Errors (see Footnote 2)

ER1 All the personal information in computer databases should be double-checked for accuracy – no matter how much this costs.

ER2 Companies should take more steps to make sure that the personal information in their files is accurate.

ER3 Companies should have better procedures to correct errors in personal information.

ER4 Companies should devote more time and effort to verifying the accuracy of the personal information in their databases.

Unauthorized Secondary Use (see Footnote 2)

US1 Companies should not use personal information for any purposes unless it has been authorized by the individuals who provided the information.

US2 When people give personal information to a company for some reason, the company should never use the information for any other reason.

US3 Companies should never sell the personal information in their computer databases to other companies.

US4 Companies should never share personal information with other companies unless it has been authorized by the individuals who provided the information.

Improper Access (see Footnote 2)

IA1 Companies should devote more time and effort to preventing unauthorized access to personal information.

IA2 Computer databases that contain personal information should be pro-

TECTED FROM unauthorized access – no matter how much it costs.

IA3 Companies should take more steps to make sure that unauthorized people cannot access personal information in their computers.

² seven-point Likert scale from “strongly disagree” (1) to “strongly agree” (7).

Acknowledgements. This work was supported by the German Federal Ministry of Education and Research's (BMBF) program "Datenschutz: selbstbestimmt in der digitalen Welt" via the SIOC project.

References

1. Abramova, S., Böhme, R.: Perceived benefit and risk as multidimensional determinants of bitcoin use: a quantitative exploratory study. In: ICIS 2016 Proceedings (2016)
2. Arias-Oliva, M., Pelegrín-Borondo, J., Matías-Clavero, G.: Variables influencing cryptocurrency use: a technology acceptance model in Spain. *Front. Psychol.* **10**, 475 (2019)
3. Arli, D., van Esch, P., Bakpayev, M., Laurence, A.: Do consumers really trust cryptocurrencies? *MIP* **39**(1), 74–90 (2021)
4. Bech, M.L., Garratt, R.: Central bank cryptocurrencies. *BIS Q. Rev.* (2017)
5. Berentsen, A., Schär, F.: The case for central bank electronic money and the non-case for central bank cryptocurrencies. *Federal Reserve Bank of St. Louis Review*, Second Quarter 2018, pp. 97–106 (2018)
6. Brandeis, L., Warren, S.: The right to privacy. *Harv. Law Rev.* **4**(5), 193–220 (1890)
7. Chen, L.: A model of consumer acceptance of mobile payment. *Int. J. Mob. Commun.* **6**(1), 32–52 (2008)
8. Chen, L.D., Tan, J.: Technology adaptation in e-commerce: key determinants of virtual stores acceptance. *EMJ* **22**(1), 74–86 (2004)
9. Cronbach, L.J.: Coefficient alpha and the internal structure of tests. *Psychometrika* **16**(3), 297–334 (1951)
10. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Q.* 319–340 (1989)
11. Dinev, T., Bellotto, M., Hart, P., Russo, V., Serra, I., Colautti, C.: Privacy calculus model in e-commerce—a study of Italy and the united states. *Eur. J. Inf. Syst.* **15**(4), 389–402 (2006)
12. Esmailzadeh, P., Hemang, S., Cousins, K.: Individuals' cryptocurrency adoption: a proposed moderated-mediation model. In: AMCIS 2019 (2019)
13. EUROSTAT: EUROSTAT 2018 (2021). <https://ec.europa.eu/eurostat/de/home>. Accessed 11 Mar 2023
14. Fabian, B., Ermakova, T., Sander, U.: Anonymity in bitcoin?—the users' perspective. In: ICIS 2016 Proceedings (2016)
15. Fishbein, M.: A theory of reasoned action: some applications and implications (1979)
16. Foley, S., Karlsen, J.R., Putninš, T.J.: Sex, drugs, and bitcoin: how much illegal activity is financed through cryptocurrencies? *Rev. Financ. Stud.* **32**(5), 1798–1853 (2019)
17. Glaser, F., Zimmermann, K., Haferkorn, M., Weber, M.C., Siering, M.: Bitcoin-asset or currency? Revealing users' hidden intentions. In: ECIS 2014 (2014)
18. Groß, T.: Validity and reliability of the scale internet users' information privacy concerns (IUIPC). In: PoPETS 2021 (2021)
19. Hair, J.F., Jr., Hult, G.T.M., Ringle, C.M., Sarstedt, M.: A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM). Sage (2021)
20. Hamm, P.: Acceptance factors for cryptocurrencies as payment systems. In: HICSS 2022 (2022)

21. Harborth, D., Pape, S.: German translation of the concerns for information privacy (CFIP) construct. Technical report, SSRN (2018)
22. Harborth, D., Pape, S.: How privacy concerns and trust and risk beliefs influence users' intentions to use privacy-enhancing technologies – the case of tor. In: HICSS 2019, pp. 4851–4860 (2019)
23. Ilhan, A., Fietkiewicz, K.J.: Data privacy-related behavior and concerns of activity tracking technology users from Germany and the USA. *Aslib J. Inf. Manag.* **73**(2), 180–200 (2021)
24. Jabbar, A., Geebren, A., Hussain, Z., Dani, S., Ul-Durar, S.: Investigating individual privacy within CBDC: a privacy calculus perspective. *Res. Int. Bus. Financ.* **64**, 101826 (2023)
25. Jalan, A., Matkovskyy, R., Urquhart, A., Yarovaya, L.: The role of interpersonal trust in cryptocurrency adoption. *J. Int. Financ. Markets Institutions Money* **83** (2023)
26. Jonker, N.: What drives the adoption of crypto-payments by online retailers? *Electron. Commer. Res. Appl.* **35**, 100848 (2019)
27. Kokolakis, S.: Privacy attitudes and privacy behaviour: a review of current research on the privacy paradox phenomenon. *Comput. Secur.* **64**, 122–134 (2017)
28. Li, Y.: Empirical studies on online information privacy concerns: literature review and an integrative framework. *Commun. Assoc. Inf. Syst.* **28**(1), 28 (2011)
29. Lustig, C., Nardi, B.: Algorithmic authority: the case of bitcoin. In: HICSS 2015, pp. 743–752 (2015)
30. Malhotra, N.K., Kim, S.S., Agarwal, J.: Internet users' information privacy concerns (IUPC): the construct, the scale, and a causal model. *ISR* **15**(4), 336–355 (2004)
31. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.* 50–60 (1947)
32. Mashatan, A., Sangari, M.S., Dehghani, M.: How perceptions of information privacy and security impact consumer trust in crypto-payment: an empirical study. *IEEE Access* **10**, 69441–69454 (2022)
33. Mendoza-Tello, J.C., Mora, H., Pujol-López, F.A., Lytras, M.D.: Disruptive innovation of cryptocurrencies in consumer acceptance and trust. *IseB* **17**, 195–222 (2019)
34. Merhi, M., Hone, K., Tarhini, A.: A cross-cultural study of the intention to use mobile banking between Lebanese and British consumers: extending UTAUT2 with security, privacy and trust. *Technol. Soc.* **59** (2019)
35. Muñoz-Leiva, F., Luque-Martínez, T., Sánchez-Fernández, J.: How to improve trust toward electronic banking. *Online Inf. Rev.* **34**(6), 907–934 (2010)
36. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. *Decentralized Bus. Rev.* 21260 (2008)
37. Poon, W.C.: Users' adoption of e-banking services: the Malaysian perspective. *J. Bus. Ind. Mark.* **23**(1), 59–69 (2008)
38. Schallehn, F., Valogianni, K.: Sustainability awareness and smart meter privacy concerns: the cases of us and Germany. *Energy Policy* **161**, 112756 (2022)
39. Schmitz, C., et al.: Limesurvey: an open source survey tool. LimeSurvey Project Hamburg, Germany (2023). <http://www.limesurvey.org>. Accessed 11 Mar 2023
40. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). *Biometrika* **52**(3/4), 591–611 (1965)
41. Smith, H.J., Dinev, T., Xu, H.: Information privacy research: an interdisciplinary review. *MIS Q.* 989–1015 (2011)

42. Smith, H.J., Milberg, S.J., Burke, S.J.: Information privacy: measuring individuals' concerns about organizational practices. *MIS Q.* 167–196 (1996)
43. Stewart, K.A., Segars, A.H.: An empirical examination of the concern for information privacy instrument. *ISR* **13**(1), 36–49 (2002)
44. Tronnier, F., Harborth, D., Hamm, P.: Investigating privacy concerns and trust in the digital euro in Germany. *Electron. Commer. Res. Appl.* **53**, 101158 (2022)
45. Udo, G.J.: Privacy and security concerns as major barriers for e-commerce: a survey study. *Inf. Manag. Comput. Secur.* **9**, 165–174 (2001)
46. Undheim, T.: Why banks fear bitcoin (2014). <https://web.archive.org/web/20180424095311/>. <https://fortune.com/2014/11/20/why-banks-fear-bitcoin/>. Accessed 11 Mar 2023
47. Venkatesh, V., Morris, M.G., Davis, G.B., Davis, F.D.: User acceptance of information technology: toward a unified view. *MIS Q.* 425–478 (2003)
48. Voskobochnikov, A., Abramova, S., Beznosov, K., Böhme, R.: Non-adoption of crypto-assets: exploring the role of trust, self-efficacy, and risk. In: *ECIS* (2021)
49. Voskobochnikov, A., Obada-Obieh, B., Huang, Y., Beznosov, K.: Surviving the cryptojungle: perception and management of risk among north American cryptocurrency (non)users. In: Bonneau, J., Heninger, N. (eds.) *FC 2020. LNCS*, vol. 12059, pp. 595–614. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51280-4_32
50. Wang, H., Lee, M.K., Wang, C.: Consumer privacy concerns about Internet marketing. *Commun. ACM* **41**(3), 63–70 (1998)
51. Westin, A.F.: Privacy and freedom. *Washington Lee Law Rev.* **25**(1), 166 (1968)
52. Wooldridge, J.M.: *Introductory Econometrics: A Modern Approach*. Cengage Learning (2015)
53. Xu, H., Dinev, T., Smith, J., Hart, P.: Information privacy concerns: linking individual perceptions with institutional privacy assurances. *J. Assoc. Inf. Syst.* **12**(12), 1 (2011)
54. Zhang, T., Lu, C., Kizildag, M.: Banking “on-the-go”: examining consumers' adoption of mobile banking services. *Int. J. Qual. Serv. Sci.* **10**(3), 279–295 (2018)



Automated Enrichment of Logical Attack Graphs via Formal Ontologies

Kéren Saint-Hilaire^{1,3} , Frédéric Cuppens^{2,3} , Nora Cuppens^{2,3} ,
and Joaquin Garcia-Alfaro¹  

¹ SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France
{keren.saint-hilaire, joaquin.garcia_alfaro}@telecom-sudparis.eu

² Polytechnique Montréal, Montréal, Canada

{frédéric.cuppens, nora.boulahia-cuppens}@polymtl.ca

³ Chair CRITiCAL, MITACS, Montréal, Canada
keren-a.saint-hilaire@polymtl.ca

Abstract. Attack graphs represent the possible actions of adversaries to attack a system. Cybersecurity experts use them to make decisions concerning remediation and recovery plans. There are different attack graph-building approaches. We focus on logical attack graphs. Networks and vulnerabilities constantly change; we propose an attack graph enrichment approach based on semantic augmentation post-processing of the logic predicates. Mapping attack graphs with alerts from a monitored system allows for confirming successful attack actions and updating according to network and vulnerability changes. The predicates get periodically updated based on attack evidence and ontology knowledge, allowing us to verify whether changes lead the attacker to the initial goals or cause further damage to the system not anticipated in the initial graphs. We illustrate our approach using a specific cyber-physical scenario affecting smart cities.

Keywords: Cybersecurity · Attack Graph · Defense Graph · Ontology

1 Introduction

Automation of cybersecurity aims to protect critical systems from cyber-attacks, i.e., from illicit activities perpetrated by adversaries who are trying to alter and disrupt normal business processes. Automated remediation of cyber-attacks is a complex task to achieve, specially under real-time constraints [6, 11, 13]. The understanding of attack realization against a system is essential to automate such tasks. This can be accomplished by adapting attack graphs to counter adversarial paths before adversaries perpetrate the final steps of a cyber-attack.

Attack graph can be classified into logical, topological, and probabilistic families [1]. Logical attack graphs represent the adversarial activities as logical predicates, requiring from successful preconditions to be considered as successful,

i.e., they accurately describe how to judge whether the attack was or not successfully perpetrated. Topological families offer a higher-level view of possible attacks in an information system, representing an attack as a way of accessing new resources. Finally, probabilistic families assign probabilities to nodes and attack steps (e.g., using Bayesian theory).

In our work¹, we choose logical attack graphs. The reason for our choice is as follows. Both topological and probabilistic models provide less precision than logical models, f.i., in terms of explainability about attacks' performance. Indeed, logical attack graphs illustrate the causes of the attacks instead of snapshots of the attack steps [8]. This offers several advantages. For instance, the size of the graph increases in a polynomial manner, whereas in other approaches, it can increase exponentially. Moreover, causality relations between adversaries and systems are already represented in the logical statements of nodes and edges in a logical attack graph. In the other approaches, one may go through Boolean variables to identify the cause of an adverse situation that allows adversaries' actions in a stage, increasing processing and inference complexity. In the case of logical attack graphs, exploiting existing vulnerabilities on an asset is the main cause of the attack.

Our work tackles the following question: *how can real-time system monitoring enrich a priori logical attack graphs by considering embedded and implied inferences on expert knowledge bases?* We validate that a posteriori enrichment of the graphs makes it possible to fulfill certain preconditions that were not considered in the initial graph's generation. Semantic information about system vulnerabilities allows us to discover whether the system is now exposed to different situations that can augment the attack surface to newer detrimental events, causing even further damage.

We also conduct experimental work using the following setup. We use a scanner of vulnerabilities to discover and list vulnerabilities in a given monitored system. The results are consumed by MulVAL [9], a logic-based attack graph engine. We add system monitoring using a SIEM (Security Information and Event Management), enhanced with additional tools to trigger and post-process attack alerts. We also instantiate precise attacks to change the state of the system (i.e., exploitation of vulnerabilities) and use a recent implementation of VDO² to enrich the initial attack graph by augmenting the predicates of the initial graph with the semantic data of VDO and the alerts from Prelude-OSS. Alerts trigger a search within the graph and expand those paths related to successful vulnerability exploitation.

The paper is organized as follows. Section 2 provides a background of the subject and some preliminaries on using attack graphs. Section 3 presents our attack-graph enrichment approach. Section 4 provides the experimental results. Section 5 surveys related work. Section 6 concludes the paper.

¹ An early version of this work is available in Ref. [12].

² <https://github.com/usnistgov/vulntology>.

2 Background

2.1 Logical Attack Graph Modelling

We define preliminary concepts, such as Graph, and AND-OR Graph, as underlying requirements for logical attack graph modeling [1].

Definition 1 (Graph). *A Graph is a set V of vertices and a set E of unordered and ordered pairs of vertices, denoted by $G(V; E)$. An unordered pair of vertices is an edge, while an ordered pair is an arc. A graph containing edges alone is non-oriented or undirected; a graph containing arcs alone is called oriented or directed.*

In a directed graph:

- The parent or source of an arc $(v_1; v_2) \in A; v_1 \in V; v_2 \in V$, is v_1 .
- The child or destination of an arc $(v_1; v_2) \in A; v_1 \in V; v_2 \in V$, is v_2 .
- The incoming arcs of a node v are all the arcs for which v is the child:
 $\forall a = (v_1; v) \in A, \text{ with } v_1 \in V$.
- The outgoing arcs of a node v are all the arcs for which v is the parent:
 $\forall a = (v; v_2) \in A, \text{ with } v_2 \in V$.
- The indegree $deg^-(v)$ of a vertex $v \in V$ is the number of arcs in A whose destination is the vertex v : $deg^-(v) = \text{Card}(\{v_i; \forall v_i \in V; (v_i; v) \in A\})$.
- The outdegree $deg^+(v)$ of a vertex $v \in V$ is the number of arcs in A whose destination is the vertex v : $deg^+(v) = \text{Card}(\{v_i; \forall v_i \in V; (v_i; v) \in A\})$.
- A root is a vertex $v \in V$ for which $deg^-(v) = 0$ (no incoming arc).
- A sink is a vertex $v \in V$ for which $deg^+(v) = 0$ (no outgoing arc).

Definition 2 (AND-OR Graph). *An AND-OR graph is a directed graph where each vertex v is either an OR or an AND. A vertex represents a sub-objective, and according to its type (AND or OR), it requires either the conjunction or disjunction of its children to be fulfilled.*

According to Definitions 1 and 2, logical attack graphs are based on AND-OR logical directed graphs. The nodes are logical facts describing adversaries' actions or the prerequisites to carry them out. The edges correspond to the dependency relations between the nodes. Depending on the approach, various operators can be considered in a logical attack graph. The most popular operators are AND and OR. The AND operator describes the achievement's requirement of all the facts of its children for the logical fact of a node to be achieved. The OR operator describes the achievement condition of at least one fact of its children for the logical fact of a node to be achieved.

3 Proposed Approach

After the attack graph generation, using a priori knowledge about vulnerabilities and network data, both networks and vulnerabilities may evolve (i.e., the

configuration of system devices may change, software updates may be enforced, etc.). Hence, the network is not exposed to the same vulnerabilities as at the beginning of the attack graph generation process. It is essential to update the attack graph according to systems' changes. When enriching a logical attack graph, causality relations between adversaries and systems shall be represented in the logical statements of nodes and edges. We propose a logical attack graph enrichment approach based on ontologies to address these requirements.

3.1 Generation of the Attack Graph

Generating a logical attack graph requires the definition of rules describing causality relations. As an example, we consider code execution. Code execution on a machine allows an adversary access to a host. This scenario corresponds to the logical implication detailed by the following rule:

$$\boxed{execCode(h, a) \rightarrow canAccesHost(h)}$$

where $canAccesHost(h)$ is a logical predicate describing the accessibility to host h , and $execCode(h, a)$ another predicate assessing that an adversary a executed code in h . The example can be extended as follows:

$$\boxed{execCode(h, a) \wedge hasCredentialsOnMemory(h, u) \rightarrow harvestCredentials(h, u)}$$

where $harvestCredentials(h, u)$ describes a series of credentials harvesting on host h , $execCode(h, a)$ the predicate that an adversary a is executing code on host h , and $hasCredentialsOnMemory(h, u)$ the predicate of storing the credentials on the memory of host h , the example describes an adversary harvesting the credentials of a previous user that logged onto the system by finding them in the memory of that precise system.

3.2 Monitoring the Information System

To update the attack graph based on the real-time state of the system, we need also to monitor the information system. The monitoring process output can get continuously mapped with the initial nodes of the attack graph to find out if a vulnerability is being exploited. The mapped information consists of the port, the IP address, and the device's protocol. The mapping is prioritized based on the severity of the alert and the Common Vulnerability Scoring System (CVSS) score of the CVE. Our approach prioritizes the CVE with the highest CVSS version 2.0 score for an alert that concerns various vulnerabilities to deduce its impacts using a vulnerability ontology. In Table 1, a system contains three vulnerabilities concerning remote desktop protocol and exploitable using port 3389 and protocol TCP. In this case, our approach prioritizes CVE-2019-0708 since its score is higher. Next, we provide more details about this process using semantic information about concrete vulnerabilities.

Table 1. Comparison between different various CVE concerning the same port and protocol based on their CVSS index.

CVE-ID	Product	Protocol	Port	CVSS
CVE-2019-0708	windows remote_desktop_protocol	tcp	3389	9.8
CVE-2012-0152	windows remote_desktop_protocol	tcp	3389	9.3
CVE-2012-0002	windows remote_desktop_protocol	tcp	3389	9.3

3.3 Vulnerabilities and Ontologies

Vulnerability information is necessary for the attack graph generation and enrichment process. Vulnerability description in standardized databases provides information about the preconditions and post-conditions and practical ways to be exploited. The vulnerability information allows semantically expressing the adversary’s actions toward the adversarial goals, such as pre- and post-conditions. It is also necessary to consider information about exploited vulnerabilities to update logical attack graphs in real time. The vulnerability information is a text written in natural language. This information needs to be transformed into machine-readable text. The use of an ontology is necessary to represent the machine-readable text and thus ensure its homogeneity. An ontology makes it possible to represent a domain in a structured way. The ontology facilitates interoperability between information from the attack graph and the extracted information from vulnerability databases. It is possible to make queries on the ontology to infer new knowledge necessary for the ontological enrichment of the attack graph.

Let us take CVE-2002-0392 as an example. The information from its description: “Apache 1.3 through 1.3.24, and Apache 2.0 through 2.0.36, allows remote attackers to cause a denial of service and possibly execute arbitrary code via a chunk-encoded HTTP request that causes Apache to use an incorrect size” can be classified as represented in Table 2. The information from the table can lead to the construction of an ontology with the classes *CVE-ID*, *Product*, *AttackType*, *Method*, *Impact* and the properties *concernsProduct*, *hasRemoteType*, *hasMethod*, *resultsInImpact*.

Table 2. Classification of CVE-2002-0392 characteristics.

CVE-ID	Product	Remote Type	Method	Impact
CVE-2002-0392	Apache	remote	Code Execution	Privilege Escalation

3.4 Enrichment of Attack Graphs

Algorithm 1 represents our proposed approach for enriching attack graphs based on a vulnerability ontology and monitoring system information. When a threat

exists on a vulnerable component of the monitored system, it is necessary to look through the vulnerability characteristics to find its post-conditions. Depending on the attack goal and the deduced impact, new rules are generated allowing the logical reasoner to find a new path from the additional consequence to the attack goal, see below.

$$\boxed{\begin{array}{l} shutdown(host) \rightarrow physicalDamage(bus) \\ execCode(host, user) \rightarrow shutdown(host) \end{array}}$$

These post-conditions allow the attack graph to be enriched with new paths. For an attack that aims to cause physical damage and an exploited vulnerability that can cause a service interruption that can be shutdown, reboot, or panic, the enrichment process adds a new path from the node consisting of the vulnerability exploitation and the node expressing the physical damage.

Algorithm 1: Attack Graph Enrichment Process

Data: System state
 $G = (V, E) \leftarrow$ AttackGraphGenerated;
 $initialImpact \leftarrow$ impact of exploited vulnerability in pro-active graph;
 $listImpact \leftarrow$ List of impacts from the SPARQL query;
 $V = \{v_0, v_1, \dots, v_n\} \leftarrow$ List of vertices of G ;
 $E = \{e_0, e_1, \dots, e_m\} \leftarrow$ List of edges of G ;
Result: G'
 initialization;
for $i=0$; $i < len(listImpact)$; $i++$ **do**
 if $listImpact[i] \neq initialImpact$ **then**
 if $impact = Shutdown \cup impact = Reboot \cup impact = Panic$ **then**
 for $z=0$; $z < len(V)$; $z++$ **do**
 if $V[z]$ is a fact node \cap attack goal is PhysicalDamage **then**
 Add new rules to the logical reasoner;
 $G' \leftarrow$ the attack graph regenerated;
 $i \leftarrow$ the number of vertices added;
 $V' = \{v_0, v_1, \dots, v_{n+i}\} \leftarrow$ List of vertices of G' ;
 $t \leftarrow$ the number of edges added;
 $E' = \{e_0, e_1, \dots, e_{m+t}\} \leftarrow$ List of edges of G' ;
 $G' = (V', E')$;

4 Experimental Approach

4.1 Use Case Scenario

Next, we describe a use case scenario provided by smart city stakeholders. A denial-of-service attack against a municipality network is perpetrated. Commu-

nication between machines and sensors is interrupted, causing further delays in the city’s transportation service. People fleeing the area start fighting, forcing the authorities to close all transportation services. The violence in public transportation on a given bus affects the health of several passengers. This scenario is taken into account in our experiments.

4.2 Setup

We instantiate the scenario depicted in Fig. 1 to validate our approach. It represents a cyber-physical system monitored by a SIEM, based on Prelude-OSS³. We use a virtual machine representing the starting device of the scenario, another machine to instantiate the breach point, and a third one representing the critical asset.

The rationale of the scenario depicted in Fig. 1 is as follows. An adversary successfully executes arbitrary code on the starting device by connecting remotely through Remote Desktop Protocol (RDP), a network service that provides users with graphical means to control computers remotely). The adversary can then read the memory of the starting device. The administrator’s credentials are saved in the memory of the starting device. Then, the adversary harvests those credentials. We assume the administrator can connect to all the machines in the domain to manage them remotely. Then, an adversary capable of reusing the credentials can log onto the breach point and remotely connect to the critical asset. To eavesdrop network traffic, the adversary perpetrates a DNS Poisoning attack [5]. The adversary also performs integrity attacks to modify application-level information, such as the bus schedule and routes, to perturb the influence of traffic and cause a congestion increase. This causes citizens to take the wrong buses at the wrong time, leading to panic and violence mentioned in Sect. 4.1. In parallel, the adversary reuses the domain credentials to steal some other access keys and impersonate other users (shown in Fig. 1 with steps *Access Keys Stealer* and *User Compromise*).

W.r.t. Fig. 2, we use Nessus Essentials⁴ scanner (Step 1) to discover and list of vulnerabilities in the monitored system. Data from Nessus is consumed by MulVAL [9], a reasoning engine based on logical programming (Step 2), to generate a logic-based attack graph (Step 3). We use Prelude-ELK⁵, an extended version of Prelude-OSS, to monitor the system in real time. When an alert is generated, the procedural processor maps the information from the alert with the attack graph nodes’ information (Step 4) to determine if the alert concerns a discovered vulnerability in the system. The procedural processor queries the impacts of the exploited vulnerability on a vulnerability knowledge graph (Step 5). For a new impact deduced (Step 6), the procedural processor adds a new rules to the logical reasoner (Step 7). The attack graph is regenerated with a new path (Step 8).

³ <https://www.prelude-siem.com/en/oss-version/>.

⁴ <https://www.tenable.com/products/nessus/nessus-essentials>.

⁵ <https://github.com/Kekere/prelude-elm>.

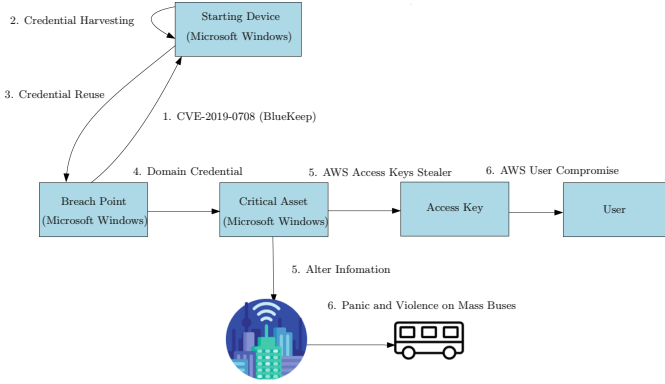


Fig. 1. Cyber-physical attack scenario. An adversary exploits the vulnerability associated with CVE-2019-0708 on a Starting Device. Then, administrator credentials are harvested from the device’s memory and reused by the adversary to take control of a critical asset. The attack affects physical and digital elements associated with the system (e.g., people and services).

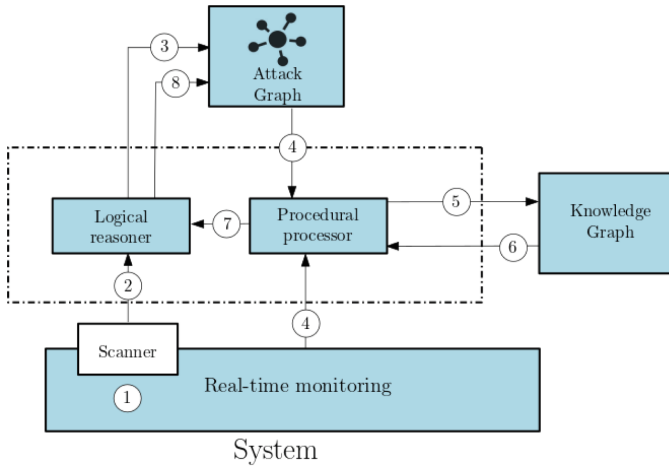


Fig. 2. The Attack Graph Enrichment Process.

MulVAL. Based on the scenario shown in Fig. 1, we create input data for MulVAL and interaction rulesets associated with the vulnerability and the proposed scenario. We encode the new interaction rules as Horn clauses [9]. The first line corresponds to a first-order logic conclusion. The remaining lines represent the enabling conditions. The clauses below correspond to the following statement from the scenario shown in Fig. 1:

“The breach point credentials can be harvested on the starting device only if there is previously an execution code exploit on the starting device and the

credentials of the administrator are saved onto the memory of the starting device.”

```
harvestCredentials(_host, _lastuser) :-
    execCode(_host, _user),
    hasCredentialsOnMemory(_host, _lastuser).
```

The clauses below represent the following facts:

“It is possible to execute code on the breach point when these credentials have been harvested and because the breach point and the starting device are on the same network and can communicate through a given protocol and port.”

```
execCode(_host, _user) :-
    networkServiceInfo(_host, _program, _protocol, _port, _user),
    hacl(_host, _h, _protocol, _port),
    harvestCredentials(_h, _user).
```

Ontology. We use VDO, an ontology of CVEs proposed by NIST. Figure 3, from [4], represents various attributes of the VDO ontology for characterizing software vulnerabilities. Various features, such as *Impact Method*, and *Logical Impact* are mandatory. *Impact Method* describes how a vulnerability can be exploited. *Logical Impact* describes the possible impacts a successful exploitation of the Vulnerability can have. For each CVE affecting the monitored system, we can fulfill the classes of information from the ontology according to the description and metrics of the CVE.

Prelude-ELK. We use an extended⁶ version of Prelude-OSS’s LML (Log Monitoring Lackey) and third-party sensors such as Suricata⁷ to monitor and process Syslog messages generated from different hosts on heterogeneous platforms. We install Rsyslog Windows Agent⁸ and Suricata on each virtual machine to monitor them with the ELK extension of Prelude-OSS. The results are processed in real-time, mapping the alerts and VDO’s data while conducting our attack graph enrichment process.

Procedural Processor. We create a procedural processor where we upload the input required for the attack graph generation. The engine displays a web visualization of the attack graph. The server matches the last alert’s IP address, port, and protocol with the attack graph. When a vulnerability is likely to be exploited, the engine consults the vulnerability ontology to deduce other impacts of the exposure. The tool generates new rules according to ontology inference.

⁶ <https://github.com/Kekere/prelude-elk>.

⁷ <https://suricata.io/>.

⁸ <https://www.rsyslog.com/windows-agent/>.

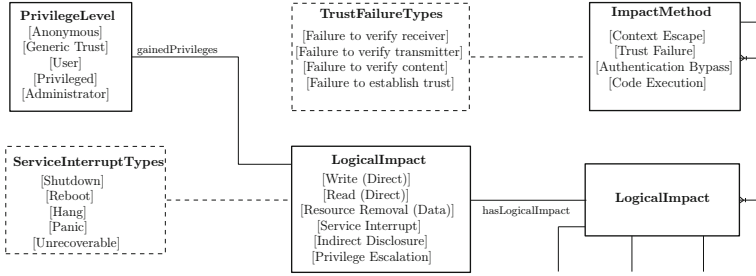


Fig. 3. Sample classes associated to VDO (Vulnerability Description Ontology).

In this paper, we do not consider the impact of an exploited vulnerability on other vulnerabilities in the system nor countermeasures actions to mitigate the attacker’s actions. We will focus on this in future work.

4.3 Results

Figure 4(a), represents the attack graph generated for the scenario depicted in Fig. 1. The goal, represented by Node 1, is to cause panic and violence (see the use-case scenario described in Sect. 4.1). A red node represents the existence of a vulnerability on a device. An orange node represents network configuration, e.g., device characteristics, the connection between two devices in the network, etc. When the preconditions are satisfied, a yellow node represents the inference rules leading to a fact. Green nodes represent facts. For instance, Node 15 represents the network access to the Starting Device, using Remote Desktop Protocol (RDP) services, from the attacker located on the Breach Point; the adversary location is represented by Node 18. Node 20 concerns the vulnerability identified as *CVE-2019-0708* on the Starting Device. Node 19 concerns the network configuration of the Starting Device; port 3389 is opened, allowing remote connection to the device using remote desktop service. Node 14 represents the rule that leads the adversary to remotely exploit the vulnerability on the Starting Device when preconditions on Nodes 19, 20, and 15 are met.

In real-time, alerts are processed with Prelude-ELK (see Sect. 4.2). The procedural processor matches precondition nodes’ information with alert information, such as IP address, protocol, and port. The processor makes a query on VDO to deduce other post-conditions associated with *CVE-2019-0708* as represented in Listing 1.1. A list of deduced impacts from the SPARQL query is represented in Table 3. One consequence concerns privilege escalation, and the other concerns interrupting the communication between the affected device and other devices. Important information is not communicated on time for the citizens, including public transportation users. The service interruption can also cause violence on public transportation. Therefore new paths are added to the attack graph based on the deduced impacts. As a result, an enriched attack graph is derived.

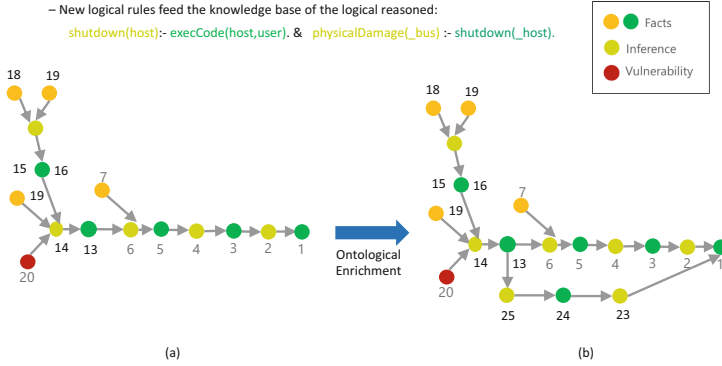


Fig. 4. Sample results. (a) Attack graph generated for the scenario leading to violence on buses. (b) The same attack graph is once enriched with data from the ontology.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/keren/ontologies/2022/6/...>

SELECT DISTINCT ?v ?impmethod ?log
WHERE {
  ?vul :hasIdentity ?vvalid .
  ?vvalid :value ?v .
  ?vul :hasScenario ?sce .
  ?sce :hasAction ?ac .
  ?sce :affectsProduct ?prod .
  ?ac :resultsInImpact ?im .
  ?im :hasLogicalImpact ?log .
  ?ac :hasImpactMethod ?imp .
  ?imp :value ?impmethod .
  FILTER ( str (?v) ="CVE-2019-0708" ) .
}

```

Listing 1.1. SPARQL query

Table 3. List of impacts deduced.

CVE-ID	Logical Impact
CVE-2019-0708	Privilege Escalation
CVE-2019-0708	Reboot

Figure 4(b) represents such an enriched attack graph. The three nodes highlighted with the red square correspond to the new nodes added to the enriched attack graph thanks to the ontology inference. Node 24 describes the service interruption. Node 25 is an interaction rule expressing the achievement of Node 24 because the precondition represented by Node 13 is satisfied. Node 23 leads

to the violence on buses scenario (i.e., by inference, Node 24 targets Node 23, a new rule concerning the attack goal). In Fig. 4, the two added nodes represent a new path the adversary can take to cause panic and violence. As we can see, the enriched graph is now acyclic. The new path is shorter than the predicted one. The adversary can reach the goal, represented by Node 1, much sooner than expected. This difference would make operators aware that applying a remediation plan is more urgent. Therefore, the experts can prioritize remediation actions that prevent the more impactful attacker’s actions.

5 Related Work

5.1 Attack Graph Generation Approaches

In [3], Gosh and Gosh propose a planning-based approach for generating minimal attack graphs. The planner generates acyclic paths from information, vulnerabilities, and initial network configuration, combining them, resulting in a minimal attack graph. Minimal attack graphs do not contain redundant nodes and edges. This approach makes it possible to generate an attack graph in polynomial time, regardless of the distribution of vulnerabilities on the attack graph. The initial network configuration and exploit description are the inputs for generating a minimal attack graph using a planner.

Roschke et al. [10] propose an approach to generate logical attack graphs based on logic programming. The input information for the generation of graphs is system and vulnerability information. They present the integration of an IDS in the graph generation process, complementing the process with data fusion and correlation. This improves the quality of the alerts and the quality of their correlation. This correlation makes it possible to prioritize and label alerts. Logic programming-based approaches are more flexible regarding semantic correspondence with other knowledge bases.

Compared to those approaches mentioned above, we monitor the network to update the attack graph based on state change of the network and generate attack graphs based on network information received from Nessus scans. We also enrich the attack graph based on vulnerability information from CVEs and alerts received from a SIEM. We use a logical attack graph generation approach. With a logical approach, the inference is more straightforward. Moreover, the semantics abilities enhance attack graph enrichment with ontology. We use a vulnerability ontology to correlate alerts with the system and vulnerability information.

5.2 Ontology and Attack Graph Generation

Recently, ontologies have been used in different approaches of attack graph generation. Falodiya et al. [2] propose an algorithm that traverses a semantic attack graph to add the information extracted from the graph into an ontology. This ontological approach makes it possible to store other information, such as the countermeasures available for a vulnerability and their cost, as well as the

anti-forensic measures that the attacker, the vulnerability information can use, and the CVSS Score. Analyzing this information using an ontological approach becomes significantly easier as the network size increases.

Lee et al. [7] propose an ontological representation of attack graphs. An ontology that represents attack graphs for a simple network environment is created using RDF schema and OWL. Classes and relationships are created from the multi-requisite graph model. States, vulnerabilities, and prerequisites are represented as classes. This approach improves the machine readability of large-scale attack graphs and thus automates network security assessment. Ontological structures facilitate security assessment. Experts can get the information needed for risk analysis without analyzing the entire attack graph. This task done by the experts can be time-consuming when the graph is large.

In our approach, ontologies contribute to attack-graph enrichment. We use a standardized ontology proposed by the National Institute of Standards and Technology (NIST), the Vulnerability Database Ontology (VDO), for the attack graph enrichment. VDO provides mandatory classes such as *Logical Impact* and *Product*, which we use to map alerts with attack graph nodes. New attack paths can be discovered for a given CVE in VDO. The semantic abilities of logical attack graphs and ontologies also allow us to update the graphs. This improves the automation of the enrichment process (i.e., cybersecurity operators do not have to modify inputs to update the attack graphs manually).

6 Conclusion

We have proposed an ontology-based approach for attack graph enrichment. We use logical graph modeling, in which attacks are represented with predicates. Successful precondition validation means successful attack perpetration. Compared to similar approaches, such as topological and probabilistic attack graphs, our approach simplifies inference since graphs' edges now specify causality. We have implemented the proposed approach using existing software. We have validated the approach based on a cyber-physical use-case proposed by smart-city stakeholders. We have validated the full approach, from generating an initial attack graph (using network vulnerability scans), to enriching the graph (mapping monitoring alerts and ontology semantics in real-time). The predictions of the initial graph are successfully updated into the enriched graph based on inferences by an ontology thanks to expert and monitoring knowledge. In the future, we will evaluate the performance of the proposed approach. We will also focus on presenting remediation actions to block the attacker's actions.

Acknowledgements. We acknowledge financial support from the European Commission (H2020 IMPETUS project, under grant agreement 883286) and the Chair CRIT-iCAL, funded by MITACS (Canada).

References

1. Aguessy, F.-X., Bettan, O., Blanc, G., Conan, V., Debar, H.: Hybrid risk assessment model based on Bayesian networks. In: Ogawa, K., Yoshioka, K. (eds.) IWSEC 2016. LNCS, vol. 9836, pp. 21–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44524-3_2
2. Falodiya, K., Das, M.L.: Security vulnerability analysis using ontology-based attack graphs. In: 2017 14th IEEE India Council International Conference, INDICON 2017, pp. 1–5 (2018)
3. Ghosh, N., Ghosh, S.: A planner-based approach to generate and analyze minimal attack graph. *Appl. Intell.* **36**(2), 369–390 (2012)
4. Gonzalez, D., Hastings, H., Mirakhorli, M.: Automated characterization of software vulnerabilities. In: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 135–139. IEEE (2019)
5. Hu, Q., Asghar, M.R., Brownlee, N.: Measuring IPv6 DNS reconnaissance attacks and preventing them using DNS guard. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 350–361. IEEE (2018)
6. Kaiser, F.K., Andris, L.J., Tennig, T.F., Iser, J.M., Wiens, M., Schultmann, F.: Cyber threat intelligence enabled automated attack incident response. In: 2022 3rd International Conference on Next Generation Computing Applications (NextComp), pp. 1–6. IEEE (2022)
7. Lee, J., Moon, D., Kim, I., Lee, Y.: A semantic approach to improving machine readability of a large-scale attack graph. *J. Supercomput.* **75**(6), 3028–3045 (2019)
8. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 336–345 (2006)
9. Xinming, O., Govindavajhala, S., Appel, A.W.: MulVAL: a logic-based network security analyzer. In: USENIX Security Symposium, Baltimore, MD, vol. 8, pp. 113–128 (2005)
10. Roschke, S., Cheng, F., Meinel, C.: Using vulnerability information and attack graphs for intrusion detection. In: 2010 6th International Conference on Information Assurance and Security, IAS 2010, pp. 68–73 (2010)
11. Roschke, S., Cheng, F., Schuppenies, R., Meinel, C.: Towards unifying vulnerability information for attack graph construction. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 218–233. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04474-8_18
12. Saint-Hilaire, K., Cuppens, F., Cuppens, N., Garcia-Alfaro, J.: Ontology-based attack graph enrichment. In: 2021 TIEMS (The International Emergency Management Society) Annual Conference, Paris, France (2021). <https://arxiv.org/abs/2202.04016>
13. Stan, O., et al.: Heuristic approach for countermeasure selection using attack graphs. In: 2021 IEEE 34th Computer Security Foundations Symposium (CSF), pp. 1–16 (2021)



Detecting Web Bots via Mouse Dynamics and Communication Metadata

August See^(✉), Tatjana Wingarz, Matz Radloff, and Mathias Fischer

Universität Hamburg, Hamburg, Germany

{richard.august.see,tatjana.wingarz,mathias.fischer}@uni-hamburg.de

Abstract. The illegitimate automated usage of Internet services by web robots (bots) is an ongoing problem. While bots increase the cost of operations for service providers and can affect user satisfaction, e.g., in social media and games, the main problem is that some services should only be usable by humans, but their automated usage cannot be prevented easily. Currently, services are protected against bots using visual CAPTCHA systems, the de facto standard. However, they are often annoying for users to solve. Typically, CAPTCHAs are combined with heuristics and machine-learning approaches to reduce the number of times a human needs to solve them. These approaches use request data like IP and cookies but also biometric data like mouse movements. Such detection systems are primarily closed source, do not provide any performance evaluation, or have unrealistic assumptions, e.g., that sophisticated bots only move the mouse in straight lines. Therefore we conducted an experiment to evaluate the usefulness of detection techniques based on mouse dynamics, request metadata, and a combination of both. Our findings indicate that biometric data in the form of mouse dynamics performs better than request data for bot detection. Further, training a mouse dynamic classifier benefits from external and not only website-specific mouse dynamics. Our classifier, which differentiates between artificial and human mouse movements, achieves similar results to related work under stricter and more realistic conditions.

Keywords: web bots · mouse dynamics · captchas

1 Introduction

Programs that can automatically request endpoints increase operating costs and can frustrate users. For example, web bots made popular items such as graphics cards and new game consoles unavailable for years because they were purchasing them automatically. Commercial countermeasures, such as IDS solutions, prove ineffective against web bots because these bots operate within the constraints of the targeted e-commerce website's existing APIs or user interface. For instance, web bots may mimic human behavior by navigating to a product and clicking on the "buy now" button. The problem is that the endpoints are used as intended. A defense that scans, e.g., for malicious payload in requests is pointless here. One solution is CAPTCHAs, which give users tasks that are difficult for a computer

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 73–86, 2024.

https://doi.org/10.1007/978-3-031-56326-3_6

but easy for a human to solve. However, this introduces user friction that can cost a company customers and thus revenue [9]. Modern CAPTCHAs are used together with risk assessment methods. Depending on the risk score, a certain hard CAPTCHA or no CAPTCHA at all is presented [15]. While we see this as a step in the right direction, the problem remains that such solutions are not privacy friendly as request data and biometric features are passed on to third parties. Commercial CAPTCHA providers, understandably, do not disclose which features they use for detection and which are most beneficial to identify bots.

Existing approaches are subject to various limitations, such as being closed-source, considering only request data or mouse dynamics, or having shallow assumptions in their evaluation. For instance, some approaches assume that advanced web bots only produce mouse movements in straight lines rather than curved human-like movements. Moreover, these approaches fail to address other real-world problems, such as whether website-specific mouse data or any mouse data can be used for bot detection. We address this in our paper.

Our main contribution is evaluating the usefulness of mouse dynamics for detecting bots in realistic settings. In more detail:

- We evaluate the performance of classifiers for bot detection based on mouse dynamics and compare them to the performance of using request data. We do this on a consistent dataset that contains mouse- and request data belonging to the same user. Our evaluation includes advanced bots that mimic human mouse movements utilizing third-party software.
- We show that bot detection based on mouse dynamics can benefit from not being solely trained on website-specific mouse movements, indicating that website operators do not need to train on the mouse movements of their users exclusively but can leverage third-party datasets.
- We investigate the relationship between the number of data points and the performance of bot detection, thus allowing us to determine the amount of data required for good performance and, consequently, the speed at which a classification can take place.

The rest of this paper is structured as follows: Sect. 2 discusses web bot detection using request data and mouse dynamics. Section 3 explains how and which features we used to train classifiers from related work. Section 4 describes our evaluation, how we created our dataset, and the limitations of our work. Finally, Sect. 5 concludes the paper.

2 Related Work

We divide the related work into approaches that detect bots based on request data and ones that detect bots based on mouse data. Note that there are also approaches that recognize bots based on other biometric data [5, 6], or approaches that are based on trusted platforms [8].

Modern CAPTCHA systems like hCaptcha and reCAPTCHA [15, 16] are already using biometric data like mouse movements in addition to request data.

However, they do not disclose if the detection of bots is website specific and how well which factor performs. This is most likely due to protecting business secrets and denying bot creators information about where improvements need to be made. Google itself doesn't even specify what data they use exactly for reCAPTCHA. However, there is related work that tries to break this down [19].

2.1 Bot Detection via Request Data

A simple way for bot detection is to block IPs that are known for spamming or cyber attacks, for example, using *abuseipdb*¹. Furthermore, there are many approaches for the detection of bots based on request data [11, 12, 14–16, 20].

Iliou et al. [11] present a comparison of different machine learning algorithms and combinations of various attributes used in previous literature. Their approach does not rely on cross-website tracking or using external resources like IP databases. This makes their approach simple to reimplement and validate. Their methods were tested on a year's worth of HTTP log data from MK-Lab's public web server². The data included IP addresses, the request method, the request path, referrers, user agent strings, and timestamps. The attributes mainly include request metadata that would be suitable for a privacy-friendly bot detection system, for example, the percentage of image requests or the number of total bytes per session. The authors split the bot data in their dataset into simple and advanced bots, which are determined by whether the requests have a browser agent name and, in case they do, whether the IPs have shown malicious activity before. Their results show that different sets of attributes perform best depending on the classification algorithm used. The best machine learning methods are Random Forest and Multilayer Perceptron, although the paper concludes that using an ensemble classifier that averages over all used methods would be more stable. Additionally, simple web bots can be detected very easily, while detecting advanced bots is significantly harder, with areas under the ROC curve of 1.00 and 0.64, respectively. Especially in false positive intolerant use cases, the performance of detecting advanced bots is too poor to be used in the real world. The authors conclude that future work would need to incorporate more advanced features that bots cannot easily simulate.

2.2 Bot Detection via Mouse Dynamics

There is a lot of related work in the area of authentication using biometric data using mouse dynamics [13, 17, 18].

The work by Shen et al. [18] shows that it is possible to use mouse and trackpad actions to verify the authenticity of users. For this purpose, they group mouse events such as single-click, double-click, or drag-and-drop. This data is combined with information about the current application type, e.g., web browsing or gaming, the screen area where the mouse movement occurred, the window position, and the timestamp. The data is transformed into a feature vector

¹ <https://www.abuseipdb.com/>.

² Multimedia Knowledge and Social Media Analytics Laboratory, <https://mklab.iti.gr/>.

containing data such as the time it took a user to click a button, the speed of movement, or the acceleration. Finally, three different one-class classifiers (Nearest Neighbor, Single-Layer Neural Network, Support Vector Machine) are compared, with the Support Vector Machine method performing best with false positive and false negative rates of 0.37% and 1.12%, respectively. This was achieved only when 3000 operations and 30 min of processing time for successful authentication are considered. With a more feasible authentication time of one minute, the values for FPR and FNR increase to 44.65% and 34.78%, respectively. This drastically limits the applicability of this approach, which the authors also note.

Acien et al. [1] show the feasibility of using biometric features for bot detection. They use both function-based and GAN-based mouse trajectory synthesis methods to generate training and evaluation data. Six different classifier types (Support Vector Machine, K-Nearest Neighbor, Random Forest, Multi-Layer Perceptron, and 2 Recurrent Neural Networks with Long Short-Term Memory and Gated Recurrent Units, respectively) are compared to each other, with Random Forest performing the best. Combined, their method can distinguish between humans and bots with up to 98.7% accuracy with only one mouse trajectory as input. They conclude that, compared to state-of-the-art works, the usage of mouse data has unexploited potential in the context of bot detection. While there are other approaches for bot detection based on mouse dynamics [5, 21], they are all quite similar to each other. There are even approaches and projects that try to synthesize human mouse movements [1, 2].

2.3 Bot Detection via Request Data and Mouse Dynamics

Iliou et al. [10] present a method of using request data together with mouse data for bot detection, building on their previous work on bot detection using requests [11]. When classifying mouse dynamics, they do not build on existing work but create a new model that performs the classification using a convolutional neural network (CNN) on the raw mouse positions. As they do not have a labeled bot dataset, they create bots themselves. While the general idea of their approach has merit, they perform their evaluation with advanced bots that only move the mouse in straight lines. Figure 1 shows an example of the advanced bot behavior used for evaluation in their paper.

The mouse movements created by such a bot are not realistic enough to challenge human behavior and are very easy to classify as a bot by, e.g., looking at the straightness/curvature or angular velocity of the purported mouse movements. While our approach is similar to the one presented by Iliou et al., we utilize more advanced bots for our evaluation that do not only move in straight lines and thus mimic human-like behavior more closely. We describe our advanced bot setup in Sect. 3.

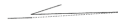
Advanced bot
✓ Heuristic hyperlink selection
✓ Advanced mouse movements


Fig. 1. Excerpt of the advanced mouse movements of [10] (Page 18, Table 7).

In summary, there is already relevant research on bot detection using biometric features, including mouse dynamics, but most of it is closed-source. Most publicly available approaches tackle the problem of bot detection independent of the website being defended, i.e., the bot detection systems are trained on a dataset that does not originate from the website being defended. Additionally, the majority of approaches consider only request data or mouse data for detection, while the ones combining both techniques only evaluate their approaches with easily detectable bots.

3 Bot Detection Using Requests and Mouse Dynamics

Our core idea is to improve the detection of web bots by using mouse dynamics in addition to request data. While request (meta)data like user-agent or screen size can easily be faked, mouse movements are continuous and contain many features, making them significantly harder to replicate. Thus an attacker would need valid, human-like mouse movements for each action. An example of several mouse movements is given in Fig. 2, which shows three different recordings of mouse movements, two from using a chat app (one activity produced by a human, the other by the advanced bot used in our evaluation) and one from using a rhythm game³ where the mouse is used heavily. This image depicts that the mouse dynamics are different per application and that advanced bots exist which do not exclusively move in straight lines.

To be consistent with prior work, we use machine learning models and request features proposed by [11] for bot detection on request data. For bot detection on mouse dynamics, we use machine learning models and mouse features proposed by [1]. Both papers are described in Sect. 2.

3.1 Request Data

Iliou et al. [11] ranked the best-performing metrics for simple and advanced bots per classification algorithm. However, some attributes used in their analysis are not suitable for this paper. For example, the authors include a Boolean indicating whether a request has a known search engine in their "Referer" header. Because we asked participants to visit the websites directly, this attribute is omitted. We use the following selection of metrics from Iliou et al.'s work [11] for our analysis:

1. The percentage of HTTP requests that led to an HTTP 4xx code response.
2. The percentage of HTTP requests that requested a CSS file.
3. The percentage of HTTP requests that requested a JavaScript file.
4. The percentage of HTTP-requested URLs that contain the previously requested URL as a subpart.
5. The total time between the first and the last HTTP request of the session.
6. Standard deviation of requested pages' depth (number of "/" in URL path).
7. Mean and Standard Deviation of times between successive requests.

³ <https://osu.ppy.sh/home>.

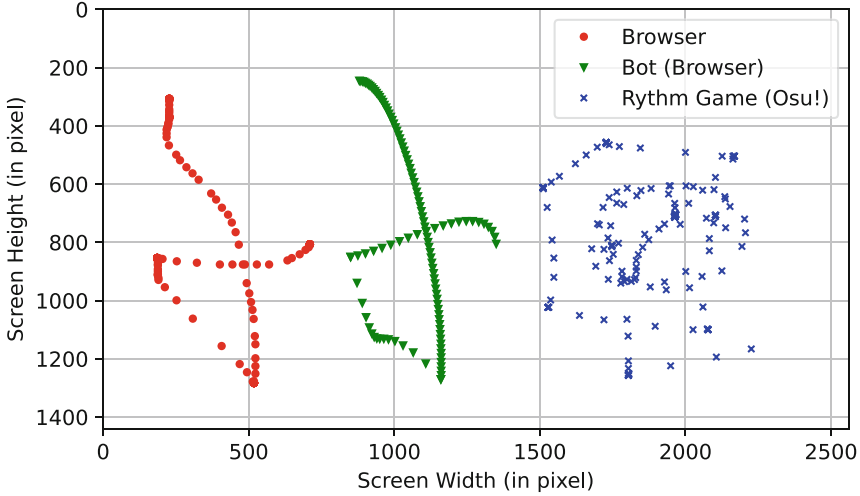


Fig. 2. Mouse dynamics example. Browser activity in a chat app on the left. In the middle movements of ghost-cursor (see footnote 7) creating a path between four given points. On the right a human playing a rythm game (Osu!).

3.2 Mouse Features

Mouse dynamic features consist of the relative x - and y -coordinates as well as a time value for each mouse event (e.g., left-click). Single mouse data points are grouped based on the following rules: They either end with a click, have a maximum of 50 data points, or span a maximum of two seconds. The features are calculated for each group. As a basis for all derived values, the time, x and y coordinates are linearly interpolated such that vectors with uniformly spaced values x'_t and y'_t every $20ms$ are generated. All indices start at zero.

In the following, we describe the additional used features engineered similarly to Gamboa et al. [7] and [4]. We include the path length from the origin, angle of the path tangent, horizontal, vertical, and overall velocity, acceleration, jerk, and angular velocity. Additionally, the type of action, length of the movement, and time needed to complete the action will be used.

The path length from the origin s'_t , i.e., the accumulated sum of previous segment lengths:

$$s'_t = \sum_{k=0}^{t-1} \sqrt{(x'_{k+1} - x'_k)^2 + (y'_{k+1} - y'_k)^2}$$

The angle of the path tangent with the x-axis θ_t is the arctangent ($atan2$ is used, which returns only values $-\pi < \theta < \pi$) of the segment at time $t > 0$. At $t = 0$ an angle of 0 is assumed.

$$\theta_t = atan2((y'_{t+1} - y'_t), (x'_{t+1} - x'_t))$$

The temporal features horizontal (v_x), vertical (v_y), tangential (v) and angular velocity (ω) as well as tangential acceleration (\dot{v}) and jerk (\ddot{v}) are computed as follows:

$$v_x = \frac{\delta x}{\delta t}; \quad v_y = \frac{\delta y}{\delta t}; \quad v = \sqrt{v_x^2 + v_y^2};$$

$$\omega = \frac{\delta \theta}{\delta t}; \quad \dot{v} = \frac{\delta v}{\delta t}; \quad \ddot{v} = \frac{\delta \dot{v}}{\delta t}$$

For each of the 9 vectors ($x'_t, y'_t, s'_t, v_x, v_y, v, \omega, \dot{v}, \ddot{v}$) the mean, standard deviation, minimum, maximum and value range (max-min) is calculated and yields the first 45 feature values.

Additionally, the time t_{total} and length s_{n-1} of the stroke (i.e. group of n data points), its straightness and jitter are computed. The time is the difference between the first and last data points' timestamps and the length can be the accumulated sum of segment lengths but using the raw instead of the interpolated data.

$$t_{total} = t_{n-1} - t_0$$

$$s_{n-1} = \sum_{k=0}^{n-1} \sqrt{(x'_{k+1} - x'_k)^2 + (y'_{k+1} - y'_k)^2}$$

Analogous to Gamboa et.al.'s definition [7], the *straightness* is defined as the ratio of the Euclidian distance between the first and last points of each group, and the total distance:

$$straightness = \frac{\sqrt{(x_0 - x_{n-1})^2 + (y_0 - y_{n-1})^2}}{s_{n-1}}$$

The *jitter* is the ratio between the original and smoothed path lengths:

$$jitter = \frac{s'_{n'-1}}{s_{n-1}}$$

In total, these 50 values make up the input vector that is computed for each mouse action group. We base our detection on whether a mouse movement is human on the features described above.

4 Evaluation

In this section, we summarize the evaluation results of our approach. We employed the best classifiers that were identified in related work. We utilized a random forest classifier [11] to analyze the request data, and for the mouse data, we also employed a random forest classifier [1]. One of the advantages of using a random forest classifier for the mouse data is its explainability, which helps in understanding how the model arrived at its predictions. We used the dataset described in Sect. 4.1 for our training and evaluation. The features used for the classifiers are described in Sect. 3.1 and Sect. 3.2. Further, we answer the following research questions:

- RQ1:** What is the performance of the detection depending on the available data, i.e., the number of requests and mouse movements?
- RQ2:** How does the performance of the machine learning model change when trained additionally with mouse dynamics from an external dataset, i.e., unrelated to mouse dynamics on our websites?

4.1 Data Collection and Augmentation

To train and evaluate bot detection approaches, we need a dataset of request data together with related mouse dynamics, i.e., the combination of requests and matching mouse movements of a user. However, such a dataset does not exist to our knowledge [10]. While some datasets on mouse dynamics exist [3], they are obtained by users that repeatedly perform specific mouse-intensive tasks. Since such a type of mouse dynamics differs from the mouse dynamics of users visiting a website, it could affect a classifier’s performance. We use this dataset in a second step to explore whether this is true.

Since no suitable dataset combining both features is publicly available [10], we need to build our dataset. For this, we invited users to visit and browse our two websites that log each request and every mouse movement. We announced our experiment with a link to our websites via a mailing list and had 322 participants visiting the first and 163 participants visiting the second website mentioned in the mail. We excluded mobile users, as well as users with no recorded mouse movements. Figure 3 shows the distribution of all users on both websites and their generated data points.

To integrate bot data we had to write our own. For this we use puppeteer. The behavior looks like this:

1. Accepting the initial prompt dialogue to start the experiment
2. Visiting the top-level pages {About, Blog, Contact/Imprint, Login, Register}
3. Visiting 10 randomly selected single blog pages
4. Visiting 100 randomly selected pages
5. Registering an account

All actions are configured to wait for the target element to be visible and clickable, scrolling it into view, if not. A random delay between 0 and 2 seconds is applied before each action. The behavior should reflect a scraper that does not scrape at full speed and in a specific order, e.g., width search.

Further, we distinguish between a basic mouse bot and an advanced mouse bot. The basic mouse bot moves the mouse on a direct path and at a constant speed to the target (links, blog posts, ...). The advanced mouse bot does not do this but uses bezier curves implemented in the popular javascript library `ghost-cursor`⁴, which promises human-like mouse movements. An example of such movements created by `ghost-cursor` is depicted in Fig. 2. We sample as many bot users as human users in the experiment.

⁴ <https://github.com/Xetera/ghost-cursor>.

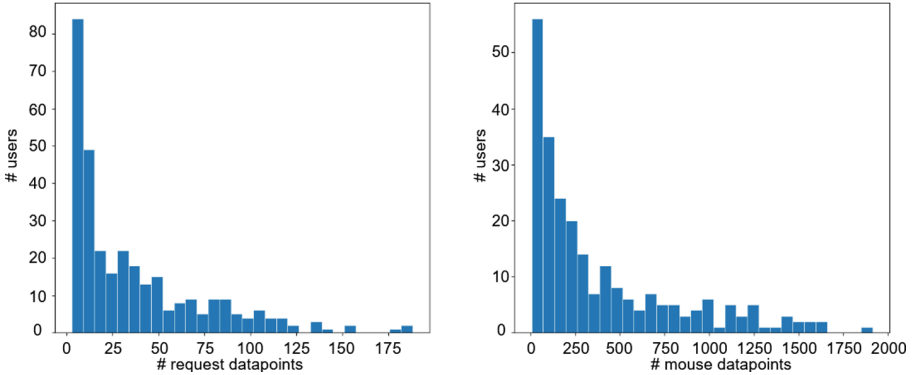


Fig. 3. Distribution of users in terms of data point count

4.2 Results

This section presents our results. Limitations are described in Sect. 4.3.

RQ1 - Bot Detection Performance. The performance of our model for the detection of bots depends on the data available. With more data available, the request data model’s evaluation metrics show increasingly good performance. Table 1 shows the detailed results. Note that there are many cases where fewer data points than the limit are available (cf. Figure 3). The mouse data model generally performs better the more data is available. Table 2 lists the performance for different amounts of data points per user. A big advantage of this approach is that more data points can potentially be acquired in a shorter amount of time compared to request data. For example, when sampling at 30 events per second, as this work’s implementation does, it only takes on average 1.66 s (50 samples) to capture the number of data points needed to surpass the request data model’s performance. When, for example, considering a potential application for a CAPTCHA, this time does not represent a significant disruption of most user interactions, e.g., filling in a registration form.

Parameter Tuning We used combinations of the following parameters to determine the best random forest parameters for bot detection empirically. Note that the number of bots and users in the dataset is the same, i.e., the same number of sessions. For mouse movements, we use the advanced bot that mimics human mouse movements.

1. Number of estimators (10, 50, 100, 150, 200, 1000)
2. Maximum number of features (None, log2, sqrt)
3. Maximum tree depth (None, 1, 2, 3, 4, 6, 7)

Tables 3 and 4 show the 10 best-performing combinations for mouse and request data. The data is sorted by accuracy. The additional scores Precision,

Table 1. Request data model performance with varying amounts of data points per user

Data Points/User	Acc	Precision	Recall	AUC	Time
200	0.980	0.980	0.980	0.982	0.209
100	0.970	0.961	0.980	0.985	0.209
No limit	0.960	0.960	0.960	0.972	0.215
50	0.950	0.941	0.960	0.976	0.208
20	0.910	0.918	0.900	0.975	0.221
5	0.900	0.885	0.920	0.945	0.220
10	0.890	0.842	0.960	0.956	0.220
4	0.880	0.913	0.840	0.922	0.211

Table 2. Mouse data model performance with varying amounts of data points per user (Advanced Mouse Bot)

Data Points/User	Acc	Precision	Recall	AUC	Time
No limit	0.966	0.964	0.968	0.993	0.449
50	0.933	0.943	0.922	0.979	0.124
200	0.930	0.951	0.906	0.979	0.189
100	0.920	0.942	0.895	0.975	0.149
20	0.855	0.846	0.868	0.949	0.114
10	0.850	0.862	0.833	0.903	0.120
5	0.846	0.909	0.769	0.916	0.099
4	0.826	0.895	0.739	0.879	0.098

Recall, AUC, and training time are computed as well. Their values of the top-performing results lie close together except for training time. The different values for the number of estimators and the maximum number of features for split consideration perform very similarly. For request data, the values for the following experiments were chosen to be 100 and *None*, respectively, as their result had the same accuracy and AUC as the top result. Analogously for the mouse data result, 200 and *sqrt* were chosen. All results have in common that no restriction to the decision trees’ maximum depth is applied, which is also the default value of *scikit-learn*’s implementation. This is expected as the tree depth is directly correlated with the ability to classify multi-dimensional input data.

Basic vs. Advanced Mouse Bot. The first direct comparison used all available human mouse data and the generated basic and advanced mouse data for training and testing datasets. Table 5 shows that the model performs better in every aspect and can classify inputs more reliably when using data generated only by the basic mouse bot with linear movements. This is expected as the bots’ linear movements are uniquely identifying properties that result in very specific out-

Table 3. Model accuracy for different parameters (request data)

Features	Estimators	Acc	Prec.	Recall	AUC	Time
None	1000	0.910	0.887	0.940	0.973	4.348
log2	200	0.910	0.887	0.940	0.973	0.732
log2	1000	0.910	0.887	0.940	0.975	3.965
None	100	0.910	0.887	0.940	0.973	0.486
sqrt	1000	0.910	0.887	0.940	0.972	3.912
log2	100	0.910	0.887	0.940	0.971	0.389
log2	150	0.910	0.887	0.940	0.972	0.694
None	50	0.900	0.870	0.940	0.971	0.273
sqrt	200	0.900	0.870	0.940	0.972	0.727
None	150	0.900	0.870	0.940	0.972	0.703

Table 4. Model accuracy for different parameters (mouse data)

Features	Estimators	Acc	Prec.	Recall	AUC	Time
sqrt	150	0.967	0.964	0.969	0.994	11.305
log2	1000	0.966	0.962	0.970	0.993	70.796
sqrt	200	0.966	0.962	0.970	0.994	5.899
sqrt	50	0.966	0.964	0.968	0.993	9.280
sqrt	100	0.966	0.962	0.969	0.994	9.815
sqrt	1000	0.966	0.960	0.971	0.994	82.889
log2	200	0.964	0.963	0.965	0.993	20.381
log2	150	0.962	0.963	0.960	0.993	5.950
None	150	0.961	0.954	0.969	0.991	91.586
None	200	0.961	0.953	0.969	0.991	110.588

comes for many input features. The model only correctly differentiated between humans and bots 96.6% of the time but still has a very high AUC.

Combining Mouse and Request Data for Advanced Bot Detection. Since we have a dataset containing request data and a user’s matching mouse dynamics, we can explore whether a bot detection system that combines both mouse and request data may improve the performance compared to using them individually. Therefore, we apply the classifiers mentioned above to the mouse and request data individually. Afterward, we combine the calculated predictions of the two classifiers and average them to determine the final result. We split the data into training and test sets on the user level, i.e., each user instance (human or bot) is only part of either the training or test set. We use two test ratios for this experiment, namely 0.1 and 0.2. Table 6 shows the overall performance, the precision of 1.0 was omitted from the table for readability. The most valuable

Table 5. Simple and Advanced Mouse Data Performance

Scenario	Acc	Prec	Recall	AUC	Time
Basic mouse	0.995	0.997	0.994	1.000	0.857
Advanced mouse	0.966	0.962	0.970	0.994	1.293

difference is that there are no false positive results, while the false negative rates of 0.272 and 0.327 are higher in contrast to using the classifiers separately. However, the lower false positive rate and same overall performance favor using the combined approach as it is a priority not to disrupt the user experience [9].

Table 6. Combined Mouse and Request Data Performance

Test ratio	Acc	Recall	AUC	TP	FP	TN	FN
0.1	0.960	0.953	0.976	122	0	22	6
0.2	0.950	0.940	0.970	252	0	49	16

RQ2 - Using Unrelated Mouse Dynamics for Training. We used Antal et al.’s dataset [3] to compare the performance to different real-world data. Inputs from 21 users were collected during their normal computer activities on one desktop and 20 laptop devices. Both mice and touchpads were used. Their raw mouse movement and interaction data are preprocessed similarly to the experiment’s data. The whole dataset yielded 1.54M input vectors. The initial assumption was that these mouse movements do not match the interaction with our website and the performance decreases when using these mouse movements. However, when trained additionally with the external dataset, the accuracy increases to 99.71%, and the values for FPR and FNR decrease to 0.55% and 0.15%, respectively. This indicates that the origin of the mouse movements is not important with regards to their effectiveness.

4.3 Limitations

The strongest limitation of our approach is that we only have access to a synthetic bot dataset, similar to approaches like [10]. Further, we create bots using fitting third-party projects. However, an external, labeled dataset with bot- and real-human traffic would be more suitable. The lack of such a realistic dataset leads to the performance of our model likely being worse outside our lab setting. Bots may act more camouflaged, e.g., by using recorded mouse movements. Those bots would probably escape detection. However, this is universal. Bots that behave completely like humans cannot be distinguished from humans. At the same time, making a bot behave like a human increases the costs for an attacker because the bot cannot work at full performance, e.g., scrape all data available or monitor a website for a long duration.

Further, while we were able to show that mouse data can easily be used for the detection of bots, another limitation is the focus on mouse data. This excludes users who interact without a mouse, e.g., only via keyboard, mobile devices, or screen readers.

5 Conclusion

We demonstrated the value of incorporating both mouse dynamics and request data when detecting bots. Unlike previous research that relied solely on one of the two data types or made unrealistic assumptions about the capability of advanced bots, we used a consistent dataset that included both mouse movements and request data belonging to the same user. Furthermore, we utilized a third-party library to create bots that performed human-like mouse movements. We used classifiers that performed best in literature for these tasks. We achieved better results with similar performance but in a more realistic setting. Thus mouse dynamics remain a useful tool for identifying even advanced bots. An interesting finding was that mouse data from third-party sources can be used to train the classifiers while achieving similar performance, thus simplifying the usage process. By leveraging third-party mouse data, operators can minimize the need to save and train on potentially sensitive user data. In the future, we intend to test our approach on a larger e-commerce dataset. Further, we want to consider more combinations of alternative approaches to bot detection, e.g., by including typing behavior as well as touch events from smartphones.

References

1. Acien, A., Morales, A., Fierrez, J., Vera-Rodriguez, R.: BeCAPTCHA-mouse: synthetic mouse trajectories and improved bot detection. [arXiv:2005.00890](https://arxiv.org/abs/2005.00890) [cs] (2021)
2. Akrouf, I., Feriani, A., Akrouf, M.: Hacking google reCAPTCHA v3 using reinforcement learning. arXiv preprint [arXiv:1903.01003](https://arxiv.org/abs/1903.01003) (2019)
3. Antal, M., Denes-Fazakas, L.: User verification based on mouse dynamics: a comparison of public data sets. In: 2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics, pp. 143–148. IEEE (2019)
4. Antal, M., Egyed-Zsigmond, E.: Intrusion detection using mouse dynamics. *IET Biomet.* **8**(5), 285–294 (2019)
5. Chu, Z., Gianvecchio, S., Wang, H.: Bot or human? A behavior-based online bot detection system. In: Samarati, P., Ray, I., Ray, I. (eds.) *From Database to Cyber Security*. LNCS, vol. 11170, pp. 432–449. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04834-1_21
6. Dee, T., Richardson, I., Tyagi, A.: Continuous transparent mobile device touchscreen soft keyboard biometric authentication. In: 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), pp. 539–540. IEEE (2019)
7. Gamboa, H., Fred, A.: A behavioral biometric system based on human-computer interaction. In: *Proceedings of the SPIE*, vol. 5404, pp. 381–392 (2004). <https://doi.org/10.1117/12.542625>

8. Gummadi, R., Balakrishnan, H., Maniatis, P., Ratnasamy, S.: Not-a-bot: improving service availability in the face of botnet attacks. In: NSDI, pp. 307–320 (2009)
9. Heath, N.: Expedia on how one extra data field can cost \$12m (2010). <https://www.zdnet.com/article/expedia-on-how-one-extra-data-field-can-cost-12m/>. Accessed 18 Oct 2021
10. Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S., Kompatsiaris, I.: Detection of advanced web bots by combining web logs with mouse behavioural biometrics. *Digit. Threats: Res. Pract.* **2**(3), 1–26 (2021)
11. Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S., Kompatsiaris, Y.: Towards a framework for detecting advanced web bots. In: Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019. Association for Computing Machinery, New York (2019)
12. Jonker, H., Krumnow, B., Vlot, G.: Fingerprint surface-based detection of web bot detectors. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019. LNCS, vol. 11736, pp. 586–605. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29962-0_28
13. Jorgensen, Z., Yu, T.: On mouse dynamics as a behavioral biometric for authentication. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 476–482 (2011)
14. Li, X., Azad, B.A., Rahmati, A., Nikiforakis, N.: Good bot, bad bot: characterizing automated browsing activity. In: 2021 IEEE symposium on security and privacy (SP), pp. 1589–1605. IEEE (2021)
15. Liu, W.: Introducing reCAPTCHA v3: the new way to stop bots (2018). <https://developers.google.com/search/blog/2018/10/introducing-recaptcha-v3-new-way-to>. Accessed 20 May 2021
16. Machines, I.: Stop more bots. start protecting user privacy (2018). <https://www.hcaptcha.com/>. Accessed 20 May 2021
17. Sayed, B., Traoré, I., Woungang, I., Obaidat, M.S.: Biometric authentication using mouse gesture dynamics. *IEEE Syst. J.* **7**(2), 262–274 (2013)
18. Shen, C., Cai, Z., Guan, X.: Continuous authentication for mouse dynamics: a pattern-growth approach. In: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), pp. 1–12 (2012). <https://doi.org/10.1109/DSN.2012.6263955>
19. Sivakorn, S., Polakis, J., Keromytis, A.D.: I’m not a human: breaking the google recaptcha. *Black Hat* **14** (2016)
20. Suchacka, G., Cabri, A., Rovetta, S., Masulli, F.: Efficient on-the-fly web bot detection. *Knowl.-Based Syst.* **223**, 107074 (2021)
21. Wei, A., Zhao, Y., Cai, Z.: A deep learning approach to web bot detection using mouse behavioral biometrics. In: Sun, Z., He, R., Feng, J., Shan, S., Guo, Z. (eds.) CCBR 2019. LNCS, vol. 11818, pp. 388–395. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31456-9_43



Practical Single-Round Secure Wildcard Pattern Matching

Jun Xu¹, Shengnan Zhao^{2,3}, Chuan Zhao^{2(✉)}, Zhenxiang Chen¹, Zhe Liu^{4,5},
and Liming Fang⁴

¹ University of Jinan, Jinan, China

² Quan Cheng Laboratory, Jinan, China
ise_zhaoc@ujn.edu.cn

³ Shandong University, Jinan, China

⁴ Nanjing University of Aeronautics and Astronaut, Nanjing, China

⁵ Zhejiang Lab, Hangzhou, China

Abstract. Secure pattern matching allows a client who holds a substring (pattern) to find all the substring's locations appearing in the long string (text) stored in a server. Meanwhile, the server should not learn any information about the pattern or the matching results. *Wildcard pattern matching* (WPM) problem, a specific variant with more realistic significance, defines that the pattern contains wildcards that can match any character in the text.

Previous studies introduce various approaches for the WPM problem but requires at least a two-round protocol or computation cost linear to input length. Oriented to applications in the client-server mode, however, existing solutions are not practical and efficient enough. Therefore we focus on the round and computation complexity of the WPM. In this paper, under the semi-honest model, we propose a single-round secure WPM protocol based on oblivious transfer (OT) and secret sharing schemes. The insight of our proposed protocol is the reduction from the WPM to the process of secret sharing and reconstruction in a novel way. We provide a customized OT construction and apply the OT extension technique to the protocol, where the client and the server need merely a constant number of public key operations in a round of communication. In addition, we prove the security of the protocol in the ideal/real simulation paradigm and evaluate the performance. Compared to existing secure WPM protocols, both theoretical and experimental results show that our protocol is more practical.

Keywords: wildcard pattern matching · oblivious transfer · secret sharing

1 Introduction

Secure multi-party computation (MPC) is an important subfield of cryptography and is first introduced by Andrew Yao [1] in the early 1980s. The goal of MPC

J. Xu and S. Zhao—Contributed equally to this work.

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 87–101, 2024.

https://doi.org/10.1007/978-3-031-56326-3_7

is to enable a set of independent mutually untrusted parties to jointly compute a function f on their private inputs, during which any additional information except for the output of that function cannot be revealed. Traditionally, two security models are mainly taken into consideration in MPC, *i.e.*, *semi-honest* model and *malicious* model. In the semi-honest adversarial model, the adversary follows the protocol instruction but tries to learn anything about the other party's input. In contrast, the adversary in the malicious model can follow any arbitrary polynomial-time strategy to deviate from the protocol.

Secure pattern matching (PM) problem is becoming a hot topic in the research field of MPC [2]. In this problem, the goal is to find all the locations of the pattern in a text for the party holding a pattern p , while the other learns nothing about the pattern. The PM problem can be formally defined as follows: given a finite alphabet Σ , a server holds a text $T \in \Sigma^n$ and a client holds a pattern $p \in \Sigma^m$ ($m < n$). The client wants to learn where its pattern is a substring of the server's text. Meanwhile, the server cannot learn any information about the pattern or the matching results. Considering a hospital holding patient genomic data, a researcher with a specific DNA sequence wishes to know the frequency and positions of the gene occurrences in the database and analyze the structure and properties of this sequence. However, the researcher does not intend to reveal its DNA sequence to the hospital. The hospital needs to prevent miscellaneous researchers from pirating the records of its genomic database on the other hand.

Wildcard pattern matching (WPM) problem, a specific variant with more realistic significance, defines that the pattern contains wildcards that can match any character of the alphabet in the text. We mainly focus on the WPM problem in this paper. In general, the wildcard character is denoted by \star , which could be any character of the alphabet. In addition, the server is not allowed to learn the locations of wildcard characters in the pattern, either. Same to the PM problem without wildcards, the client can only obtain information where the occurrence of the pattern in the server's text. Pattern matching is widely applied in text retrieval, computational biology, DNA analysis [3], intrusion detection systems [4], and other fields. Prior studies [5–8] introduce various approaches for the WPM problem but requires at least a two-round protocol or expensive computation cost (linear to input length).

Our Contributions. We design an efficient protocol that addresses the WPM problem in the presence of semi-honest adversaries. The proposed protocol is extremely competitive for lightweight devices in many scenarios such as Information Processing Systems. Our contributions can be summed up as follows:

- We provide a novel combination between the secret share scheme and a customized oblivious transfer protocol, which would be building blocks for the pattern matching problem.
- We propose an efficient single-round protocol with semi-honest security for wildcard pattern matching, which requires $O(\kappa)$ exponentiation computation and $O(mn)$ communication, where κ is the security parameter and independent of the input length m and n .

- We evaluate the performance and apply the precomputation technique and proxy OT to our contribution. We prove the security of the protocol in the ideal/real simulation paradigm.

2 Related Works

There are three main approaches to constructing secure pattern matching protocols: oblivious automaton evaluation, homomorphic encryption (HE), and Yao’s garbled circuit. To our knowledge, the protocols based on oblivious automaton evaluation are often used to solve the approximate/exact pattern matching problem [9–11]. Yao’s garbled circuits is a generic approach for secure computation, which can be used to evaluate arbitrary functions, given a description of the function as a fixed-size circuit. In 2010, Katz and Malka [12] showed how to modify Yao’s garbled circuits to obtain a secure pattern matching protocol where the size of the circuit is linear in the number of matched locations. Later, Kolesnikov *et al.* [7] believed that the protocol [12] can be extended to solve wildcard pattern matching, while there may be a requirement that it should provide a priori bound on the number of matches for the circuit construction. Secure wildcard pattern matching protocols based on homomorphic encryption schemes have been extensively studied in the past decade.

The first work of secure wildcard pattern matching was considered by Hazay and Toft [5] in 2010. In their scheme, the client is required to encrypt the wildcard locations using an additively homomorphic variation of ElGamal encryption and then supply the ciphertext to the server. In the meanwhile, the substrings of the server’s text must be modified to match the pattern at those positions. Baron *et al.* [13] suggested an efficient pattern matching protocol entitled 5PM. The core idea of their work is to reduce the problem of pattern matching with single-character wildcards to a sequence of linear operations, which can be efficiently computed in the malicious model using additively homomorphic encryption schemes. Therefore, they employed homomorphic encryption in an insecure pattern matching algorithm to support basic linear operations. Yasuda *et al.* [14] adopted a packing method and somewhat homomorphic encryption technique to address both approximate and wildcard pattern matching for non-binary inputs, where the encryption scheme supports a limited number of polynomial additions and multiplications on encrypted data. Their proposed packing method is applied to compute multiple Hamming distance values between the pattern and text in encrypted form.

Recently, Zarezadeh *et al.* [6] firstly resolved the parameterized pattern matching problem in the semi-honest and malicious setting where there exists a renaming bijection on the alphabet such that a pattern can be transformed into a substring of the text. Their proposed protocol supports wildcard and approximate pattern matching. Subsequently, they extended their construction to the multi-pattern matching scenario [15] that the pattern owner can find the matching locations in multiple texts and presented an efficient solution for parameterized matching of multiple patterns in the semi-honest adversary model.

In addition to the above methods, several works based on oblivious transfer for wildcard pattern matching were studied in [7, 8, 16]. The scheme of Kolesnikov *et al.* [7] called SWiM is a simple and fast protocol for wildcard pattern matching in a semi-honest setting, which converts the problem of wildcard pattern matching into the problem of secure equality test of strings. On the basis of this idea, Qin *et al.* [16] also presented a pattern matching protocol by combining oblivious transfer with secret sharing. However, these two works require an additional secure string equality test protocol.

3 Preliminaries

Throughout the paper, we use the following notation: The length of the text T is n , while the length of the pattern p is m . The notation t_i denotes the m -bit substring of the text T from the i -th location. The wildcard is denoted by \star . We use $[m]$ to denote a set $\{1, \dots, m\}$. We denote vectors in bold, and matrices in capitals. For a vector \mathbf{x} , we let $\mathbf{x}[k]$ denote the k -element of vector \mathbf{x} , $x_{i,j}$ denote the j -th share in the i -th secret sharing. For a matrix A , we let \mathbf{a}_i denote the i -th row of A , \mathbf{a}^j denote the j -th column of A . We use the notation $\vec{\mathbf{k}}$ to denote a tuple which contains three vectors.

3.1 Oblivious Transfer

Oblivious Transfer [17] is an essential cryptographic primitive that is used as a fundamental building block for MPC protocols. The standard definition of 1-out-of-2 OT involves two participants, a sender (denoted by \mathcal{S}) holding two inputs (m_0, m_1) and a receiver (denoted by \mathcal{R}) holding a choice bit $b \in \{0, 1\}$. After the transfer is completed, \mathcal{R} learns m_b without learning anything about the other input m_{1-b} , while \mathcal{S} has no output and learns nothing about b . The efficient 1-out-of-2 OT extension technique (also known as IKNP OT) is introduced by Ishai *et al.* [18], which can achieve an arbitrarily large number of OTs by executing a small (relying on security parameter) number of OT instances, and a number of symmetric key primitives. In 2013, Kolesnikov and Kumaresan [19] presented an optimization and generalization of IKNP OT extension protocol, which offers the sublinear communication/computation cost in the security parameter.

In this paper, our building block of independent interest is 1-out-of-3 OT, which is used to design a secure wildcard pattern matching protocol.

3.2 Secret Sharing

Secret sharing is a fundamental primitive, that is at the core of many MPC protocols. The idea of secret sharing was introduced by Shamir [20] and Blakey [21], and they constructed specific threshold secret sharing schemes based on Lagrange's interpolation theorem and projective geometry theory, respectively. Informally speaking, in a (t, n) -secret sharing scheme ($t \leq n$), the secrets s

can be split into n shares, which would be distributed among several parties, such that any $t - 1$ of the shares cannot leak anything about s , while any t shares allow complete reconstruction of the secret s . In secret-sharing-based MPC protocol, the target is then to obtain a secret-shared representation of the inputs to the computation, such that any possible set of adversarial parties reveals no information about the underlying secret. A (t, n) -secret sharing scheme needs to satisfy these two properties: *correctness* (meaning that any $k \geq t$ shares can completely determine the secret) and *privacy* (meaning that any set of shares of size less than t does not leak anything about the secret).

In our discussion, we will use (n, n) -secret sharing schemes, where all n shares are required and sufficient to reconstruct the secret.

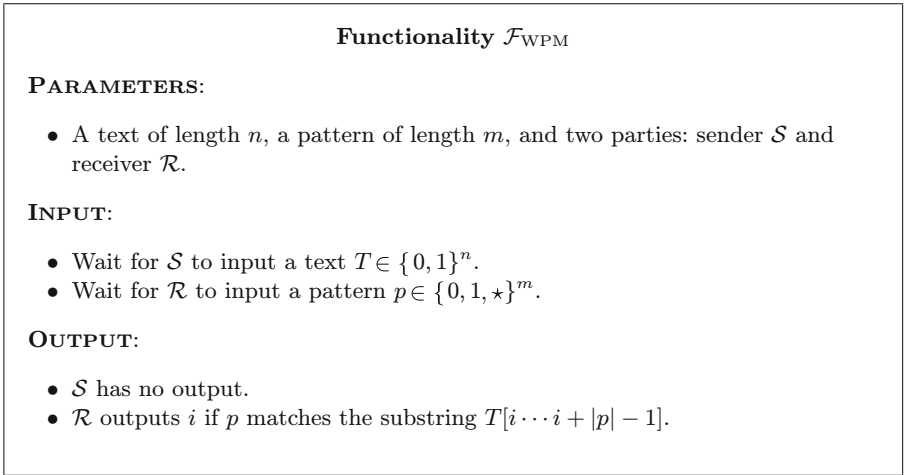


Fig. 1. Wildcard Pattern Matching Functionality

4 Secure Wildcard Pattern Matching Scheme

In this work, we present a secure two-party protocol for wildcard pattern matching based on two cryptographic tools: oblivious transfer and secret sharing. The rationale behind our secure wildcard pattern matching scheme is described in the following. Then, we explain how to construct a secure scheme for semi-honest parties.

4.1 Overview of Techniques

Before discussing our contributions and technical approach, we review the privacy-preserving wildcard pattern matching protocol in [16]. They transformed the secure wildcard pattern matching problem into reconstruction of a shared secret and presented a secure two-party wildcard pattern matching protocol

based on oblivious transfer and secret sharing. As usual, the sender \mathcal{S} has a text $T \in \{0, 1\}^*$ and the receiver \mathcal{R} holds a pattern $p \in \{0, 1, \star\}^*$. The intuition idea behind their protocol is that, (1) the pattern p is represented by the shares of secret s ; (2) the two parties with reverse roles invoke the standard 1-out-of-2 OT protocol, where \mathcal{R} as the sender inputs these shares and some random shares and \mathcal{S} as the receiver inputs the text T . (3) \mathcal{S} receives shares corresponding to its own input T from OT protocol and reconstructs the secret s' . If $s' = s$, this means \mathcal{S} receives all the valid shares. Therefore, the pattern p matches the text T . However, their proposed protocol is inapplicable to many application scenarios due to the limited computing power of the receiver (for example, if it is a mobile device). Furthermore, the sender and receiver in their protocol are required to perform a string equality test protocol relying on OT, which causes additional overhead. We modify their protocol and use 1-out-of-3 OT protocol to achieve wildcard pattern matching, where the two parties always maintain a single role during the whole execution of our protocol. In addition, we transfer most of the computational cost to the sender \mathcal{S} and achieve better efficiency by dispensing with the string equality test technique.

Our method can be depicted as follows. Suppose the sender \mathcal{S} holds a text $T \in \{0, 1\}^*$ and the receiver \mathcal{R} holds a pattern $p \in \{0, 1, \star\}^*$. To achieve secure pattern matching, the key is how to judge the i -th bit of p matches the i -th bit of T securely. As a very simple warm up, consider the case that $|T| = |p| = 1$. \mathcal{R} wants to know whether the pattern p matches the text T or not and \mathcal{S} can not obtain any information about p . Initially, \mathcal{S} will first choose a public string \mathbf{r} of length κ as a substitute of T and another string \mathbf{s} of the same length as \mathbf{r} randomly. According to the value of T , \mathcal{S} sets a tuple $\vec{\mathbf{k}}$ as follows: if $T = 0$, $\vec{\mathbf{k}} = (\mathbf{r}, \mathbf{s}, \mathbf{r})$; Otherwise, $\vec{\mathbf{k}} = (\mathbf{s}, \mathbf{r}, \mathbf{r})$. Then \mathcal{S} and \mathcal{R} jointly execute a 1-out-of-3 OT protocol, where \mathcal{S} gives input $\vec{\mathbf{k}}$ and \mathcal{R} gives input $c \in \{0, 1, 2\}$. After that, \mathcal{R} determines if p and T are matched based on whether its output from OT protocol is \mathbf{r} . Note if $p = \star$, then \mathcal{R} sets $c = 2$ and always can obtain \mathbf{r} . In this process, the only new information that \mathcal{R} obtains is output \mathbf{r} or \mathbf{s} , which leaks no information about the text T of \mathcal{S} .

Next, we extend this approach to the case $|T| = |p| = m$ by combining secret sharing. Specifically, \mathcal{S} represents its text T on shares of \mathbf{r} using a secret sharing scheme, *i.e.*, each bit of T is denoted by a secret share r_j . In the meantime, a random share s_j is selected for every secret share of \mathbf{r} . After doing so, \mathcal{S} will set a tuple $\vec{\mathbf{k}}_j$ for each bit of T using s_j and r_j in the same way as above. Then \mathcal{S} and \mathcal{R} invoke 1-out-of-3 OT protocol m times, where \mathcal{R} takes as input $c[j]$ for every time. T matches p if and only if \mathcal{R} obtains all shares of \mathbf{r} from OT protocol and reconstructs \mathbf{r} . An example is given in Fig. 2. During the whole execution, \mathcal{R} either receive all the valid shares of \mathbf{r} or at least a random share. In the latter case, \mathcal{R} can not reconstruct the publicly shared string \mathbf{r} . The security of the protocol is obvious in the semi-honest setting: \mathcal{R} can not obtain any information about the text T of \mathcal{S} but know whether the pattern p matches T .

The idea of the general case of wildcard pattern matching with $|T| > |p|$ is natural: simply perform the above method on each substring $T[i \dots i + |p| - 1]$

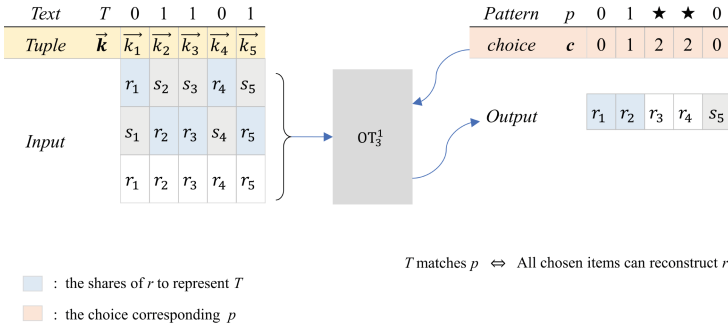


Fig. 2. Illustration of the core idea of our protocol

and p . Besides, note that in every matching of substring $T[i \dots i + |p| - 1]$ and pattern p , \mathcal{R} 's OT choice is always the same selection integer vector c . Hence instead of $|p|(|T| - |p| + 1)$ instances of string-OT where the string is κ bits long, we can use $|p|$ instances of string-OT, with string of length $\kappa(|T| - |p| + 1)$ to reduce computation cost.

4.2 Secure Wildcard Pattern Matching Protocol

In this section, we present a new secure wildcard pattern matching protocol Π_{WPM} based on 1-out-of-3 OT first and then give an analysis of correctness and a formal proof of security. The wildcard pattern matching functionality, denoted by \mathcal{F}_{WPM} , is formally defined in Fig. 1. Recall that the discussion in Sect. 4.1, T matches p if and only if \mathcal{R} obtain all shares of r from OT protocol. The detailed protocol is presented in Fig. 3 and is proven secure in the presence of semi-honest adversaries.

Correctness. Our goal here is to prove that in an honest execution of the protocol the output of \mathcal{R} is indeed the locations in the text T where the pattern p appears. Note that in the OT phase \mathcal{S} takes as input shares of r and corresponding random shares. The pattern p matches the substring t_i if only if \mathcal{R} can get all the valid shares of r and reconstruct r . For non-wildcard bits in pattern p , the relevant valid shares would be received by \mathcal{R} when these bits are equal to the values at the corresponding locations. In the case of wildcard bits in p , \mathcal{R} can always obtain the corresponding valid shares from OT protocol. Therefore, \mathcal{R} can reconstruct the publicly shared string r based on these valid shares and obtain the matching location. And if the match between t_i and p is unsuccessful, there is at least one random share will be outputted to \mathcal{R} . Under the circumstance, Bob cannot reconstruct the string r because of the property of the secret sharing scheme.

Security. We are now ready to prove the security of our protocol Π_{WPM} in the presence of semi-honest adversaries.

Wildcard pattern matching protocol Π_{WPM}

PARAMETERS:

- Two parties: *sender* \mathcal{S} and *receiver* \mathcal{R} .
- Text length n , pattern length m . Define $n' = n - m + 1$.
- Both parties share a public random string $\mathbf{r} \leftarrow \{0, 1\}^\kappa$ as auxiliary input.
- Ideal functionality OT-functionality $\mathcal{F}_{\text{OT}_3^1}$.

PROTOCOL:

1. **[secret sharing]** For each $i \in [n']$, \mathcal{S} invokes the secret sharing scheme to share the public random string \mathbf{r} into $r_{i,j}$, where $j = 1, \dots, m$ and $\mathbf{r} = \bigoplus_{j=1}^m r_{i,j}$.
2. \mathcal{S} chooses values $s_{i,j} \leftarrow \{0, 1\}^\kappa$ at random for $i \in [n']$ and $j \in [m]$.
3. Let \mathbf{t}_i denote the m -bit substring of \mathcal{S} 's text T for $i = 1, \dots, n'$. Then \mathcal{S} generates the values tuple for each $j \in [m]$ as follows:

- (1) if $\mathbf{t}_i[j] = 0$, the values tuple is in the order of $(r_{i,j}, s_{i,j}, r_{i,j})$;
- (2) if $\mathbf{t}_i[j] = 1$, the values tuple is in the order of $(s_{i,j}, r_{i,j}, r_{i,j})$;

Without loss of generality, let $\vec{k}_{i,j} = (k_{i,j}^0, k_{i,j}^1, k_{i,j}^2)$ denote each tuple. Then \mathcal{S} forms $n' \times m$ matrix U below using these tuples.

$$\begin{pmatrix} (k_{1,1}^0, k_{1,1}^1, k_{1,1}^2) & (k_{1,2}^0, k_{1,2}^1, k_{1,2}^2) & \cdots & (k_{1,m}^0, k_{1,m}^1, k_{1,m}^2) \\ (k_{2,1}^0, k_{2,1}^1, k_{2,1}^2) & (k_{2,2}^0, k_{2,2}^1, k_{2,2}^2) & \cdots & (k_{2,m}^0, k_{2,m}^1, k_{2,m}^2) \\ \vdots & \vdots & \ddots & \vdots \\ (k_{n',1}^0, k_{n',1}^1, k_{n',1}^2) & (k_{n',2}^0, k_{n',2}^1, k_{n',2}^2) & \cdots & (k_{n',m}^0, k_{n',m}^1, k_{n',m}^2) \end{pmatrix}$$

4. **[OT]** For each $j \in [m]$, \mathcal{S} and \mathcal{R} invoke 1-out-of 3 OT-functionality $\mathcal{F}_{\text{OT}_3^1}$
 - (1) \mathcal{R} acts as *receiver* with input a selection integer $c[j]$ corresponding to the j -th bit of p .
 - (2) \mathcal{S} acts as *sender* with input \mathbf{u}^j as the j -th column of U
 - (3) \mathcal{R} receives output $\mathbf{v}^j = (k_{1,j}^{c[j]}, k_{2,j}^{c[j]}, \dots, k_{n',j}^{c[j]})$
5. **[secret reconstruction]** \mathcal{R} forms $n' \times m$ matrix V using \mathbf{v}^j as follows.

$$\begin{pmatrix} k_{1,1}^{c[1]} & k_{1,2}^{c[2]} & \cdots & k_{1,m}^{c[m]} \\ k_{2,1}^{c[1]} & k_{2,2}^{c[2]} & \cdots & k_{2,m}^{c[m]} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n',1}^{c[1]} & k_{n',2}^{c[2]} & \cdots & k_{n',m}^{c[m]} \end{pmatrix}$$

For $i \in [n']$, \mathcal{R} reconstructs the public string \mathbf{r} using \mathbf{v}_i and outputs $\{i \in [n'] \mid \text{the values in } \mathbf{v}_i \text{ can reconstruct } \mathbf{r}\}$

Fig. 3. Secure Wildcard Pattern Matching Protocol

Theorem 1. *Assuming that the 1-out-of-3 oblivious transfer is secure against semi-honest adversaries and the secret sharing scheme satisfies that all shares are necessary and sufficient to reconstruct the secret, the protocol Π_{WPM} securely computes the functionality \mathcal{F}_{WPM} in the semi-honest setting.*

Proof. We prove Theorem 1 in a hybrid model where a trusted party is used to compute the oblivious transfer functionality $\mathcal{F}_{OT_3^1}$. We separately prove the case that \mathcal{S} is corrupted and the case that \mathcal{R} is corrupted. The formal proof is available on <https://github.com/Cathysrm/Proof-of-security> for the lack of space.

5 Performance Evaluation

5.1 Complexity Analysis

In this section, we analyze the efficiency of our scheme by comparing it with the most representative secure wildcard pattern matching protocol, SWiM [7], the recently proposed Zarezadeh *et al.*'s protocol [22] and [6].

The main cost of our protocol appears in all OTs in step 4. We use the notation $OT_3^1\text{-}\binom{m}{l}$ to denote m instances of 1-out-of-3 string-OT where the string is l bits long. Consider the case of $|T| = |p| = m$, $OT_3^1\text{-}\binom{m}{\kappa}$ are required in our protocol. According to the OT extension technique [19], any number of OTs can be obtained with communication proportional to the total size of parties' inputs and computation proportional to the size of the security parameter. Thus it is easy to see that the total communication cost of $OT_3^1\text{-}\binom{m}{\kappa}$ is the communication cost of implementing "base OT" instances plus $2\kappa m$ bits transferred for symmetric-key operations between \mathcal{S} and \mathcal{R} . These base OTs has $O(\kappa m)$ communication complexity. Therefore, we can conclude that the communication cost of $OT_3^1\text{-}\binom{m}{\kappa}$ is $O(\kappa m)$ bits. The computation cost can be reduced to $O(\kappa)$ exponentiations. As to $|T| > |p|$, we make use of a batched version of 1-out-of-3 OT to transfer the string of length $\kappa(n - m + 1)$ and thus avoid the requirement of $m(n - m + 1)$ instances of OT_3^1 on κ -bit strings. Consequently, our proposed protocol in the semi-honest model has $O(\kappa)$ computation complexity, and the total cost of communication of executing m OTs each of length $\kappa(n - m + 1)$ would be $O(\kappa mn)$ (the security parameter κ can be viewed as a constant).

As can be seen in Table 1, we summarize the performance of the above three protocols and our scheme. Both the research [22] and [6] focus mainly on homomorphic encryption. Compared with our scheme, their protocol has lower communication costs. But in terms of computation, in addition to exponentiations, it usually involves complicated encryption operations and massive multiplications on encrypted data, which requires higher computing capacity for a client. The SWiM protocol is also based on OT and requires only a small number of public-key primitives plus some symmetric-key operations from OT extension. It involves m instances of 1-out-of-2 OT on $(n - m + 1)$ -bit strings in the OT phase where the communication cost is $O(mn)$ bits. However, we note that the SWiM protocol still needs to perform private equality tests (PEQT) of strings

behind the OT phase and thus this finally leads to two rounds of communication, while the proposed protocol requires only single-round of communication. The remarkable thing here is that we consider merely the communication rounds in the online phase. Furthermore, in the most basic PEQT protocol, \mathcal{S} and \mathcal{R} check whether their l -bit strings x and y are equal by executing random $\text{OT}_2^1(\binom{l}{\kappa})$, where \mathcal{R} utilizes $y[i]$ as its choice and \mathcal{S} acts sender with input random κ -bit strings (s_0^i, s_1^i) . After that, \mathcal{R} obtains $s_{y[i]}^i$ and then computes $str_{\mathcal{R}} = \bigoplus_{i=1}^l s_{y[i]}^i$. \mathcal{S} computes $str_{\mathcal{S}} = \bigoplus_{i=1}^l s_{x[i]}^i$ where $s_{x[i]}^i$ corresponds to the i -bit of x and sends this value to \mathcal{R} . \mathcal{R} determines that $x = y$ iff $str_{\mathcal{R}} = str_{\mathcal{S}}$. Therefore, for each PEQT instance of l -bit strings, it requires l instances of OT_2^1 on κ -bit strings and causes $O(\kappa l)$ communication costs. In the case of the SWiM protocol, there are $n - m + 1$ instances of PEQT on m -bit string, so $O(\kappa m(n - m + 1))$ communication costs are required. Overall, we have the same communication and computation complexity to the SWiM protocol but better communication rounds.

5.2 Experimental Performance

We analysis the efficiency of the proposed protocol through some experimental results in this section.

Since the main cost of our protocol comes from 1-out-of-3 OT, we performed the experiments on m instances of OT_3^1 on $\kappa(n - m + 1)$ -bit strings instead of implementing the whole protocol. Our implementation was done in libOTe library [23]. All runs have been taken on a virtual machine with 4GB RAM and 8 cores (the host machine is Intel Core i5-10210U 2.11 GHz with 20 GB RAM). The running time of our scheme compared with SWiM is shown in Table 2 and all running times are reported as the average over 20 trials.

As experimental results show, the proposed scheme is more significantly efficient than the protocols of SWiM in small-scale text/pattern, taking 0.034 seconds to conduct a wildcard pattern matching operation for text length $|T| = 10^3$ and pattern length $|p| = 10^2$. We see a $27.4\times$ improvement in running time compared to SWiM. However, considering the larger values for size of text/pattern, our performance improvement is unsatisfactory or even worse in the extreme case where the length of the text is much greater than that of the pattern. Due to the limited computing power, the detailed results of our performance at scale can not be provided. We can conclude that the proposed scheme is optimized for the case where the size of the text is approximate to that of the pattern. For instance $|T| = |p| = 10^3$, our scheme needs to generate 1000 instances of OT_3^1 on κ -bit strings and finally takes 0.039 s. Using the same parameters, the SWiM protocol results in 1.019 seconds on executing 1000 instances of OT_2^1 on κ -bit strings and one PEQT instance of 1000-bit strings. This is a $26.1\times$ improvement.

Table 1. Complexity comparison for text length = n , pattern = m , computation security parameter κ , a finite alphabet Σ

work	security	method	communication	round	computation		
					expo.	multi.	enc.
[7]	semi.	OT	$O(mn)$	2	$O(\kappa)$	–	–
[22]	mal.	HE	$O(m+n)\kappa$	2	–	$O(mn)$	$O(m \Sigma)$
[6]	semi.	HE	$O(n)$	2	$O(n)$	$O(mn)$	$O(m(\Sigma +m))$
Ours	semi.	OT+SS	$O(mn)$	1	$O(\kappa)$	–	–

6 Optimizations

In this section, considering that the main computing and communication overhead in the WPM protocol comes from OT, we give two optimizations on securely computing functionality $\mathcal{F}_{\text{OT}_3^1}$.

6.1 Online/Offline OT

We briefly describe how the protocol can be modified so that most of the cost can be incurred in an offline phase before the parties' inputs are known. The idea of precomputing OT could trace back to Beaver's work [24] where a precomputing 1-out-of-2 OT construction is given, we apply this idea to a more general case. See Fig. 4 for a full description.

On the basis of the precomputing protocol, we are ready now to briefly describe how the protocol Π_{WPM} can be modified so that most of the cost can be incurred in an offline phase before the parties' inputs are known. In brief, we are able to run all OTs in Step 4 of the protocol by invoking our precomputing protocol. First, the receiver uses a random $\sigma[j] \in \{0, 1, 2\}$ as its OT choice. The sender chooses a $n' \times m$ matrix W randomly and takes as OT input \mathbf{w}^i denoting the i -th column of matrix W . The each element of this matrix $\mathbf{w}_{i,j}$ is denoted by a tuple $\vec{\gamma}_{i,j} = (\gamma_{i,j}^0, \gamma_{i,j}^1, \gamma_{i,j}^2)$. Later, upon learning p , the receiver sends $\theta = \mathbf{c}[j] - \sigma[j] \pmod 3$ to the sender, where $\mathbf{c}[j]$ denotes the i -th bit of p . As the sender learns its input T , it prepares the value tuple $\vec{\mathbf{k}}_{i,j} = (k_{i,j}^0, k_{i,j}^1, k_{i,j}^2)$ and computes $\vec{\mathbf{z}}_{i,j} = \vec{\mathbf{k}}_{i,j} \oplus \vec{\gamma}_{i,j}$ as follows: $z_{i,j}^a = k_{i,j}^a \oplus \gamma_{i,j}^b$ where $b = a - \theta \pmod 3$. The receiver can compute $k_{i,j}^{\mathbf{c}[j]} = \gamma_{i,j}^{\sigma[j]} \oplus z_{i,j}^{\mathbf{c}[j]}$ after receiving $\vec{\mathbf{z}}_{i,j}$ from the sender. By the precomputation technique, we are able to transfer most of the $O(nm)$ communication to the offline phase, and the resulting protocol is still secure.

6.2 Proxy OT

So far as we know, all known oblivious transfer protocol relies on a large number of asymmetric-key operations, which are typically implemented by modular

Table 2. Comparison of the total runtime (in seconds) for the text of length $|T| = n$, the pattern of length $|p| = m$

$ p $	$ T $	SWiM	Ours	Improved
10	10^3	0.846	0.025	33.8×
	10^4	0.953	0.031	30.7×
	10^5	1.073	0.098	10.9×
10^2	10^3	0.930	0.034	27.4×
	10^4	1.011	0.097	10.4×
	10^5	2.052	0.684	3.0×
10^3	10^3	1.019	0.039	26.1×
	10^4	1.122	0.627	1.8×
	10^5	2.227	–	–
2^8	2^{12}	0.993	0.098	10.1×
	2^{14}	1.139	0.334	3.4×
	2^{16}	1.771	1.563	1.1×
2^{10}	2^{12}	1.072	0.348	2.3×
	2^{14}	1.596	1.388	1.2×
	2^{16}	2.139	–	–

PARAMETERS:

- Two parties: *Sender* \mathcal{S} and *Receiver* \mathcal{R} .
- A security parameter κ .
- Ideal functionality $\mathcal{F}_{\text{OT}_3^1}$ primitive.

INPUT OF \mathcal{S} : Message set $M = \{m_0, m_1, m_2\}$.

INPUT OF \mathcal{R} : Choice σ .

PROTOCOL:

- Precomputing Phase:
 1. \mathcal{S} generates random message set $M' = \{m'_0, m'_1, m'_2\}$ and sends $\mathcal{F}_{\text{OT}_3^1}$ M' . Here $|m'_i| = |m_i| = l$.
 2. \mathcal{R} sends $\mathcal{F}_{\text{OT}_3^1}$ a random choice σ' and receives message $\hat{m} = m'_{\sigma'}$.
- Online Phase:
 1. \mathcal{R} computes $\delta = \sigma - \sigma' \pmod 3$ and sends δ to \mathcal{S} .
 2. After receiving δ , \mathcal{S} computes $x_i = m_i \oplus m'_j$, here $j = i - \delta \pmod 3$.
 3. \mathcal{R} receives m_σ by computing $x_\sigma \oplus \hat{m}$.

Fig. 4. Generic Precomputing OT_3^1 Protocol

exponentiations, that are dense computational tasks. The computational overhead of oblivious transfer is usually more critical than that in communication. The construction of Ishai *et al.* [19] shows an efficient extension technique with

the additional symmetric-key operation to achieve massive effective OT, which reduces the number of asymmetric operations. However, in many certain scenarios, the receiver with limited computational resources *e.g.*, a handheld device can not undertake such intensive computational tasks even using OT extension technology. We wish to minimize the computational task of the receiver. Therefore, we use *proxy oblivious transfer*, a variant of oblivious transfer proposed in [25], to further reduce the computation overhead of the receiver.

In *proxy oblivious transfer* protocol, there are three parties: A sender that holds two messages m_0 and m_1 , and a receiver with a choice $\sigma \in \{0, 1\}$, as well as a third party, the proxy, which has no inputs and serves as the receiver's proxy to learn the chosen item. At the end of the protocol, the proxy receives the output m_σ without learning the choice σ , while the sender and receiver learn nothing.

Our protocol can be implemented on the basis of the 1-out-of-3 proxy OT. It is remarkable that this improvement is particularly useful for the receiver with low computational power, since most of the computational overhead is transferred to the proxy, and it can actually compute all the exponentiations in the preprocessing phase.

7 Conclusion

In this paper, we transformed the wildcard pattern matching problem into reconstruction of a shared secret by combining XOR-secret-sharing and 1-out-of-3 OT and presented an efficient protocol with security against semi-honest adversary. The proposed protocol has the same communication and computation complexity to the state-of-the-art solutions but better round complexity.

Acknowledgement. This work is supported by the Taishan Scholars Program, National Natural Science Foundation of China (No. 61702218, 62172258, 61672262), Shandong Provincial Natural Science Foundation (No. ZR2021LZH007, ZR2019LZH015), Shandong Provincial Key Research and Development Project (No. 2019GGX101028, 2018CXGC0706, 2021SFG C0401), Project of Independent Cultivated Innovation Team of Jinan City (No. 2018GXRC002).

References

1. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982), pp. 160–164. IEEE (1982)
2. Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C.-Z., Li, H., Tan, Y.: Secure multi-party computation: theory, practice and applications. *Inf. Sci.* **476**, 357–372 (2019)
3. Xu, G., Li, H., Ren, H., Lin, X., Shen, X.S.: DNA similarity search with access control over encrypted cloud data. *IEEE Trans. Cloud Comput.* **10**, 1233–1252 (2020)
4. Namjoshi, K., Narlikar, G.: Robust and fast pattern matching for intrusion detection. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9. IEEE (2010)

5. Hazay, C., Toft, T.: Computationally secure pattern matching in the presence of malicious adversaries. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 195–212. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_12
6. Zarezadeh, M., Mala, H., Ladani, B.T.: Secure parameterized pattern matching. *Inf. Sci.* **522**, 299–316 (2020)
7. Kolesnikov, V., Rosulek, M., Trieu, N.: SWiM: secure wildcard pattern matching from OT extension. In: Meiklejohn, S., Sako, K. (eds.) FC 2018. LNCS, vol. 10957, pp. 222–240. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-58387-6_12
8. Wei, X., Zhao, M., Xu, Q.: Efficient and secure outsourced approximate pattern matching protocol. *Soft. Comput.* **22**(4), 1175–1187 (2018)
9. Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient DNA searching through oblivious automata. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 519–528 (2007)
10. Frikken, K.B.: Practical private DNA string searching and matching through efficient oblivious automata evaluation. In: Gudes, E., Vaidya, J. (eds.) DBSec 2009. LNCS, vol. 5645, pp. 81–94. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03007-9_6
11. Gennaro, R., Hazay, C., Sorensen, J.S.: Text search protocols with simulation based security. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 332–350. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_20
12. Katz, J., Malka, L.: Secure text processing with applications to private DNA matching. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 485–492 (2010)
13. Baron, J., El Defrawy, K., Minkovich, K., Ostrovsky, R., Tressler, E.: 5PM: secure pattern matching. *J. Comput. Secur.* **21**(5), 601–625 (2013)
14. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Privacy-preserving wildcards pattern matching using symmetric somewhat homomorphic encryption. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 338–353. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08344-5_22
15. Zarezadeh, M., Mala, H.: Secure parameterized multi-pattern matching in multi-text owner setting. In: 2021 18th International ISC Conference on Information Security and Cryptology (ISCISC), pp. 6–12. IEEE (2021)
16. Qin, H., Wang, H., Wei, X., Xue, L., Lei, W.: Privacy-preserving wildcards pattern matching protocol for IoT applications. *IEEE Access* **7**, 36094–36102 (2019)
17. Rabin, M.O.: How to exchange secrets with oblivious transfer. *Cryptology ePrint Archive* (2005)
18. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
19. Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 54–70. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_4
20. Shamir, A.: How to share a secret. *Commun. ACM* **22**(11), 612–613 (1979)
21. Blakley, G.R.: Safeguarding cryptographic keys. In: International Workshop on Managing Requirements Knowledge, p. 313. IEEE Computer Society (1979)
22. Zarezadeh, M., Mala, H., Ladani, B.T.: Efficient secure pattern matching with malicious adversaries. *IEEE Trans. Dependable Secure Comput.* **19**, 1407–1419 (2020)

23. Rindal, P.: libOTe: an efficient, portable, and easy to use oblivious transfer library (2018)
24. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_8
25. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce, pp. 129–139 (1999)



Efficient Non-interactive Anonymous Communication

Sigurd Eskeland^(✉) and Svetlana Boudko

Norsk Regnesentral, Postboks 114 Blindern, 0314 Oslo, Norway
sigurd@inbox.com

Abstract. Methods for untraceable and anonymous communication, such as anonymous routing networks and dining cryptographers networks, are in general very complex and suffer from high performance overhead of a minimum order of N^2 encryptions for N participants. In this paper, we propose an original approach to untraceable communication that avoids some of the significant shortcomings of existing methods. Using non-interactive privacy-preserving aggregation as an underlying building block we achieve attractive features, including unsurpassed low computational and transmission overhead of only 3 encryptions per participant in only a single round.

Keywords: Anonymous communication · Secure multiparty computation · Privacy-preserving aggregation

1 Introduction

The necessity for privacy exists in numerous ICT applications. Anonymous communication refers to sender untraceability in which transmitted data elements cannot be linked to the sender. Sometimes receiver untraceability is also desired, in which the receiver of transmitted data elements cannot be determined. Sender anonymity, untraceable data transmission, data collection, and message submission are desired core privacy features of many privacy-related applications, such as voting systems, anonymous surveys, social networks, and more. Anonymous communication systems can be divided into categories including anonymous routing networks (AR nets) and dining cryptographers networks (DC nets) [1, 6, 9, 15, 18, 20, 23, 27, 28]. AR nets cover approaches such as mix nets [7, 10–12, 29, 30], onion routing [17], and crowds [24]. These are end-to-end-oriented, in which a sender communicates with a receiver through a network of designated proxies (message forwarding nodes) by which the communication becomes untraceable w.r.t. the senders. AR nets have generally low to medium latency.

DC nets differ from AR nets by not using message-forwarding and having no trusted proxies. Rather they are group- and broadcast-oriented, assuming a closed group of N participants, where each group member is able to communicate anonymously with the other group members. To realize untraceable transmission of a single message, the collaboration of all group participants is necessary.

An inherent implication is that the sending order must be determined anonymously, and thus an additional synchronization protocol is necessary, adding to the complexity.

The existing approaches have significant disadvantages regarding high complexity and high transmission overhead of an order of minimum N^2 messages and a correspondingly high computational overhead. All existing approaches are interactive, which impacts latency and incurs in general very complex protocol designs. A performance comparison with some prominent and well-performing anonymous communications schemes is shown in Table 1.

Contributions. In this paper, we propose a novel and original approach to untraceable communication that avoids the mentioned significant shortcomings of existing methods. Using non-interactive privacy-preserving aggregation [4, 8, 16, 19, 25] as a building block we achieve the following attractive features:

- Unsurpassed low transmission and computational overhead: Each participant computes and broadcasts only 3 encryptions.
- Non-interactive: All participants transmit (broadcast) in a single round. No particular sending order or synchronization is necessary, and there are no user interactions where one transmission triggers off another transmission.
- Simple design: There is only a single round. The scheme is simple to implement.

Outline. Related work is discussed in Sect. 2. Some background on privacy-preserving aggregation is presented in Sect. 3. In Sect. 4 the novel anonymous communication scheme is presented, and Sect. 5 concludes the paper.

2 Related Work

In this section we look into techniques for anonymous communication that we divide into the two categories of anonymous routing networks and dining cryptographers networks. For a comprehensive survey, see Shiraz et al. [26]. Existing techniques are in common complex and have low performance with transmission and computational overhead of an order of minimum N^2 .

2.1 Anonymous Routing Networks

Anonymous routing techniques comprise mix nets [10], including variants such as [7, 29] discussed below, onion routing [17], and crowds [24]. These are anonymization networks, assuming a number of intermediate computers (or nodes) forwarding messages on behalf of a sender and a receiver. The intermediate nodes are third parties that may be trusted or untrusted.

Mix nets were originally proposed by Chaum [10], having a number of N intermediates (called *mix nodes* or *mixes* by Chaum) in a fixed chain configuration (*mix cascade*). The senders encrypt their messages N times using the public key of each intermediate. Each intermediate waits for a predefined number (threshold) of ciphertexts to be received (into a *batch*). When the threshold

is reached, it decrypts each ciphertext, thus removing one layer of encryption, and forwards the partially decrypted ciphertexts in a random (shuffled) order to the next intermediate. Assuming that the ciphertexts have the same lengths, the reshuffling breaks the input-output-relation, creating unlinkability. In conclusion, mix nets have a high transmissions and computation cost of N^2 transmission and N^2 public key encryptions. The threshold criterion combined with the serial mix node configuration incur high latency.

A mix net variant was proposed by Yang et al. [29] for anonymous (oneway) data collection, enabling a fixed group of *respondents* to anonymously send to a single receiver (the *miner*). It has two operational distinctions from [10]; instead of a chained *mix cascade*, the participants interact in parallel with the receiver; and instead of multiple encryptions, each sender encrypts only once using an aggregated Diffie-Hellman-type public key, $y = \prod_{i=1}^N y_i = g^x$, of which the sum of the corresponding private keys is $x = \sum_{j=1}^N x_j$. Each responder P_i produces an ElGamal ciphertext $(c_i = m_i y^{r_i}, d_i = g^{r_i})$, where r_i is an ephemeral private key. The miner sends all N encryptions to respondent P_1 , who blinds them by multiplying an ephemeral random encryption factor δ_k to both ElGamal cipher elements $(c_j y_k^\delta, d_j g_k^\delta)$, $1 \leq k \leq N$, before returning the modified ciphertexts in a random order to the miner, thus achieving untraceability. This process is repeated until the last respondent P_N . Next, each respondent P_i computes N partial decryptions, i.e., the ElGamal decryption factor $e_{j,i} = (d_j g_k^\delta)^{x_i}$, $1 \leq j \leq N$. By combining the partial decryptions pertaining to a given ciphertext, the plaintexts $m_i = (c_i y_k^\delta) \prod_{j=1}^N e_{i,j} = (m_i y^{r_i} y_k^\delta) \prod_{i=1}^N g^{-r_j x_i} g^{\delta_k x_i}$, $1 \leq i \leq N$, are restored due to the mutual elimination of the respective encryption factors. The transmission overhead is high, by $4N^2 + N$ transmitted ElGamal encryptions.

Brickell and Shmatikov [7] proposed a mix net scheme providing fault detection, but instead of third party mixes, a group of N sending participants form a chain configuration, using layered encryption. Each participant encrypts his or her message N times using the public key of each of participant in order P_1, P_2, \dots, P_N , resulting in C_i . This encryption process is then repeated, but now N encryption layers are added using a secondary public key of each participant, resulting in C'_i . The reason for the repeated encryptions is for detecting faulty behavior. An intermediate waits for a number of ciphertexts to be received, decrypts, thus removing one layer of encryption, and forwards the new ciphertexts in a random order to the next intermediate. The participants send their multi-layered ciphertexts to P_1 who decrypts all ciphertexts, randomly and secretly rearranges their order. P_1 sends the shuffled ciphertexts \vec{C}_1 to P_2 who decrypts and shuffles the ciphertexts in \vec{C}_1 , resulting in \vec{C}_2 . This is done by all, ending with P_N . The decryptions are checked by all participants who checks that his encryption C'_i is included. If alright, the participants remove the remaining N layers of encryption.

Crowds [24] was intended for anonymous web browsing. In crowds, messages are routed by the intermediates by means of random paths to and from the intended web servers. When a web browser makes a request, an intermediate flips a biased coin to determined whether to forward the request to another

intermediate or directly to the web server. In the former case, the intermediate selects another intermediate at random to which it forwards the request. Subsequent requests and server responses follow the same path. Crowds assume symmetric key encryption, and each pair of intermediates share a symmetric encryption key. To reduce latency, communication is encrypted using a *path key*. This key is generated by the first intermediate, and is forwarded to the next intermediate in the path by encrypting it with the key it shares with the next intermediate. Crowds does not provide anonymity against global eavesdroppers.

2.2 Dining Cryptographers Networks

The idea of dining cryptographers networks (DC nets) was proposed by Chaum [9]. The concept was later revised by Waidner [27] using additive groups \mathbb{G} modulo p , instead of bitwise XOR operations as originally proposed by Chaum. DC nets assume a (closed) group $\mathcal{U} = \{P_1, \dots, P_N\}$ of N participants arranged in a logical *ring* topology or a *complete graph* (or star) topology, where a node symbolizes a user and an edge represents a secret key that is shared by its two adjacent users. Specifically, in a ring arrangement, each user P_i shares the secret ephemeral key $k_{i-1,i}$ with the adjacent user P_{i-1} and the ephemeral key $k_{i,i+1}$ with P_{i+1} , so that each user holds in total two shared secrets. In a star arrangement, each user shares a unique secret with each other user, so there are in total $N(N-1)$ shared secrets. Application of the Diffie-Hellman-primitive is convenient for establishing shared ephemeral DH keys. DC nets provide unconditional security, which may not be provided by the chosen key establishment method.

For each round, only one participant can send an encrypted plaintext, but the participation of the remaining users is required. To send a message anonymously (in a ring topology), $P_i \in \mathcal{U}$ blinds the plaintext m_i by means of the shared secrets, and broadcasts the encryption $c_i = m_i \oplus k_{i-1,i} \oplus k_{i,i+1}$. Each of the other participants $P_j \in \mathcal{U} \setminus \{P_i\}$ broadcasts an “empty” encryption $c_j = k_{j-1,j} \oplus k_{j,i+1}$. The plaintext is anonymously obtained by combining all encryptions $m_i = \bigoplus_{j=1}^N c_j$, in which the pairwise shared keys cancel each other out. As the encryptions are indistinguishable, it is not possible to determine the originator of m_i .

DC nets can be realized by additive groups [27] for integrating number-theoretic primitives such as Diffie-Hellman key establishment. An (non-empty) encryption on this form becomes $c_i = m_i k_{i-1,i}^{-1} k_{i,i+1} \pmod p$, where $k_{i-1,i}$ and $k_{i,i+1}$ are secret ephemeral DH blinding factors. Aggregating the encryptions by multiplication restores $m_i = \prod_{j=1}^N c_j \pmod p$.

If two or more encrypted plaintexts are sent in one round, a *collision* occurs resulting in an aggregated plaintext, in which case the plaintexts cannot be restored. To prevent collisions, it is necessary to establish an anonymous sending order before the sending can take place. This is known as *scheduling* or *slot reservation*, of which the participants agree on a sending schedule in an anonymous way so that each participant learns when to send but not who is sending in the other respective rounds.

In addition to the mentioned disadvantages, the pairwise shared encryption factors must be ephemeral to prevent plaintext leakage. Depending on the key establishment scheme, this incurs extra transmission and computational overhead. Other shortcomings of DC nets are “churn”, scalability, and high latency.

2.3 Disruption

DC nets are vulnerable to disruption, in which a participant sends an invalid encryption or a random message, or modifies somebody else’s valid encryption, with the result that legitimate plaintexts cannot be recovered. This begs the questions of how disruptions be can detected and how invalid encryptions can be identified. Disruption detection can be done *subjectively*, meaning that the originator of an encryption checks whether his or her encryption has been tampered with, in which case that user must take action and inform the remaining participants. “Universal” disruption detection capability is a stronger feature that allows anybody to check the correctness of any encryption.

Disruption identification is considered a difficult problem in DC nets, since this should be resolved without compromising anonymity of honest participants. Disruption is related to *collision detection* and *collision resolution*, in the context of slot scheduling [5,6,27,28], in the sense that collisions may render colliding encryptions invalid.

Chaum was the first to address this problem by proposing a *trap* mechanism, in which the participant declare a *trap* or *non-trap* for his or her reserved sending slot. If a trap is declared, the message is random, and a cryptographic commitment for the random message is computed. If a dishonest party tries to disrupt the communication, this is detected by the sending party who proves this by opening the commitment by sending the encryption key. Disadvantages are the added transmission overhead and that this mechanism is probabilistic, effective in some but not all cases.

Golle and Juels [18] proposed using non-interactive zero-knowledge proofs based on the bilinear Diffie-Hellman key exchange primitive. Their approach allows anybody to (universally) detect invalid encryptions by means of additional verification elements. Long-terms encryption keys are generated using a polynomial function (cf. (k, n) -threshold cryptography) to enable recovery from disruption or network faults. A variant of zero-knowledge proofs using Pedersen commitments was proposed by Franck and van de Graaf [15].

Nosouhi et al. [23] describe a straight-forward XOR-type DC-net scheme with subjective disruption management. If a user detects that his or her encryption has been tempered with or corrupted, everybody has to publicize their pairwise keys pertaining to the relevant slot. Then each user P_i checks if the keys shared with the adjacent users P_{i-1}, P_{i+1} are different from the publicized keys. Since the message cannot be recovered, it can be said that anonymity is preserved in such situations.

2.4 Hybrid Approaches

Corrigan-Gibbs and Ford [11] proposed a hybrid approach (*Dissent*) combining mix nets (the shuffle scheme by Brickell and Shmatikov [7] described in Sect. 2.1) and XOR-type DC nets. Its goals are moderate-sized groups, variable-length message support, and disruption support. Despite that the bulk scheme builds on DC nets, the authors claim that their approach is “efficient enough for latency-tolerant messaging in small distributed groups”, a claim that is most reasonably disputed by Bauer and Staudemeyer [1].

Each user establish two ephemeral key pairs, for inner and outer encryption, and all public keys are distributed to all users. By means of the shuffle scheme, each user P_i anonymously distributes a message descriptor d_i , whose purpose is to facilitate variable length encryptions w.r.t. the subsequent invocation of the DC net protocol. It indicates the size of his or her message m_i . Regarding the subsequent DC net encryption, it contains N “seeds”, each seed is encrypted with the public key of the respective user.

P_i encrypts d_i N times using the “inner” public key of each of participant in order P_1, P_2, \dots, P_N , resulting in the ciphertext C'_i , and N more times by means of the N outer public keys, resulting in C_i . The reason for the outer encryption layers is to detect faulty behavior (disruption). After the N outer encryption layers have been removed during the shuffle, P_N broadcasts the inner-encrypted message descriptors. Everybody checks that his C'_i is included or not. If there are no complaints about missing or invalid encryptions, then all participants broadcasts the inner private key, allowing all participants to remove the remaining N layers of encryptions from each C'_i , obtaining the message descriptor plaintexts in random order. Each user recognizes its own descriptor and its position, knowing when to send the genuine and empty encryptions. XOR DC net sending is then conducted. By means of d_i , the participants compute and broadcast a variable length encryption for each round.

3 Preliminaries

Our new approach to untraceable communication is based on privacy-preserving aggregation. A key observation is that *sender untraceability* is implicitly provided by privacy-preserving aggregation, as explained further in Sect. 4. In this chapter we will provide a brief background on this topic.

3.1 Privacy-Preserving Aggregation

Secure multiparty computation (SMC) is a class of cryptographic protocols that enable two or more participants, not trusting each other, to jointly compute a function over their inputs while keeping those inputs private. Privacy-preserving aggregation is a subfield of SMC that enables a group of participants to compute an aggregation (sum or product) without having to disclose their inputs to any party. Many privacy-preserving aggregation schemes assume homomorphic

cryptography. The inputs are encrypted, and by aggregating the encryptions only a plaintext sum (or product) is produced. Depending on protocol designs, aggregation is performed by a single (untrusted) central entity or locally by each participant, in which all encryptions are broadcasted and received by the participants themselves.

Furthermore, we can distinguish between interactive and non-interactive privacy protocols. *Non-interactive* protocols are characterized by unidirectional communication or broadcasting, and there are no interactions in which one transmission triggers off another transmission [2, 13, 19, 25]. *Interactive* protocols are in contrast characterized by multiple rounds of communication [3, 14, 16, 21, 22]. Consequently, the former approaches are more efficient and have lower transmission and computational overhead than the latter.

The non-interactive privacy-preserving aggregation scheme of Shi et al. [25] (cf. Appendix A) is very efficient, having only one round of communication and one modular exponentiation per user. For this reason, we suggest a modification (cf. Appendix B) as a building block for the proposed scheme.

The security property achieved by privacy-preserving aggregation is often referred to as *aggregator obliviousness*. The following informal definition is adapted from [19]:

Definition 1 (Aggregator obliviousness). *This notion requires that the aggregator \mathcal{A} cannot learn any more than the aggregated value from the encrypted inputs of the (honest) users. If there are dishonest users (i.e., users sharing their private information with the aggregator), the notion only requires that the aggregator gets no extra information about the values of the honest users beyond their aggregated value. It is assumed that each user encrypts only one value per time period.*

We observe that if the aggregation scheme is secure, it is not possible to disclose individual plaintext inputs, which means that untraceability is inherently achieved.

3.2 Performance Goals

Transmission and computational overhead should be as low as possible and the protocol design should not be complex, which infer that interaction should be avoided. The proposed protocol has an unsurpassed efficiency, with a transmission overhead of only three group elements and a computational overhead of only three modular exponentiations per user.

3.3 Security Goals

The security properties we wish the protocol to satisfy is foremost sender anonymity (untraceability), although receiver anonymity (untraceability) is also provided due to broadcasting. Sender untraceability means that it must be infeasible to link (or trace) any plaintext message to its originator (the sender)

with more than $1/N$ probability, assuming no colluding participants. Likewise, receiver untraceability means that a certain message cannot be linked to a particular receiver with more than $1/N$ probability.

3.4 Adversarial Model

The adversarial model is *honest-but-curious*, a.k.a. semi-honest, meaning that the participants will not deviate from the protocol specification in how they compute and exchange messages. They may attempt to learn all possible information from the legitimately computed messages. An adversary can consist of up to $N - 2$ colluding participants or an external party. We assume the attacker is computationally polynomial-time limited and can monitor all network traffic.

4 Non-interactive Anonymous Communication Using Privacy-Preserving Aggregation

To achieve untraceable communication, we assume a predefined group of N individuals. For any given round, each participant can anonymously send a message by means of three encryptions that are broadcasted and received by everyone else in the group. There is no single aggregator and no decryption key, although this could be possible by a minor modification of the scheme. No particular transmissions order, synchronization, or coordination is necessary. Given that it is not computationally feasible to recover any plaintext in other ways than by aggregating the complete set of the pertaining encryptions, a participant may contribute with “empty” encryptions in case there is nothing particular to send.

We will next present some key observations that are the basis for the anonymous communication scheme:

1. Sender untraceability is implicitly provided by privacy-preserving aggregation as it is not possible to disclose individual plaintext inputs due to the aggregation obliviousness security property (Definition 1).
2. A privacy-preserving product (PPP) aggregation protocol enables a group of participants, each holding a secret prime, to compute a product \hat{p} in an untraceable manner, since no prime factor can be traced to a specific participant due to the aggregation obliviousness security property (Definition 1).
3. The (prime) factors of \hat{p} have no particular order.

In summary, application of a PPP protocol enables a group of participants to send and distribute primes in an untraceable manner, so that the originator of any given prime cannot be determined when the product \hat{p} is factorized, assuming that the PPP protocol is secure.

For sending arbitrary messages and not only primes, we consider *Goldbach’s conjecture*, which is one of the oldest and best-known unsolved problems in number theory and all of mathematics.¹ Goldbach’s conjecture states that every

¹ https://en.wikipedia.org/wiki/Goldbach's_conjecture.

even natural number greater than 2 is the sum of two prime numbers, and has been shown to hold for all integers less than 4×10^{18} . In agreement with Goldbach's conjecture, the idea is to represent a plaintext m_i as a sum $m_i = p_i + q_i$ of two primes (p_i, q_i) . The sum of two odd primes is always even. Taking arbitrary (odd and even) values of m_i into account, simply divide by two, i.e., $m_i = \frac{p_i + q_i}{2}$.

Using a secure PPP protocol, each participant P_i computes the encryptions $c_i = E(p_i)$ and $c'_i = E(q_i)$. After respectively aggregating the two sets of encryptions, $(\{c_i\}_{i=1}^N, \{c'_i\}_{i=1}^N)$, the primes $P = \{p_1, \dots, p_N\}$, $Q = \{q_1, \dots, q_N\}$ are restored by factorization.

Since there is no particular order to the primes in (P, Q) and the correct associations between the corresponding prime pairs $(p_i \in P, q_i \in Q)$ that constitute a given plaintext m_i are broken and unknown, a measure is necessary to correctly identify and restore the pertaining prime pair associations. For this purpose, we use a similar approach by privately aggregating plaintext hashes $z = \prod_{j=1}^N f(m_j)$. The plaintexts are restored by a simple exhaustive search of at most $N!$ trials, finding the pertaining prime pairs.

Our scheme can handle a limited number of users and data sizes, where basically the applied PPP is the main limiting factor. However, it can fit well as a *building block* and in this regard be used for larger groups and bigger data sizes if some few adjustments are carried out, cf. Sect. 4.3.

4.1 Hardness Assumption

The security of our protocol relies on the security of the underlying PPP aggregation scheme. The suggested PPP aggregation scheme (cf. Appendix B) relies on the difficulty of the Decisional Diffie-Hellman problem.

4.2 The Novel Non-interactive Anonymous Communication Scheme

The proposed anonymous communication scheme consists of the following steps:

Setup. Each user P_i is assigned a group $\mathcal{U} = \{P_1, \dots, P_N\}$. A large public prime p is selected that will be used as a modulus. For each user $P_i \in \mathcal{U}$, a key center randomly generates a long-term secret encryption key s_i , $1 < s_i < p - 1$, in which the sum is zero:

$$0 = \sum_{i=1}^N s_i \bmod p - 1 \quad (1)$$

Encryption. P_i selects two primes (p_i, q_i) , so that $m_i = \frac{p_i + q_i}{2}$ and $(p_i, q_i) < b$, where $b = \frac{p}{N}$ is an upper bound. P_i encrypts $(p_i, q_i, f(m_i))$:

$$c_i = E_{s_i}(p_i, 1), \quad c'_i = E_{s_i}(q_i, 2), \quad d_i = E_{s_i}(f(m_i), 3)$$

where $E_{s_i}(x, j) = xf(t, j)^{s_i} \bmod p$, f is a secure hash function, t is a timestamp or unique counter, j is a constant, and $f(t, j)^{s_i}$ is a secret encryption factor. The encryptions (c_i, c'_i, d_i) are broadcasted to the other members in \mathcal{U} .

Aggregation and Restoration. After receiving all encryptions, each participant combines the respective encryptions ($\{c_i\}_{i=1}^N, \{c'_i\}_{i=1}^N, \{d_i\}_{i=1}^N$), yielding the three products (x, y, z) , where

$$x = \prod_{i=1}^N c_i = \prod_{i=1}^N p_i \bmod p, \quad y = \prod_{i=1}^N c'_i = \prod_{i=1}^N q_i \bmod p, \quad z = \prod_{i=1}^N d_i = \prod_{i=1}^N f(m_i) \bmod p$$

Notice that all encryption factors $f(t, j)^{s_i}$, $1 \leq i \leq N, j \in \{1, 2, 3\}$, are cancelled out in agreement Eq. 1, resulting in the intended products as shown above. By factorizing (x, y) , the pertaining primes $P = \{p_1, \dots, p_N\}$ and $Q = \{q_1, \dots, q_N\}$ are restored.

Lastly, by means of the cryptographic checksum z all plaintexts are restored by finding the prime pairs, $(p_i \in P, q_i \in Q)$, constituting $m_i = \frac{p_i + q_i}{2}$. Let π denote a permutation of a vector \vec{v} of monotonic increasing elements $\vec{v} = (1, 2, \dots, N-1, N)$, of which there are $N!$ permutations. For each permutation π , check if

$$\prod_{i=1}^N f\left(\frac{p_i + q_{\pi(i)}}{2}\right) \bmod p \stackrel{?}{=} z$$

is true, then all N plaintexts $\{m_i = \frac{p_i + q_{\pi(i)}}{2}\}_{i=1}^N$ are recovered.

When Less than N Plaintexts. Although it is not possible to recover any plaintexts unless the complete set of N encryptions for a given round is present, a participant may contribute with an “empty” encryption in case he or she has nothing particular to send. An empty encryption is only the encryption factor $E_{s_i}(1, j) = f(t, j)^{s_i}$. Since the encryptions are in effect probabilistic due to the randomly chosen secret exponent s_i , it cannot be determined whether a given encryption is empty or not.

4.3 Scalability

The scalability of the above protocol depends heavily on the selected PPP scheme. The suggested PPP scheme was considered due to its simple non-interactive design and having low transmission and computational overhead. The size of the modulus p imposes a bound on the group size N and data size m due to the bound $p_i, q_i < \frac{p}{N}$, $1 \leq i \leq N$.

The untraceability scheme can be extended to larger data sizes by segmenting the plaintext into blocks of a fixed size b , in which the untraceability scheme is invoked for each block. The blocks must be equipped with a small header indicating a unique message identifier and block number. To restore the plaintexts after decryption, the blocks pertaining to the same message are joined in the right order.

4.4 Performance

A comparison of some prominent anonymous communications schemes is shown in Table 1, with regard to number of rounds, transmission complexity, and com-

putational complexity. Transmission complexity is the total number of transmitted data elements (encryptions, hashes, and signatures), while computational complexity refers to the total number of encryptions, decryptions, and signature computations and verifications. A significant improvement of the proposed scheme is that it has linear transmission and computational complexity, while this is quadric for existing work. As noted, we suggest a non-interactive privacy-preserving aggregation subprotocol to keep the overhead at a minimum.

Table 1. Performance comparison

Scheme	Number of rounds	Trans. complexity	Comp. complexity
Brickell and Shmatikov [7]	$2N + 5$	$2N^2 + 3N$	$5N^2 + 3N$
Corrigan-Gibbs and Ford* [11]	5	$2N^2 + 19N$	$5N^2 + N$
Basic DC nets** [9, 27]	N	N^2	N^2
Proposed scheme	1	$3N$	$3N$

4.5 Security Analysis

In this section we show that the proposed scheme ensures sender untraceability under the assumption that the underlying PPP is secure. Note that receiver untraceability is implicitly provided due to broadcasting. We also show that the suggested PPP is secure given that Decision Diffie-Hellman problem is hard to solve.

Theorem 1. *The proposed scheme provides sender untraceability provided that the applied PPP protocol ensures aggregator obliviousness.*

Proof. Only an informal proof is necessary. If the applied PPP protocol provides the *aggregator obliviousness* security property (Definition 1), no other information than the product can be deduced. Therefore, it is infeasible to link a prime factor to a particular encryption. Thus, sender untraceability is ensured. \square

Lemma 1. *The PPP protocol ensures aggregator obliviousness assuming that the Decision Diffie-Hellman problem is hard.*

Proof. The aggregator obliviousness security property is based on the assumption that the encryption factors, here denoted $\vec{h} = (h_1, \dots, h_N) \in \mathbb{Z}_p$, $N \geq 2$, are indistinguishable from random integers $\vec{h}' = (h'_1, \dots, h'_N)$ in the same domain \mathbb{Z}_p , cf. the Decision Diffie-Hellman problem. Let g be a generator to \mathbb{Z}_p .

Setup. Assume a challenger and an adversary. The challenger picks a random generator g for the group \mathbb{Z}_p and the random vector $(s_1, \dots, s_N) \in \mathbb{Z}_{p-1}$, such that $\sum_{i=1}^N s_i = 0$. Let $h = g^x$ for some random undisclosed integer $x \in \mathbb{Z}_{p-1}$. The challenger sends $(g, h, g^{s_1}, \dots, g^{s_N})$ to the adversary.

Challenge. The challenger flips a random coin b . If $b = 0$ then the challenger sends $\vec{h} = (h^{s_1}, \dots, h^{s_N})$ to the adversary. If $b = 1$ then the challenger picks N random elements $\vec{h}' = (h'_1, \dots, h'_N)$ in \mathbb{Z}_p , such that $\prod_{i=1}^N h'_i = \prod_{i=1}^N h^{s_i}$. The challenger sends \vec{h}' to the adversary.

Guess. The adversary has now the distribution $(g, g^x, g^{s_i}, t_i, 1 \leq i \leq N)$, where either $t_i = h^{s_i} = g^{x \cdot s_i}$ or $t_i = h'_i$. The adversary guesses either $b = 0$ or $b = 1$. The adversary wins the game if he successfully guesses b . If the adversary has a non-negligible advantage in guessing the correct b , he would be able to solve the DDH problem with non-negligible advantage as well. The proposed scheme is therefore secure assuming this is a computationally infeasible problem. \square

5 Conclusion

Anonymous routing networks and dining cryptographers networks (DC nets) are wellknown approaches to untraceable and anonymous communication, but these methods are in general very complex and suffer from high performance overhead. In this paper, we have proposed an original approach to untraceable communication that avoids some of the significant shortcomings of existing methods. Using non-interactive privacy-preserving aggregation as an underlying building block we achieve attractive features, such as unsurpassed low computational and transmission overhead of only 3 encryptions per participant in only a single round. No particular sending order, synchronization, or user interactions are necessary.

Acknowledgements. Parts of this research have been supported by basic institute funding at Norsk Regnesentral, RCN grant number 342640, and the NORCICS project, RCN grant number 310105.

A Shi et al. Privacy-Preserving Aggregation

A non-interactive privacy-preserving sum aggregation was proposed Shi et al. [25]. It meets the *aggregator oblivious* security property under the DDH hardness assumption, and has therefore smaller ciphertexts than in [19]. Similar to the Joye and Libert scheme, it was originally proposed for the smart meter setting, and transmissions are limited to each user broadcasting a single encryption for each round, resulting in a low computational load and bandwidth. The Shi et al. scheme comprises the following steps:

Setup. A key center (KC) establishes a large public prime p . For each user $P_i \in U$, KC randomly generates an encryption key $s_i \in \mathbb{Z}_{p-1}$ in agreement with $0 = -\sum_{1 \leq i \leq N} s_i \pmod{p-1}$.

Encryption. $P_i \in U$ samples a timeseries consumption value m_i at time interval t , and computes the ciphertext:

$$c_i = g^{m_i} f(t)^{s_i} \pmod{p}$$

where f is a secure hash function.

Aggregation and Decryption. After having received all N ciphertexts, they are multiplied according to

$$\hat{M} = \prod_{1 \leq i \leq N} c_i = \prod_{1 \leq i \leq N} g^{m_i} f(t)^{s_i} = \prod_{1 \leq i \leq N} g^{m_i} \pmod{p}$$

cancelling out the encryption factors $f(t)^{s_i}$, $1 \leq i \leq N$. The aggregated plaintext is then found by computing the discrete logarithm of \hat{M} w.r.t. to the base g .

B A Privacy-Preserving Product Protocol

The non-interactive privacy-preserving sum aggregation was proposed Shi et al. [25], shown in Appendix A, can conveniently be simplified to a privacy-preserving product protocol simply by neglecting the final step of resolving the aggregated sum by computing the discrete logarithm. Therefore, this simplification meets the *aggregator oblivious* security property under the DDH hardness assumption.

Setup. Each user P_i is assigned a group U of exactly N members. For each user $P_i \in U$, the key center randomly generates a secret encryption key $s_i \in \mathbb{Z}_{p-1}$ in agreement with $0 = \sum_{1 \leq i \leq N} s_i \pmod{p-1}$.

Encryption. $P_i \in U$ encrypts the prime p_i :

$$c_i = E_{s_i}(p_i, t) = p_i f(t)^{s_i} \pmod{p}$$

where h is a secure hash function and t is a timestamp. The ciphertext is transmitted to the AC.

Aggregation and Decryption. Each user aggregates the received ciphertexts according to

$$\hat{p} = \prod_{1 \leq j \leq N} c_j = \prod_{1 \leq j \leq N} p_j f(t)^{s_j} = \prod_{1 \leq j \leq N} p_j \pmod{p} \quad (2)$$

The multiplication is hence cancelling out the encryption factors $f(t)^{s_j}$, $1 \leq j \leq N$, yielding the product \hat{p} . As can be seen, the procedure is identical to that of [25], with the exception of omitting the final discrete logarithm computation.

References



1. Bauer, J., Staudemeyer, R.: From dining cryptographers to dining things: unobservable communication in the IoT, pp. 1–7 (2017)
2. Benhamouda, F., Joye, M., Libert, B.: A new framework for privacy-preserving aggregation of time-series data. *ACM Trans. Inf. Syst. Secur.* **18**(3), 1–21 (2016)
3. Borges, F., Demirel, D., Böck, L., Buchmann, J., Mühlhäuser, M.: A privacy-enhancing protocol that provides in-network data aggregation and verifiable smart meter billing. In: 2014 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6 (2014)

4. Borges, F., Mühlhäuser, M.: EPPP4SMS: Efficient privacy-preserving protocol for smart metering systems and its simulation using real-world data. *IEEE Trans. Smart Grid* **5**(6), 2701–2708 (2014)
5. Bos, J.: Practical privacy. J.N.E. Bos [Leiden] (1992)
6. Bos, J., den Boer, B.: Detection of disrupters in the DC protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) *EUROCRYPT 1989*. LNCS, vol. 434, pp. 320–327. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_33
7. Brickell, J., Shmatikov, V.: Efficient anonymity-preserving data collection. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006*, pp. 76–85. Association for Computing Machinery, New York (2006)
8. Busom, N., Petrljic, R., Sebe, F., Sorge, C., Valls, M.: Efficient smart metering based on homomorphic encryption. *Comput. Commun.* **82**, 95–101 (2016)
9. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptol.* **1**, 65–75 (1988)
10. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **24**(2), 84–90 (1981)
11. Corrigan-Gibbs, H., Ford, B.: Dissent: accountable anonymous group messaging. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security Security, CCS 2010*, pp. 340–350. Association for Computing Machinery, New York (2010)
12. Corrigan-Gibbs, H., Wolinsky, D.I., Ford, B.: Proactively accountable anonymous messaging in verdict. In: *22nd USENIX Security Symposium (USENIX Security 2013)*, pp. 147–162. USENIX Association, Washington (2013)
13. Emura, K., Kimura, H., Ohigashi, T., Suzuki, T.: Privacy-preserving aggregation of time-series data with public verifiability from simple assumptions and its implementations. *Comput. J.* **62**, 614–630 (2019)
14. Erkin, Z., Tsudik, G.: Private computation of spatial and temporal power consumption with smart meters. In: Bao, F., Samarati, P., Zhou, J. (eds.) *ACNS 2012*. LNCS, vol. 7341, pp. 561–577. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31284-7_33
15. Franck, C., van de Graaf, J.: Dining cryptographers are practical. *arXiv Cryptography and Security* (2014)
16. Garcia, F.D., Jacobs, B.: Privacy-friendly energy-metering via homomorphic encryption. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) *STM 2010*. LNCS, vol. 6710, pp. 226–238. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22444-7_15
17. Goldschlag, D., Reed, M., Syverson, P.: Onion routing. *Commun. ACM* **42**(2), 39–41 (1999)
18. Golle, P., Juels, A.: Dining cryptographers revisited. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 456–473. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_27
19. Joye, M., Libert, B.: A scalable scheme for privacy-preserving aggregation of time-series data. In: Sadeghi, A.-R. (ed.) *FC 2013*. LNCS, vol. 7859, pp. 111–125. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_10
20. Krasnova, A., Neikes, M., Schwabe, P.: Footprint scheduling for dining-cryptographer networks. In: Grossklags, J., Preneel, B. (eds.) *FC 2016*. LNCS, vol. 9603, pp. 385–402. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_23

21. Leontiadis, I., Elkhiyaoui, K., Molva, R.: Private and dynamic time-series data aggregation with trust relaxation. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 305–320. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12280-9_20
22. Li, F., Luo, B., Liu, P.: Secure information aggregation for smart grids using homomorphic encryption. In: 2010 First IEEE International Conference on Smart Grid Communications, pp. 327–332 (2010)
23. Nosouhi, M., Yu, S., Sood, K., Grobler, M.: HSDC-net: secure anonymous messaging in online social networks, pp. 350–357 (2019)
24. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.* **1**(1), 66–92 (1998)
25. Shi, E., Hubert Chan, T.-H., Rieffel, E.G., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: NDSS, vol. 2 (2011)
26. Shirazi, F., Simeonovski, M., Asghar, M.R., Backes, M., Díaz, C.: A survey on routing in anonymous communication protocols. *ACM Comput. Surv.* **51**(3), 51:1–51:39 (2018)
27. Waidner, M.: Unconditional sender and recipient untraceability in spite of active attacks. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 302–319. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_32
28. Waidner, M., Pfitzmann, B.: The dining cryptographers in the disco: unconditional sender and recipient untraceability with computationally secure serviceability. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, p. 690. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_69
29. Yang, Z., Zhong, S., Wright, R.N.: Anonymity-preserving data collection. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD 2005, pp. 334–343. Association for Computing Machinery, New York (2005)
30. Zhao, X., Li, L., Xue, G., Ahn, G.-J.: Efficient anonymous message submission. *IEEE Trans. Dependable Secure Comput.* **15**(2), 217–230 (2018)



PointPuff: An Ed25519 Optimization Implementation

Mengqing Yang^{1,2,3} , Chunxiao Ye^{1,2} , Yuanmu Liu³, Yan Jin^{1,2},
and Chunming Ye^{1,2}

¹ College of Computer Science, Chongqing University, Chongqing, China
{yecx, jiny}@cqu.edu.cn

² Key Laboratory of CPS-DSC, MoE, Chongqing University, Chongqing, China

³ Blockchain Application Department, Jingdong Technology, Beijing, China
yangmengqing@pku.edu.cn, liuyuanmu@jd.com

Abstract. Data transmission and interaction in a network can not be separated from the digital signatures. In recent years, Ed25519 algorithm has attracted extensive attention for its “High-speed and High-security” features. However, as shown by some test data, the performance of Ed25519, especially in terms of signature verification, remains unsatisfactory. Therefore, we improved the algorithm of Ed25519 batch verification in all three layers of elliptic curve arithmetic. We put forward a new point structure called PointPuff to accelerate the point-checking and point add processes, improve the traditional elliptic curve multi-scalar multiplication operation, and design a new finite-field large integer multiplication operation. In our test, the optimized batch verification performance was 50.04% higher than the existing algorithm, which was consistent with the theoretical analysis and within the error range.

Keywords: Digital signature · Edwards Curve · Public-key cryptography · Finite-field

1 Introduction

Digital signatures are used widely in networks. Among all digital signature algorithms, the most widely adopted algorithms are RSA, DSA, ECDSA, and EdDSA. The RSA algorithm is the most widespread signing algorithm in the world. By the end of 2021, RSA continues to be preferred in digital signature algorithms, with 50.47% of the sites using it [20]. Unfortunately, signing with an RSA is prohibitively expensive [14]. The ECDSA signatures are much shorter than the RSA keys. Specifically, to achieve the same security, the difference is 256-bit versus 3072-bit at the size [10]. However, RSA and EdDSA provide the best security [13]. EdDSA solves the same discrete log problem as DSA/ECDSA but uses a different family of elliptic curves known as the twisted Edwards Curve. As a result, an increasing number of protocols and software packages have begun to use EdDSA, especially Ed25519 [11].

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 117–130, 2024.

https://doi.org/10.1007/978-3-031-56326-3_9

EdDSA was extended in 2015 [4] and set as the standard using RFC8032 [12]. In February 2017, Ed25519 was standardized for use in DNSSEC through RFC8080 [22]. We choose Ed25519 as our research object because of its “High-speed and High-security” features and more importantly because it can be designed to accomplish aggregation signing and batch verification [3]. Ed25519 is regarded as the most successful digital signature algorithm that balances efficiency and security well and is four times faster than ECDSA. The elementary optimization of batch verification was discussed in 2020 [6] but was not sufficient. In the testing of a modern CPU, the signing speed of Ed25519 has a significant advantage over that of ECDSA. However, compared to ECDSA, the speed of Ed25519 signature verification is not obvious [24]. The real advantage of Ed25519 is that it can perform batch signature verification or signature aggregation and ensure security in the same time.

Multi-scalar multiplication, curve point arithmetic, and finite-field arithmetic are three layers of elliptic curve arithmetic. Layer 1 encapsulates Layer 2, and Layer 2 encapsulates Layer 3. We performed optimizations in all three layers in our study.

- The Bos-coster, Straus, and Pippenger algorithms are three traditional types of multi-scalar multiplication, and we chose the latter two for further optimization. Depending on the different calculation situations, we accelerate the precomputation process in the Straus algorithm and the “bucket initialization” process in the Pippenger algorithm.
- For point arithmetic, we chose a suitable point addition arithmetic for each situation. Furthermore, we created a new high-performance point addition arithmetic called “PointPuff” defined in group \mathbb{P}^3 for the most commonly used situation.
- For field arithmetic, we propose a new finite-field multiplication arithmetic in a new representation called *Radix-2*^{25.6} representation.
- Finally, we fit our test data to the hyperbolic curves generated by the theoretical analysis and performed some error analyses. It turns out that they all fit well and are within the error margin. We compared our algorithm with two open-source implementations (rust: dalek-25519¹, go: tendermint²), and it turned out that our work was 50.04% faster than the existing algorithms.

2 Preliminaries and Related Works

We define some parameters first:

- \mathcal{A} refers to a point addition in Curve25519,
- \mathcal{D} refers to a point double in Curve25519,
- \mathbf{A} refers to an addition in \mathbb{F}_p ,
- \mathbf{M} refers to a multiplication in \mathbb{F}_p ,
- \mathbf{S} refers to a square in \mathbb{F}_p ,
- \mathbf{d} refers to a multiplication with parameter d in \mathbb{F}_p .

¹ <https://github.com/dalek-cryptography/ed25519-dalek>.

² <https://github.com/tendermint/tendermint/blob/main/crypto/ed25519/ed25519.go>.

2.1 Ed25519 Batch Verification

Ed25519 Signature Generation. Two inputs: private key sk , message m . The small letter refers to scalar and the capital letter refers to point on Curve25519, and the capital letter with underline refers to the y -axis of the point in affine coordinates. G is the base point in Curve25519.

- Generate the private key and public key.

$$\begin{aligned} e &= SHA512(sk) = (e_0, e_1, \dots, e_{511}), \\ s &= (0, 0, 0, e_3, \dots, e_{253}, 1, 0), \\ A &= s \times G, pk = \underline{A}. \end{aligned} \quad (1)$$

- Sign the message with the actual private key s .

$$\begin{aligned} r &= SHA512(e_{256}, e_{257}, \dots, e_{511} || m), \quad R = r \times G, \\ k &= SHA512(\underline{R} || pk || m) \pmod{p}, \quad q = r + k \cdot s \pmod{p}, \\ Sig &= (\underline{R} || q). \end{aligned} \quad (2)$$

Ed25519 Signature Single Verification. Three inputs: public key pk , signature Sig and message m .

$$Verify \quad q \times G == R + Hash(\underline{R} || pk || m) \times A. \quad (3)$$

Ed25519 Signature Batch Verification. Batch verification involves inputting n sets of triplets (n public keys pk_i , signatures (\underline{R}_i, q) , messages m_i) and performing verification for one time. To do this, we need to generate n independent 128-bit random numbers and multiply each (3) by them.

- Decode point A_i, R_i according to pk_i, \underline{R}_i .
- Generate n 128-bit random numbers z_i .
- Calculate scalars: $h_i = Hash(\underline{R}_i || pk_i || m_i)$.
- Verify:

$$\textcircled{D} == \left(- \sum_{i=1}^n z_i \cdot q_i \pmod{l}\right) \times G + \left(\sum_{i=1}^n z_i \times R_i\right) + \sum_{i=1}^n (z_i \cdot h_i \pmod{l}) \times A_i. \quad (4)$$

2.2 Multi-scalar Elliptic Curve Multiplication

Bos-Coster. Bos-coster [19] is a multi-scalar elliptic curve multiplication optimization algorithm recommended in the original Ed25519 essay [3]. However, it can be proven that it only provides little optimization. The Bos-coster in Ed25519 decreases the scalar size from 256-bit to $(256 - \log_2 n)$ -bit by sorting.

Straus. The Straus algorithm [21] is an old optimal scheme that was published in 1964. It combines the non-adjacent form (NAF) of integer and double-and-add method to save cycles in scalar multiplication. The Straus applied in $\sum_{i=1}^n s_i P_i$ can be separated into three parts.

- Generate precomputation table for all points P_i .
- Calculate the integer expansion by NAF of window 5 for each scalar.
- Add parameters by column, and then double.

Pippenger. The Pippenger algorithm has been published and modified in [16–18]. It also contains three parts and is similar to that of Straus. The only two differences between Pippenger and Straus are the integer expansion and column addition. The Pippenger uses signed 2^w -radix expansion rather than NAF. It uses 2^{w-1} buckets to help with column additions without precomputation.

To compare the original single verification (using the Straus algorithm with the non-adjacent form, but verifying signatures one by one) and the three multi-scalar algorithms, we list four functions below in Table 1, each of which represents the total time needed to verify n signatures.

Table 1. Computational overhead.

Algorithm	Time cost function
Single	$n2^{w-2}\mathcal{A} + (n\mathcal{A} + nw\mathcal{D})\frac{256}{w}$
Bos-coster	$n2^{w-2}\mathcal{A} + (n\mathcal{A} + nw\mathcal{D})\frac{256 - \log_2 n}{w}$
Straus	$n2^{w-2}\mathcal{A} + (n\mathcal{A} + w\mathcal{D})\frac{256}{w}$
Pippenger	$2^w \frac{256}{w} \mathcal{A} + (n\mathcal{A} + w\mathcal{D})\frac{256}{w}$

Compared with single verification, the Bos-coster contributes only 2.4% improvement when $n = 64$, whereas Straus and Pippenger can realize at least two times improvement. Therefore, the Bos-coster is not a good choice. For the Straus, it is easy to prove that the computational cost achieves a minimum value when $w = 5$. For the Pippenger, the minimum value of the computational cost depends on both w and n . Specifically, when n is small, the minimum can be reached when $w = 6$. As n increases, w gradually increases to 7, 8, 9 and so etc.

2.3 Point Additions

Dr. Bernstein introduced several types of twisted Edwards curve calculations in 2008 [2]. We only focus on point addition in this paper.

- The affine coordinate is a classic twisted Edwards curve point expression form with two axes.

$$(x, y) \in \mathbb{E} : -x^2 + y^2 = 1 + dx^2y^2 \quad (5)$$

The point addition of affine coordinates is shown in (6). With two divisions in the finite-field, it provides brief knowledge of twisted Edwards curve addition.

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1 y_2 + x_2 y_1}{1 + dx_1 x_2 y_1 y_2}, \frac{y_1 y_2 + x_1 x_2}{1 - dx_1 x_2 y_1 y_2} \right) \quad (6)$$

- The extended Edwards coordinate [9] is considered the fastest form for point addition in \mathbb{E} with an $8\mathbf{M} + 1\mathbf{d}$ cost. Even better, a similar form provides an $8\mathbf{M}$ cost method but cannot work when the addition points are equal.

$$(x, y) \rightarrow (X : Y : Z : T), \\ x = X/Z, y = Y/Z, T = XY/Z, Z \neq 0$$

Point addition can be displayed by the following equation:

$$(X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2) = (X_3 : Y_3 : T_3 : Z_3). \quad (7)$$

Extended Edwards Point Addition 1:

$$\begin{aligned} X_3 &= (X_1 Y_2 + Y_1 X_2)(Z_1 Z_2 - dT_1 T_2), \\ Y_3 &= (Y_1 Y_2 + X_1 X_2)(Z_1 Z_2 + dT_1 T_2), \\ T_3 &= (Y_1 Y_2 + X_1 X_2)(X_1 Y_2 + Y_1 X_2), \\ Z_3 &= (Z_1 Z_2 - dT_1 T_2)(Z_1 Z_2 + dT_1 T_2). \end{aligned} \quad (8)$$

Extended Edwards Point Addition 2 (two points cannot be equal):

$$\begin{aligned} X_3 &= (X_1 Y_2 - Y_1 X_2)(T_1 Z_2 + Z_1 T_2), \\ Y_3 &= (Y_1 Y_2 - X_1 X_2)(T_1 Z_2 - Z_1 T_2), \\ T_3 &= (T_1 Z_2 + Z_1 T_2)(T_1 Z_2 - Z_1 T_2), \\ Z_3 &= (Y_1 Y_2 - X_1 X_2)(X_1 Y_2 - Y_1 X_2). \end{aligned} \quad (9)$$

2.4 Finite-Field Multiplication

The arithmetic in the finite-field \mathbb{F}_p ($p = 2^{255} - 19$) is another important part of Ed25519 signature calculations. The multiplication of two 256-bit integers in \mathbb{F}_p accounts for most calculations. [3] raised the *Radix-2⁵¹* representation and [5] raised the *Radix-2^{25.5}* representation, both of which are good solutions. However, even the optimized algorithm [7] based on [5] contained more than 100 64-bit multiplications.

2.5 Other Related Works

In recent years, the optimization of Ed25519 has been discussed in some specific areas. Reference [23] and [15] focused on the performance of hardware architecture implementations of Ed25519. Reference [8] focused on the mobile wireless network security of Ed25519. Reference [1] focused on optimizations in the finite-field calculation using the Fast Fourier Transform. As you can see, batch verification and point arithmetic technologies have not been discussed. Our finite-field optimization focuses on integer calculation in the matrix. Therefore, our work is unique.

3 Our Optimizations

3.1 PointPuff

We defined the new point structure PointPuff in \mathbb{P}^3 .

$$\begin{aligned}(x, y) &\rightarrow (X, Y, U, V), \\ X &= x, Y = y, U = xy, V = 2d \cdot U\end{aligned}$$

When we calculate $P_3 = P_1 + P_2$, with P_1 and P_2 in the extended Edwards point structure and P_3 in the PointPuff structure. The arithmetic function is as follows:

$$\begin{aligned}(X_1:Y_1:T_1:Z_1) + (X_2:Y_2:U:V) &= (X_3:Y_3:T_3:Z_3). \\ A &= (Y_1 - X_1)(Y_2 - X_2), \quad B = (Y_1 + X_1)(Y_2 + X_2), \\ C &= T_1V_2, \quad D = Z_1 + Z_1, \\ X_3 &= (B - A)(D - C), \quad Y_3 = (B + A)(D + C), \\ Z_3 &= (B + A)(B - A), \quad T_3 = (D + C)(D - C)\end{aligned}\tag{10}$$

This method only costs 7M and can be applied even when P_1 equals P_2 . We initialize all the points in the PointPuff structure and most of the intermediate points in the extended Edwards point structure. Thus, it can save most of the point addition process from 9M to 7M.

3.2 Restructuring of Signature

The process of Subsect. 3.2 is displayed in Fig. 1.

The inputs for verification usually include three parameters: signature, public key, and message. The signature structure traditionally includes (R, q) . The classic input contains all the information needed to recover the two points (the public key point “A” and the “R” point) and verify the signature.

However, a considerable amount of time is required to recover points using only the y parameters in the affine coordinate. The decoding point process costs 9.0% of the total time in the classic algorithm, and up to 35.2% in the pipeline process in our test.

We modify the structure of the signature in a particular way, which not only accelerates the point-check and point-recovery processes, but also accelerates the point addition in multi-scalar elliptic curve multiplications. We maintain the structure of the public key and message and add some bytes to the end of the signature. Specifically, the bytes we add are three parameters X, U, V of point A (public key) and point R in their PointPuff structure.

With the information of public key and \underline{R} , which are the axis y of point A and point R, we can easily get the PointPuff structure of those two points.

Moreover, when it comes to point-checking, the classic method usually recovers the affine structure of the point. That is, to calculate

$$x = \sqrt{\frac{y^2 - 1}{1 + dy^2}}. \quad (11)$$

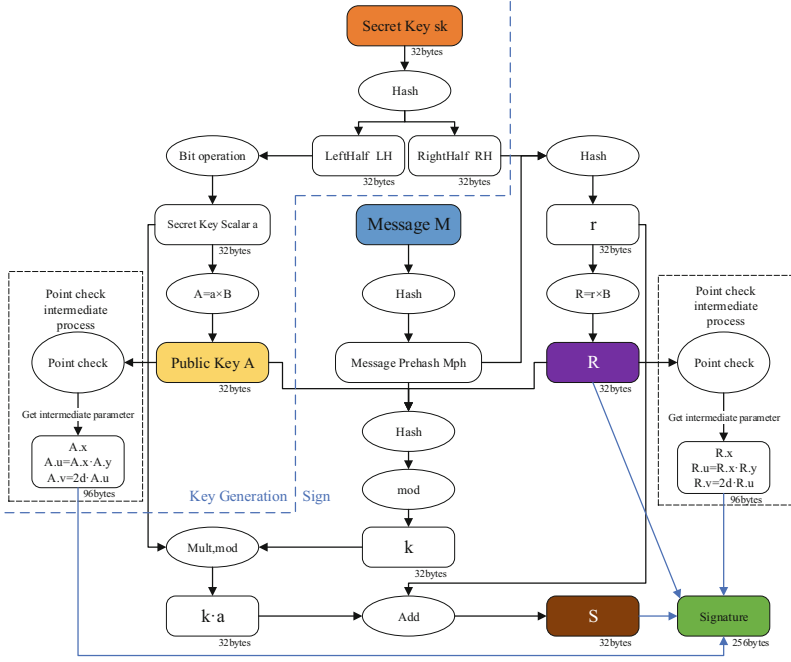


Fig. 1. Process of restructured signature generation.

Then, we check whether (x, y) fits (5). The process in (11) costs $18M$ and $15S$. However, with our method, the same function only costs $2M$. Knowing that, we can easily obtain four parameters (X, Y, U, V) with $X = x$, $Y = y$, $U = xy$ and $V = 2dx y$, we can check this point using the following equations:

$$\begin{aligned} a &= (Y + X)(Y - X), \\ b &= UV, \\ 2a &\equiv b + 2 \pmod{p}. \end{aligned} \quad (12)$$

The process of decoding the point and point-checking costs only approximately 0.7% of the total time.

3.3 Pruning and Renovating in Straus and Pippenger

Optimization in Straus. We find that in most Ed25519 implementations, the Extended Edwards Addition is applied in the precomputation process in the Straus algorithm. We use (10) to replace the classic method (8) when calculating $2P = P + P$, which reduces the finite-field multiplication from 9 to 7. Moreover, although (9) cannot be used when $P_1 = P_2$, it is an efficient algorithm if we can ensure $P_1 \neq P_2$. Fortunately, in the precomputation process in the Straus algorithm, when we calculate the precomputation table $\{P, 3P, 5P, \dots, 15P\}$, it is clear that the cofactors of equation $(2i - 1) \cdot P + 2P = (2i + 1) \cdot P$ are not equal. Therefore, we apply (9) to this process, which reduces the finite-field multiplications from 9 to 8.

Optimization in Pippenger. The Pippenger algorithm focuses on the construction and calculation of the intermediate parameter Bucket. Nevertheless, all the current implementations directly add the initialized Bucket with points to the best of our knowledge, which seems correct but not efficient. When a Bucket is empty, it is better to copy the point parameters into the Bucket rather than performing an addition. Typically, Buckets are set in extended Edwards form. When the first round of point addition is performed, it turns out to be a calculation of $Bucket \leftarrow (0, 1, 1, 0) + Point$, which costs $4M$ whatever the structure of the point is. However, the operation of a point copy costs only some array copy operations. And (10) can be used in nearly all addition processes in this section.

3.4 Improvement in Finite-Field Calculations

We propose a new representation called *Radix-2^{25.6}* representation in \mathbb{F}_p , which reduces the number of 64-bit multiplications from over 100 to 75.

Radix-2^{25.6} Representation. The number 25.6 is the average of $\{26, 26, 26, 26, 24\}$, which indicates the structure of our arithmetic.

We represent an integer x module $2^{255} - 19$ as:

$$\begin{aligned} x &= x_0 + 2^{26}x_1 + 2^{52}x_2 + 2^{78}x_3 + 2^{104}x_4 + 2^{128}x_5 + 2^{154}x_6 \\ &\quad + 2^{180}x_7 + 2^{206}x_8 + 2^{232}x_9 \end{aligned}$$

$$= \vec{x} \cdot \vec{f};$$

$$\vec{x} = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9)^T, \quad x_i \in [0, 2^{26} - 1];$$

$$\vec{f} = (1, 2^{26}, 2^{52}, 2^{78}, 2^{104}, 2^{128}, 2^{154}, 2^{180}, 2^{206}, 2^{232})^T.$$

So the multiplication of x and y can be represented as the matrix below:

$$\begin{bmatrix}
 x_0 & 152x_9 & 152x_8 & 152x_7 & 152x_6 & 38x_5 & 152x_4 & 152x_3 & 152x_2 & 152x_1 \\
 x_1 & x_0 & 152x_9 & 152x_8 & 152x_7 & 38x_6 & 38x_5 & 152x_4 & 152x_3 & 152x_2 \\
 x_2 & x_1 & x_0 & 152x_9 & 152x_8 & 38x_7 & 38x_6 & 38x_5 & 152x_4 & 152x_3 \\
 x_3 & x_2 & x_1 & x_0 & 152x_9 & 38x_8 & 38x_7 & 38x_6 & 38x_5 & 152x_4 \\
 x_4 & x_3 & x_2 & x_1 & x_0 & 38x_9 & 38x_8 & 38x_7 & 38x_6 & 38x_5 \\
 x_5 & 4x_4 & 4x_3 & 4x_2 & 4x_1 & x_0 & 152x_9 & 152x_8 & 152x_7 & 152x_6 \\
 x_6 & x_5 & 4x_4 & 4x_3 & 4x_2 & x_1 & x_0 & 152x_9 & 152x_8 & 152x_7 \\
 x_7 & x_6 & x_5 & 4x_4 & 4x_3 & x_2 & x_1 & x_0 & 152x_9 & 152x_8 \\
 x_8 & x_7 & x_6 & x_5 & 4x_4 & x_3 & x_2 & x_1 & x_0 & 152x_9 \\
 x_9 & x_8 & x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0
 \end{bmatrix} \cdot \vec{y}.$$

Define function F :

$$F(\hat{x}, \hat{y}) = \begin{bmatrix}
 x_i \\
 x_{i+1} & x_i \\
 x_{i+2} & x_{i+1} & x_i \\
 x_{i+3} & x_{i+2} & x_{i+1} & x_i \\
 x_{i+4} & x_{i+3} & x_{i+2} & x_{i+1} & x_i \\
 & 4x_{i+4} & 4x_{i+3} & 4x_{i+2} & 4x_{i+1} \\
 & & 4x_{i+4} & 4x_{i+3} & 4x_{i+2} \\
 & & & 4x_{i+4} & 4x_{i+3} \\
 & & & & 4x_{i+4}
 \end{bmatrix} \cdot \hat{y},$$

$$\hat{x} = (x_i, x_{i+1}, x_{i+2}, x_{i+3}, x_{i+4}). \tag{13}$$

Thus, optimized finite-field multiplication can be executed using Algorithm 1. All 64-bit multiplications in our algorithm are included in the three F functions, each of which contains 25 64-bit multiplications.

4 Results an Analysis

4.1 Data Fitting

We tested our code on *Intel(R) Xeon(R) Platinum 8338C 8-core @ 2.60 GHz* single thread. We tested five curves, including one Straus algorithm and four Pippenger algorithms with window width 6, 7, 8, 9. In the test, we generated the signature data to be verified randomly in advance, performed signature verification repeatedly, and obtained the average value. To reduce error, we repeated 100000 samples for each test point. For example, when the number of batches tested was 128, 12.8 million signatures were formed, and 128 signatures were used as a group to test the signature verification performance when $n = 128$. Finally, all the results were averaged. This experimental method is time-consuming. To save time, accurately fit each curve, and calculate the curve intersection point, we first took a larger step to roughly test the location of the curve intersection point and the approximate trend of the curve. Then, in the second test, intensive sampling was conducted at the intersection to determine the exact location of

Algorithm 1: Optimized Finite-field Multiplication

Calculate $x \cdot y = z$, x and y are 256-bit integers in $\mathbb{F}_p(p = 2^{255} - 19)$.

Initialize parameters:

1. Let $\vec{x} = \langle \hat{x}_1, \hat{x}_2 \rangle, \vec{y} = \langle \hat{y}_1, \hat{y}_2 \rangle$;
2. $\hat{r} = F(\hat{x}_1, \hat{y}_1)$; $\hat{s} = F(\hat{x}_2, \hat{y}_2)$;
3. $r_0 = r_0 - 152 \cdot s_5$; $r_1 = r_1 - 152 \cdot s_6$; $r_2 = r_2 - 152 \cdot s_7$; $r_3 = r_3 - 152 \cdot s_8$;
4. $r_5 = r_5 - s_0$; $r_6 = r_6 - s_1$; $r_7 = r_7 - s_2$; $r_8 = r_8 - s_3$;
5. $\hat{x}_3 = \hat{x}_1 + \hat{x}_2$; $\hat{y}_3 = \hat{x}_1 + \hat{y}_2$;
6. $\hat{t} = F(\hat{x}_3, \hat{y}_3)$;
7. $g = r_8 + (t_3 - r_3)$;
8. $cache_1 = g \bmod 2^{26} - 1$; $g \gg= 26$; $g = g + t_4 - r_4 - s_4$;
9. $cache_2 = g \bmod 2^{24} - 1$; $g \gg= 24$; $g = r_0 + (g + t_5 - r_5) \cdot 38$;
10. $z_0 = g \bmod 2^{26} - 1$; $g \gg= 26$; $g = g + r_1 + (t_6 - r_6) \cdot 38$;
11. $z_1 = g \bmod 2^{26} - 1$; $g \gg= 26$; $g = g + r_2 + (t_7 - r_7) \cdot 38$;
12. $z_2 = g \bmod 2^{26} - 1$; $g \gg= 26$; $g = g + r_3 + (t_8 - r_8) \cdot 38$;
13. $z_3 = g \bmod 2^{26} - 1$; $g \gg= 26$; $g = g + r_4 + s_4 \cdot 38$;
14. $z_4 = g \bmod 2^{24} - 1$; $g \gg= 24$; $g = g + r_5 + (t_0 - r_0)$;
15. $z_5 = g \bmod 2^{26} - 1$; $g \gg= 26$; $g = g + r_6 + (t_1 - r_1)$;
16. $z_6 = g \bmod 2^{26} - 1$; $g \gg= 26$; $g = g + r_7 + (t_2 - r_2)$;
17. $z_7 = g \bmod 2^{26} - 1$; $g \gg= 26$; $g = g + cache_1$;
18. $z_8 = g \bmod 2^{26} - 1$; $g \gg= 26$;
19. $z_9 = g + cache_2$;

Return \vec{z} .

the intersection. Intensive sampling was conducted when the curve slope was large, and sparse sampling was conducted when the curve slope was small.

The test data are listed in Table 2. As shown in the table, when the batch size is 5000, the performance of batch signature verification (35123 TPS) is 5.56 times that of the single signature verification (6318 TPS).

Table 2. TPS data of our work.

Batch Size	TPS	Batch Size	TPS
1	6318	128	22582
2	9539	256	25557
4	12851	512	28140
8	16030	1024	30141
16	18120	2000	33239
32	19349	2600	33994
64	20016	5000	35123

Theoretically, TPS-n plots fit hyperbolic curves. Then we get 5 curves in our test. We aggregate all the five curves and test data to establish Fig. 2 to conveniently identify the intersection points and the tendency.

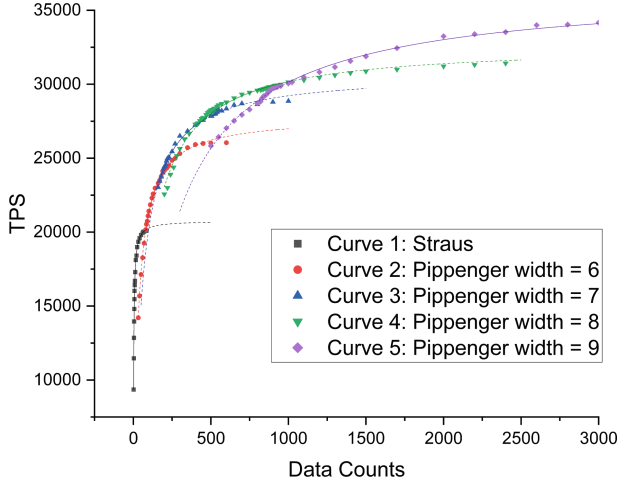


Fig. 2. Test data aggregation.

We use hyperbolic curve function (14) to fit our test data. And we get the results of values and errors of each parameters in Table 3.

$$TPS = \frac{P_1 n}{P_2 + n} \quad (14)$$

Table 3. Curve Parameter List

Curve name		P_1	P_2
Straus	Value	20765.32118	2.42110
	Error	21.68154	0.01442
Pippenger w = 6	Value	27913.52689	30.97788
	Error	38.47435	0.22242
Pippenger w = 7	Value	30895.49039	53.47907
	Error	47.06186	0.55382
Pippenger w = 8	Value	32701.01963	83.62811
	Error	65.16361	1.27968
Pippenger w = 9	Value	36499.00921	211.17442
	Error	67.36150	2.33933

To get the curve intersection point, we respectively set up simultaneous functions using parameters from Table 3, and subsequently obtained the curve intersection points N_1 , N_2 , N_3 , N_4 .

$$N_1 = 80.535, N_2 = 179.651, N_3 = 462.419, N_4 = 1014.557. \quad (15)$$

Therefore, we draw a conclusion:

when $n \in [2, 80]$, choose the Straus algorithm;
 when $n \in [81, 179]$, choose the Pippenger algorithm with window width 6;
 when $n \in [180, 462]$, choose the Pippenger algorithm with window width 7;
 when $n \in [463, 1014]$, choose the Pippenger algorithm with window width 8;
 when $n \in [1015, \infty)$, choose the Pippenger algorithm with window width 9.

4.2 Error Analysis

Because each curve is fitted, all the curve equations have errors. At this time, the calculation of the curve intersection also transfers the errors. To determine the error range of the curve intersection points, we set up two curves simultaneously and reorganized them to obtain the function of the intersection points. Subsequently, the partial differential of the function is calculated, and the value is assigned at the intersection to obtain the error value.

$$TPS = \frac{P_1 n}{P_2 + n}, TPS' = \frac{P_1' n}{P_2' + n}.$$

While $TPS = TPS'$,

$$N = \frac{P_1 P_2' - P_1' P_2}{P_1' - P_1} = f(P_1, P_2, P_1', P_2');$$

$$\Delta N = \left| \frac{\partial f}{\partial P_1} \right| \cdot \Delta P_1 + \left| \frac{\partial f}{\partial P_2} \right| \cdot \Delta P_2 + \left| \frac{\partial f}{\partial P_1'} \right| \cdot \Delta P_1' + \left| \frac{\partial f}{\partial P_2'} \right| \cdot \Delta P_2'; \quad (16)$$

$$\Delta N = \frac{P_1' \cdot |P_2' - P_2|}{(P_1' - P_1)^2} \cdot \Delta P_1 + \frac{P_1'}{|P_1' - P_1|} \cdot \Delta P_2$$

$$+ \frac{P_1 \cdot |P_2' - P_2|}{(P_1' - P_1)^2} \cdot \Delta P_1' + \frac{P_1}{|P_1' - P_1|} \cdot \Delta P_2'. \quad (17)$$

The errors in the curve intersection point can be calculated by substituting the relative parameters into (17):

$$\Delta N_1 = 1.487, \quad \Delta N_2 = 13.821, \quad \Delta N_3 = 64.780, \quad \Delta N_4 = 72.948. \quad (18)$$

So we get the real curve intersection points:

$$N_1 = 80.535 \pm 1.487, \quad N_2 = 179.651 \pm 13.821,$$

$$N_3 = 462.419 \pm 64.780, \quad N_4 = 1014.557 \pm 72.948. \quad (19)$$

The errors of N_1, N_2, N_4 are acceptable, whereas the N_3 error is large. The reason for this is analyzed in the next section.

Overall, the test results prove our advantage over the other implementations (see Table 4). The advantage of our work grows as the batch size increases. When the batch size was 256, our work had a 50.04% advantage over the existing Ed25519 batch verification implementation.

Table 4. Improvement rate list.

Batch Size	Dalek25519	Tendermint	Our work
1	1.000	1.000	1.000
4	1.561	1.477	2.034
8	1.926	1.805	2.537
16	2.041	1.920	2.868
32	2.101	1.994	3.062
64	2.250	2.038	3.168
128	2.432	2.304	3.574
256	2.696	2.469	4.045

4.3 Cost Analysis

Table 1 lists the cost functions of each multi-scalar elliptic curve multiplication algorithm. For Straus, the addition cost \mathcal{A} in the time-cost function can be separated into two parts: part 1, $n2^{w-2}\mathcal{A}$; part 2, $256n/w\mathcal{A}$. The first part contains 1 point addition in (10) and 7 point additions in (9) in a $2^{w-2}\mathcal{A}$ process ($w = 5$). That is one-time 7M and seven-time 8M, in which all additions in the second part are (8).

As for the Pippenger, in addition to the factors mentioned above, the time cost also depends on pruning in the Bucket addition process. As mentioned in Sect. 3.3, when a Bucket is empty, we just copy the point parameters into Bucket, which requires minimal time. Thus, the Bucket addition process depends on the number of empty Buckets at the end of each cycle. When $w = 6, 8, 9$, the effect of empty Buckets can be ignored in their valid ranges. However, this cannot be ignored when $w = 7$. Therefore, the error of N_3 is nonnegligible.

5 Conclusion

In this study, we performed optimizations in three layers of the Ed25519 curve algorithm to accelerate the calculation process. With the upgrading of digital signatures in the network, the support of equipment and policies, and the further application of blockchain and the IoT, this work will be useful for further theoretical analysis and practical applications.

References

1. Ben-Sasson, E., Carmon, D., Kopparty, S., Levit, D.: Elliptic curve fast Fourier transform (ECFFT) part I: fast polynomial algorithms over all finite fields. arXiv preprint [arXiv:2107.08473](https://arxiv.org/abs/2107.08473) (2021)
2. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards curves. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 389–405. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68164-9_26

3. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. *J. Cryptogr. Eng.* **2**(2), 77–89 (2012)
4. Bernstein, D.J., Josefsson, S., Lange, T., Schwabe, P., Yang, B.Y.: EdDSA for more curves. *Cryptology ePrint Archive* (2015)
5. Bernstein, D.J., Schwabe, P.: NEON crypto. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 320–339. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_19
6. Chalkias, K., Garillot, F., Nikolaenko, V.: Taming the many EdDSAs. In: van der Merwe, T., Mitchell, C., Mehrnezhad, M. (eds.) SSR 2020. LNCS, vol. 12529, pp. 67–90. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64357-7_4
7. Chou, T.: Sandy2x: new curve25519 speed records. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 145–160. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31301-6_8
8. Das, M., Wang, Z.: Ed25519: a new secure compatible elliptic curve for mobile wireless network security. *Jordan. J. Comput. Inf. Technol.* **8**(1) (2022)
9. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Twisted Edwards curves revisited. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 326–343. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_20
10. Hoffman, P., Wijngaards, W.C.: Elliptic curve digital signature algorithm (DSA) for DNSSEC. Technical report, RFC6605 (2012)
11. Ianix: Things that use Ed25519 (2022). <https://ianix.com/pub/ed25519-deployment.html>
12. Josefsson, S., Liusvaara, I.: Edwards-curve digital signature algorithm (EdDSA). Technical report, RFC8032 (2017)
13. Kontsevov, E.: Comparing SSH keys - RSA, DSA, ECDSA, or EdDSA? (2022). <https://goteleport.com/blog/comparing-ssh-keys/>
14. Krasnov, V.: ECDSA: the missing piece of DNSSEC (2022). <https://www.cloudflare.com/dns/dnssec/ecdsa-and-dnssec/>
15. Mehrabi, M.A., Doche, C.: Low-cost, low-power FPGA implementation of Ed25519 and curve25519 point multiplication. *Information* **10**(9), 285 (2019)
16. Pippenger, N.: On the evaluation of powers and related problems. In: 17th Annual Symposium on Foundations of Computer Science (SFCS 1976), pp. 258–263. IEEE Computer Society (1976)
17. Pippenger, N.: The minimum number of edges in graphs with prescribed paths. *Math. Syst. Theory* **12**(1), 325–346 (1978)
18. Pippenger, N.: On the evaluation of powers and monomials. *SIAM J. Comput.* **9**(2), 230–250 (1980)
19. Rooij, P.: Efficient exponentiation using precomputation and vector addition chains. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 389–399. Springer, Heidelberg (1995). <https://doi.org/10.1007/BFb0053453>
20. Security, H.N.: EV certificate usage declining: is the internet becoming more secure? (2021). <https://www.helpnetsecurity.com/2021/12/13/newer-tls-protocols/>
21. Straus, E.G.: Addition chains of vectors (problem 5125). *Amer. Math. Monthly* **70**(806–808), 16 (1964)
22. Sury, O., Edmonds, R.: Edwards-curve digital security algorithm (EdDSA) for DNSSEC. Technical report, RFC8080 (2017)
23. Yu, B., Huang, H., Liu, Z., Zhao, S., Na, N.: High-performance hardware architecture design and implementation of Ed25519 algorithm. *J. Electron. Inf. Technol.* **43**(7), 1821–1827 (2021)
24. Yu, J.: Is there a case to prefer ed25519 over ECDSA p-256 for DNSSEC?



Detecting Web Tracking at the Network Layer

Maximilian Wittig^(✉) and Doğan Kesdoğan

University of Regensburg, Regensburg, Germany
{maximilian.wittig,dogan.kesdogan}@ur.de

Abstract. Third-party tracking allows companies to identify users and track their online activity across different websites or digital services. This paper presents a first experimental study to detect advertisements and tracker by inspecting fully encrypted network transactions at the TCP/IP network level associated with a website. The first results are encouraging and motivate to extend this first proof-of-concept study even further in the future. A classical application area in the future would be the use in areas where communication can only be accessed on encrypted TCP/IP level (keyword secure IoT environments) or the presented approach is used simply to enable a classical extension of the portfolio for tracker detection.

Keywords: Online tracking · Network Traffic Classification · Privacy

1 Introduction

Advertising is essential to a free market economy: it enables commerce by providing consumers with product and service information and encouraging competition. However, online marketing has emerged as a new independent business model, described in the literature as “surveillance capitalism” [32]. To protect themselves from advertising and online tracking, users rely on privacy-enhancing blocking tools, such as Adblock Plus or uBlock Origin. Many of them are based on manually created blacklists, which leads to scalability issues. Hence, the research community has applied machine learning to automate the creation of blacklists. To the best of our knowledge, these approaches solely leverage features at the application layer of the network stack [5, 13, 16, 24]. Although the models perform well with an accuracy up to 98.1%, they assume access to the application layer. Consequently, related work proposes client-side tools that cannot be easily extended to the network-level as the trend shifts towards encrypted data transmission.

This raises the question whether the purpose of a communication (here advertising and tracking) can be inferred despite encryption. Regarding this question, there are many traffic classification problems that gain information by applying pattern recognition to encrypted traffic traces. For instance, in [3, 12, 18] the encrypted traffic is categorized into several application types, ranging from

chat application to streaming service. Another prominent example is website fingerprinting, which aims to determine the website a user has accessed via an anonymized channel [14, 21, 26, 27]. These attacks have proven to be effective on HTTPS because traffic features, such as size, timing, and order of network packets, are unique to each website. To the best of our knowledge, we are the first to adapt traffic analysis to tracking detection. This use case entails additional challenges compared to the aforementioned traffic classification problems, since we intend to classify individual elements within a website in contrast to its cumulative traffic flow. Furthermore, modern websites contain various types of potentially harmful resources (images, JavaScript, etc.) which show diverse behavior patterns. Considering that the information about the resource type is not available in encrypted traffic, we face a binary classification problem involving heterogeneous target classes (tracking or non-tracking). Notably, our approach does not break encryption but exploits that privacy-invasive resources, or even communications with the serving host, follow an observable tracking protocol which is distinct from benign traffic traces. In fact, our model learns some meta-information to match observed patterns with known patterns. Traffic classification applied to the advertising and tracking problem potentially opens up network-wide detection and blocking of unwanted resources, which is becoming increasingly relevant as Internet of Things proliferate [20, 30]. As a starting point, we seek to answer whether tracking resources are detectable in an encrypted traffic flow, and if so, what features best describe them.

In summary, our contributions are as follows:

- For our empirical study, we create a dataset of the top 1K most popular websites according to the Majestic list. Our dataset covers the entire network-stack from L1 to L7 and maximizes the number of trackers by automatically accepting the consent management platform.
- We train a classifier based on fully encrypted network transactions to identify tracking resources with an accuracy and F_1 score of about 90% and 88%, respectively. Given that the tracking portion of most domains is highly homogeneous, features related to the whole conversation exhibit high discriminatory power.
- We show that traffic features are particularly useful for detecting third-party tracker, as well as unwanted resource types such as documents (HTML), images, and scripts.

The remainder of the paper is structured as follows: Sect. 2 provides background information on cross-site online tracking and corresponding blocking measures. Section 3 presents our experimental design and Sect. 4 showcases our dataset. We show our results on feature importance and classifier performance in Sect. 5. Section 6 concludes our study.

2 Background and Related Work

2.1 Cross-Site Online Tracking

The beauty of the Web lies in the hyperlinking of objects, which allows third-party elements to be easily included into first-party websites. However, the

analysis in [2] shows the picture of a hidden market whose actors collect personal data without the user’s influence by exploiting third-party content. Today’s online marketing is mainly driven by advertisers (demand for product placements), publishers (supply of advertising space) and advertising networks (intermediaries).

When a website is visited, the available advertising space is auctioned to the highest bidder by the ad network [31]. For the financial evaluation of an auction, bidders are provided with a detailed user profile containing information on the geography, demographics and preferences of the respective website visitor. This is achieved by using various tracking techniques that uniquely identify the user and expose information, such as the browsing history [7, 19, 23]. In collaboration with several publishers, third-party trackers scale across a variety of sites to create a comprehensive browsing profile. As more browser vendors discontinue support for third-party cookies, recent studies show a vast majority of websites are using first-party tracking techniques (e.g., first-party cookies or CNAME cloaking) and sharing them with other parties to circumvent blocking [4, 8, 9]. They conclude that tracking prevention should be extended to first parties as well.

2.2 Existing Blocking Techniques

Broadly speaking, several strategies exist to mitigate online tracking. One strategy might be to spoof web tracker by providing incorrect information. A prominent example is Brave’s randomization to some fingerprinting endpoints [6]. However, blocking unwanted content according to known signatures is much more common. In contrast to spoofing, blocking requires the detection of trackers. This is done either by manually created blacklists or through machine learning to automate the signature creation process.

Blacklists. Popular browser extensions like uBlock Origin rely on blacklists. These lists are manually curated based on user feedback and suffer from scalability and robustness issues. First, blacklists struggle to keep up with the ever expanding advertising and tracking ecosystem. For example, the adoption of anti-adblocker rules took around 90 days [15]. To make matters worse, there are several evasion strategies for advertisers to avoid blacklists, such as changing domains or using the CNAME cloaking technique [10, 29].

Machine Learning. To address both, scalability and robustness, researchers have applied machine learning (ML) for automated ad and tracker blocking. The first generation of ML approaches detects trackers based on a single dimension of the application layer. These approaches featurizing the content of either URLs, HTTP headers, or request and response payloads [5, 13, 24]. Since they mimic blacklists, they inherit their shortcomings (i.e. the presence of a specific keyword is susceptible to trivial evasion). Therefore, the second generation involves a number of graph-based approaches that capture the interactions among HTML

elements, JavaScript, and HTTP requests [16, 25]. They leverage rich cross-layer features and thus claim to be robust to evasion attempts. However, the aforementioned approaches require access to the application layer, which is usually encrypted. Thus, they cannot efficiently prevent tracking at the network-level, but only on a per-host basis. In contrast, we leverage traffic aspects of TCP/IP layer, which are hardly affected by encryption because the traffic shape is always available. Lastly, we argue that traffic flow statistics are robust to evasions. One could obfuscate such information by sending redundant traffic or delaying packets to manipulate the time series. However, this would have a negative impact on bandwidth, latency, and quality of service.

3 Experimental Design

This section provides an overview of our study design. First, we present our assumed attack scenario. Second, we report on our methodology for creating and preprocessing the dataset, as well as parameter selection and training of the classifiers and their evaluation. Last, we present our feature engineering process.

3.1 Scenario

Consider the following scenario (cf. Fig. 1). Alice wants to visit a website (here `example.org`). The initial response from the web server contains the web page, which links to new resources, leading to subsequent requests. These requests typically involve several hosts (third parties). The HTTP packets are then transmitted over Ethernet and are split into multiple TCP packets due to the maximum transmission unit (in our example from packet no. 20 to 23). In addition, all communication between Alice and the web servers is encrypted. However, for labeling purposes, we assume the network traffic in plain. To achieve this, we observe and experiment locally on Alice’s host. This gives us also the opportunity to study the upper bound of tracking defenses based on network features with perfect information. One information that might be useful is the relative arrival time of packets since the site was accessed, or knowledge about individual resources. In our model, tracking is defined as a host providing tracking resources to Alice. While some web services are used exclusively for tracking purposes, others fall into a gray area because portions of the communication provide benefits. Therefore, a fine distinction of tracking is necessary. Since current tracking techniques follow fixed protocols, these may show different traffic characteristics than regular web services and thus differ in their communication patterns, even if encrypted. Furthermore, the question arises how well suited these structures are to determine the semantics of the data flow, in this case tracking. Towards answering these questions, we present a machine learning approach for the automated identification of privacy-intrusive services. We classify resources into two classes: (i) tracker and (ii) non-tracker (other). Here, the challenge is to assess the intent of a service with limited knowledge.

In our attack scenario, we run an ex-post traffic analysis for each website. Meaning, we expect Alice to visit each web page sequentially, observe all traffic,

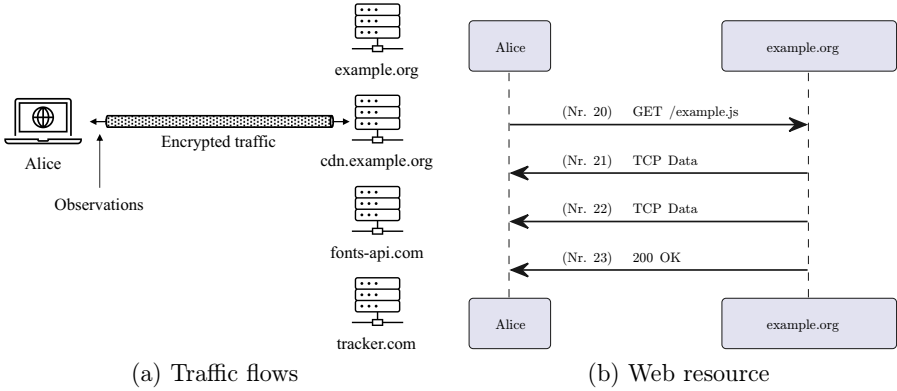


Fig. 1. Example scenario for the website call example.org.

and eventually estimate trackers independently of other pages. Our methodology prevents traffic features tailored to a host from being biased. For instance, extensive communication might result from a chatty host or because the host is present at many first parties. In a practical environment, a similar outcome is achieved by using a short time window. It is worth noting that isolating the network traffic of a website preserves the partyness of a resource, which is necessary for labeling the dataset. Furthermore, we discard signaling packets (e.g. ACK packets) since they provide no useful information for our classification task. A similar optimization is done in [21] with the usage of a minimum byte size filter.

3.2 Methodology

As depicted in Fig. 2, our study consists of three components: Web Crawler, Preprocessing and Evaluation. For our empirical study¹, we first build an appropriate dataset of network traces. For this purpose, we developed a tool based on Python-Selenium to automatically crawl websites and capture the entire network stack using Tcpdump. The crawler is containerized in Docker. This allows parallel browsing of websites while keeping the network traffic of each website isolated.

Web Crawler. The homepage of the 1000 most popular websites from the Majestic Million list is automatically visited and the web traffic to the destination port 80 and 443 is filtered. The Majestic 1K list ranks the top websites based on the number of backlinks and can be freely downloaded². We use Google Chrome in headless mode to disable telemetry data. To evade bot detection, we obfuscate the user agent and set the language to en-US. Furthermore, the homepage is

¹ The source code of our study is available at: <https://github.com/wim50594/network-traffic-tracker-observer>.

² <https://majestic.com/reports/majestic-million>.

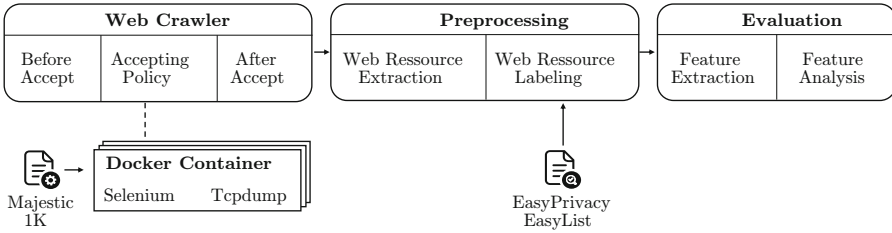


Fig. 2. Abstract overview of our experiment.

scrolled vertically to simulate organic user interaction and trigger additional requests. The environment variable `SSLKEYLOGFILE` is set to allow TLS data to be decrypted during preprocessing. The `sslkeylog` contains keying material of the TLS handshake. Each website is accessed up to three times for 20 s, with varying degree of cookie policy acceptance: before accept, accepting policy and after accept. Cookie consent is important because more and possibly different trackers may be loaded after acceptance. We only visit a website a third time if it offers a cookie policy and is discovered by us. To identify a cookie policy, we scan the Document Object Model for typical phrases to accept all cookies. We adopt the word list from preliminary work [17]. For all websites, we conduct a stateless crawl, meaning the cache is cleared after each visit.

Preprocessing. Since most requests are protected by HTTPS, the pcap files are decrypted using the `sslkeylog`. After that, we extract web resources from the packet flow via the `tshark` command line tool. During this process, an HTTP request and response, as well as their associated TCP segments, are reassembled into a resource (cf. Fig. 1b). Subsequently, each resource is labeled based on the blacklists `EasyList` (advertisement) and `EasyPrivacy` (tracking). Both lists³ were originally developed for the browser extension `Adblock` and are used in numerous studies [11, 13]. Although our main goal is to study tracker detection, we include `EasyList` in the ground truth because digital advertising is fundamentally linked to tracking and cannot be considered separately. In our case, the well-known advertising network `doubleclick.net` is only exposed by `EasyList`. To apply the filter rules to the web resources, we utilize Brave’s `adblock` engine⁴. Last, we determine the party level of a resource based on `eTLD+1` as well as the content type from the HTTP header.

Evaluation. First, we extract several network features with respect to IP address, size, direction, and timing. Next, we analyze the feature importance by plotting their distribution and measuring the classification performance of a decision tree and logistic regression. Both models were chosen because of their

³ <https://easylist.to/>.

⁴ <https://github.com/brave/adblock-rust>.

interpretability of the classification decision. The task of binary classification is to assign a web resource to one of the two categories (i) tracker or (ii) non-tracker. Both classifiers are trained in a supervised learning environment (ground truth is given by blacklists). For this, we use the Python library scikit-learn with default hyperparameters, except for a regularization term to prevent overfitting. These are configured by a grid search. The maximum depth of the decision tree is limited to $\lfloor \frac{\text{number of features}}{4} \rfloor$ with minimum samples at a leaf node set to 5, and L1 penalty term is set to $C = 10^{-4}$ for logistic regression. The performance is evaluated using 5-fold cross-validation and repeated 5 times unless otherwise stated. The split into training and test data is based on website visits, so communication to hosts remains compact. As the features are different in scale, they must be standardized before passed to the logistic regression.

3.3 Network Traffic Features

Based on observations during preliminary work regarding the prevalence and operation of trackers [1, 8, 23], we know that tracking services seldom deliver large responses (e.g., tracking pixel). At the same time, large amounts of data are supposed to flow to privacy-intrusive services. Hence, we expect the data flow to tracking parties to be characterized by small responses with low volume compared to large client requests. With these observations in mind, we develop statistical features for binary classification of web resources.

Table 1. Network traffic features for this study. The resource level refers to the associated packets of a resource. The conversation level comprises the entire packet flow to a host during a page visit.

A: Resource level (packets associated with a resource)	
packet size	Packet sizes of a resource
in/out packet size	Directional packet sizes of a resource (from client's perspective)
relative time	Relative arrival time of the packets since website call
delta resource time	Delta time to the previous resource of a host
B: Conversation level (separated by IP address)	
packet size	Packet sizes of a conversation
in/out packet size	Directional packet sizes of a conversation (from client's perspective)
relative time	Relative arrival time of the packets since website call
prevalence(ip)	No. of observed IP address on first-party websites

The network features explored in our experiment are summarized in Table 1. We use two different granularity levels of features, namely the conversation level and the resource level. The former describes the communication behavior to a host during a page visit (e.g., in Fig. 1 the traffic to the IP host of

fonts-api.com). At this level, we are only considering the packet flow without any resource information. However, since web services offer both tracking and non-tracking content (especially in the case of a first-party), we further explore the characteristics of individual resources. To build our features, we aggregate the network packets and compute the main descriptive statistics count, sum, min, max, mean, rsd, and span. The latter takes the difference between max and min. The relative standard deviation is standardized by std/mean (coefficient of variation). At both levels of granularity, we consider TCP packet size, TCP packet size of incoming and outgoing traffic, and the relative arrival time of packets since the site was visited. The split into in and out packets encodes directional features as shown to be important in previous work [26, 27]. Since third-party trackers require a high penetration rate to create a comprehensive browsing profile, we expect them to be widely spread on first-party websites. Therefore, we calculate the frequencies of IP addresses on first-party websites within the training data. Unknown IP addresses within the test data are encoded as -1 . Last, we measure the time intervals between resources (delta resource time).

Since we introduced several variants of features measuring similar properties, we expect that an automatic classifier will only need a subset of the proposed features. However, we examine all features to understand which specific features are most relevant for classifying tracking web services.

4 Dataset

We conducted our experiment on November 9, 2022, for which we obtained the most current state of the Majestic list and the two blacklists EasyList and EasyPrivacy. We ran our crawl at a German university. Accordingly, we were often redirected to a German version of the website due to IP geolocating.

Table 2. Characteristics of the Majestic-Million 1K dataset

Characteristic	
Websites successfully crawled	933
Cookie-acceptance rate	0.46
Count unique domains	first-party=827/third-party=2161
Average parties per domain	parties=25.52/tracking-parties=22.3
Count resources	first-party=83,993/third-party=182,001
Count tracking resources	is-tracker=105,586/not is-tracker=160,408
Count packets	in=3,258,443/out=320,468

The characteristics of the dataset are summarized in Table 2. In total, 933 websites were successfully visited, approximately 266 thousand resources were requested, and over 3.5 million packets were exchanged. Not all websites could be accessed because the Majestic list contains entries whose domain could not be

resolved, or the connection was refused. Most websites (70%) fall under the categories of Computers & Technology, Education, News or Business. The number of first-parties is lower than successfully crawled websites, because some have multiple domains from which they are redirected to the main site. For example, `business.site` becomes `www.google.com`, which is already ranked 1. Likewise, typical properties of web traffic are observed. As expected, we find a significant preponderance of third-parties per domain on average, with 22.3 hosts sending at least one tracking-related resource. Consequently, only one third of the resources originates from the first-party. Furthermore, an imbalance exists between incoming packets to the client and outgoing packets to the web server by a factor of 10. On 46% of the websites, we accepted all cookies. Unfortunately, the ground truth of consent banner is unknown, but in a similar study a detection rate of 63.2% for European websites was reported [17]. Since our experiment crawls the top websites worldwide, with some providing no cookie policy (e.g., `apple.com` or `wikipedia.org`), the result looks reasonable. In 10% of the cases, a cookie banner could not be found within the time window of 20 s because the web pages consisted of extensive HTML elements. The five most common consent texts are: “accept all cookies”, “accept all”, “accept”, “i accept”, and “alle akzeptieren”.

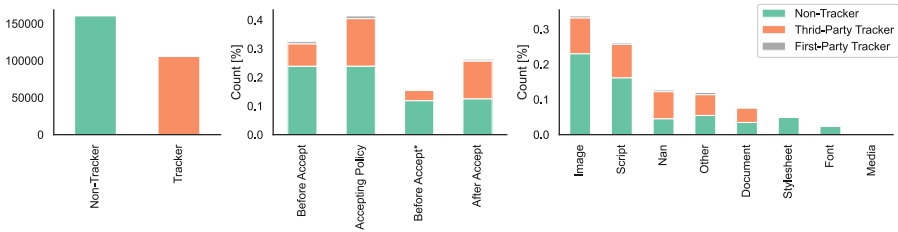


Fig. 3. Tracker distribution in Majestic 1K dataset.

Figure 3 presents a weak imbalance between our target classes with a distribution of about 60/40% harmless resources. However, only 6.7% of first-party resources are tracking-related, as most trackers are included by third parties (94.6%). When the cookie policy is accepted, we measure a remarkable increase of tracking resources. This indicates that the user’s consent is only partially taken into consideration, since trackers are already loaded even without consent⁵. The number of resources after accepting is lower, because we visit web pages a third time only when a cookie policy is detected. For a better comparison, Before Accept* denotes a subset of websites whose cookie banners we have identified. The comparison with the samples of after accept shows a significant jump in resources as well as third-party trackers, while first-party trackers slightly increase. Most requested elements are images and scripts (59.9%). If no content type is provided in the HTTP header, it is labeled as “none”. The resources marked as “other” include json files in most cases (65.3%).

⁵ The interested reader can find an in-depth analysis in [22].

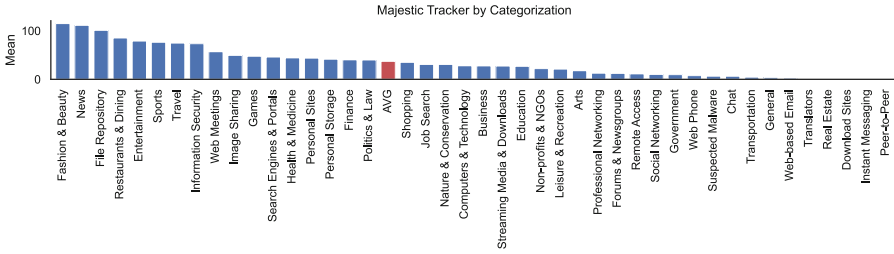


Fig. 4. Majestic 1K top tracked websites categorization by Cyren URL Lookup API⁹. Some were assigned to multiple categories and therefore counted repeatedly.

The amount of tracking varies considerably among different categories of websites. In Fig. 4 we show the average number of tracking resources across all three page visits for each web category. Websites at the lower end of the spectrum are mostly service providers that require authentication, as well as government and non-profit organizations. On average the web pages of the last 20 categories consist of 130.42 non-tracking resources vs. 171.93 overall. Meaning, they are simpler in design and require fewer resources. In contrast, monetization plays a major role for websites at the top of the spectrum, as they primarily offer editorial content with no external funding.

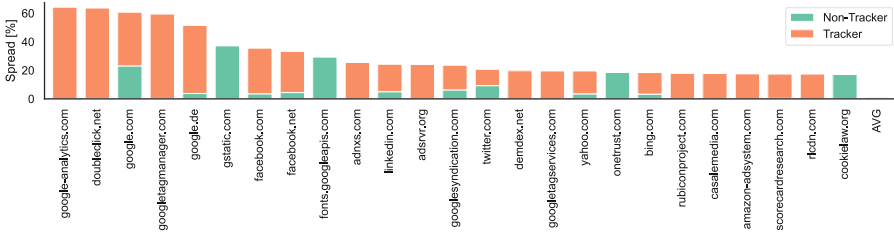


Fig. 5. Top 25 most widespread third parties.

The tracking ecosystem involves only a handful of third parties. Figure 5 shows the percentage of third parties on first-party sites, most of which are for tracking purposes. A regular user is exposed to at least one of the top 25 third parties with 84.89% chance. On average, third parties are only present on about 1% of the pages. The dominance of a few providers becomes even greater when you consider that many of the domains are owned by the same organization.

5 Analysis of Network Traffic

In this section, we analyze characteristic patterns in network flow to tracking or non-tracking services. To do this, we visualize the distribution of network

features and then evaluate them using statistical methods. Lastly, we evaluate the classification power and how its performance depend on the website context and resource type.

5.1 Feature Distribution

Since a host may exchange a mix of tracking and non-tracking resources, we measure the purity of the communication channels by computing its service entropy. For our two classes, the entropy varies between 0 and 1. The overall entropy is 0.028 on average and 0.023 for third parties, indicating that communication is mostly pure. This can be explained by the fact that blacklists often only block at a coarse domain level (in our experiment 51.5% of the rules). However, for first-party communication, the mean entropy slightly increases to 0.072. While this suggests that detection at the granularity of communication is promising, the associated resources share the same conversational context. Therefore, it is necessary to distinguish individual resources more precisely to minimize the impact on functionality, which is especially important for the first-party.

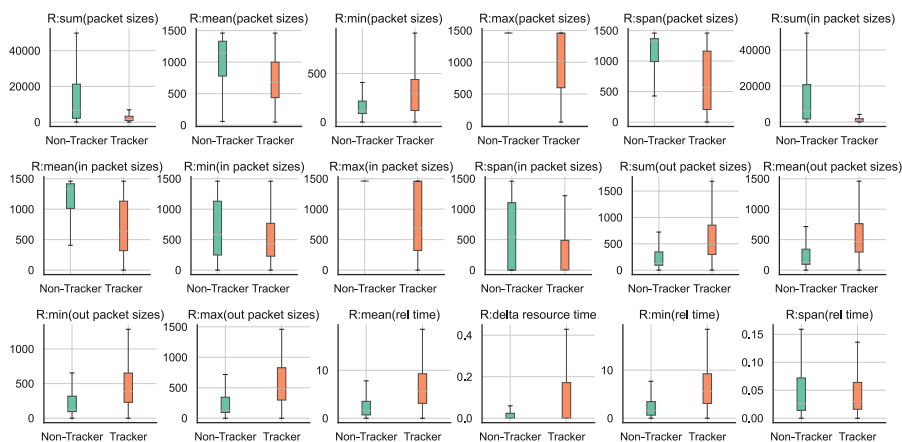


Fig. 6. Feature distribution of resources by size, direction, and time. As a rule of thumb, the less the boxes overlap, the more distinctive the feature.

This finding motivates us to look at the network properties of resources. For this purpose, we aggregate the packet attributes of resources by statistical location parameters (cf. Sect. 3.3). Figure 6 visualizes the discrimination power of the resource features. To achieve good classification results, the features should follow different distributions. For readability reasons, we only plot the relevant features without outliers. Overall, tracking resources tend to be smaller in size (sum) which also applies to individual packets (mean). A small span (max - min) represents equally sized packets. While the span of privacy-invasive resources is dispersed, non-trackers are more concentrated at the larger end. Harmless traffic

is typically characterized by a small request and large responses. In contrast, a tracking pixel that only triggers requests to set or synchronize cookies, causes both short outbound and inbound traffic. The opposite case is primarily a result of tracking scripts that have small requests and large responses. Overall, trackers are significantly smaller than non-trackers in terms of packet size, except for `R:min(packet sizes)`, which varies substantially. This observation could possibly be explained due to the information leakage (e.g., via URL parameter [23]). Incoming traffic makes up the majority of packets (91%), so the feature distribution looks similar to the previous one. The opposite is true for outgoing packets. These are shifted further to the top due to the loss of information caused by either multiple parameters in the URL or POST data. The packet arrival time reveals that trackers are loaded later and with greater pauses between individual requests, while the total loading time of a resource (`R:span(rel time)`) is similarly short for both categories. Notably, the discrimination power of outgoing packet size and relative time is limited by a large interquartile range.

5.2 Feature Importance

Next, we look at the feature importance to discuss what features most accurately describe trackers. To begin with, we show in Fig. 7 a simplified version of the decision tree with a depth of 3, which still achieves an accuracy of 85.3% and a F_1 score of 0.823. It is an excellent tool for generalizing typical characteristics of tracking traffic. The tree consists exclusively of conversation features, as they are best suited to classify third-party trackers. Most resources (71%) can be described by two paths. Accordingly, a conversation is classified as tracker if it generates less than 53.8 kB of incoming traffic, it starts after 0.8 s, and its IP address has been observed over 3 times at a first-party. In contrast, harmless network traffic is characterized by at least 234.1 kB of incoming packets and an IP address prevalence of at most 29. While both paths lead to pure leaves, with Gini coefficients of 0.213 and 0.061, respectively, there are also minorities that do not yield good classification results. For example, if the incoming packets range from 53.8 KB to 234.1 kB and the conversation starts after 1.3 s, 48.6% of them are misclassified as trackers. Compared to most trackers, this class contains mostly advertisements, large JavaScript files, and twice as many requests to the host, resulting in a large communication volume.

To further understand which network features are most indicative of tracking, we compute the Gini importance for the decision tree and report the weights for the logistic regression in Fig. 8. For reference, we compare both scores with the point biserial correlation coefficient, which measures the individual predictive ability of each attribute. To improve readability, only significant features are shown.

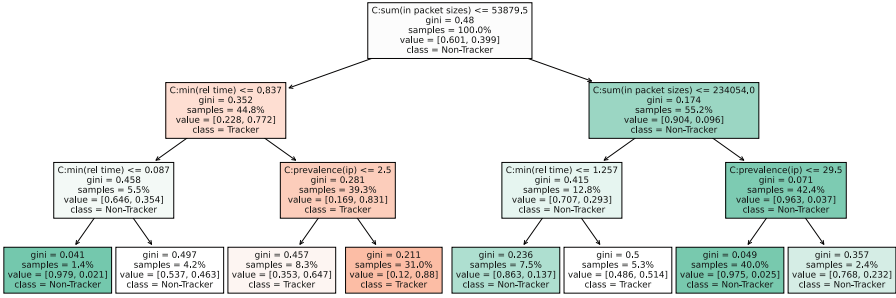


Fig. 7. Simplified decision tree with maximum depth of 3.

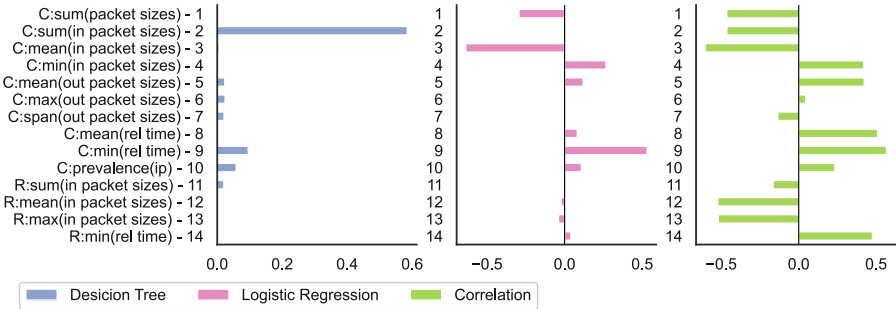


Fig. 8. Feature importance for different model.

Due to the regularization and the strong correlation between features, only a few features are selected. The most informative characteristics relate to the incoming packet size as well as the relative time. These results are in line with Fig. 6, as privacy-intrusive services generally do not provide useful web content ($r = -0.47$) and conversations with them tend to start after the main page has finished loading ($r = 0.57$), e.g., after an event has been triggered. In particular, the resource-specific features are less important, although the classification task involves the assessment of a resource. One explanation might be that web services provide very homogeneous content (either tracking or non-tracking) and therefore classification based on conversation is plausible. In addition, conversation features accumulate more information over the time horizon, leading to less variation compared to their resource counterpart. The importance of `C:prevalence(ip)` is consistent with our intuition that a higher frequency of a host on first-party websites is an indicator of tracking, since cross-site tracking requires widespread services ($r = 0.23$).

5.3 Classifier Performance

In this section, we evaluate the performance of our classifiers. To reduce variations, we test our classifier with 5-fold cross-validation and repeat this process

5 times, since results can vary depending on composition of training and test data. We are particularly interested in the distinctiveness of features at different levels of granularity, as perfect information may not be needed.

Table 3. Comparison of classification performance.

	All features		Conv. level		Res. level	
	Acc.	F1	Acc.	F1	Acc.	F1
Decision tree	0.899	0.875	0.882	0.860	0.853	0.824
Logistic regression	0.862	0.833	0.859	0.829	0.776	0.733

Overall Performance. The results in Table 3 show better overall performance for the decision tree than for the logistic regression. The decision tree can reproduce the blacklist labels 89.9% of the time. When looking at only a subset of features, conversational features perform better than resource-based features. As the communication flow contains more information, the conversational features exhibit more discriminative power. But the resource-specific features are still valuable, in detecting trackers for an impure communication with high service entropy. Therefore, combining both levels of granularity provides the best result.

First-Party vs. Third-Party. Even though the conversation features contribute the most to the classification, they are inadequate for detecting at first-party context. We observe different network characteristics for resources depending on the context of the party. For example, the average on `R:sum(out packet sizes)` is for first-party tracker 1.9 kB and for third-party tracker 0.9 kB, while `R:sum(in packet sizes)` is for non-tracker on average 27.5 and 21.7, respectively. Since first parties are only responsible for 5.6% of tracking, our classifiers are not able to generalize first-party trackers well. To compensate for the imbalance, we apply random undersampling to the majority classes. Each sample takes 30% of the dataset and is 5-fold cross-validated. This process is repeated 10 times, and we report the average in Table 4.

As the performance of first-party trackers is significantly lower than that of third-party trackers, correct detection in the first-party context remains a challenging task. While the decision tree based on resource features is best suited for first-party context, with a true positive rate of 78.2%, only 59.5% of the predicted positives are correct.

Table 4. Classification performance for first-party/third-party.

		First-Party		Third-Party	
		Precision	Recall	Precision	Recall
Decision Tree	All features	0.668	0.661	0.850	0.906
	Conv. level	0.688	0.657	0.811	0.926
	Res. level	0.595	0.782	0.802	0.882
Logistic Regression	All features	0.713	0.639	0.822	0.775
	Conv. level	0.715	0.638	0.823	0.775
	Res. level	0.464	0.745	0.768	0.752

Table 5. Classification performance for different resource types.

	Decision Tree			Logistic Regression		
	F1	Precision	Recall	F1	Precision	Recall
document	0.891	0.859	0.926	0.800	0.739	0.874
font	0.073	0.048	0.220	0.104	0.063	0.360
image	0.919	0.944	0.897	0.907	0.948	0.869
media	0.462	0.381	0.755	0.174	0.142	0.315
nan	0.876	0.842	0.914	0.835	0.761	0.925
other	0.838	0.811	0.868	0.777	0.731	0.830
script	0.847	0.881	0.816	0.524	0.857	0.378
stylesheet	0.302	0.221	0.525	0.147	0.103	0.272

Resource Types. To further uncover what our classifiers are able to detect, we present the classification results over various tracking object types in Table 5. Again, we apply undersampling due to heavy imbalance of types. Our approach performs equally well in distinguishing the different tracking types, except media (aka. video or audio), stylesheet, and font. However, they account for only 13.5% of the tracking and contribute little to the overall performance. The misclassifications are caused because the network flow of regular traffic is not sufficiently distinctive from tracking. The point biserial correlation coefficient is close to 0 for the three types for all network features.

6 Conclusion

In this work, we present preliminary experimental findings on identifying tracking resources based on TCP/IP features. Our classifier achieves a promising precision and recall of up to 85% and 90%, and is particularly suitable for third-party and image tracker detection. This shows that properties of tracking protocols are reflected on the network layer, which enables network-wide tracking protection without breaking encryption.

We conclude by discussing the limitations and outline future research directions. First, we assume a local attacker scenario with perfect information that offers a broad selection of features. However, our results suggest that analyzing the packet flow to a host is sufficient, as the performance improves marginally by including resource features. Besides the size of incoming packets, temporal features relative to the web page call provide useful information, but are usually only known to the client. Second, we only consider basic flow statistics (count, sum, mean, etc.) instead of conducting a time series analysis, though the latter may result in better performance [3, 27]. Third, our experiment is limited by the low level of interaction on the homepage and focuses solely on web traffic. In future work, we plan to extend our analysis to mobile and IoT traffic.

References

1. Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., Diaz, C.: The web never forgets: persistent tracking mechanisms in the wild. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 674–689 (2014). <https://doi.org/10.1145/2660267.2660347>
2. Agogo, D.: Invisible market for online personal data: an examination. *Electron. Mark.* **31**(4), 989–1010 (2020). <https://doi.org/10.1007/s12525-020-00437-0>
3. Akbari, I., et al.: A look behind the curtain: traffic classification in an increasingly encrypted web. In: Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 5, no. 1 (2021). <https://doi.org/10.1145/3447382>
4. Bekos, P., Papadopoulos, P., Markatos, E.P., Kourtellis, N.: The Hitchhiker’s guide to facebook web tracking with invisible pixels and click IDs. In: Proceedings of the ACM Web Conference 2023 (2023). <https://doi.org/10.1145/3543507.3583311>
5. Bhagavatula, S., Dunn, C., Kanich, C., Gupta, M., Ziebart, B.: Leveraging machine learning to improve unwanted resource filtering. In: Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, pp. 95–102 (2014). <https://doi.org/10.1145/2666652.2666662>
6. Brave: Fingerprint randomization (2020). <https://brave.com/privacy-updates/3-fingerprint-randomization/>
7. Bujlow, T., Carela-Español, V., Solé-Pareta, J., Barlet-Ros, P.: A survey on web tracking: mechanisms, implications, and defenses. *Proc. IEEE* **105**(8), 1476–1510 (2017). <https://doi.org/10.1109/JPROC.2016.2637878>
8. Chen, Q., Ilia, P., Polychronakis, M., Kapravelos, A.: Cookie swap party: abusing first-party cookies for web tracking. In: Proceedings of the Web Conference 2021, pp. 2117–2129 (2021). <https://doi.org/10.1145/3442381.3449837>
9. Demir, N., Theis, D., Urban, T., Pohlmann, N.: Towards understanding CNAME based first-party cookie tracking in the field. In: GI Sicherheit 2022 (2022)
10. Dimova, Y., Acar, G., Olejnik, L., Joosen, W., Van Goethem, T.: The CNAME of the game: large-scale analysis of DNS-based tracking evasion. *Proc. Priv. Enhanc. Technol.* **3**, 394–412 (2021). <https://doi.org/10.2478/popets-2021-0053>
11. Englehardt, S., Narayanan, A.: Online tracking: a 1-million-site measurement and analysis. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1388–1401 (2016). <https://doi.org/10.1145/2976749.2978313>

12. Fathi-Kazerooni, S., Kaymak, Y., Rojas-Cessa, R.: Tracking user application activity by using machine learning techniques on network traffic. In: 2019 International Conference on Artificial Intelligence in Information and Communication, pp. 405–410 (2019). <https://doi.org/10.1109/ICAIIIC.2019.8669040>
13. Gugelmann, D., Happe, M., Ager, B., Lenders, V.: An automated approach for complementing ad blockers' blacklists. In: Proceedings on Privacy Enhancing Technologies, pp. 282–298. No. 2 (2015). <https://doi.org/10.1515/popets-2015-0018>
14. Hayes, J., Danezis, G.: K-Fingerprinting: a robust scalable website fingerprinting technique. In: Proceedings of the 25th USENIX Conference on Security Symposium, pp. 1187–1203 (2016)
15. Iqbal, U., Shafiq, Z., Qian, Z.: The ad wars: retrospective measurement and analysis of anti-adblock filter lists. In: Proceedings of the 2017 Internet Measurement Conference, pp. 171–183 (2017). <https://doi.org/10.1145/3131365.3131387>
16. Iqbal, U., Snyder, P., Zhu, S., Livshits, B., Qian, Z., Shafiq, Z.: AdGraph: a graph-based approach to ad and tracker blocking. In: 2020 IEEE Symposium on Security and Privacy, pp. 763–776 (2020). <https://doi.org/10.1109/SP40000.2020.00005>
17. Jha, N., Trevisan, M., Vassio, L., Mellia, M.: The internet with privacy policies: measuring the web upon consent. *ACM Trans. Web* **16**(3) (2022). <https://doi.org/10.1145/3555352>
18. Ma, Q., Huang, W., Jin, Y., Mao, J.: Encrypted traffic classification based on traffic reconstruction. In: 2021 4th International Conference on Artificial Intelligence and Big Data, pp. 572–576 (2021). <https://doi.org/10.1109/ICAIBD51990.2021.9459072>
19. Mayer, J.R., Mitchell, J.C.: Third-party web tracking: policy and technology. In: 2012 IEEE Symposium on Security and Privacy, pp. 413–427 (2012). <https://doi.org/10.1109/SP.2012.47>
20. Mohajeri Moghaddam, H., et al.: Watching you watch: the tracking ecosystem of over-the-top TV streaming devices. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 131–147 (2019). <https://doi.org/10.1145/3319535.3354198>
21. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website fingerprinting in onion routing based anonymization networks. In: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, pp. 103–114 (2011). <https://doi.org/10.1145/2046556.2046570>
22. Papadogiannakis, E., Papadopoulos, P., Kourtellis, N., Markatos, E.P.: User tracking in the post-cookie era: how websites bypass GDPR consent to track users. In: Proceedings of the Web Conference 2021, pp. 2130–2141 (2021). <https://doi.org/10.1145/3442381.3450056>
23. Roesner, F., Kohno, T., Wetherall, D.: Detecting and defending against third-party tracking on the web. In: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pp. 155–168 (2012)
24. Shuba, A., Markopoulou, A., Shafiq, Z.: NoMoAds: effective and efficient cross-app mobile ad-blocking. *Proc. Priv. Enhanc. Technol.* (4) (2018)
25. Siby, S., Iqbal, U., Englehardt, S., Shafiq, Z., Troncoso, C.: WebGraph: capturing advertising and tracking information flows for robust blocking. In: 31st USENIX Security Symposium, pp. 2875–2892 (2022)
26. Siby, S., Juarez, M., Diaz, C., Vallina-Rodriguez, N., Troncoso, C.: Encrypted DNS – > privacy? A traffic analysis perspective. In: Network and Distributed System Security Symposium (2020)

27. Sirinam, P., Juarez, M., Imani, M., Wright, M.: Deep fingerprinting: undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (2018). <https://doi.org/10.1145/3243734.3243768>
28. Sjösten, A., Snyder, P., Pastor, A., Papadopoulos, P., Livshits, B.: Filter list generation for underserved regions. In: Proceedings of The Web Conference 2020, pp. 1682–1692 (2020). <https://doi.org/10.1145/3366423.3380239>
29. Snyder, P., Vastel, A., Livshits, B.: Who filters the filters: understanding the growth, usefulness and efficiency of crowdsourced ad blocking. SIGMETRICS (2020). <https://doi.org/10.1145/3392144>
30. Varmarken, J., Le, H., Shuba, A., Markopoulou, A., Shafiq, Z.: The TV is smart and full of trackers: measuring smart TV advertising and tracking. Proc. Priv. Enhanc. Technol. **2**, 129–154 (2020). <https://doi.org/10.2478/popets-2020-0021>
31. Yuan, S., Abidin, A.Z., Sloan, M., Wang, J.: Internet advertising: an interplay among advertisers, online publishers, ad exchanges and web users. arXiv: 1206.1754 (2012)
32. Zuboff, S.: The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power. PublicAffairs (2018)



What's Inside a Node? Malicious IPFS Nodes Under the Magnifying Glass

Christos Karapapas¹ , George C. Polyzos¹ ,
and Constantinos Patsakis^{2,3} 

¹ Athens University of Economics and Business, Athens, Greece
karapapas@aueb.gr

² University of Piraeus, Piraeus, Greece

³ Athena Research Center, Marousi, Greece

Abstract. InterPlanetary File System (IPFS) is one of the most promising decentralized off-chain storage mechanisms, particularly relevant for blockchains, aiming to store the content forever, thus it is crucial to understand its composition, deduce actor intent and investigate its operation and impact. Beyond the network functionality that IPFS offers, assessing the quality of nodes, i.e. analysing and categorising node software and data, is essential to mitigate possible risks and exploitation of IPFS. To this end, in this work we took three daily snapshots of IPFS nodes within a month and analysed each node (by IP address) individually, using threat intelligence feeds. The above enabled us to quantify the number of potentially malicious and/or abused nodes. The outcomes lead us to consider using a filter to isolate malicious nodes from the network, an approach we implemented as a prototype and used for assessment of effectiveness.

Keywords: InterPlanetary File System · Web3 Security

1 Introduction

Web 1.0 is known as the *read-only* web. For many years the World Wide Web had an informative and educative role presented through static content. Few people generated content that was read by many people. From the constant interaction as well as the expanding familiarity that users had with the web, the number of users who also wanted to create content grew. Thus, the need for a more participative web arose, giving birth to Web 2.0. The latter, despite its shortcomings, is massively adopted. Single points of failure due to security incidents and control from a single organisation along with privacy violations for, e.g. marketing purposes, by centralised data storage facilities have been two of the most thorny issues in Web 2.0 for years.

Lately, there has been a lot of discussion around Web3. One of the pillars of Web3 is the decentralisation of the web to allow users to regain control over their data and selectively share and monetise the information they create. An integral

part of attaining these goals are distributed ledger and blockchain technologies, and token-based economics [6, 12]. Web3 promises to offer decentralised services, meeting the needs of the Internet of Things (IoT) era and introducing a financial aspect of the web-user relationship through cryptocurrencies.

Web3 is considered to consist of different stacks, and these, in turn, of different protocols that cooperate with each other in order to provide services to the user. Some of them are data storage, domain name resolution, decentralised identities or, at a higher level, social media, gaming and marketplaces. All these different protocols, as well as the bridges between them, are still in their making; thus, their shortcomings may be exploited by attackers or utilised for malicious purposes. Indeed, ransomware and dark web marketplaces use cryptocurrencies to make siphon their payments, while blockchains and IPFS are used for the coordination of malware as C2 servers or to store malicious payloads.

In this work, we focus on distributed data storage and, more specifically, the InterPlanetary File System (IPFS), a cornerstone of the decentralised storage component of Web3. IPFS claims to have 2 million unique weekly users¹, and it has certainly caught the eye of the scientific community, as reflected by a total of more than 1160 papers found on Scopus with the search term “IPFS” in title and/or abstract. As IPFS is a collection of sub-protocols, it can be exploited by malicious users in a variety of ways. Immutability and decentralisation create a very dangerous mix that can be abused in various ways [5]. Karapapas et al. [8] illustrated how cybercriminals could exploit IPFS to set up an anonymous malware C2 facility alongside smart contracts. Patsakis et al. [13] showed that it could be abused to provide a robust malware C2 server infrastructure. Moreover, it is known to have been utilised by the Storm botnet [14] while new evidence has come to light linking IPFS to phishing².

To this end, we aim to unravel the structural elements of the IPFS network, and the nodes, focusing on suspicious activity. Initially, we crawl the IPFS network to enumerate it and make the first contact with the nodes. Following that, we collect intelligence from different sources regarding the aforementioned nodes. Moreover, we collect the exchanged data by nodes and analyse them to have a deeper understanding of the consistency of the network. Finally, we try to determine the extent of possible abuse of IPFS for copyright infringement.

The remainder of this paper is structured as follows: In Sect. 2, we present the required background information and overview of technologies. In Sect. 3, we present related research regarding IPFS monitoring. Section 4 focuses on nodes, presenting our data collection methodology and our findings regarding the nodes that constitute the IPFS network. Then, Sect. 5 goes a step higher in terms of abstraction, focusing on the content stored in IPFS. In Sect. 6, we discuss possible countermeasures to isolate malicious nodes. Finally, in Sect. 7, we summarise our findings and contributions, discussing possible future research directions.

¹ <https://decrypt.co/resources/how-to-use-ipfs-the-backbone-of-web3>.

² <https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/ipfs-the-new-hotbed-of-phishing>.

2 Background

2.1 IPFS

InterPlanetary File System (IPFS) [3] is a peer-to-peer file-sharing system, consisting of many novel technologies, aiming to achieve decentralised data storage and low latency file distribution. Some of its main goals are to foster censorship circumvention and to avoid a single point of failure. E.g., in 2017 it was utilised to disseminate and store data regarding the Catalan independence referendum³ when the Spanish government attempted to censor it.

Contrary to traditional file systems, in IPFS files are addressed by their content and each one is assigned a unique content ID (CID). One of the main IPFS components is `libp2p` [9], an umbrella term for many underlying network protocols. IPFS uses Distributed Hash Table (DHT), a highly scalable coordinator of data lookup among the different nodes. `libp2p` provides IPFS with the KAD-DHT, a Kademlia [11] variant. The latter is responsible for storing three types of mappings: 1. Provider Records, i.e., what content is hosted by whom, 2. Peer Records, i.e., who (PeerID) has what address and finally, 3. InterPlanetary Name System (IPNS) records, i.e., static names pointing to varying data. Another noteworthy component is BitSwap, which is a data-exchanging protocol based on `want-have content` and `have content` messages [15]. Moreover, Merkle DAG, a combination of Merkle Tree and Directed Acyclic Graph (DAG), is used to certify that the data exchanged are unique and IPFS does not store any duplicates. Finally, users can have access to files stored on IPFS, through HTTPs, by visiting public gateways. Public gateways have been provided not only by Protocol Labs, which is the main developer of IPFS but also by various companies embracing Web3, like Cloudflare, Pinata, etc. (<https://ipfs.github.io/public-gateway-checker/>). As of July 2022, i.e., v0.14, the implementation of IPFS is known as Kubo⁴.

3 Related Work

P2P networks have been of interest to the scientific community for many years, and while their popularity fluctuates, they have never been outdone. In recent years, the advent of cryptocurrencies and blockchain technology has brought them back into the limelight. Thus, while P2P node profiling has been extensively studied in the past, research in the context of Web3 is minimal. Web3 is in a very early phase and its decentralised components are still under heavy development. Hence, the current research regarding its nodes is still in its infancy. Henningsen et al. in [7] make one of the first attempts to explore the IPFS network. Adopting a hybrid design, passively and actively, they aim to enumerate the IPFS network and profile its nodes. The authors note that the overlay network outperforms the

³ <https://edri.org/our-work/no-justification-for-internet-censorship-during-catalan-referendum/>.

⁴ <https://github.com/ipfs/kubo/blob/master/docs/changelogs/v0.14.md>.

overlay induced by buckets. Furthermore, they observe that an overwhelming percentage of nodes, i.e. 94%, did not react to the authors' attempt to connect to them. The reason this happens is twofold. The first is because many nodes are behind NAT and thus advertise their local IP address. The second is that a large portion of users uses IPFS in an opportunistic way, therefore their footprint remains in buckets for longer than they remain online and connected.

Recently, researchers discovered a botnet hiding in the IPFS ecosystem [14]. The latter, named InterPlanetary Storm (IPStorm) and estimated size of 9000 devices, utilises IPFS at multiple levels. Initially, the researchers found that it uses the libp2p DHT to discover nodes. Bots identify each other with the attribute **Agent Version**: "storm". In addition, the botnet utilises the Pub/Sub protocol as a communication channel over specific topics. Finally, the botnet uses IPFS to share files so that it can be updated to a newer version.

Trautwein et al. [16] further to providing a basic guide of IPFS' design, they collected data from three different sources to shed light on various metrics related to IPFS performance. Initially, they crawled the IPFS network to gather information about peers. Among the conclusions drawn is that IPFS nodes are geographically distributed in 152 countries, yet more than 50% are located in just two countries, US and China. Furthermore, more than 50% of the IPs are covered by five automated systems, yet only 2.3% of the nodes are in some cloud infrastructure. The last insight extracted from this dataset is that the IPFS network suffers from high rates of churn, with 87.6% of peers having an uptime of less than 8 h. Finally, the authors wanted to study the time performance in downloading data. To this end, they experimented with different AWS regions and recorded the download duration from the data they produce each time. In 50% of the cases, the download took less than 3 s, and in 90% of the cases, less than 4.5 s.

4 Profiling IPFS Nodes

4.1 Data Collection Methodology

To enumerate the IPFS network we used the IPFS Crawler [7]. The IPFS crawler is a tool written in Go and is based on libp2p (v0.11.0). Acting as a Kademlia node the crawler uses precomputed keys to extract all the entries from most buckets for every node it encounters. In essence, it invokes `FINDNODE` actions repeatedly using the appropriate precomputed keys. Finally, the crawler produces two files: (i) a JSON file storing the tuple `<PeerID, multiaddress, agent, reachability>` for every distinct node met, and (ii) a CSV file containing all the pairs of connected nodes.

We conducted a series of consecutive crawls. Initially, the crawls were performed iteratively, every ten days during the period from March to April 2022. Each crawl series spanned over a day (24h) totalling about 360 crawls in a row per day. From the data in the JSON file, for each PeerID we extracted the IP addresses. Each IPFS node maintains an address book retaining information for the nodes it encounters. If any of the encountered nodes advertises a new

address, then it is appended in the address book for reachability purposes. As a result, a single PeerID may correspond to more than one IP address. We studied each different address considering it as a unique node. Moreover, nodes behind a firewall or NAT use `p2p-circuit`, a `libp2p` relay transport protocol, to avoid connectivity barriers. In essence, these nodes advertise addresses through relay nodes. As a consequence, they do not reveal their real IP address but the IP address of the relay. The aforementioned peers as well as those which advertise only local IP addresses are excluded from our analysis. Clearly, the absence of such IP addresses prevents us from studying or fingerprinting the corresponding hosts.

4.2 Node Profiling

In this section, we present general information regarding the IPFS network and its nodes. We should mention that in the following findings, every different IP address is considered a different node. Although, we found that unique Peer IDs advertise multiple IP addresses since our study focuses on the “fabric” of the IPFS network. Thus, we want to enumerate and analyse every different IP address.

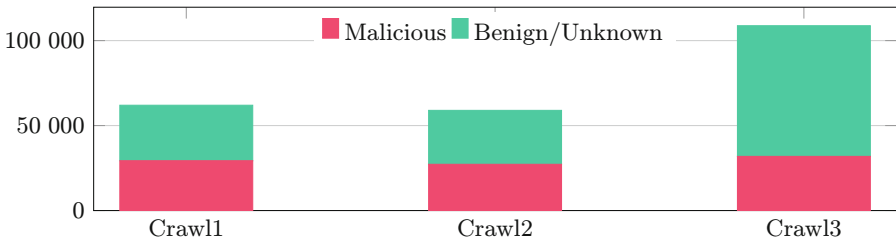


Fig. 1. Malicious nodes per crawl.

Figure 1 illustrates the nodes per crawl and the count of malicious nodes for which we collected intelligence. In Fig. 2a, the exact results of IP addresses per crawl can be found. Moreover, from the same figure, we can observe that 16783 were found online in all three crawls. We can assume that the aforementioned nodes were found online at least once a day in the span of the whole month. Given the periodic changes of IPs, we can assume that most of these IPs belong to some infrastructure that has been devoted to constantly working with IPFS.

A node’s agent version can be an indication of malicious activity. Nodes’ agent version is public and advertised, thus, it can act as an identifier for malicious nodes to discover and track each other. The latter is a technique already implemented by “storm” agents. Figure 3 illustrates the ten most used agent versions we found in each crawl. We should highlight that the counts depicted correspond to the agents from the nodes we managed to connect to. In each crawl we found 50%, 61%, 49% respectively, unreachable peers, i.e., we found their address

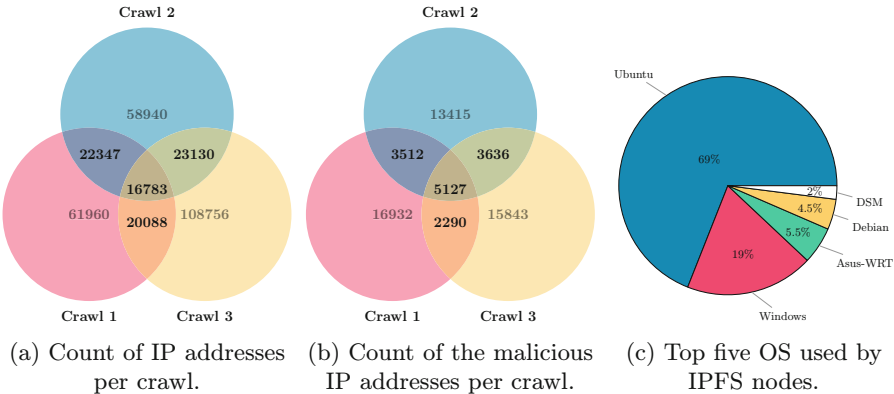


Fig. 2. Crawl statistics.

stored in the DHT but they were offline. Moreover, IPFS is open-source software; therefore, it is at the user’s discretion whether to display the agent version. The latter results are aligned with the ones in [16]. In the third crawl we observe that there is an increase in nodes using the agent called **Hydra Booster**⁵. Hydra Booster is a node having many different Peer IDs over a common routing table. It is designed to accelerate IPFS’ processes carried out through DHT-like content resolution, routing and discoverability. The existence, as well as the operation of these nodes, brought about an increase in the number of nodes of the third crawl. One of the features of open software, which has been hotly debated lately, is that upgrading to a newer version is at the user’s discretion. Observing the crawling results of Fig. 3, one can observe that there are many different software versions running and communicating simultaneously. For example, `go-ipfs 7.0` was released in July of 2020 while `go-ipfs 11.0` in August of 2021. Moreover, although the measurements were made in mid-2022, and version `go-ipfs 12.0` had already been released, we can conclude from the bar charts that the versions which are more widely used are the older ones. In addition, we must mention that agent storm, which has been found in all three crawls with a non-negligible number, is characteristic of the nodes belonging to the IPStorm botnet we have already mentioned.

In what follows, we study the maliciousness of nodes, so we used Virus Total to assess the corresponding IPs. Nevertheless, Virus Total also provides valuable insights regarding the geographic distribution of the various nodes, regardless of whether they are malicious or not. The vast majority of the nodes are located in two countries, namely the United States and China. We notice that our results are aligned with [16].

To conduct a more in-depth analysis, we passed the crawling results to intelligence services. Namely, we used Shodan, a network monitoring tool, to fingerprint each node. Shodan returned intelligence for approximately 40960 unique

⁵ <https://github.com/libp2p/hydra-booster/>.

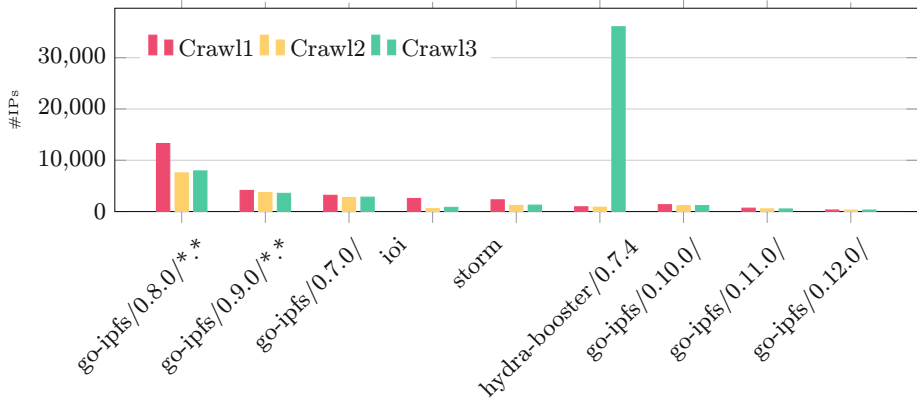


Fig. 3. The ten most commonly used agent versions in each crawl. The ****** denotes varying subversions combined.

nodes. Figure 4 illustrates the ten most commonly used ports by the total of nodes we examined. Port 22, the most widely used port by IPs related to IPFS, is typically used for Secure Shell (SSH) connections, which allow users to log in to a host and execute commands remotely. Port 80 is used as the default port for HTTP (Hypertext Transfer Protocol) traffic, port 8080 is an alternative to port 80 and moreover the default port of the IPFS gateway, and port 443 for HTTPS. Port 3389 is typically used by hosts running Microsoft Remote Desktop Protocol (RDP) to allow remote access to the host’s desktop. Finally, port 4001 is used by default for IPFS traffic, but users can also set up a custom port. Regarding the operating system running on IPFS nodes, Shodan’s results, depicted in Fig. 2c, indicate that the lion’s share uses Ubuntu Linux. The next runner-up is Microsoft Windows 10, followed by Debian Linux. The latter is also exhibited by the most used services, Fig. 5, where most hosts appear to be using SSH as opposed to RDP. Moreover, most of them seem to have a web server (nginx and then Apache).

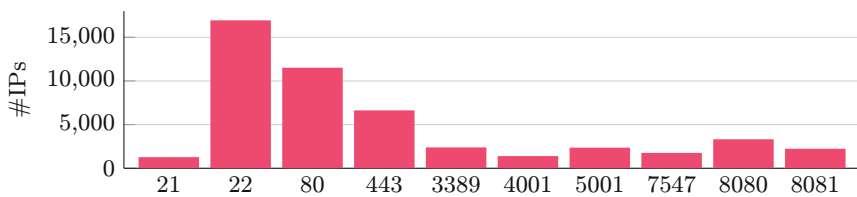


Fig. 4. The ten most common ports.

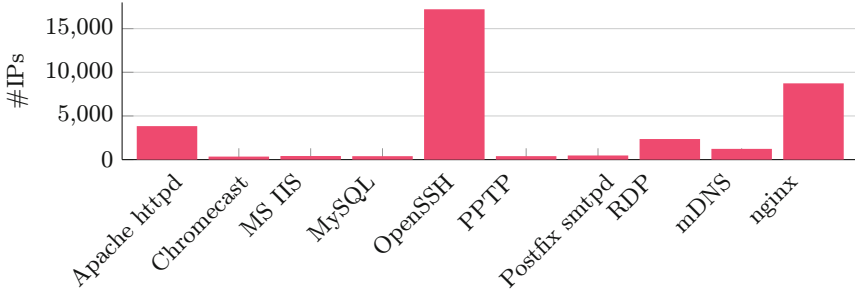


Fig. 5. The ten most commonly used services.

JARM [1] is an open-source fingerprinting tool that generates a string based on the response of the host to ten TLS packets. JARM is used by the community as a software-wise host clustering tool, therefore it is also eligible to detect malware Command & Control (C2). We use JARM strings, extracted from Shodan and Virus Total, to detect any similarities among the different nodes. Finally, we combined them since for the same IP different services can provide varying information. For 1002 IP addresses, we found information in both services, so we considered both records. The JARMS indicate that there are several clusters of IPs in which servers have the same TLS configuration, which implies that the same entity is behind them. The most common ones are illustrated in Table 1.

Table 1. Most common JARMS.

JARM	# IPs
2ad2ad0002ad2ad00042d42d0000008aec5bb03750a1d7eddfa29fb2d1deea	2070
2ad2ad16d2ad2ad22c2ad2ad2ad2adfd9c9d14e4f4f67f94f0359f8b28f532	1378
15d3fd16d29d29d00042d43d000000fe02290512647416dcf0a400ccbc0b6b	577
15d3fd16d29d29d00042d43d0000009ec686233a4398bea334ba5e62e34a01	562
15d3fd16d21d21d00042d43d000000fe02290512647416dcf0a400ccbc0b6b	489

4.3 Malicious Activity

In this section, we investigate the moral character of IPFS nodes, i.e., we examine whether and to what extent there are malicious nodes. To this end, we collect and leverage existing intelligence to create and present their profile. Our goal is to assess the network structure, keeping IPFS users and the related community alert to the existence of malicious activity in the IPFS network. Due to the current IPFS rules, every node maintains several active connections varying from 600 to 900 peers. Thus, we argue that it is very important for each node to know what kind of alignment, i.e. neutral or malicious, the node it interacts with has.

Initially, we leveraged the intelligence provided by two popular services, namely Virus Total (<https://virustotal.com/>) and SpamHaus (<https://www.spamhaus.org/>), to get a baseline for the reputation and past activity of nodes. SpamHaus uses several methods to find information about an internet resource. It uses sensors in large networks, i.e. a data-sharing community, from which it collects data about network traffic. In addition, SpamHaus deploys honeypots to attract malicious users. Along the same lines as SpamHaus and VT, in addition to monitoring more than 70 anti-malware and IP blocking services, it relies on data generated and shared by an already large community. Both the aforementioned services provide APIs to interact with their knowledge base and generate a JSON formatted output for each request. We combine the extracted output information with the SpamHaus output and we consider malicious those nodes with at least one record in one of the aforementioned services.

Moreover, in Fig. 2a, we notice that from the 27861 different IP addresses we encountered during the first crawl, 5126 of them, $\approx 18\%$ remained online throughout the whole month. The latter indicates that there is a number of nodes that constantly utilise the IPFS network for malicious purposes. Compared to the 16783 found online in all three crawls, as depicted in Fig. 2, a significant part of them, i.e., 30.5%, are known to be malicious. Based on SpamHaus' results, we conclude that the majority of malicious nodes were discovered using the DNS Sinkhole technique. According to this technique, security researchers create, at various levels, a DNS record of a known malicious URL pointing to an address they own, usually a sinkhole server. The gain from applying this technique is twofold: On the one hand, they prevent communication between bot and C2, and on the other hand, researchers can find which computers are infected, i.e. ask to connect to known malicious URLs.

In Table 2a, the five most commonly requested and sinkholed URLs in the number of unique IP addresses are illustrated. Note that several URLs such as `differentia.ru`, `atomictrivia.ru`, `amnsreiuojy.ru` and `restlesz.su` are known to be leveraged as C2 by malware. `disorderstatus.ru` is a relatively newly created domain reported to be mostly used for spamming. To draw deeper conclusions about the URLs, we isolated the Top Level Domain (TLD) of the different requested URLs. To our surprise, while most requested URLs have a ".ru" TLD, this is not reflected among the unique TLDs. On the contrary, we notice that the most commonly encountered is ".xyz", a relatively new TLD offering many domains that would traditionally be registered by legitimate users. The fact that they are new and cheap and that traditional domain names are available has led xyz domains to be widely exploited⁶. Given that 11227 xyz domains are hosted by these addresses makes us conclude that some adversaries use nodes of IPFS for hosting malicious domains in addition to C2 infrastructure. Tinba a portmanteau of the words Tiny Banker, is a trojan that leverages packet sniffing to determine whether the user visits a bank's webpage. In that case, the trojan

⁶ <https://www.spamhaus.com/resource-center/getting-the-low-down-from-xyz-registry-on-combating-domain-abuse/https://www.bleepingcomputer.com/news/security/these-are-the-top-level-domains-threat-actors-like-the-most/>.

tries to steal the keystrokes and sends them to a C2. Nymaim and Ranbyus are well-known trojans, which steal information from the user and consequently send them to a C2. Some of their variants have been found to use domain fluxing to communicate with their orchestrator, and some have been found in DoS attacks. Mirai is used to infect Internet of Things (IoT) devices and turn them into bots that can be used to launch large-scale network attacks. The Mirai botnet was initially discovered in 2016 and was part of various high-profile cyberattacks, including distributed denial-of-service (DDoS) attacks that brought down popular websites and online services. The most frequently displayed campaigns are gathered in Table 2b.

Table 2. Extroversion of malicious nodes: Which groups do they belong to and what webpages they seek to visit.

URL	Count	Campaigns Count	
differentia.ru	38681	tinba	30019
disorderstatus.ru	15504	conflicker	22650
atomictrivia.ru	7049	nymaim	22228
amsreiujy.ru	5662	andromeda	6403
restlesz.su	2180	ranbyus	4845
		mirai	3750

(a) The five most sinkholed URLs and the number of unique requests.

(b) Malware campaigns with the largest participation from the encountered nodes.

Finally, we studied the JARMs of malicious nodes to better frame our research. As we have already mentioned, we combined knowledge from all intelligence services to produce the results. Notably, among them, we found a cluster of 68 nodes corresponding to the JARM fingerprint 15d3fd16d29d29d00042d43d000009ec686233a4398bea334ba5e62e34a01 which is attributed to the notorious *emotet* botnet.

As already mentioned, the crawler we used, in addition to information about the nodes encountered, produces an edge list with each pair of connected nodes. Based on this, we constructed a mapping from one PeerID to the several PeerIDs we found connected during the second day. In essence, we built for each peer its buckets expanded to the span of a day. Consequently, we converted the aforementioned mapping to the corresponding IP addresses. This way, we can investigate whether there is a clique between the malicious nodes. The findings indicate that there is no such clique, as the median percentage of malicious nodes in the buckets of a malicious node is 7%, and the average is 9.5%. Along the same lines, the median percentage of nodes in the buckets of a benign node is also 7%, with the average being 9.2%.

5 File Investigation

Despite the processes and functionality IPFS offers through libp2p and its other components, its main purpose is undeniably storage-related. The largest NFT marketplaces use IPFS for the data storage and integrity it provides, while its widespread utilisation has already brought about the need for cooperation with other Web3 layers, such as ENS, which natively offers names corresponding to CIDs. No wonder the increasing popularity has also caught the eye of cyber criminals. A recent research⁷ highlights that the volume of malware samples hosted in IPFS has increased during 2022. Moreover, researchers report the **Agent Tesla** malware, which using phishing techniques, leads to an IPFS public gateway, disguising the download of malicious content. To better frame our research into the storage of the IPFS ecosystem, we also researched the file side. Our research is twofold, in the first case, we eavesdropped on the files requested by IPFS users, while in the second, more actively, we searched for files we randomly downloaded from well-known torrent sites.

5.1 Bitswap Eavesdropping

According to the operating rules of IPFS, when a user searches for a file, a one-hop inquiry is first performed through Bitswarm, requesting it from nodes with an active connection to the initiator. If none of them responds, the query is then served by the DHT. To collect data, we tweaked our node so that it maintains active connections with around 4000 nodes; that is, according to our measurements, approximately 20% of the network's active nodes at that time. So when one of those nodes was looking for a file, thanks to Bitswap's functionality, that information would also go through us. This way, we could eavesdrop on about 20% of the network's requests and, in turn, request back to retrieve them. In total, we monitored the requests for 24 h while we set each request to last no more than 15 s. This way, we avoided downloading very large files while, on the other hand, we cancelled the search in case it was routed through the DHT. In total, we collected 49155 files with a size of about 13.7 GB. To have a more complete picture of the type of files requested, we used the Python `mimetypes` module⁸ to find the MIME type of each file. We shall mention that it managed to classify 13691 of the files. The latter can be attributed to Bitswap's design. When a user requests a file from Bitswap, the search is performed by the root CID of the file. The aforementioned file contains links to the chunks of which it is composed. Thus, when the requester receives the root CID and learns the CIDs of the chunks that make up the file, it requests through Bitswap consecutively all the chunks, which are essentially blocks of data. The file results illustrate that 3716 are image files with MIME types "image/png", "image/gif", "image/jpeg", and 9148 are JSON files, which is the most common format for NFT metadata. The latter clearly demonstrates and confirms our initial statement that IPFS

⁷ <https://blog.talosintelligence.com/ipfs-abuse/>.

⁸ <https://docs.python.org/3/library/mimetypes.html>.

is a cornerstone of NFT data storage and Web3 in general. Among others, we fetched 177 Javascript files and 27 videos of type “video/mp4”. We then fed the image files to the Python Not Suitable For Work (NSFW) Detector module to determine whether IPFS is being used for inappropriate content. From the 1636 image files it examined successfully, it found 33 unsuitable⁹. The above indicates that some users leverage IPFS’ anonymity to host inappropriate content that is difficult for LEAs to track and take down.

5.2 Torrent Files

Very often, inappropriate files are found in the form of torrent files disseminated through torrent search engines. We downloaded a sample from various widespread torrent sites, ten popular torrents in total. We computed their CIDs locally to determine whether they are shared on the IPFS. This way, not only did we not add any illegal files to the IPFS network, but we also limited the possibility of tampering with the results of our upcoming searches. The ten different torrent files yielded 72 different root CIDs. Each torrent file can contain a video file, a cover image for the video file, a text file with information about the file, etc. In turn, we made 72 requests to the DHT for providers of these CIDs. We found providers for seven of them, and in fact, for most of them, more than one. The latter implies that IPFS users may also share the same content in torrents and that intellectual infringement content is also distributed through IPFS.

6 Countermeasures

The amount of malicious nodes connected to IPFS is alarmingly high. Given the P2P nature of IPFS and its continuous exploitation, we believe that pruning nodes from the network might provide an initial measure of sanitising the network; otherwise, the benign peers facilitate the malicious ones. To this end, we opt for a periodical blacklist approach that is resolved through InterPlanetary Name System (IPNS). In essence, we propose using the proposed data crawling methodology to monitor the nodes on a daily basis, the IPs are collected and using intelligence services, we determine whether the IP should be blocked or not. Each IP is four bytes long, so the expected size is rather small and easy to manage. For instance, using our experiments as a baseline, using the worst estimate of 32000 malicious nodes, the blacklist would be around 125KB if the IPs were directly stored (4 bytes per IP). Given its size and possible optimisations (e.g. use binary search over the sorted list), searching whether the connected peers are malicious can be very efficient. Moreover, since the amount of nodes is tolerable, the collection of data from intelligence services can be rather fast. Of course, one could hide the IPs using approaches based on Bloom filters [4]. In this case, one would need less than half of this storage (almost 56KB) to store these IPs with 0.01% possible false positive. However, the issue is that

⁹ <https://pypi.org/project/nsfw-detector/>.

this error would be persistent, meaning that the nodes that would be false positives would be considered malicious by everyone without being able to rectify this error. Nevertheless, with the growth of IPFS and the increase of malicious nodes, probabilistic structures such as Bloom filters might be more optimal.

IPFS is becoming institutional, after all, many organisations are participating in it and supporting it. Recent research efforts indicate that it could frame the existing banking system [10], while at the same time, it constitutes a cornerstone of Decentralised Finance (DeFi). Our research does not intend to act as a brake on its use; on the contrary, it intends to inform, alert and promote its secure use. For instance, the network administrator of an organisation participating in the IPFS network can block the traffic towards and from a suspicious IP address by adding a rule to the firewall. Note that it can also remove alert fatigue from SOCs who might observe malicious IPs connected to the monitored infrastructure due to IPFS traffic. Finally, while IPFS provides the ability to disconnect from a node, it does not provide natively the option for the user to maintain a blacklist.

7 Conclusions

Open and decentralised systems are, by their very nature, prone to several attacks. However, given the crucial role of IPFS for Web3, it is essential to protect the ecosystem. Our measurements indicate that an alarming number of IPs reported as malicious through intelligence services are using IPFS. Rather than making it centralised, we opt for soft measures that allow nodes to isolate malicious ones selectively. We argue that this isolation can significantly benefit the network as the content of most of these nodes may be malicious, leading legitimate ones to facilitate nefarious acts and malicious campaigns. Therefore, their isolation, in the long run, may increase the robustness of the network and trust in it.

IPFS seems to have sacrificed part of the privacy to succeed in terms of performance, speed, and robustness [2]. This shortcoming can be exploited for malicious purposes, but it can also be leveraged by security analysts to monitor malicious nodes. Thus, apart from the fact that we can obtain critical information regarding a malicious node, such as its IP address, we can also monitor it from a content point of view, i.e., its requests as well as what it provides. Therefore, a future direction of this work is an extension of the implementation of the proposed filter so that it associates malicious nodes with the corresponding content.

Acknowledgements. The authors would like to thank Dennis Trautwein for his insightful comments. This work was supported in part by the European Commission under the Horizon Europe Programme, as part of the project LAZARUS (<https://lazarus-he.eu/>, Grant Agreement no. 101070303) and the Horizon 2020 Programme, as part of the project HEROES (<https://heroes-fct.eu/>, Grant Agreement no. 101021801) and was also supported in part by the Research Center of the Athens University of Economics and Business.

The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

References

1. Althouse, J.: Easily Identify Malicious Servers on the Internet with JARM (2020). <https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a>
2. Balduf, L., Henningsen, S., Florian, M., Rust, S., Scheuermann, B.: Monitoring data requests in decentralized data storage systems: a case study of IPFS. In: 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), pp. 658–668. IEEE (2022)
3. Benet, J.: Ipfs-content addressed, versioned, p2p file system. arXiv preprint [arXiv:1407.3561](https://arxiv.org/abs/1407.3561) (2014)
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13**(7), 422–426 (1970)
5. Casino, F., Politou, E., Alepis, E., Patsakis, C.: Immutability and decentralized storage: an analysis of emerging threats. *IEEE Access* **8**, 4737–4744 (2019)
6. Cook, A.V., Bechtel, M., Anderson, S., Novak, D.R., Nodi, N., Parekh, J.: The spatial web and web 3.0: what business leaders should know about the next era of computing. Deloitte Insights (2020)
7. Henningsen, S., Florian, M., Rust, S., Scheuermann, B.: Mapping the interplanetary filesystem. In: 2020 IFIP Networking Conference (Networking), pp. 289–297 (2020)
8. Karapapas, C., Pittaras, I., Fotiou, N., Polyzos, G.C.: Ransomware as a service using smart contracts and IPFS. In: 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 1–5. IEEE (2020)
9. Labs, P.: Libp2p (2021). <https://libp2p.io/>
10. Mamun, A.A., Hasan, S.R., Bhuiyan, M.S., Kaiser, M.S., Yousuf, M.A.: Secure and transparent KYC for banking system using IPFS and blockchain Technology. In: 2020 IEEE Region 10 Symposium (TENSYMP), pp. 348–351 (2020). <https://doi.org/10.1109/TENSYMP50017.2020.9230987>
11. Maymounkov, P., Mazières, D.: Kademia: a peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_5
12. Murray, A., Kim, D., Combs, J.: The promise of a decentralized internet: what is web 3.0 and how can firms prepare? *Business Horizons* (2022)
13. Patsakis, C., Casino, F.: Hydras and IPFS: a decentralised playground for malware. *Int. J. Inf. Secur.* **18**(6), 787–799 (2019)
14. Pripoae, S.: Looking Into the Eye of the Interplanetary Storm (2020). <https://www.bitdefender.com/files/News/CaseStudies/study/376/Bitdefender-Whitepaper-IPStorm.pdf>
15. De la Rocha, A., Dias, D., Psaras, Y.: Accelerating content routing with bitswap: a multi-path file transfer protocol in IPFS and Filecoin (2021)
16. Trautwein, D., et al: Design and evaluation of IPFS: a storage layer for the decentralized web. In: Proceedings of the ACM SIGCOMM 2022 Conference, pp. 739–752 (2022)



Quantum-Secure Communication for Trusted Edge Computing with IoT Devices

George Kornaros^{1,2}(✉) , Georgia Berki^{1,2}, and Miltos Grammatikakis¹

¹ Hellenic Mediterranean University, 71410 Iraklio, Greece
kornaros@hmu.gr

² Intelligent Systems and Computer Architecture Lab, 71410 Iraklio, Greece
<https://isca.hmu.gr>

Abstract. Internet-of-Things(IoT)-based edge computing in smart factories, smart grid, agriculture, constructions and autonomous vehicles include service-oriented gateways that connect with the cloud, perform machine-to-machine communication, often transmitting large amount data up and down the network, performing time-sensitive processing and involving intelligent local decision-making. In view of a sharp increase in cyberattacks today targeting edge computing, these gateways need to provide digital signing and key negotiation for ensuring reliable data sources, trusted applications and authentic devices and connections. In contrast to common perception, we show that post-quantum cryptography methods do not necessitate extensive modification to adopt in such environments; further, the cryptography algorithm's hardness is preserved while fulfilling the IoT device's resource limitations. In particular, we demonstrate an efficient method and an implementation on a 32-bit ARM Cortex-M4, 64KB memory microcontroller, based on post-quantum key encapsulation mechanisms (KEMs), for secure communication and authentication in an industrial IoT environment.

Keywords: trustworthy edge computing · IoT secure communications · post-quantum KEM · FrodoKEM · firmware-over-the-air updating

1 Introduction

With the trend toward increasing computing power while consuming less energy, a challenge of edge computing is to ensure the integrity of a trustworthy source of information over the lifetime of the IoT/edge deployments [13]. For many edge and IoT use cases, the data source is not physically integrated with the edge computation hardware. Despite providing a hardware-based root of trust,

This work was supported in part by the European Union (EU) Horizon 2020 Project FLUIDOS (Flexible, scaLable, secUre, and decentralIseD Operating System) under GA No. 101070473.

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 163–176, 2024.

https://doi.org/10.1007/978-3-031-56326-3_12

methods for signatures/certificates, and smart safeguards for device and firmware protection [14], it is challenging to verify the authenticity of a sensor or other device linked to an IoT gateway, yet it is required for developing a trusted edge data source.

New quantum-resistant public key encryption, key encapsulation, and signature algorithms are being developed in response to recent breakthroughs in quantum computer technology. These are increasingly endorsed in post-quantum public key cryptosystems for potential IoT applications [12]. Lattice-based public-key encryption (PKE) schemes hold a great promise for post-quantum cryptography. Its security is dependent on lattices' worst-case computing assumptions, which continue to be regarded as being tough even for quantum computers. Lattice-based cryptographic methods benefit from very strong security proofs based on worst-case hardness, relatively efficient implementations, as well as great simplicity and, lately, their promising potential as a platform for constructing advanced functionalities. The increasingly real threat of quantum computers breaking all widely-deployed public-key cryptography has driven research in new paradigms for building core public-key primitives like signatures, public-key encryption, and key encapsulation mechanisms (KEMs) from problems that are computationally intractable even for quantum computers. An umbrella term for this is Post-Quantum Cryptography (PQC). The US National Institute of Standards and Technology (NIST) is in the process of selecting new standards which will be used for decades to come. The process has reached its third round with four finalists in the KEM/PKE category: Classic McEliece [1], Kyber [20], NTRU [6] and Saber [10].

Key encapsulation methods (KEMs), also known as key encapsulation techniques, are an asymmetric encryption technique that improves the safe transmission (or production) of symmetric keys by eliminating the need for randomly generated padding in short messages. This is the case for the algorithm of FrodoKEM which is induced from an INDistinguishability under Chosen Plaintext Attack (IND-CPA)-secure public-key encryption scheme called FrodoPKE [2,5]. FrodoKEM is a Chosen Ciphertext Attack (CCA)-secure and Chosen Plaintext Attacks (CPA)-secure lattice-based cryptosystem that relies on Learning with errors (LWE) problem solving for its protection. It has slightly larger key sizes and slower performance as compared to other lattice-based models, based on LWE rings. As constructed, FrodoKEM is also "constant-time". To prevent some forms of eavesdropping attacks, it does not need to be reoptimized in terms of security. Constant-time is a cryptographic security feature that protects against a variety of side-channel timing attacks.

IoT devices are now commonly pre-provisioned with digital certificates that have been issued by the manufacturer (or configured in collaboration with the client) and are used for key exchange and authentication, which is a standard practice in IoT security today. Even with immutable hardware, such as ROM, second (or higher) stage boot loaders can rely on memory that is programmed only later in the process by Original Equipment Manufacturers (OEMs). As an example, in complex ecosystems, different parties may be in charge of the

various stages in the chain. Therefore, automotives and Industrial IoT (IIoT) environments increasingly use firmware updating or patching with limited-life session keys for authentication and key-exchange protocols. In order to secure firmware updates of industrial deployments with IoT devices, it is crucial to apply cryptographic techniques that are immune to quantum technology-based attacks.

1.1 Secure Communication in Industrial IoT-Based Environments

Edge computing is becoming increasingly important in supporting the discovery and authentication of infrastructure resources such as compute, network, and storage, as well as other resources such as IoT devices, sensors, data, code units, services, applications, or users interacting with the system. Today, centralized gateway-based systems commonly rely on the installation of a secret on IoT and computing devices for device authentication (e.g., a device certificate stored in a hardware security module, or a combination of code and data stored in a trusted execution environment).

Figure 1 shows a microfactory setup to provide improved preventive maintenance through data collection for monitoring of manufacturing processes. To develop security measurements in each layer of the IoT technology stack and effectively protect and recover from potential security threats and attacks, it is crucial to establish authenticity and secure communicating with all IoT devices. Most IoT devices are low-energy embedded devices that lack the computing resources needed to support the implementation of advanced and effective authentication and encryption algorithms because they are unable to perform complex processing operations in real-time. At the same time, modern implementations of RSA-based security protocols, including the latest Transport Layer Security (TLS) 1.3 standard (which does not even support RSA key exchange) attempting to protect against microarchitectural and timing side channel attacks, are shown to be at risk as new side-channel attack techniques are demonstrated to overcome countermeasures in force [18].

In this paper, we demonstrate the integration of FrodoKEM in modest IoT devices while extending firmware updates for microcontrollers in a microfactory environment that incorporates IoT devices for monitoring the factory machinery. Constrained industrial IoT devices, such as LoRa end-nodes, are first authorized by a FrodoKEM implementation in case of a two-way authentication scenario. The successful authentication/authorization enables the firmware updating of the embedded devices inside each island, through integrating KEM/PKE for server-client authenticated key exchange. This is achieved by employing a constant-time implementation of FrodoKEM, which follows the 1st-round specification, tailored for a resource-constrained STM32WL55JC1 LoRa device.

Despite the fact that the proposed framework is useful for the wider context of establishing secure communication in constrained machine-type communication environments, we present next a specific use-case.

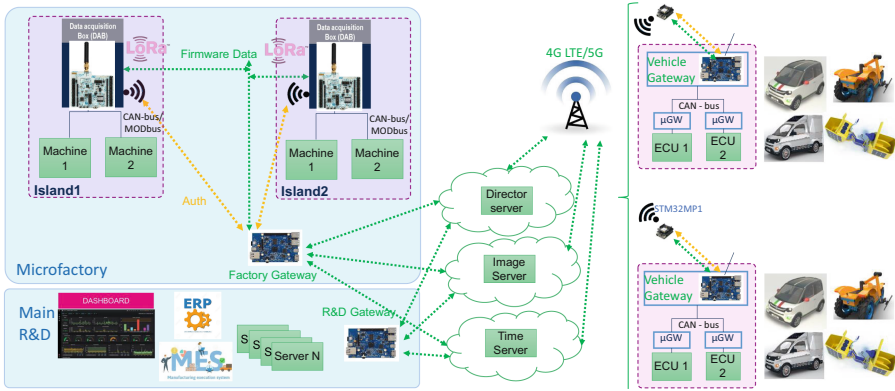


Fig. 1. Security in a microfactory organization integrating quantum-secure island authentication and uptane-based firmware updating.

Use-Case. Providing security to IoT systems is a challenge [22], especially as new wireless communication technology, low power wide area network (LPWAN) has emerged as industrial IoT applications get closer to Industry 4.0 automation. Due to the lack of security concerns against quantum attackers, previous research has shown weaknesses in many existing IoT systems and services [16, 19]. It is critical to offer device security and authentication, together with remote detection of technical issues. In this scope, Firmware-Over-The-Air (FOTA) updates for Industrial IoT devices since IoT service management is becoming a part of the new technological innovation connected to the emerging industrial IoT, with features like adaptive security, scalable and efficient routing, among others.

The Uptane standard [8] has already been used successfully in the automotive industry to mitigate risks, including local and remote assaults, that aim to intercept and tamper with the new firmware in order to update the IoT device with a modified and purposefully defective firmware image. Similarly, in an IoT-enabled factory infrastructure, adversaries may deny a device’s functions via a rollback assault, or drain a microcontroller’s resources in a denial-of-service manner, or, in the most severe case, control an IoT device with malicious software.

Using an Edwards-curve Digital Signature Algorithm (EdDSA) variant, the ED25519, which is not quantum resistant, ASSURED, a framework for over-the-air (OTA) software update [3], provided end-to-end authentication and integrity so that only authorized devices could install the update. Heterogeneity of IoT devices and very constrained devices are not considered though. Examples of employing ARM TrustZone technology have been suggested to facilitate a secure firmware-over-the-air (FOTA) update [11]. The image signature is verified using the RSA technique, while the validity and the corruption of the image are checked using an application that is operating in a secure OP-TEE. However, the solution is not generic and not tailored for IoT devices with very constrained capabilities.

A firmware-over-the-air solution was also built on top of a real-time OS utilizing an STM32F779NI MCU vehicle gateway, where both the Primary and Secondary client functions were combined into a single entity [17]. Internally, the vehicle's gateway Electronic Control Unit (ECU)'s secure CAN-bus interface for firmware transfer was accomplished, however the ECUs lacked robustness against quantum attacks.

PQC-Based Extensions for FOTA Updating. Figure 2 presents an overview in a PQC-strengthened device authentication setup, through a two-factor authentication of gateway-devices inside IoT-islands. An ephemeral key can be shared by two parties via KEM, a one-round protocol. In particular, a sender creates a ciphertext of an ephemeral key using the public key of a recipient. The sender cannot specifically select the ciphertext. The receiver then uses the same ephemeral key to decipher the ciphertext. Encapsulation and decapsulation, represented by Encaps and Decaps, respectively, are the terms used to describe the algorithms used by the transmitter and the receiver. The restrictions imposed by IoT devices and communications need to be taken into account in order to provide a realistic implementation when this approach is utilized for ephemeral key exchange whenever a new FOTA update is launched. Specifically, the size of PQ public key or signature can become an additional constraint. In particular, FrodoKEM public key (pk) and ciphertext (ct) is 19336 bytes. Since, the maximum payload size of a LoRa packet cannot exceed 222 bytes considering the Eu863-870 region, we use the LoRa established (and AES-encrypted) link to send part or parts of the pk and ct. The remaining bytes are sent via the gateway-to-gateway link and assembled in place.

Gateways provide secure communication by utilizing TLS 1.3, and ephemeral key exchange in TLS 1.3, as standardized, is based entirely on elliptic curve Diffie-Hellman, while the transition to post-quantum cryptography promises to provide great potential for both secrecy (through post-quantum key exchange) and authentication (by post-quantum digital signatures) [9].

2 FrodoKEM Implementation on Low-End IoT Devices

When a large number of devices are active at once, it becomes challenging to identify security risks and adversaries who can infect authorized devices and obfuscate the enormous volumes of data being transmitted across the network. To provide authentication and authorization without having to invest in new deployments, common approaches involve adding a security device in front of each traditional industrial IoT object. For resource-constrained devices and LPWAN settings, we provide the FrodoKEM implementation to extend this strategy and make the devices quantum resistant (i.e., secure against the quantum computers being developed today).

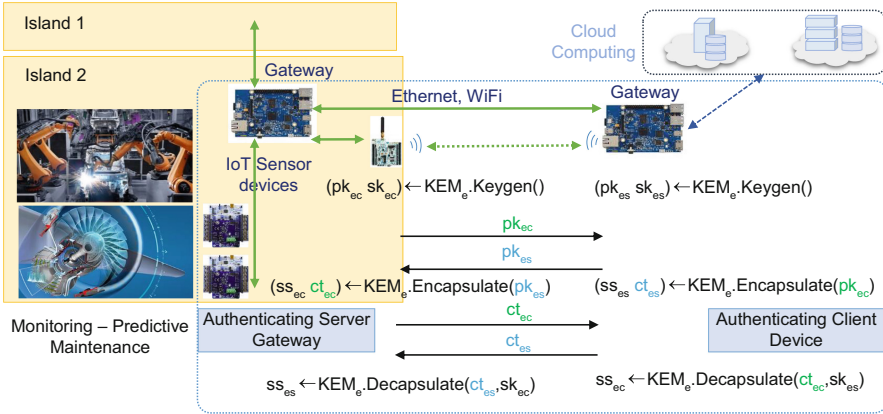


Fig. 2. Microfactory organization integrating quantum-secure gateway-device key exchange with authentication for firmware updating.

STM32WL55JC1 multiprotocol LPWAN includes a dual core microcontroller, a 32-bit Arm Cortex-M4 and an ARM Cortex-M0, operating at a frequency up to 48 MHz, while it accommodates a Flash memory of 256 KB and an SRAM of 64 KB. The adaptive real-time accelerator (ART Accelerator) allowing 0-wait-state execution from Flash memory, can enable efficient storage and retrieval of firmware data, not able to sustain during firmware execution due to restricted memory space.

The FrodoKEM-640 scheme is realized to establish an authentic LoRa end-device for initiating the Uptane-based protocol between the devices behind the LoRa end-device and inside a microfactory island. FrodoKEM-640 can use either advanced encryption standard, AES (e.g., 128, 256), or SHA-3-based extendable-output function SHAKE to create internal sample matrices during encryption or decryption. The algorithm has three basic functions: the KeyGen(), the Encaps() and the Decaps(). The KeyGen() is the function which generates the two keys: the first key is the public key and the second one is the secret key. The Encaps() function uses the public key and generates the ciphertext and the shared_secret of encaps. Decaps() uses the secret key and ciphertext as parameters; with the processing of these functions, another share_secret is generated for Decaps(). If the two share_secrets are equal then the gateway – LoRa end-device link is secured.

KeyGen() refers to certain parts of code and its function is under the name Keypair [2]. The function parameters include the public key pk and the secret key sk . First, the public key is created and then, with the usage of this key, the secret key is generated.

This process involves the generation of a random matrix called seedA and is composed of 16 bits; then, with the help of these 16 bits (i.e., seedA), the function Frodo.Gen generates the matrix A. Frodo.Gen or seedA has two forms. The first one uses the crypto algorithm SHAKE128 and the other one the AES.

Moreover, a pseudorandom bit string is generated, to create two arrays, the array S and the array E, with Frodo.SampleMatrix. The next step is to compute the arrays A, S and E with the combination $B = AS + E$, with b being equivalent to Frodo.Pack(B). Furthermore, it is important to create $pkh = \text{SHAKE}(seedA || b, len(pkh))$, where ‘||’ denotes the concatenation operation. The last step is to return the public key or $pk = seedA || b$ and the secret key or $sk = (s || seedA || b, S, pkh)$. This process is summarized next.

$$\begin{aligned}
 seed_A &\leftarrow U(0, 1)^{len(seed_A)} \\
 A &\leftarrow Frodo.Gen(seed_A) \\
 S, E &\leftarrow \chi \left(Z_q^{\binom{n \times n}{n}} \right), \chi \text{ is a Gaussian distribution over} \\
 &\quad Z \text{ with center zero and standard deviation } \sigma \text{ (2.8)} \\
 B &\leftarrow AS + E \\
 b &\leftarrow Frodo.Pack(B) \\
 pkh &\leftarrow \text{SHAKE}(seedA || b, len(pkh)) \\
 pk &\leftarrow seedA || b, sk \leftarrow (s || seedA || b, S, pkh)
 \end{aligned}$$

After the receiving entity gets the public key pk, i.e., $(seedA, b)$, the Encaps() (encryption) process generates the cipher text and the share_secret. The Encaps parameters are the public key, the cipher text and the share_secret. Initially, the algorithm creates a key matrix, called m. Next, two pseudorandom arrays are generated via SHAKE to create the seed SE and a random bit string. This random bit string is separated in three parts and each part is processed with the Frodo function SampleMatrix which normalizes the arrays; more specifically, it samples the error matrix and creates the three matrices: E’, S’ and E”.

Moreover, a new array called A is generated. This implementation includes giving the first 16 bits of the public key to the Frodo.Gen(seedA) function. Next, the $B' = AS' + E'$ is computed and B' produces $c1 = \text{Frodo.Pack}(B')$. After, b, which is part of the public key, is unpacked and B is created, as $B = \text{Frodo.Unpack}(b)$. The B, S’ and E” are computed and produce $V = S'B + E''$ and the array C, as $C = V + \text{Frodo.Encode}(m)$. C is computed in order to create c2 with the function pack, as $c2 = \text{Frodo.Pack}(C)$. The last step is to compute the share_secret which is named ss, and the computation is implemented with the function SHAKE128 where $ss = \text{Shake}(c1 || c2 || k, len_{ss})$. The Encaps() algorithm is summarized next.

$$\begin{aligned}
m &\leftarrow \text{Frodo.Gen}(\text{seed}_A) \\
S', E' &\leftarrow \chi \left(Z_q^{\binom{nx\bar{m}}{}} \right), \chi \text{ is a Gaussian distribution over} \\
&\quad Z \text{ with center zero and standard deviation } \sigma \text{ (2.8)} \\
A &\leftarrow \text{Frodo.Gen}(\text{seed}_A) \\
B' &\leftarrow AS' + E' \\
E'' &\leftarrow \chi \left(Z_q^{\binom{\bar{nx}\bar{m}}{}} \right) \\
V &\leftarrow S'B + E'' \\
c1 &\leftarrow \text{Frodo.Pack}(B') \\
C &\leftarrow V + \text{Frodo.Encode}(m) \\
c2 &\leftarrow \text{Frodo.Pack}(C) \\
\text{ciphertext } c &\leftarrow (c1||c2), \\
\text{share_secret } ss &\leftarrow \text{SHAKE}(c1||c2||\text{len}_{ss})
\end{aligned}$$

Decaps() or decryption is required to ensure the validity of the ciphertext with the evaluation of the share_secret. The Decaps function operates on the secret key, ciphertext and share_secret. The objective of the decryption algorithm is to check that share_secret1 is equal to the share_secret2 and thus to verify that there is no attack [2].

Initially, the ciphertext produces c1 and c2 by unpacking as $B' = \text{Frodo.Unpack}(c1)$ and $C = \text{Frodo.Unpack}(c2)$. B' and C with a part of secretkey, which is called S , generate the array M , by computing $M = C - B'S$, and then $\text{Frodo.Decode}(M)$ produces m . Pk includes the $\text{pk} = \text{seed}_A || b$ from the secret key. Afterwards, two pseudorandom arrays, $\text{seed}_{s_e'}$, k' and a random bit of string are generated [2]. This random bit of string is separated in three parts and the three matrices E' , S' and E'' are created by sampling. The algorithm also creates a new array A . After, the computation $B'' = AS' + E'$, the goal is to get B by unpacking b and calculate the $V = S'B + E''$. Then, V is used to produce array C' , as $C' = V + \text{Frodo.Encode}(m)$. The last step is to check if $B' || C$ is equal to $B'' || C'$. If the arrays are equal, then the share_secret, ss , is created with the right variables, otherwise the share_secret includes a variable error.

$$\begin{aligned}
B' &\leftarrow \text{Frodo.Pack}(c1) \\
C &\leftarrow \text{Frodo.Pack}(c2) \\
M &\leftarrow C - B'S \\
m &\leftarrow \text{Frodo.Decode}(M) \\
A &\leftarrow \text{Frodo.Gen}(\text{seed}_A) \\
S' &\leftarrow \chi \left(Z_q \binom{(n, x\bar{m})}{\bar{m}} \right) \\
E' &\leftarrow \chi \left(Z_q \binom{(n, x\bar{m})}{\bar{m}} \right) \\
B'' &\leftarrow AS' - E' \\
B &\leftarrow \text{Frodo.UnPack}(b) \\
E'' &\leftarrow \chi \left(Z_q \binom{(\bar{n}, x\bar{m})}{\bar{m}} \right) \\
V'' &\leftarrow S'B - E'' \\
C' &\leftarrow V + \text{Frodo.Encode}(m) \\
ss &\leftarrow \text{SHAKE}(c1||c2||k', \text{len}_{ss}), \text{if } B'||C == B''||C' \\
ss &\leftarrow \text{SHAKE}(c1||c2||s, \text{len}_{ss}), \text{if } B'||C! = B''||C'
\end{aligned}$$

3 Adjusting Implementation to Resource-Constrained Microcontrollers

Table 1 shows the basic data structures size in bytes. In addition, the required memory for the execution of the encryption function peaks to 51248 bytes, as measured by using an STM32L552ZE nucleo device that includes 512 Kbytes of Flash memory and 256 Kbytes of SRAM. By adding the variables allocated at runtime, the required memory grows even to 101368 bytes. It is therefore prohibitive for a microcontroller with modest SRAM, such as the STM32WL55JC1 LoRa device, to run even the encryption algorithm.

Table 1. Memory requirements of FrodoKEM640 in bytes (note that secret key size is the sum of the sizes of the actual secret value (10272 bytes) and of the public key (9616 bytes))

secret key (sk)	public key (pk)	ciphertext (c)	shared secret (ss)
19888	9616	9720	16

The memory footprint of FrodoKEM640 by using AES and SHAKE algorithms on STM32L552ZE board is summarized in Table 2.

Table 2. Memory requirements (program RAM) of FrodoKEM640 in bytes on STM32L552ZE board.

FrodoKEM640	Keypair	Encaps	Decaps
AES (1-way, 2-way cache)	41160	82032	102680
SHAKE (1-way, 2-way cache)	36064	57728	78376
AES with Flash	30952	51832	61232
SHAKE with Flash	25840	27528	37928

Hence, to minimize memory usage, memory space is reused whenever possible. For instance, B and Bp and BBp data structures require 10240 bytes and thus, all share the space called *sp*. For instance, variables k' and *ct* (ciphertext) are allocated as follows:

```
uint8_t * Fin_ct = &sp[0];
uint8_t * Fin_k = &sp[4860];
```

Figure 3 shows the performance of the FrodoKEM640 on STM32WL55JC1 through using two methods. First, by using the board’s builtin 32-bit timer TIM-2, and second, the debug and trace unit (DWT), which contains a cycle count register (DWT_CYCCNT). Comparatively, the Shake option of Encaps function gives 7.22% improved latency against the AES version.

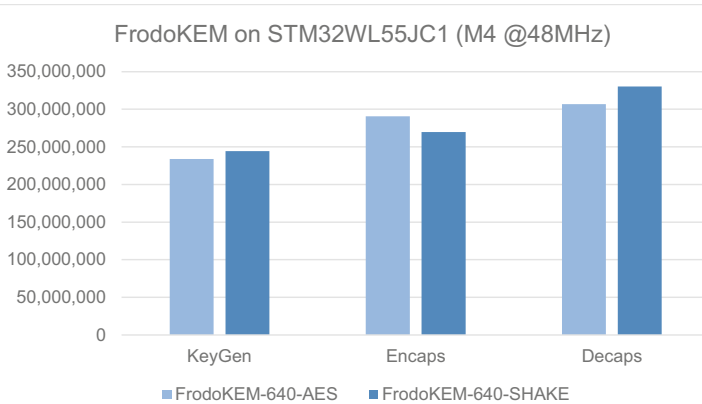


Fig. 3. Latency of keypair(), encaps() and decaps() functions on STM32WL55JC1 through using the Flash to store big data structures; Measurements in M4 MCU clock cycles (i.e., 48 MHz) are captured via TIMER-TIM2.

The STM32L552ZE platform, which has sufficient hardware resources but lacks the LoRa controller, was used as a reference point. Figure 4 summarizes the latency of FrodoKEM640 by using AES and SHAKE algorithms on the STM32L552ZE platform. The -Ofast compiler option is used to optimize for speed, and comparison results are gathered when alternative cache configurations are chosen. The Shake option of Encaps function delivers 1.14% smaller latency (1.309 s vs. 1.495 s) than the AES option when the 1-way cache is activated and an additional 0.9% improved latency (1.296 s) when the 2-way cache is active. Hence, no significant gain is observed.

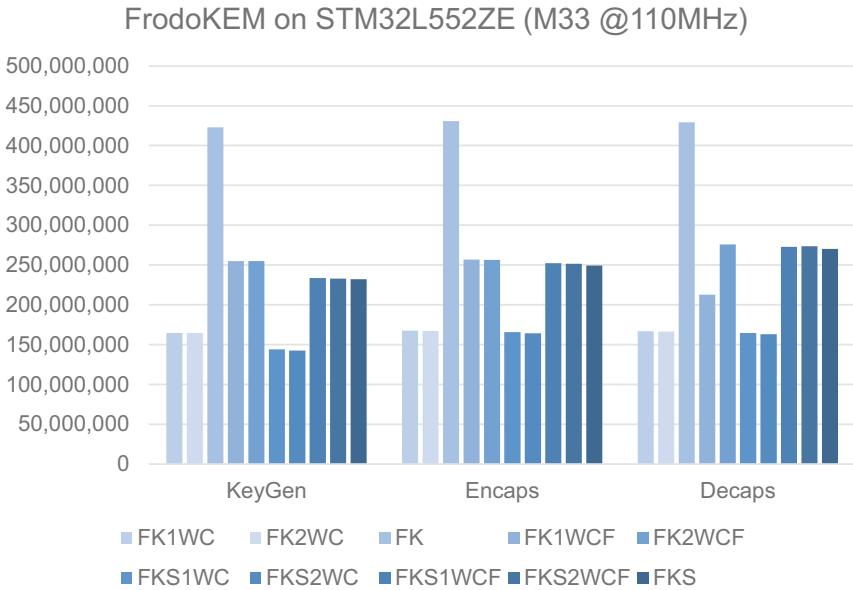


Fig. 4. Latency of `keypair()`, `encaps()` and `decaps()` functions on STM32L552ZE; Measurements in M33 MCU clock cycles (i.e., 110 MHz) are captured via TIM2. The different versions of FrodoKem (FK) include 1-way or 2-way cache (1WC, 2WC) without and with involving Flash(F), the first five versions use AES and the next five versions use Shake (S).

Figure 4 depicts the latency also of FrodoKEM (FK) without using cache, not only as a reference, but most importantly as a potential countermeasure method against return-oriented Flush-Reload cache side-channel attacks on ARM processors [23]. As suggested, such attacks on ARM can be completely eliminated if no memory sharing is allowed between apps. When considering other side channel threats [7, 21], however, trustworthy application execution is recognized as being more crucial than speed.

While the major goal is security and incorporating PQC KEM mechanisms in a restricted capacity LoRa platform, our encapsulation time and decapsulation

time are longer than those of previous post-quantum lattice-based techniques [15] but are still adequately efficient. For example, considering a common X.509 certificate verification with a latency of more than 3.3 s (ECDSA with SHA256), as shown in Fig. 5, the proposed KEM implementation delivers promising results. Additionally, note that a single LoRa packet payload of maximum size, 222 bytes at the fastest SF8, requires 655.9 ms, while the SF12 mode (but with a 51 byte payload) needs 2.793 s.

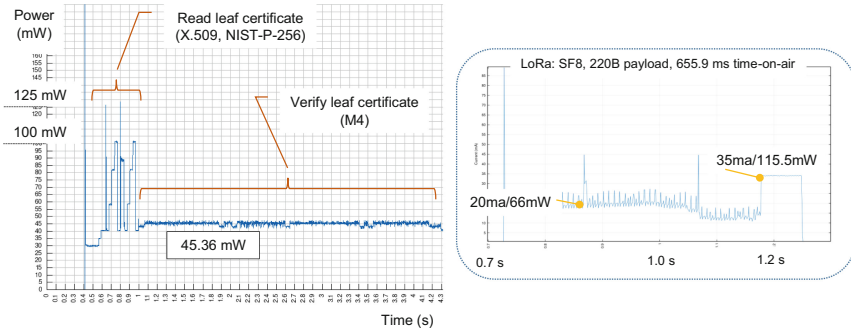


Fig. 5. Power consumption of STM32WL55JC1, for ARM Cortex-M4 (at 48 MHz) computing STM manufacturer provided X.509 certificate, ECDSA signature, and a single LoRa packet Tx of 222 bytes; measurement are captured via external INA219-based board sensor.

By examining memory needs, we observe that post-quantum flash requirements can increase by more than 10 times the size of pre-quantum flash. The requirement for stack memory is also significantly increased by post-quantum techniques. As a result, switching to post-quantum signatures necessitates an increase in memory (stack and flash) and bandwidth (for keys and signatures).

4 Conclusion

In this paper, we show that quantum-resistant solutions in IoT-based edge and fog computing paradigms with resource-limited industrial IoT devices are future-proof and post-quantum security is cost-effective and controllable. Due to its anti-quantum attack properties and shorter, quicker calculation processes, FrodoKEM, a Ring-LWE based encryption method that relies on hard problems on the lattice, is practically realized for data security between IoT nodes. We demonstrated an efficient realization of FrodoKEM640 with a reduced memory footprint to fit the STM32WL55JC1 ARM Cortex-M4 microcontroller board with just 64KB of RAM, to make the point that lattice-based cryptography can practically enhance security of resource-limited IoT devices. Even though finely-tuned version of the algorithm (i.e., written in assembly), which performs better, or hardware-accelerated functions (e.g., integrated hardware AES), or other

optimized method (e.g., SABER [4]) can be employed, ease of realization, fast-time-to market and development effort are essential factors in industry. These options are included in our future plans.

References

1. Albrecht, M.R., et al.: Classic mceliece, merger of classic mceliece and nts-kem, May 2021. <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions>
2. Alkim, E., et al.: Frodokem learning with errors key encapsulation. NIST, Gaithersburg, MD, USA, Technical report (2021). <https://frodokem.org/files/FrodoKEM-specification-20210604.pdf>
3. Asokan, N., Nyman, T., Rattanavipanon, N., Sadeghi, A.R., Tsudik, G.: Assured: architecture for secure software update of realistic embedded devices. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **37**(11), 2290–2300 (2018). <https://doi.org/10.1109/TCAD.2018.2858422>
4. Barton., J., Buchanan., W., Pitropakis., N., Sayeed., S., Abramson., W.: Post quantum cryptography analysis of TLS tunneling on a constrained device. In: Proceedings of the 8th International Conference on Information Systems Security and Privacy - ICISSP, pp. 551–561. INSTICC, SciTePress (2022).<https://doi.org/10.5220/0010903000003120>
5. Bos, J., et al.: Frodo: take off the ring! Practical, quantum-secure key exchange from LWE. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1006–1018. CCS '16 (2016). <https://doi.org/10.1145/2976749.2978425>
6. Chen, C., et al.: NTRU (2020). <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions>
7. Chen, Q.A., Qian, Z., Mao, Z.M.: Peeking into your app without actually seeing it: UI state inference and novel android attacks. In: 23rd USENIX Security Symposium (USENIX Security 14), pp. 1037–1052. USENIX Association, San Diego, CA, August 2014. <https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/chen>
8. Community, U.: Uptane standard for design and implementation v1.2.0. <https://uptane.github.io/papers/uptane-standard.1.2.0.html>
9. Crockett, E., Paquin, C., Stebila, D.: Prototyping post-quantum and hybrid key exchange and authentication in tls and ssh. *Cryptography ePrint Archive*, Paper 2019/858 (2019). <https://eprint.iacr.org/2019/858>
10. D’Anvers, J.P., et al.: Saber (2020). <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions>
11. Dhobi, R., Gajjar, S., Parmar, D., Vaghela, T.: Secure firmware update over the air using trustzone. In: 2019 Innovations in Power and Advanced Computing Technologies (i-PACT), vol. 1, pp. 1–4 (2019).<https://doi.org/10.1109/i-PACT44901.2019.8959992>
12. Fernández-Caramés, T.M.: From pre-quantum to post-quantum IoT security: a survey on quantum-resistant cryptosystems for the internet of things. *IEEE Internet Things J.* **7**(7), 6457–6480 (2020). <https://doi.org/10.1109/JIOT.2019.2958788>
13. Hopkins, K., Bergquist, J., Ortner, B., Kröger, M., Wong, S.: Edge security challenges. *Kubernetes IoT Edge Working Group, Whitepaper* (2019). <https://github.com/kubernetes/community/tree/master/wg-iot-edge/whitepapers/edge-security-challenges>

14. Kornaros, G.: Hardware-assisted machine learning in resource-constrained IoT environments for security: review and future prospective. *IEEE Access* **10**, 58603–58622 (2022). <https://doi.org/10.1109/ACCESS.2022.3179047>
15. Lee, J., Kim, D., Lee, H., Lee, Y., Cheon, J.H.: RLizard: post-quantum key encapsulation mechanism for IoT devices. *IEEE Access* **7**, 2080–2091 (2019). <https://doi.org/10.1109/ACCESS.2018.2884084>
16. Liu, Z., Azarderakhsh, R., Kim, H., Seo, H.: Efficient software implementation of Ring-LWE encryption on IoT processors. *IEEE Trans. Comput.* **69**(10), 1424–1433 (2020). <https://doi.org/10.1109/TC.2017.2750146>
17. Mbakoyiannis, D., Tomoutzoglou, O., Kornaros, G.: Secure over-the-air firmware updating for automotive electronic control units. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 174–181. SAC '19 (2019). <https://doi.org/10.1145/3297280.3297299>
18. Ronen, E., Gillham, R., Genkin, D., Shamir, A., Wong, D., Yarom, Y.: The 9 lives of bleichenbacher's cat: new cache attacks on TLS implementations. In: *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 435–452 (2019). <https://doi.org/10.1109/SP.2019.00062>
19. Sajid, A., Abbas, H., Saleem, K.: Cloud-assisted IoT-based SCADA systems security: a review of the state of the art and future challenges. *IEEE Access* **4**, 1375–1384 (2016). <https://doi.org/10.1109/ACCESS.2016.2549047>
20. Schwabe, P., et al.: Crystals-kyber (2020). <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions>
21. Sepúlveda, J., Gross, M., Zankl, A., Sigl, G.: Beyond cache attacks: exploiting the bus-based communication structure for powerful on-chip microarchitectural attacks. *ACM Trans. Embed. Comput. Syst.* **20**(2) (2021). <https://doi.org/10.1145/3433653>
22. Yunakovsky, S.E., et al.: Towards security recommendations for public-key infrastructures for production environments in the post-quantum era. *EPJ Quantum Technol.* **8**(1) (2021). <https://doi.org/10.1140/epjqt/s40507-021-00104-z>
23. Zhang, X., Xiao, Y., Zhang, Y.: Return-oriented flush-reload side channels on arm and their implications for android devices. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 858–870. CCS '16 (2016). <https://doi.org/10.1145/2976749.2978360>



Evaluation of a Red Team Automation Tool in Live Cyber Defence Exercises

Hannes Holm and Jenni Reuben^(✉)

Division for Cyber Defense and C2 Technology, Swedish Defence Research Agency,
Olaus Magnus väg 42, Linköping, Sweden
{hannes.holm,jenni.reuben}@foi.se

Abstract. This paper presents an evaluation of the red team automation tool Lore in two live-fire cyber defense exercises (CDX). During the CDXs, Lore and manual “red” teams subjected 72 network security analysts (i.e., defenders; the “blue” side) to various threats such as software exploits and shell commands. Ten hypotheses related to how the actions by manual red teams and Lore are perceived and managed by the security analysts are examined. Evaluations were made by studying the subjective judgements of the analysts and by comparing the objective ground truth to their submitted incident reports. The results show that none of the null hypotheses could be rejected. In other words, the security analysts could not tell the difference between the actions made by the manual red team and those made by Lore, and their performance was similar regardless of the source of the threats.

Keywords: Red team automation · Cyber defence exercises · Network Security

1 Introduction

Offensive network cyber security assessments are useful for a range of purposes, in particular, to identify vulnerabilities in systems and to train cyber security analysts on how to best identify and respond to threats.

While offensive assessments are useful, they are costly, time consuming and personnel constrained. Enterprise-level network vulnerability assessments can require thousands of hours, and a large cyber defence exercise (CDX) such as Locked Shields¹ can involve hundred red team members (i.e., simulated threat agents) that each spend weeks to prepare and execute the CDX.

As a consequence, automated offensive assessments of networks and systems have become a trending research area [22]. This paper describes an evaluation of the automated red team tool Lore [8] from two live-fire CDXs.

While the costs to prepare and execute Lore scenarios is far cheaper than manual red team campaigns - it can take as little as a few minutes to prepare and

¹ CCDCOE, “Locked Shields” (2022), <https://ccdcoe.org/exercises/locked-shields/>.

execute a campaign that would take a hundreds of hours to manually conduct - it is unknown how it affects a CDX as observed by its security analyst participants. Ideally, a red team campaign conducted by Lore would be indistinguishable to that made by a manual red team, i.e., a sequence of actions that have been manually determined by red team members. This is summarized by the following research question:

Research Question (RQ): *Do the autonomous nature of Lore impact the security analysts' perception of cyber situations enacted in CDXs?*

Ten hypotheses [H1-H10] were formulated to answer the research question:

- **H1:** The perceived abilities of the threat agents depend on whether the threats are performed manually or autonomously.
- **H2:** The perceived realism of the cyber situations depends on whether the threats are performed manually or autonomously.
- **H3:** The perceived knowledge gained by the participants depends on whether the threats are performed manually or autonomously.
- **H4:** The perceived effort of the participants in understanding the threat agents' knowledge gathering activities depend on whether the threats are performed manually or autonomously.
- **H5:** The perceived effort of the participants in understanding the threat agents' attacks depends on whether the threats are performed manually or autonomously.
- **H6:** The perceived effort of the participants in understanding the threat agents' command and control mechanisms depends on whether the threats are performed manually or autonomously.
- **H7:** The perceived effort of the participants in understanding the threat agents' goals depends on whether the threats are performed manually or autonomously.
- **H8:** The perceived effort of the participants in understanding which resources are compromised depends on whether the threats are performed manually or autonomously.
- **H9:** The perceived effort of the participants in understanding which countermeasures are required depends on whether the threats are performed manually or autonomously.
- **H10:** The ability of security analysts to accurately identify and report red team actions depends on whether the threats are performed manually or autonomously.

These hypotheses were examined through experiments conducted during the CDXs Safe Cyber 2020 [20] and Safe Cyber 2022 [8]. The security analysts who have participated in these CDXs were subjected to campaigns performed by both Lore and manual red teams using a 2×2 experimental design. The hypotheses H1-H9 are analyzed using the data collected from a questionnaire, while H10 is studied by comparing the data collected from the security incident reports submitted by the analysts with that of the ground truth provided by Lore.

The remainder of the paper is structured as follows: Sect. 2 presents related work. Section 3 describes Lore. Section 4 describes Safe Cyber 2020 and Safe Cyber 2022. Section 5 describes the methodology of the study and Sect. 6 its results. Finally, Sect. 7 concludes the paper.

2 Related Work

In this section we describe two different groups of literature that are relevant to the scope of this work. Section 2.1 describes the literature that have empirically evaluated CDX participants' performances and Sect. 2.2 presents the literature on red team automation.

2.1 Evaluation of CDX

Henshel et al. [7] investigate the factors that impact the success of a CDX programme, where the success is measured in terms of team performance. The authors conducted an empirical evaluation study similar to ours in a live CDX called Cyber Shield 2015². The data for the evaluation comprises of responses from a pre-event survey, observer-assessment survey, chat logs and other recorded activities of the participants. Similar to our objective method to analyze the performances of the security analysts, the authors employed different timestamps from the reported activities of the participants to assess the team's performances. Their proficiency assessment consists of the following temporal dimensions: i) detection of an incident ii) obtaining team controller's approval ii) resolving an incident and iv) percentage of injects accurately identified according to the NSIT-inject category.

Having established an understanding of the relationship between proficiency in cyber security topics to task performances with the context of cyber security training, Abott et al. [1] extended their previous results to empirically study the factors that contribute to superior task performances within a CDX. However, unlike our aim to demonstrate the relationship between participants' performances and autonomously determined red team campaigns, the aim of their work is to lay the foundation for development of automated performance assessment.

Maennel et al. [12] attempt to devise a generic task performance model for common tasks that are related to cyber-security incident responder role. The idea is to use the generic model to assess the learning attained by the participants of a live CDX. They have empirically evaluated their model in the 2017 Locked Shields CDX. The model defines the relationships between different timestamps inferred from the logs that are captured when the participants perform tasks such as changing firewall configurations, etc. The proficiency of the participants in solving an incident is measured by taking the time intervals between different yet related time stamps. For e.g., the measure of a Blue Team's (BT)

² See the National Guard Bureau home page for further information, <https://www.nationalguard.mil/>.

task performance is the difference in time taken by the red team in successfully compromising the BT’s network to the time taken for the BT to restore their network’s essential digital services. The purpose of their evaluation study is to objectively determine the performances of the participants. In our analysis too, we study the objective actions of participants from their activity logs within the context of accurately reporting the cyber incidents.

Similarly, Mäss et al. [16] conceptualize a task performance model that interconnects various indicators of completion of tasks that are needed for certain cyber-security roles specified in the NIST cyber security competence framework³. The authors refer to the cyber security competence framework for mapping the tasks within CDX to certain skill sets and further connect the skill sets to certain roles in the framework. The authors further discuss various techniques for measuring participants’ task performances, the example measures include, i) participants’ choice of tools and programming languages and ii) Levenshtein distance between the performance of a trainee to the performance of an expert for a given task.

2.2 Automated Red Team Tools

A plethora of automated red team tools have been developed by industry and academia.

Most tools developed by the industry, such as APTSimulator, Atomic Red Team and Infection Monkey, provide low-effort execution and customization of realistic offensive techniques [22]. However, they at best automate sequences of actions selected based on boolean expressions, and at worst require manually executing each action.

Academics, on the other hand, typically focus on action selection through models trained using reinforcement learning (RL). RL can provide great results given large samples sizes. Samples are unfortunately difficult to obtain in the cyber security domain as existing valid data, such as observations from network vulnerability tests and real intrusions, are scarce, and creation of new data is costly (a single action can sometimes require hours to complete). As a consequence, many researchers have developed and employed network cyber operation simulators that try to mimic how threat agents would behave given different goals and network environments.

Dutta et al. [5] present a data-driven deep reinforcement learning (DRL) framework to learn defense countermeasures that dynamically adapt to evolving adversarial behaviors while minimizing loss of cyber system operations. To accomplish this, the authors train an adversary model and a defense model in a simulator. They formulate the cyber defender’s optimization problem using a Sequential Decision Process (SDP) with 17 states. Andrew et al. [2] present a similar work where the interaction between an adversary and a defender is modeled as a Bayesian optimization problem with weights computed using a simulator.

³ NIST, “NICE Framework Resource Center”, <https://www.nist.gov/itl/applied-cybersecurity/nice/nice-framework-resource-center>.

Miehling et al. [14] propose modeling the actions of the adversary using Bayesian attack graphs and apply dynamic programming to solve the optimal countermeasures that should be employed by the defender. The authors test their approach in a simulator built for their purpose. Sultana et al. [21] investigate the effectiveness of AI-empowered autonomous cyber attacks by training an adversary model in a simulator using two RL algorithms - Proximal Policy Optimization (PPO) and Deep Q-Network (DQN). The authors' found that while both the PPO and DQN algorithms showed promising results, the PPO agent was easier to train and more stable when applied to different scenarios.

While employing a simulator address the sample size issue, it is difficult to ensure its validity: there are a vast variety of different threat agent actions and a single bit can be the difference between a successful and failed exploit. Some papers propose handling this issue by conducting real adversary actions in virtual or emulated environments. Sarraute et al. [19] present a partially observable markov decision process (POMDP) model that includes 50 possible actions and train their model against 7 machines. Li et al. [10] propose extending the practical tool CALDERA [15] with an action selector trained through RL and the DQN algorithm. The authors train two example models using an environment based on Mininet switches and virtual machines. The first example had an action space of 10 and required that the model select a sequence of four actions. The second example had an action space of 16 and required that the model select a sequence of six actions. The first example required 4600 actions, and the second 25000 actions, to receive an accuracy deemed sufficient according to the authors. Hoang et al. [17] propose an automated penetration testing tool trained through RL and the Asynchronous Advantage Actor Critic (A3C) algorithm. The model has a state space consisting of five features and is able to automatically select server exploits available in the Metasploit framework against application servers identified using the tool nmap. The authors train the tool against five identical virtual machines especially configured to be very vulnerable. The results showed that their tool could exploit all the known vulnerabilities within an hour.

The tool Lore provides low-effort execution and customization of realistic offensive techniques, similar to the tools developed by the industry. Its actions are selected based on trained classifiers and regressors, similar to most academic efforts. As far as we know, Lore is the only automated red team tool that has been used to fully automate a CDX.

3 Lore

Lore is a tool that is able to automatically execute offensive network assessments [8]. An overview of Lore is presented in Fig. 1.

At its core, Lore is an OODA (Observe, Orient, Decide, Act) [13] loop that involves:

- creating logs during action execution (observe)
- processing logs to update the red team knowledge database (orient)

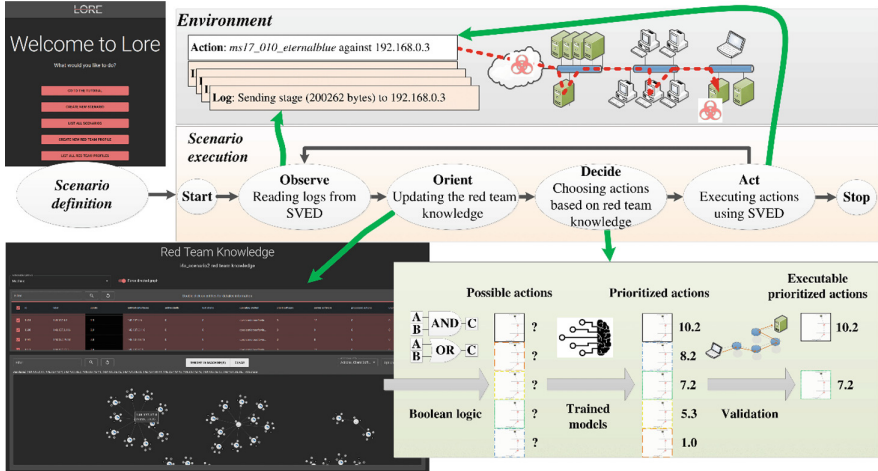


Fig. 1. Overview of Lore.

- choosing which actions to conduct based on the state of the knowledge database (decide)
- executing actions (act)

These steps are performed until the goals of the scenario have been fulfilled, or alternatively, until the action space has been exhausted.

A separate software tool called SVED (Scanning, Vulnerabilities, Exploits and Detection) [9] handles action execution and log creation. For example, executing a software exploit and reporting its result. During the execution of a scenario, Lore builds a red team knowledge database based on log data as well as any information manually given by its operators. This knowledge database is then queried to identify suitable actions. Actions are selected using a combination of boolean expressions and models trained through supervised learning. The action-space varies and it depends on the current red team knowledge. For example, three hours into the CDX Safe Cyber 2020, Lore had an action space of ~100 000, and a state-space of ~20 million [8].

Lore is capable of executing any action in SVED. At the time of writing this paper, this means 4728 different action types, for example, nmap, Virtual-box (e.g., restoring a snapshot), BloodHound, Responder, THC hydra, hashcat, CrackMapExec and any module in Metasploit. A subset of these (various action types corresponding to 68 different ATT&CK techniques) are automated by Lore’s default automation profile [8].

Lore reasons with incomplete information - the default case is that the only known information is the information gathered during the red team operation. Lore is usable with low overhead and is easily customizable. The process of choosing which actions to conduct is based on models trained through empirical tests in the cyber range CRATE [6]. Lore is capable of automatically navigating arbitrarily deep networks through pivot chains of compromised machines created based on network connectivity tests.

4 Studied Cyber Defence Exercises

This section describes Safe Cyber 2020 and Safe Cyber 2022, the two CDXs that provide the data analyzed in the paper. Both CDXs were organized by the Swedish Defence Research Agency (FOI) using the cyber range CRATE [6].

The primary purpose of these CDXs is to train security analysts on how to detect and report cyber threats and the secondary purpose is to examine the applicability of Lore. For the fulfillment of the later purpose, the participants were placed in two groups during each CDX, to conduct a 2×2 experimental designs that involved comparing how they reacted to the attack scenarios run by Lore as compared to those run by the manual red team. The employment of Lore was unknown to the participants - they were given the same background information and training as for a typical CDX.

The manual red teams also used SVED to execute all actions. Consequently, SVED contained complete details of all conducted threats during the CDXs. The CDXs emulated end-user behaviour through bots that performed simple actions such as browsing the web, opening files and sending email.

Safe Cyber 2020 [8] was carried out during March 16th to 18th 2021. The first day introduced the participants to the CDX, the second day executed it, and the third day involved conclusions and feedback. The CDX itself took place between 08:00 h and 17:00 h and involved 17 security analysts that were placed in two groups (the blue teams). Each group was tasked to defend 437 machines located on 21 network segments. These machines employed various operating systems and configurations, in particular, Windows 7 for office clients and Windows 2016–2018 for servers. The Lore scenario involved a threat agent located on the internet that exploited server vulnerabilities to pivot to internal networks, and remote access services such as ssh and psexec to move laterally. During Safe Cyber 2020, the manual red team conducted less than 100 actions, while Lore conducted thousands of actions. The manual red team compromised 10 machines against each group. Lore compromised 69 machines against the first group and 119 against the second group.

Safe Cyber 2022 [20] was carried out during September 20th to 22nd 2022. Similar to Safe Cyber 2020, the CDX took place during the second day (between 08:30h and 16:15h). It involved 55 security analysts placed in two groups (the blue teams), with each group tasked to defend a total of 477 machines. The machines employed similar but newer kinds of operating systems as was used during Safe Cyber 2020. For example, Windows 10 had replaced Windows 7 as the main office workstation image. Similar to Safe Cyber 2020, the Lore scenario involved a threat agent located on the internet. Dissimilar to Safe Cyber 2020, the threat agent pivoted to internal networks through machines compromised using USB drives rather than server exploits. It also employed more limited network scanning. During Safe Cyber 2022, the manual red teams conducted a total of 236 actions and compromised 19 machines, while Lore conducted 1890 actions and compromised 113 machines in total.

5 Data Collection Methodology

This section describes the data collection method employed during the CDXs described in Sect. 4 in order to answer the research question. Data for testing the hypotheses was acquired by means of the following two different collection instruments; i) similar to [3], a survey questionnaire collected the participants reactions during the CDXs to evaluate H1-H9 and ii) the cyber incident reports submitted by each group in order to evaluate H10.

An additional survey was employed during the Safe Cyber 2022 to collect expertise data about the CDX participants to provide context to the results presented in Sect. 6.1. The design of this survey was based on the work by Rajivan et al. [18].

5.1 Survey Questionnaire Design

Each of the hypotheses from H1 to H9 were realized as questions in the survey that measured the CDX participants perception of Lore. Given the Swedish context of Safe Cyber 2020 and Safe Cyber 2022, Swedish was employed for the survey questionnaire. A few example English translations of questions in the survey (realization of H1, H2 and H7) are given below:

- How you rate the skill level of the threat agent?
- How realistic did you find the cyber-attacks?
- How challenging was it to understand the threat agent’s goals?

5.2 Incident Reports

The security analysts were required to submit incident reports describing their judgments during both CDXs. The ground truth was obtained from CRATE (information about machines and networks) and SVED (information about conducted threats). The general incident reporting template used during the CDXs is described in [11]. Structured fields and formatting required to correlate incident reports to the ground truth given by SVED were, however, only present during Safe Cyber 2022. Consequently, the results regarding H10 is based on the incident reports gathered during Safe Cyber 2022.

Incident reports were corrected based on four fields:

- the reported time of the incident (date-time)
- the reported attackers (a list of hostnames, FQDNs or IPv4 addresses)
- the reported victim machines (a list of hostnames, FQDNs or IPv4 addresses)
- the type of incident (either shellcode, software exploit, online password guessing or network scanning)

Of these fields, respondents rarely employed other categories than shellcode (i.e., threat agent commands on remotely controlled machines). Thus, correcting a report essentially involved measuring if the victims and attackers mentioned

in it were true, given some acceptable error related to stated time of compromise. That is, analysts are seldom able to completely identify the actual time of compromise. In some systems, such as most mission-critical systems, an imprecise estimate of time of compromise can be devastating; in other systems, such as general office machines, it might be of little consequence. For this reason, reports were corrected given a range of acceptable errors related to reported time of compromise.

6 Results and Analysis

We employed a pair of statistical methods for analyzing the data from the survey questionnaire and the incident reports. We present the analysis and results from the survey questionnaire related to testing of the hypotheses H1 to H9 in Sect. 6.1. In Sect. 6.2, we report the analysis of the incident reports that is relevant to study the hypothesis H10.

Out of the 45 participants that responded to the survey concerning their cyber security expertise, responses of 2 participants were excluded due to bad input formats. The 43 included participants on average had 4 years of work experience in the information security field dealing with cyber security related problems on a day-to-day basis. Many of them (76%) had taken more than one formal course in information security subject and half of them were university graduates.

6.1 Analysis of Questionnaire

We received in total 90 responses to the questionnaire (see Sect. 5.1) from both the groups within Safe Cyber 2022. Half ($n = 45$) of these samples pertain to the participants that were subjected to Lore, and the other half ($n = 45$) pertains to the participants that were subjected to manual red team campaigns. In the case of Safe Cyber 2020, we received 34 responses in total for the questionnaire from the two groups.

As proposed by Marie et al. [4], in conditions where assumptions concerning homogeneity and variance are not met, Welch's t-test should be employed instead of Student's t-test. Accordingly, for the data analyses we conducted hypothesis testing using Welch's t-test method. Table 1 presents the t-test results for hypotheses H1 to H9 concerning the difference in participant's perceptions of manual red team's actions versus Lore's actions. As described by Table 1, the p-values for the hypotheses H1 to H9 are all greater than 0.05. Therefore, we fail to reject the null hypotheses w.r.t H1 to H9 - there are no differences in participants perceptions relating to realism, learning and understanding concerning cyber attacks enacted in live CDX regardless of whether the attacks are executed manually or autonomously. Table 1 also reports the mean and the variance of the sample data corresponding to H1 to H9 that are reported separately for two instances of Safe Cyber. Furthermore, the Confidence Interval (CI) of the estimated mean values are provided in Fig. 2. We present the 95% confidence interval of the estimated mean values in terms of Standard Mean Error (SME),

Table 1. Results from Welch’s t-test for H1 to H9 separated by the year that the instance of Safe Cyber was conducted.

Hypothesis	Safe Cyber 2020						Safe Cyber 2022					
	T-Test	p-value	Manual		Lore		T-Test	p-value	Manual		Lore	
			Mean	Variance	Mean	Variance			Mean	Variance	Mean	Variance
H1	0.23	0.82	4.29	2.22	4.53	2.12	0.11	0.92	4.95	1.89	4.93	1.42
H2	-1.33	0.20	5.24	1.44	4.47	1.39	0.73	0.47	4.34	2.66	4.6	2.23
H3	-0.06	0.95	4.47	1.76	4.71	2.10	-0.53	0.6	5.02	2.12	4.89	2.43
H4	0.25	0.81	3.40	3.11	3.6	2.83	-0.95	0.35	5.11	2.33	5.18	2.10
H5	-0.37	0.71	3.65	1.87	3.06	1.81	1.8	0.08	4.12	1.96	4.6	1.7
H6	0.12	0.91	3.21	1.41	3.06	3.13	-0.05	0.96	4.80	2.28	4.81	1.65
H7	-1.74	0.10	4.12	1.74	2.81	1.10	0.78	0.44	3.71	3.4	4.6	2.77
H8	0.67	0.51	4.06	1.43	3.82	1.15	0.56	0.58	3.95	2.2	4.31	2.76
H9	-1.25	0.22	4.53	2.89	4.12	1.49	0.28	0.78	3.7	2.56	4.2	2.25

thus the margin of error in the estimation is ± 2 SME. In Fig. 2, we report the resulting differences in the confidence interval of the estimated means when the participants experienced manual red team campaign versus Lore campaign. As seen in the figure, the confidence intervals of the two estimated means (i.e. of the manual red team vs Lore experiment group) overlap, thus indicating that there are no differences in the participants’ experiences when they were subject to manual red team as opposed to Lore. Due to space constraints, the Figs. 2a to 2f, show the differences in the confidence intervals of the mean values estimated for some of the studied hypotheses.

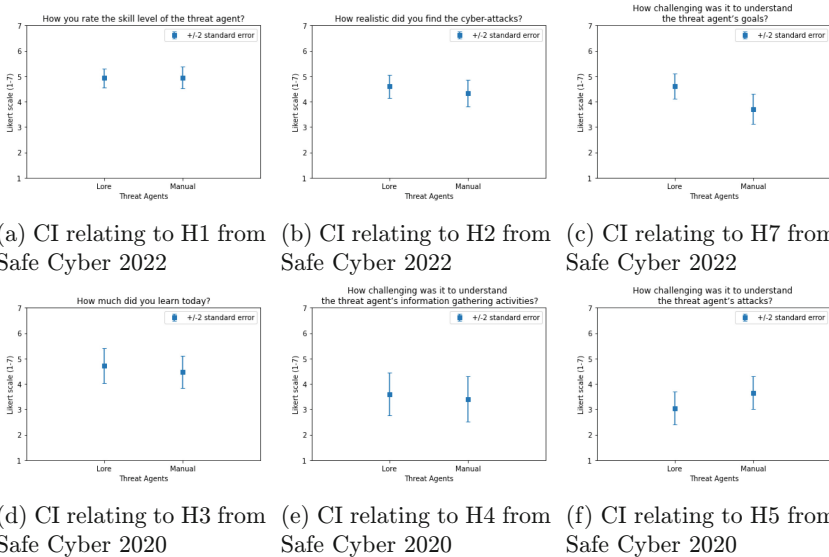


Fig. 2. Differences in the confidence intervals of estimated means of manual red campaign versus Lore campaign.

6.2 Analysis of Incident Reports

An overview of the results is presented by Table 2. In Table 2, “Error” refer to the tolerated analyst error measured as the reported time of detection minus the actual time of compromise. For example, an error of 20 means that a report stating that a machine was compromised at 15:50h only would be considered correct given that the machine indeed was compromised between 15:30h and 15:50h.

True positives (TP) are the number of correctly reported compromised machines. False positives (FP) are the number of incorrectly reported compromised machines. True negatives (TN) are the number of machines that were not compromised, and also not reported as such by the analysts. False negatives (FN) are the number of machines that were compromised, but the analysts failed to report. The total number of defended machines by each group was 477 (TP+FP+TN+FN).

Table 2. The results for four common metrics related to prediction quality, as well as their underlying convolutional matrices.

Group	Threat	Time	Reports	Error	Accuracy	f1 score	Precision	Recall	TP	FP	TN	FN
1	Lore	12–16	6	1	0.89	0.00	0.04	0.03	1	23	422	31
				20	0.89	0.00	0.04	0.03	1	23	422	31
				40	0.89	0.04	0.17	0.13	4	20	425	28
				–	0.91	0.13	0.29	0.22	7	17	428	25
	Manual	8–12	12	1	0.96	0.00	0.00	0.00	0	8	460	9
				20	0.97	0.11	0.25	0.22	2	6	462	7
				40	0.97	0.11	0.25	0.22	2	6	462	7
				–	0.97	0.11	0.25	0.22	2	6	462	7
2	Lore	8–12	6	1	0.83	0.08	0.44	0.09	7	9	387	74
				20	0.84	0.19	0.69	0.14	11	5	391	70
				40	0.84	0.19	0.69	0.14	11	5	391	70
				–	0.84	0.19	0.69	0.14	11	5	391	70
	Manual	12–16	12	1	0.96	0.00	0.00	0.00	0	11	456	10
				20	0.96	0.02	0.09	0.10	1	10	457	9
				40	0.96	0.02	0.09	0.10	1	10	457	9
				–	0.96	0.02	0.09	0.10	1	10	457	9

If including all reports regardless of the error, group 1 correctly identified 7 of the 32 machines compromised by Lore, and 2 of the 9 machines compromised by the manual red team; group 2 identified 11 of the 81 machines compromised by Lore, and 1 of the 10 machines compromised by the manual red team. As can be seen, while the analysts on overall produced more reports when faced by Lore (24 compared to 18), the primary factor was time: the groups produced 2–3 times more reports during the afternoon session than during the morning session. This is expected as the groups during the afternoon were more comfortable with their defended systems as well as the incident reporting tool.

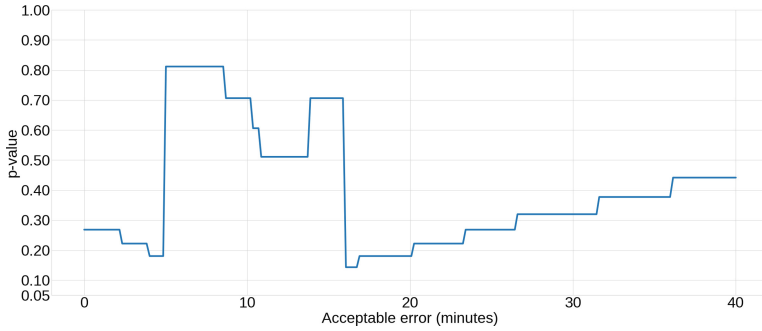


Fig. 3. p-values computed by Welch t-tests for correctness of reports produced by security analysts when faced by Lore and the manual red team.

Increasing the tolerated error has a positive effect on all metrics. The largest effect comes from raising the tolerated error from 1 to 20 min for all combinations apart from that of the first group being subjected to Lore. For some reason, the first group did a rather poor job at providing accurate time stamps for their reported intrusions. For three entries, their reported time of compromise was actually before the actual time of compromise. Based on the analysts opened feedback, this was likely due to human error when the corresponding reports were created.

In the context of H10: There are no clear differences for any of the metrics apart from accuracy: accuracy is similar for both the groups when subjected to the manual red team, and higher than when they were subjected to Lore. This is due to that the groups produced roughly the same amount of incident reports regardless of threat, even though the number of compromised machines significantly differed depending on the source of the threat. In fact, there is a 96% accuracy for both the groups when subjected to the manual red team even though they did not detect a single compromised machine (given a tolerated error of 1 min). None of the other metrics (f1 score, precision or recall) show any trend between Lore and manual red teams.

Figure 3 shows the p-values computed by Welch t-tests for correctness of reports produced by security analysts when faced by Lore and the manual red team given tolerated errors sampled in 5-minute intervals. The data in Fig. 3 was computed by grouping all reports based on threat source (24 reports for Lore and 18 for the manual red team) and scoring each submitted report as correct (1) or incorrect (-1) based on the designated tolerated error. A t-tests could thus be made for each sampled tolerated error.

As can be seen, the null hypothesis for H10 cannot be rejected regardless of tolerated error: the lowest observed p-value is 0.14, the highest is 0.81, and the mean 0.40. In other words, the analysts performed as well when subjected to Lore as when subjected to the manual red team.

7 Discussion and Conclusions

The results of this study indicate that the application of Lore do *not* impact security analysts' perception of cyber situations enacted in CDXs. All ten studied hypotheses could be rejected. There is no statistical difference between Lore and a manual red team campaigns with respect to analyst self-assessments (H1-H9) or the quality of their provided incident reports (H10).

There are, however, various threats to the validity and reliability of these results, and the analyses should be viewed with this in mind. The biggest issue is likely the small sample sizes. Among other things, this prohibited reliable analyses of the answers to the questionnaires for each cell of the 2×2 design in isolation. It is preferable to analyze each cell of a factorial design in isolation due to the possibility of a group changing its behaviour between sessions. For example, in the present research the analysts were better at responding to threats during the second session (see Sect. 6.2). Additionally, while the systems and software used for the studied CDXs were designed to mimic real-world systems, their implementation were simplified due to resource constraints. For example, many Windows "office" machines primarily differed in terms of user accounts. The lack of variety is a product of the significant costs required to create a realistic IT architecture for a CDX. Lastly, none of the participating analysts had previously employed the incident-reporting template in practice. In combination with the unrealistic time-constraints (three hour sessions), this likely resulted in fewer and less accurate reports than ideal.

The magnitude of these issues should decrease along future research on the topic, which we plan to continue whenever Lore is employed in the future. Finally, in addition to evaluating Lore, this paper also illustrate the great benefits of using an autonomous red team in a CDX. Apart from being far less expensive, it enables repeatable experiments with high-precision data that can be used to answer important research questions.

References

1. Abbott, R.G., McClain, J., Anderson, B., Nauer, K., Silva, A., Forsythe, C.: Automated performance assessment in cyber training exercises, p. 7
2. Andrew, A., Spillard, S., Collyer, J., Dhir, N.: Developing optimal causal cyber-defence agents via cyber security simulation. arXiv preprint [arXiv:2207.12355](https://arxiv.org/abs/2207.12355) (2022)
3. Bashir, M., Lambert, A., Guo, B., Memon, N., Halevi, T.: Cybersecurity competitions: the human angle **13**(5), 74–79. <https://doi.org/10.1109/MSP.2015.100>
4. Delacre, M., Lakens, D., Leys, C.: Why psychologists should by default use Welch's *t*-test instead of student's *t*-test. **30**(1), 92–101. <https://doi.org/10.5334/irsp.82>, <http://www.rips-irsp.com/articles/10.5334/irsp.82/>
5. Dutta, A., Chatterjee, S., Bhattacharya, A., Halappanavar, M.: Deep reinforcement learning for cyber system defense under dynamic adversarial uncertainties. arXiv preprint [arXiv:2302.01595](https://arxiv.org/abs/2302.01595) (2023)

6. Gustafsson, T., Almroth, J.: Cyber range automation overview with a case study of CRATE. In: Asplund, M., Nadjm-Tehrani, S. (eds.) NordSec 2020. LNCS, vol. 12556, pp. 192–209. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-70852-8_12
7. Henshel, D.S., et al.: Predicting proficiency in cyber defense team exercises. In: MILCOM 2016 - 2016 IEEE Military Communications Conference, pp. 776–781 (2016). <https://doi.org/10.1109/MILCOM.2016.7795423>
8. Holm, H.: Lore A Red Team Emulation Tool, p. 1. <https://doi.org/10.1109/TDSC.2022.3160792>
9. Holm, H., Sommestad, T.: SVED: scanning, vulnerabilities, exploits and detection. In: 2016 IEEE Military Communications Conference, pp. 976–981. IEEE (2016)
10. Li, L., Fayad, R., Taylor, A.: Cygil: a cyber gym for training autonomous agents over emulated network systems. arXiv preprint [arXiv:2109.03331](https://arxiv.org/abs/2109.03331) (2021)
11. Lif, P., Varga, S., Wedlin, M., Lindahl, D., Persson, M.: Evaluation of information elements in a cyber incident report. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 17–26. IEEE (2020)
12. Maennel, K., Ottis, R., Maennel, O.: Improving and measuring learning effectiveness at cyber defense exercises. In: Lipmaa, H., Mitrokotsa, A., Matulevičius, R. (eds.) NordSec 2017. LNCS, vol. 10674, pp. 123–138. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70290-2_8
13. McIntosh, S.E.: The wingman-philosopher of mig alley: John boyd and the ooda loop. *Air Power History* **58**(4), 24–33 (2011). <http://www.jstor.org/stable/26276108>
14. Miebling, E., Rasouli, M., Teneketzis, D.: Optimal defense policies for partially observable spreading processes on Bayesian attack graphs. In: Proceedings of the Second ACM Workshop on Moving Target Defense, pp. 67–76 (2015)
15. Miller, D., Alford, R., Applebaum, A., Foster, H., Little, C., Strom, B.: Automated adversary emulation: a case for planning and acting with unknowns. Technical report, MITRE CORP MCLEAN VA MCLEAN (2018)
16. Mäses, S., Hallaq, B., Maennel, O.: Obtaining Better Metrics for Complex Serious Games Within Virtualised Simulation Environments, p. 9
17. Nhu, N.X., Nghia, T.T., Quyen, N.H., Pham, V.H., Duy, P.T., et al.: Leveraging deep reinforcement learning for automating penetration testing in reconnaissance and exploitation phase. In: 2022 RIVF International Conference on Computing and Communication Technologies, pp. 41–46. IEEE (2022)
18. Rajivan, P., Moriano, P., Kelley, T., Camp, L.J.: What can johnny do?—Factors in an end-user expertise instrument. In: HAISA, pp. 199–208 (2016)
19. Sarraute, C., Buffet, O., Hoffmann, J.: Penetration testing== pomdp solving? arXiv preprint [arXiv:1306.4714](https://arxiv.org/abs/1306.4714) (2013)
20. Stupp, C.: Sweden tests cyber defenses as war and nato bid raise security risks. <https://www.wsj.com/articles/sweden-tests-cyber-defenses-as-war-and-nato-bid-raise-security-risks-11663925402>
21. Sultana, M., Taylor, A., Li, L.: Autonomous network cyber offence strategy through deep reinforcement learning. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III, vol. 11746, pp. 490–502. SPIE (2021)
22. Zilberman, P., Puzis, R., Bruskin, S., Shwarz, S., Elovici, Y.: Sok: a survey of open-source threat emulators. arXiv preprint [arXiv:2003.01518](https://arxiv.org/abs/2003.01518) (2020)



Automated and Improved Detection of Cyber Attacks via an Industrial IDS Probe

Almamy Touré^{1,2(✉)}, Youcef Imine¹, Thierry Delot¹, Antoine Gallais¹, Alexis Semmont², and Robin Giraud²

¹ Univ. Polytechnique Hauts-de-France, INSA Hauts-de-France, LAMIH, CNRS, UMR 8201, 59313 Valenciennes, France

almamytoure27@gmail.com

² IBM Security, Paris, France

Abstract. Network flow classification allows to distinguish normal flows from deviant behaviors. However, given the diversity of the approaches proposed for intrusion detection via IDS probes, an adequate fundamental solution is required. Indeed, most of existing solutions address a specific context which does not allow to assess the efficiency of the proposed models on a different context. Therefore, we propose in this paper an approach for malicious flow detection based on One Dimensional Convolutional Neural Networks (1D-CNN). Our solution extracts features based on the definition of network flows. Thus, it can be common to any network flow classification model. This feature engineering phase is coupled to CNN's feature detector in order to provide an efficient classification approach. To evaluate its performance, our solution has been evaluated on two different datasets (a recent dataset extracted from a real IBM industrial context and the NSL-KDD dataset that is widely used in the literature). Moreover, a comparison with existing solutions has been provided to NSL-KDD dataset. Attacks in both datasets have been defined using the globally-accessible knowledge base of adversary tactics and techniques MITRE framework. The evaluation results have shown that our proposed solution allows an efficient and accurate classification in both datasets (with an accuracy rate of 94% at least). Moreover, it outperforms existing solutions in terms of classification metrics and execution time as well.

Keywords: Intrusion Detection System · Deep Learning · MITRE ATT&CK · Features Engineering · Cyber attack

1 Introduction

Every day, new applications are connected to the Internet which increasingly complicates the management of the network in an information system traffic. Indeed, this growing number of applications diversifies the attack techniques

targeting information systems [1]. Collecting and analyzing network flows facilitates traffic management, ensures better network trending and helps to identify known intrusions and new ones [2]. Intrusion detection via IDS probes are mainly based on static correlation rules that are manually updated. Moreover, it usually does not take into account neither recent attack scenarios nor the detection of zero-day vulnerability exploitation attempts [3]. Consequently, organizations must now rely on dynamic security solutions to adapt to the changing nature of current attacks. The diversity of applications used and obfuscation techniques also limits the detection capabilities of IDS probes.

Although, the use of dynamic attack detection approaches with Machine Learning (ML) and Deep Learning (DL) algorithms has allowed to improve the identification and classification on these attacks [4]. However, the proposed solutions are not generalized because of the engineering of the features manner used for the detection and classification of attacks. Indeed, it is clearly noted in the literature that these characteristics vary from one solution to another and therefore unlikely to be reproduced elsewhere. In addition, the datasets used for the testing phases do not have a benchmark to better evaluate the accuracy of the proposed solutions.

In view of the mentioned improvement axes and weaknesses observed in the state of the art, in this paper we present as part of the improvement of an existing IDS solution at IBM, a new solution for malicious network flows detection and classification in which:

- we propose a standard and producible engineering of relevant and necessary feature classes that can be extracted and used in any network flow classifier,
- we propose a deep learning model that combines our feature extractor and the CNN's feature detector to achieve an efficient classification model,
- we perform an evaluation of our model on two datasets from different contexts while validating their content using MITRE ATT&CK framework,
- we propose a comparison between our model, machine learning approaches and deep learning solutions defined in the state of the art.

The rest of the paper is organized as follows. In Sect. 2, we discuss the recent intrusion detection works in the literature. We describe our proposed solution in Sect. 3 and present its performance evaluation results in Sect. 4. Finally, we conclude in Sect. 5.

2 Related Work

Intrusion Detection Systems (IDS) aim to detect malicious activities that could compromise a Host (HIDS) or a Network (NIDS) [5]. Obfuscation techniques such as encryption, steganography, tunneling, anonymization, mutation, morphing, physical obfuscation allow for better data protection of information system and harden the traditional methods of incident detection via network flow collectors [6]. Thus, the application of automatic and deep learning techniques allows better detection of intrusions.

Different research works have considered the use of ML techniques for intrusion detection including, e.g., Decision Trees [7], Random Forest [8], Naives Bayes (NB) [9] and Hidden Naïve Bayes (HNB) [10]. These different approaches aim to detect intrusions, by classifying network flows by scenario, via ML algorithms.

Deep Learning extends ML principles, relies on hidden layers to learn in depth and process information and allows to process large volumes of data. In [11], a sequential classifier for decreasing the false positive rate in this large amount of data to process is proposed. They used Artificial Neural Network (ANN) to reduce false positives and negatives. The accuracy was measured over the KDD99 dataset and varies according to the number of ANN classifiers chosen. In [12], an approach using several layers with hierarchy for complex feature extraction is defined to improve the effectiveness of Artificial Neural Networks (ANN) in NIDS. The solution can be evaluated by comparing traditional supervised ML classifiers and ANN techniques. However, the measured accuracy of the model varies highly upon application to two distinct datasets (KDDCup 99 and UNSW-NB15), hence the need to investigate more generic solutions able to homogeneously perform on a large variety of datasets.

Other approaches focus on the use of Recurrent Neural Networks (RNN), models adapted to sequential data. In [4], they propose an RNN model that aims to improve both the accuracy of intrusion detection and the ability to recognize the type of intrusion. The reduction of the learning time and the decrease of the overlearning, the optimization of the search for hyper-parameters can be posed as axes of improvement of the proposed approach, in order to improve the proposed algorithm and the accuracy rate.

Binary classification and multiclass classification, allow to outperform other deep learning (ANN) and ML approaches (J48, Random Forest, SVM), by reducing the learning time and the overlearning. Using such approach imposes to finely determine hyper-parameters in order to reach a good accuracy.

Deep neural networks based on a Bidirectional Long Short-Term Memory (BLSTM) can increase this accuracy by circulating the input data in both directions. In [13], the most optimal hyper-parameters are identified and tested on the CICIDS2017 dataset, while Random Forest and Principal Component Analysis (PCA) algorithms allow to select features. Feature selection is indeed a matter of prime importance. In [14], a two-layer approach achieves both a spatial and temporal feature extraction from raw data with Convolutional Neural Networks (CNN) and LSTM respectively. However, the accuracy rate decreases for unbalanced data.

A similar combination of CNN and LSTM model has been used to serialize TCP/IP packets in a predetermined time range as a traffic model in [15]. Normal and abnormal network traffic is categorized and labeled for supervised learning in 1D-CNN and applied to the UNSW-NB15 IDS dataset, using CUDA GPU acceleration to reduce time consumption. Combinations of methods can be effective, in the classification. Deep Learning has also been combined with binary algorithms (e.g., Binary Algorithm, Binary Genetic Algorithm, Binary Gravitational Search Algorithm, Binary Bat Algorithm) as optimizers in order

to increase the accuracy of detection and reduce the error rate [16]. Although inputs of the proposed hybrid IDS anomaly classification method are defined with eighty flow features from the CICIDS2017 dataset, the choice of these features remains specific to each situation.

Overall, our literature review emphasizes the lack of generic solutions to perform classification. Existing approaches remain specific to each dataset and so do the accuracy rate and the rate of false positives. The extraction and definition of features (feature extraction algorithms, extraction of spatial, temporal or vector features by deep learning) used by the proposed algorithms also highly depend on the targeted application case.

However, as observed, the solutions designed for a given dataset can hardly be transposed to others, especially given the constantly increasing diversity and complexity of cyber attacks. The MITRE framework allows to categorize the attack according to the Technical Tactics and Procedures (TTPs) their use [17]. This globally recognized knowledge base provides broad visibility into the detection perimeter. It also allows the evaluation of the dataset used and provides guidance for future research based on expanding IDS coverage. For all these reasons, we propose to use this framework to enhance the network flows we collected for further classification, thus easing the application of our approach in other situations.

3 Our Proposed Approach

In this section, we present our supervised learning approach for network intrusion detection. Network events in our approach are classified based on the features of network flows. Our approach uses CNN to achieve a fast and efficient classification and can be applied in any network context.

Thus, as we can see in the Fig. 1 the proposed approach can be broken down into three steps: the first part will deal with feature engineering which will be developed in Sect. 3.1 and will focus on the operation of convolution neural networks, in particular the feature detector and feature map proposed in this algorithm; the last step will highlight the classification phase of the flows for the identification of the adequate classes to which the flows are associated.

3.1 Our Feature Engineering

As shown in Fig. 2, a flow represents a communication session between two assets during a given time interval. Flow characteristics (e.g., IP addresses, ports, communication protocols) can be found in any network activity. We propose feature engineering based on the definition of a network flow. This aims at extracting from the raw data, relevant features useful for any network flow classification activity, reducing the processing time of a model to feed a network activity classification algorithm.

Our feature engineering aims at finding a set of universal minimalist features and facilitating several network flow analysis tasks. Thus, we believe it can be

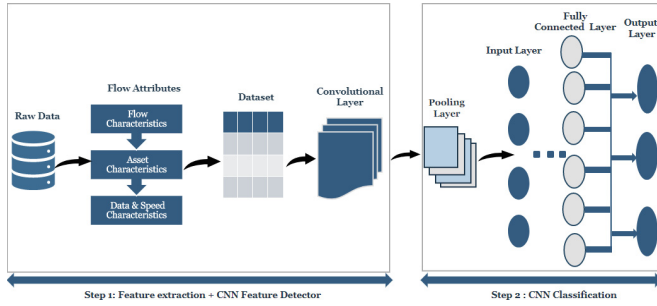


Fig. 1. Our proposed model.

used to classify network activity. As shown in Table 1, the features of any flow can be ranged in three main categories, namely:

- **Asset characteristics** that gather features related to Network, Transport and Application layers of the OSI model;
- **Flow characteristics** regroup features related to layer 5 (session) of the OSI model, including the type of session established, the direction and duration of the communication;
- **Exchanged data and rate characteristics** in which we find features related to exchanged contents (e.g., amount of communicating data).

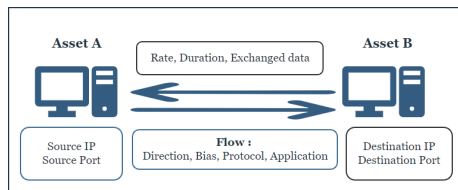


Fig. 2. Flow definition.

3.2 Our Model

Once the features are selected, the data of these features are injected into a classification algorithm. For this purpose, there are different ML and DL approaches and methodologies, as we have seen in the state of the art, most of which aim to improve the accuracy of the model. We have a part of the attack categories with similar feature values, which may decrease the models based on the ML algorithm and target an application of our model to an industrial context with high data volumes. Therefore, we propose a classification based on deep learning where some features are learned during the learning phase.

We choose CNN among other DL approaches because of its unique feature detector on the convolutional layer. The feature detector detects the features that

will match the highest values of the input data in the feature map. A convolution operation is performed between the input data (dataset) and the feature detector in order to determine this feature map that is fairly representative of the relevant data.

Our model aims at coupling this feature map formation process with the feature engineering proposed in Sect. 3.1 in order to define a robust method for classifying the network flows.

Table 1. Features family.

Flow Characteristics	
Flow-ID	Flow Type
Flow-Aggregation-Count	Flow Direction
Flow-Bias	Flow Duration
Asset Characteristics	
Source-IP	Source-Port
Destination-IP	Destination-Port
Protocol	Category
Application	Application-Group
Data and Speed Characteristics	
Source-Bytes	Destination-Bytes
Bit-Per-Second	Total Bytes

Thus, we propose a One-Dimensional CNN (1D-CNN) approach with parameters of 64 filters to extract 64 different features on the first convolution layer of the network and a kernel-size of 3 (3×3 matrix). This size corresponds to the size of the feature detector that allows to determine the number of feature maps created in our convolution layer. In this layer, we use the ReLu function to add non-linearity in our model. The max-pooling ensures a spatial invariance property and decreases the risk of overlearning by removing the unimportant information and keeping only the most relevant ones to generalize the model. Thus, we have chosen a value of two for the pool-size [18]. And, we add a pooling layer with two pool-size, followed by a flattening phase applied in all previously defined pool features maps. The fully connector layer is added with 128 units corresponding to the average number of neurons in the input and output layers, while maintaining the ReLu function. For the output layer, we give as value to the parameters eight output neurons that correspond to the classes of attacks present in our model. We rely on the cross-entropy cost function for this network flow classification problem, with an Adam optimizer, and accuracy to measure the performance of our model. The activation function Softmax evaluates the output probabilities of each class.

Table 2. Our CNN parameters.

Parameter	Value
Convolution Layer/Max-pooling	3 layers/2 layers
Dropout	0.3
Fully Connected	2 layers
Output Layer/Optimizer	Softmax/Adam
Activation Function	ReLu and Softmax
Epoch/Batch Size	200/3

In order to optimize the model, we will make it deeper while taking into account that the complexity of the model increases the computation time. Thus, we can add layers at the convolution layer or the fully connected layer or both layers. We define a loop that adds three convolution layers, and do max-pooling on these new layers while maintaining the same parameters as in the first convolution layer. However, to minimize the over-training, we will use dropout to disable some neurons. Thus, a probability of 0.3 is considered in our model to disable the neurons. Table 2 summarizes the different settings of our proposed convolution neural network model.

4 Evaluation - Results

In this section, we will evaluate the accuracy of our model in network flow classification. To do so, we first apply our feature engineering coupled with our CNN-based model on an IBM dataset extracted from a real industrial context. We compare then our solution to ML algorithms in terms of binary and multi-class classification. Since we aim to propose a general solution for network flow classification, we show, in the second part of this section, the effectiveness of our solution against existing solutions while considering, this time, the well known NSL-KDD dataset that is widely used in the literature.

4.1 IBM Dataset

IBM QRadar is a security appliance that is built on Linux. QRadar allows to collect, store and correlate logs for the detection of security incidents. To prove the effectiveness of our approach, we extracted raw data from a real-time industrial context, with an IBM proprietary IDS probe (QRadar Network Insight, QNI), which extends QRadar by providing a detailed view of real-time network communications [19].

Our dataset includes data collected in 2021, and thus recent attack chains (e.g., exploit log4Shell vulnerability). The flows used in our classification have been extracted during an interval of one week. This extraction contains legitimate flows and non-legitimate ones that we categorize using the framework MITRE ATT&CK [20]. This content is presented in table 3.

Table 3. IBM Dataset content validated with MITRE ATT&CK.

Flow-class	Target-ID	Tactics	Techniques	Size (#rows)
Normal	0	–	–	10.000
Large-Leakage	2	TA0010	T1567	10.000
Stealthy-leakage	1	TA0010	T1030	10.000
Web-Exploit	7	TA0002	T1204	5.000
DOS-Attack	3	TA0040	T1498	150
Indicator of Compromise Inbound	4	TA0001	T1189	500
Indicator of Compromise Outbound	5	TA0011	T1102	300
Malicious-Website	6	TA0011	T1102	15

4.2 Preprocessing

In this phase, we make transformations on our raw data to allow its processing by the learning algorithms that we used. The raw data has an average of 150 features by default. However, using all these features for classification is not an optimal approach. Therefore, in our solution, we reduce the number of features to 14 (at most) according to our feature engineering approach (Sect. 3.1) in addition to one target that describes the flow class.

After this extraction, all decimal entries (e.g., rate, amount of data, source/destination port numbers) remain unchanged. On the other hand, each byte in a field related to an IP address is converted to its hexadecimal value (excluding the dots that separate the bytes). After that, these values are concatenated which gives us a unique value that will be converted to decimal. The fields related to “flow direction” has at most four possible values in the raw data. Therefore, these values are mapped into integers in the set $\{1, \dots, 4\}$. The fields related to “flow bias” has at most five possible values in the raw data. Therefore, these values are mapped into integers in the set $\{1, \dots, 5\}$. Finally, the type of the communication protocol used is replaced by its corresponding number assigned by the Internet Assigned Numbers Authority (IANA) (i.e., 6 and 17 for TCP and UDP respectively).

4.3 Results

To determine the quality of our classification model and to evaluate its performance against various existing learning algorithms, we used the following four popular evaluation metrics:

$$Accuracy = \frac{(TP+TN)}{(FP+FN+TP+TN)}, \quad Precision = \frac{TP}{TP+FP}, \quad Recall = \frac{TP}{TP+FN},$$

$$ScoreF1 = 2 * \frac{(Precision*Recall)}{(Precision+Recall)}$$

where, for a given flow class C : True Positive (TP) represents the number of flows correctly classified in the given class C ; True Negative (TN) represents

the number of flows correctly classified outside the class C ; False Positive (FP) represents the number of flows wrongly classified in the class C ; and finally False Negative (FN) represents the number of flows wrongly classified outside the class C .

4.3.1 The Evaluation of Binary and Multi-class Classification Using Two-Feature Families As a first step to validate our approach, we started with a binary classification which should allow us to distinguish legitimate flows from illegitimate ones. To do so, we first assign all legitimate flows (Normal flows in Table 3) to the target 0 while all illegitimate flows will be assigned to the target 1. After that, we select ten features in each entry in our dataset. These features correspond to two features families among the three highlighted in Sect. 3.1, namely, Flow features (the number of records in a flow, the type of flow, the direction of the asset that initiated the communication, the duration of the communication, the data transfer bias) and Asset features (the source IP address, the destination IP address, the source and destination ports, and the protocol used). This data has then been injected into our model and the results have been compared in Table 4 with those of four ML algorithms (K-Nearest Neighbors (KNN), Naive Bayesian (NB), Random Forest (RF), Decision Tree (DT)).

Table 4. A comparison table in terms of binary classification using 2 families of features.

T	KNN			NB			DT			RF			Our model		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
0	1	1	1	1	0.99	0.99	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Acc	100%			99%			100%			100%			100%		

As we can see in the results of Table 4, our solution can indeed distinguish legitimate network traffic from an abnormal one, where we notice of a 100% accuracy rate for our model. The other ML algorithms also achieve good results except for Naive Bayesian that has 99% accuracy and which is more likely due to its probabilistic nature. However, in a real industrial context, it is also important to distinguish the nature of the attack flows in order to take proper counter-measurements. Therefore, we present in Table 5 a comparison of the results of a multi-class classification of illegitimate flows between our model and the above ML algorithms.

Table 5. Multiclassification with 2 families of features.

T	KNN			NB			DT			RF			Our model		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.74	0.76	0.75	0.94	0.15	0.26	0.70	0.69	0.69	0.71	0.75	0.73	1	0.96	0.97
2	0.75	0.74	0.75	0.55	0.79	0.71	0.68	0.69	0.68	0.72	0.68	0.70	1	0.92	0.94
3	1	1	1	1	1	1	1	1	1	1	1	1	1	0.95	0.92
4	0.83	0.31	0.45	0.40	0.67	0.50	0.93	1	0.96	0.93	1	0.96	0.91	0.91	0.94
5	1	1	1	1	0.45	0.62	1	0.86	0.93	1	0.91	0.94	1	0.96	0.96
6	1	1	1	1	1	1	1	1	1	1	1	1	0.85	0.89	0.92
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Acc	88%			79%			85%			86%			92%		

As shown in Table 5, we notice that our solution offers a better accuracy and a very good precision, recall and F1 scores in all attack targets. As a reminder, the F1 score is an arithmetic average between precision and recall, so it will allow us to measure the ability of the learning algorithms to better associate flows with legitimate traffic classes or corresponding target attacks.

The first general observation is a confusion between targets 1 and 2 in almost all ML algorithms and a better classification for targets 3, 6 and 7 compared to the remaining targets. We note that targets 1 and 2 correspond to data leak flows including data leaks in short time intervals, and data leaks in much longer ones. For these flow types, the same IP address can be spotted in both scenarios, which may cause the confusion between the data in these two classes in ML algorithms. In KNN, the F1 score is 0.75 for target 1 and 2, and quite low, especially for target 4. In the NB, we notice the lowest accuracy among all ML algorithms that we tested, which strongly impacts the F1 score which is 0.26 for target 1. A significant confusion was also found in the NB between targets 4 and 5 (corresponding to the inbound and outbound compromise indicator data) with F1 scores of 0.5 and 0.62 respectively. This is explained by the fact that the flow duration in these two attack classes is relatively close, due to the probabilistic operation of NB. The DT and RF improve these F1 scores at target 4 and 5 since when an attribute has the same approximate value for two given classes in an internal node, these two algorithms dissociate the given targets in the following nodes by comparing other attributes.

Our solution, which combines our feature extraction approach with the feature detector proposed in CNN, solves the confusion observed in ML algorithms. Indeed, a neural layer overlay, including the layer filtering principle, is well adapted to our context of multi-class classification of confused output layers. Convolutional neural networks learn the values of the weights in the same way that they learn the filters of the convolution layer, which can allow it to distinguish between confused classes 1 and 2, and other classes (like target 4 and 5) with medium or low F1 scores which explains the improvement of accuracy highlighted in Table 5.

4.3.2 The Evaluation of Multi-class Classification Using Three-Feature Families In order to investigate the impact of the third family of features (highlighted in Sect. 3.1) on the accuracy of our model, we select the minimalist features mentioned above, the characteristics related to “Data and Speed” in each entry considered in our dataset (e.g., the number of bytes sent by the source, the number of bytes received, etc.). Therefore, the number of characteristics increases from 10 to 14 features that we use in both our model and ML algorithms to classify malicious network flows. The results of this evaluation are presented in Table 6.

As we can observe in Table 6, an improvement of the F1 score of all targets is noticed compared to the Table 5 in both our model and ML algorithms, with an exception for NB. In this model, the F1 scores are more or less the same as the ones in Table 5 for all targets, except for target 4 (incoming traffic related to indicators of compromise) where a sharp decline is observed. This decreases the accuracy of this model from 79% to 73%.

Overall, we can clearly deduce that the additional information provided by the third family of features gives much more context and allows us to dissociate certain types of attacks. Our solution appears as the best approach with an average F1-score of 0.96 and an overall accuracy of 94.84% which is better than the ones presented in Table 5.

Table 6. Multiclassification with of all families of features.

T	KNN			NB			DT			RF			Our model		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.80	0.78	0.79	0.93	0.17	0.28	0.86	0.85	0.85	0.86	0.91	0.88	1	0.95	0.96
2	0.78	0.81	0.80	0.48	0.74	0.58	0.84	0.85	0.85	0.90	0.94	0.87	0.98	0.98	0.95
3	1	1	1	1	1	1	1	1	1	1	1	1	1	0.91	0.97
4	1	0.60	0.75	0.02	0.82	0.03	1	0.95	0.97	96	0.95	0.96	0.95	0.90	0.89
5	1	1	1	1	0.44	0.62	1	0.73	0.84	0.95	0.82	0.88	1	1	1
6	1	0.67	0.80	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Acc	90%			73%			91%			92%			94.84%		

4.3.3 The Evaluation of Epochs According to Accuracy and Loss Values The convergence of a model is determined by the analysis of the error rate and accuracy curves. Thus, we will focus on determining the optimal number of epochs in order to find a trade-off between the execution time of our CNN model and its overall accuracy because one of the drawbacks of deep learning based solutions is the execution time. In Table 2, we have initially defined a baseline value of two-hundred epochs which we believe is more than enough for our model to converge. However, we went through the **earlyStopping** technique to explore the possibility of stopping the learning, as soon as the model starts to

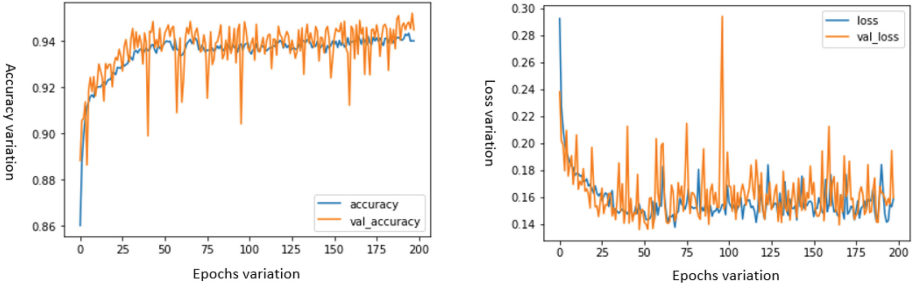


Fig. 3. Number of epoch with accuracy and loss

overlearn and thus to reduce the execution time. To do so, we present in Fig. 3, the accuracy and loss in the training data (blue curve) and in the test data (orange curve) according to the number of epochs used in our model.

As we can see in Fig. 3, from the first to the fiftieth epoch, the accuracy increases with the number of epochs and the loss decreases accordingly, which indicates that the model is still continuing to learn and converges. Between the fiftieth and seventieth epochs, both metrics start to stagnate. This stagnation continues until the 200th epoch, with some fluctuations observed in some epochs due to confusions between some neurons.

4.4 A Comparative Analysis Using the Benchmarking Dataset NSL-KDD

There have been many intrusion detection approaches proposed in the literature. However, these approaches are evaluated in different datasets. Therefore, we tested our model on the NSL-KDD dataset (that has been used in several works of the literature) in order to check its efficiency and establish a fair comparison with existing solutions that adopted the same dataset.

The NSL-KDD dataset [21] is an enhanced version of the KDDCup99 dataset [22] which is a standard dataset consisting of a wide variety of simulated intrusions in a military network environment, defined in 1998 by “MIT Lincoln Labs” of the U.S. DARPA agency. NSL KDD has a KDDTrain+ set (125,973 records) divided into 22 types of attacks and normal traffic and a KDDTest+ test set (22,544 records) which contains 14 additional attacks not included in the KDDTrain+. The set of attack classes is divided into four main families: Denial of Service (DOS), Remote To Local (R2L), User To Root (U2R), Probe. The content of NSL-KDD is presented in the Table 7.

NSL-KDD dataset provides 41 features grouped into 3 families: basic features, traffic features to the same host and traffic features to the same service. However, the application of our feature engineering (proposed in Sect. 3.1) will allow us to reduce the number of features used for training to the following 12 features (excluding labels): **duration**, **protocol-type**,

Table 7. The NSL-KDD dataset content.

Flow-class	Tactics	Techniques	Train(#rows)	Test(#rows)
Normal	–	–	67 343	9 711
DOS	TA0040	T1498	45 927	7 458
R2L	TA0001	T1133/T1199	995	2 754
U2R	TA0004	T1548	52	200
Probe	TA0043	T1589/T1590	11 656	2 421

service, land, src-bytes, dst-bytes, count, srv-count, same-srv-rate, srv-diff-host-rate, wrong-fragment and flag.

Moreover, to reduce the impact of inconsistencies between the training and test data in terms of targets, we chose to concatenate the two by keeping only 22 targets common to both sets, then splitting this concatenated set at 80% for the training sets and 20% for the test data. Then, we proceeded to a preprocessing and normalization of the extracted data, in which all numerical values are kept unchanged while attributes with categorical values like services and protocol-type have been mapped to decimal values, as it has been done with IBM dataset. Finally, targets have been labeled as follows: normal:0, DOS:1, Probe:2, R2L:3, U2R:4; and the resulting data is injected into our model. The results of the tests of our model on the NSL-KDD are presented in the Table 8.

Table 8. The evaluation results of our Model on NSL-KDD dataset

Target	Our model (NSL-KDD)		
	Precision	Recall	F1-score
Normal	0.99	0.99	0.99
DOS	0.99	0.99	0.99
Probe	0.97	0.97	0.97
R2L	0.94	0.79	0.86
U2R	0.50	0.13	0.21
Accuracy	98,7%		

Overall, as we can see in Table 8, our model achieves a global accuracy of 98%. We also notice a high precision and a good F1-Score for the targets “Normal”, “Probe” as well as “DOS” and to a lesser extent “R2L”. However, the target “U2R” has a low score compared to the remaining targets. Actually, the “U2R” target relates to an attempt to elevate privileges which does not correspond to the definition of network stream. Thus, its classification cannot rely solely on the characteristics of a network flow, which explains in addition to the low number of inputs of that target the gap with the other targets that are in fact network flows and have many more entries in the dataset.

Moreover, based on the NSL-KDD dataset, we compare in Table 9 our model with other solutions proposed in the state of the art according to the confusion matrix (precision, recall, F1-score and accuracy). Note that the chosen solutions use the following algorithms on NSL-KDD dataset: ANN [12], Deep learning with ANN (Auto-encoder) [23] and RNN+LSTM, CNN [24].

Table 9. Comparison with the state of art based on NSL-KDD dataset

Model	Accuracy	Precision	Recall	F1-score
Our model	0.98	0.88	0.78	0.80
R. Vinayakumar et al. (2019) [12]	0.785	0.810	0.785	0.765
C. Zhang et al. (2019) [23]	0,7974	0,8222	0,97974	0,7647
Z. Tauscher et al. (2021) [25]	0,8047	N/A	N/A	0,7839
L. Liu et al. (2020) [24]	0,7824	0,7838	0,7823	0,7503

As we can see in Table 9, our solution significantly enhances the results of the existing solutions especially in terms of classification accuracy. In addition, it reduces the preprocessing time applied to the raw data compared to some existing solutions. For instance, in [23], the authors run a deep learning algorithm to extract the most relevant features for classification, which is a time consuming task compared to our preprocessing phase that yet allows to achieve better classification results. Moreover, our solution also reduces the model execution time compared to the remaining solutions, since it uses less features for the training (12 in ours, 122 in [23], 41 in [12, 24, 25]).

5 Conclusion

In this paper, we propose an intrusion detection solution for network flow collectors using 1D Convolutional Neural Networks. First, our solution presents a feature engineering for the extraction of data features according to the definition of a network flow. The proposed engineering can be used in any network flow classification, since it is based on criteria that can be identified in any network event. Our feature extraction is then associated with a feature detector functionality defined in CNN, which guarantees an accurate determination of normal network flows and a robust multi-class classification of malicious ones. Our model has been evaluated on two datasets, the first one has been extracted from a real IBM industrial context while the second one is the public dataset NSL-KDD. Both sets were validated using the MITRE ATT&CK framework, and the results have proved that our model distinguishes normal behaviors from deviant ones, and efficiently identifies the attack classes. Moreover, our model significantly improves the accuracy of the classification process when it is compared to other existing solutions of the state of the art. It also reduces the number of extracted features and thus the execution time of the global model compared to existing

solutions. In the future, we intend to focus on zero-day vulnerability exploitation attacks. We also plan to work on a data generator composed of recent attack families, classified by the MITRE ATT&CK framework, given the obsolescence of some data sets.

References

1. Lin, P., et al.: A novel multimodal deep learning framework for encrypted traffic classification. *IEEE/ACM Trans. Network.* **31**, 1369–1384 (2022)
2. Zhu, X., et al.: Machine-learning-assisted traffic classification of user activities at programmable data plane. In: 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS) (2022)
3. Xin, S.: Research of intrusion detection system. In: International Conference on Computational and Information Sciences, pp. 1460–1462 (2013)
4. Yin, C., et al.: A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **5**, 21954–21961 (2017)
5. Kılıç , H., et al.: Evasion techniques efficiency over the IPS/IDS technology. In: 4th International Conference on Computer Science and Engineering (UBMK), pp. 542–547 (2019)
6. Salman, O., et al.: A review on machine learning-based approaches for Internet traffic classification. *Ann. Telecommun.* **75**(11), 673–710 (2020)
7. Jabbar, M.A., et al.: Intelligent network intrusion detection using alternating decision trees. In: International Conference on Circuits, Controls, Communications and Computing (I4C) (2016)
8. Sharmila, B.S., et al.: Intrusion detection system using naive bayes algorithm. In: IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE) (2019)
9. Meena, G., et al.: A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. In: International Conference on Computer, Communications and Electronics (Comptelix), pp. 553–558 (2017)
10. Koc, L., et al.: Network intrusion detection using a HNB binary classifier. In: 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim), pp. 81–85 (2015). <https://doi.org/10.1109/UKSim.2015.37>
11. Varanasi, V., et al.: Network intrusion detection using machine learning, deep learning - a review. In: 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 1618–1624 (2022)
12. Vinayakumar, R., et al.: Deep learning approach for intelligent intrusion detection system. *IEEE Access* **7**, 41525–41550 (2019)
13. Sivamohan, S., et al.: An effective recurrent neural network (RNN) based intrusion detection via bi-directional long short-term memory. In: International Conference on Intelligent Technologies (CONIT) (2021)
14. Wang, W., et al.: HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* **6**, 1792–1806 (2018)
15. Azizjon, M., et al.: 1D CNN based network intrusion detection with normalization on imbalanced data. In: International Conference on Artificial Intelligence in Information and Communication (ICAIC), pp. 218–224 (2020)

16. Atefi, K., et al.: A hybrid anomaly classification with deep learning (DL) and binary algorithms (BA) as optimizer in the intrusion detection system (IDS). In: 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), pp. 29–34 (2020)
17. Rajesh, P., et al.: Analysis of cyber threat detection and emulation using MITRE attack framework. In: International Conference on Intelligent Data Science Technologies and Applications (IDSTA), pp. 4–12 (2022)
18. Zheng, W.-F.: Intrusion detection based on convolutional neural network. In: International Conference on Computer Engineering and Application (ICCEA), pp. 273–277 (2020)
19. Sekharan, S.S., et al.: Profiling SIEM tools and correlation engines for security analytics. In: International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 717–721 (2017)
20. MITRE ATTA&CK. <https://attack.mitre.org/>
21. Tavallae, M., et al.: A detailed analysis of the KDD cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009)
22. Shah, B., et al.: Reducing features of KDD cup 1999 dataset for anomaly detection using back propagation neural network. In: 2015 Fifth International Conference on Advanced Computing & Communication Technologies, pp. 247–251 (2015)
23. Zhang, C., et al.: A deep learning approach for network intrusion detection based on NSL-KDD dataset. In: 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), pp. 41–45 (2019)
24. Liu, L., et al.: Intrusion detection of imbalanced network traffic based on machine learning and deep learning. *IEEE Access* **9**, 7550–7563 (2021)
25. Tauscher, Z., et al.: Learning to detect: a data-driven approach for network intrusion detection. In: 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), pp. 1–6 (2021)



An Accurate and Real-Time Detection Method for Concealed Slow HTTP DoS in Backbone Network

Jinfeng Chen¹, Hua Wu^{1,2(✉)}, Suyue Wang¹, Guang Cheng^{1,2},
and Xiaoyan Hu^{1,2}

¹ School of Cyber Science and Engineering, Southeast University, Nanjing, China
{chenjf, hwu, wangsuyue, chengguang, xyhu}@seu.edu.cn

² Jiangsu Province Engineering Research Center of Security for Ubiquitous Network,
Nanjing, China

Abstract. Slow HTTP DoS (SHD) is a type of DoS attack based on HTTP/HTTPS. SHD traffic at the application layer may be encrypted. Besides, the interval between packets can reach tens of seconds or more due to its slow sending rate. Therefore, SHD is concealed for detection. The methods for detecting high-speed DoS are not suitable for detecting the attack, making detection for SHD a challenging problem. Some existing SHD detection methods are complex and computationally intensive, making it hard to meet the demand for real-time in backbone networks. In addition, most of these methods are based on bidirectional traffic and do not consider the asymmetry of routing on the Internet. In this paper, based on the traffic characteristics of the most common types of SHD, we extract several representative features from unidirectional flows. These features can still work well under sampling and asymmetric routing scenarios. We also use Slow HTTP DoS Sketch to record the features quickly and accurately. In experiments that used public backbone datasets as background traffic, the results show that even with a large number of unidirectional flows and a sampling rate of 1/64, our method can still accurately detect SHD traffic within 2 min.

Keywords: Slow HTTP DoS · Backbone network · Asymmetric routing · Sampling · Intrusion detection

1 Introduction

Today, distributed denial of service (DDoS) is one of the most harmful and wide-ranging attacks on the Internet. Since detection for traditional DDoS with massive traffic such as SYN Flood is nowadays more mature [1], attackers have devised new types of DDoS. These new attacks are more advanced and favorable to attackers than traditional DDoS.

Slow HTTP DoS (SHD) is a new type of DDoS attack that utilizes the HTTP/HTTPS and has a high potential for harm [2]. It maliciously maintains

long but low-speed legitimate web connections through unique means and consumes all connections the server can support. The server will hold the session in its capped connection pool. Once the number of connections in the pool reaches the maximum, the server will not serve new legitimate requests, leading to a denial of service. The common SHD can be classified into three types [3] according to the way they exploit the protocol: Slow Header (Slowloris), Slow Message Body (RUDY), and SlowREAD. SHD utilizes application layer protocols and may be encrypted. Thus SHD traffic looks similar to normal traffic. Besides, SHD traffic is sent at a slow rate, and the packet interval can be more than 10 s. Compared to traditional DDoS, the traffic characteristics are not obvious. So SHD is more concealed than traditional DDoS.

Several methods have been proposed for SHD detection. Most methods require unsampled traffic. Besides, some methods need to extract many features to maintain the state of each flow as detailed as possible [4]. They can work in small-scale networks with little traffic. However, since the targets of SHD are usually located at critical nodes of the backbone network like large web servers, proxy servers, etc. Traffic passing through these observation points is massive. These methods require many resources to compose packets into flows, which is impractical in the backbone network. Even if it were possible to store the data, these methods would not be able to process it in a timely manner.

Therefore, in backbone networks, the following difficulties exist in detecting SHD: (1) High-speed network environment which further reduces the percentage of SHD traffic, makes it more concealed. (2) Most existing methods are only suitable for small-scale networks. In backbone networks, they incur large resource consumption, are difficult to meet real-time requirements, and are ineffective in sampling conditions [4–7]. (3) The monitor on the backbone network can often collect unidirectional traffic because of asymmetric routings [8]. However, most existing methods utilize bidirectional flows. These difficulties determine that SHD can easily bypass existing intrusion detection systems.

In order to solve these difficulties, we propose an accurate and real-time detection method for SHD in the backbone network. The main contributions can be described as follows.

- To accommodate massive data in the backbone network, we select a small number of features that can be extracted from the sampled traffic for detection. Also, we use a fast data structure, Slow HTTP DoS Sketch (SHD-Sketch), to record traffic information, avoid composing packets to flows and reduce overhead.
- To accommodate backbone scenarios with a large number of asymmetric routings, we extract different types of representative SHD features obtained from unidirectional flows.
- To evaluate the effectiveness of our method, we train and test on two datasets using public backbone datasets as background traffic. The results show that our method can accurately detect SHD traffic even in the presence of a large number of asymmetric routings. Furthermore, our method can still detect SHD attacks within 2 min, even at a sampling rate of 1/64.

2 Related Work

SHD is one of the most challenging types of DDoS attacks to detect. Due to slow speed and little traffic, SHD is more concealed than traditional DDoS in backbone networks.

To detect SHD traffic, researchers have proposed different schemes. Some schemes detected SHD based on potential traffic patterns. Hossein et al. [7] proposed a method based on the nonparametric cumulative sum. They also analyzed the effect of sampling methods and rates. The results show that the detection rate decreases sharply when the sampling rate is lower than 30%. Reed et al. [9] observed the potential relationship of length and interval time between packets. They found a specific pattern in the unsampled case which results in good accuracy even with a few features. These methods require all traffic to extract the patterns, which cannot be retained in sampling conditions. Therefore, these methods are hard to be applied to backbone networks.

Some schemes detected SHD attacks based on artificial intelligence. Xu et al. [6] analyzed SHD parameters such as frequency from the perspective of the signal system. They proposed a hybrid deep neural network method using CNN and GRU. Rani et al. [5] extracted 83 features from bidirectional flows and applied some ML algorithms like XGB for SHD detection in device-to-device communications. Similarly, Garcia et al. [4] proposed an AI-based detection system, which analyzes flows to obtain 57 features for detection and finally uses GMM clustering to improve precision. However, these methods require a large amount of computation which are hard to achieve real-time detection in backbone networks. Besides, both methods [4,5] need to extract many features in advance, which will further increase computation. Moreover, all of them capture features from bidirectional flow data. None of them has ever considered the effect of asymmetric routing on features and detection results when only unidirectional flow data can be captured.

In summary, the existing methods have several shortcomings making them hard to detect SHD in backbone networks. Some methods can only be applied to the case with no or small sampling ratios and thus cannot be applied to massive traffic in the backbone network. Some methods are too complex, leading to excessive resource consumption and difficulty detecting SHD in real-time. In addition, almost all methods do not consider asymmetric routings in the backbone networks and thus cannot accurately detect SHD traffic in this case. In this paper, we propose an accurate and real-time detection method for SHD in backbone networks.

3 Methodology

3.1 General Design

As shown in Fig. 1, the overall architecture of the proposed method can be divided into the following three phases: data acquisition and processing phase, offline training phase, online detection and threat alert phase.

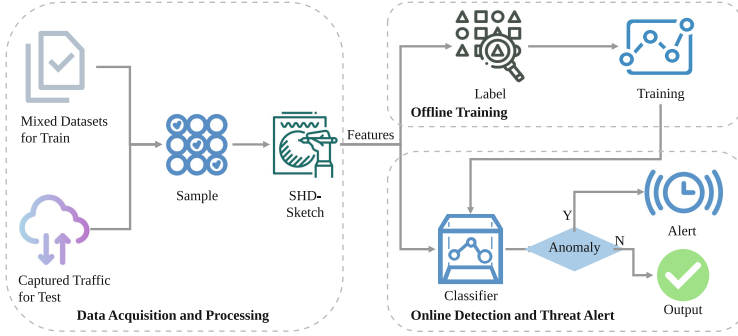


Fig. 1. General design

In the data acquisition and processing phase, for offline training data, we use traffic from public backbone datasets and traffic generated by the public tool `slowhttptest` [11] to get the mixed datasets. For online detection data, traffic is captured from the monitor located in the backbone network. Then we sample the packets and hash the IP to update SHD-Sketch. Finally, we extract the statistical features of traffic from SHD-Sketch for subsequent use.

In the offline training phase, each statistical feature is labeled with the type when the attack traffic was generated. Then the labeled features are used as the input to the machine learning algorithm for training to obtain the classifier.

In the online detection and threat alert phase, the statistical features of the captured traffic under sampling conditions are input into the classifier to distinguish the SHD traffic. The decision to generate an alert is based on the classification results. After alerting, the related IPs can be listed if necessary.

3.2 Feature Selection

Feature selection for SHD traffic is vital to overcome the difficulties presented above. Analyzing SHD attacks' characteristics helps to detect attacks more accurately and quickly with fewer features and less consumption. The three most common types of SHD attacks are Slowloris, RUDY and SlowREAD [3]. Both Slowloris and RUDY continuously send incomplete HTTP requests to consume victims' connection resources. SlowREAD can consume victims' connection and memory resources by receiving data at a low speed.

In this section, we analyze the statistical characteristics implied within the SHD traffic. Then we propose a series of representative features and optimize them. Besides, the features proposed are all obtained from unidirectional traffic to cope with asymmetric routing.

Common Characteristic of SHD Packets. We select some features according to the attack characteristics of SHD, which are effective for these types of SHD.

- Quantitative characteristics. Although SHD will not continuously generate much traffic like traditional DDoS, SHD traffic will show abrupt changes in specific periods. Because the attacker needs to send packets to maintain connections, the number or speed of packets during that period will be larger than normal traffic. Therefore, sending rate of TCP packets (Pck_Spd) is chosen as the feature. For Slowloris and RUDY, the attacker will continuously send specific attack packets to the server, and the server will return the ACK packets. For SlowREAD, since the attacker controls the read rate, the sending rate of the victim will be kept within a certain range. So the packet length in all three types of SHD tends to be more stable, i.e., the TCP packet length standard deviation (Len_Std) will be smaller than that of normal traffic. For similar reasons, the number of packets with payload (Pck_1) and without payload (Pck_0) are chosen as characteristics.
- Packet IP and Port distribution. In SHD, attackers will communicate with the victim by sending numerous packets with different $\langle IP, Port \rangle$ to occupy the victim's HTTP connection pool, resulting in the scattered $\langle IP, Port \rangle$ distribution of the attackers. The victim will only listen to one $\langle IP, Port \rangle$ pair to provide web services in public, so the distribution of the victim's $\langle IP, Port \rangle$ is fixed. Therefore, the $\langle IP, Port \rangle$ dispersion of source (S_Disp) and that of destination (D_Disp) are chosen as characteristics.

Specificity Characteristics of Different SHD Attacks. Although SHD has the above common characteristics, these characteristics may exist in other traffic. To reduce misclassification, we also selected specific characteristics according to the types of SHD.

- In the detection of Slowloris and RUDY, scanning attacks are easily misclassified as SHD attacks if only the features above are used. The scanning attackers send packets to different $\langle IP, Port \rangle$ pairs to probe live IPs and exploited ports. The victim in the SHD sends many packets to reply attackers who also have different IPs and ports. Therefore, SHD and scanning attacks have strong similarities in statistical characteristics. To distinguish between SHD and scanning attacks, we choose the number of TCP packets whose SYN flag equals 1 (Pck_SYN) as the feature. The reason is that attackers in scanning attacks send packets with the SYN flag to establish connections, yet the victim in SHD sends packets without the SYN flag to reply.
- In the detection of SlowREAD, since the attacker needs to pretend that the read rate is slow to reduce the victim's sending rate, it will send many packets with a receive window of 0. However, in normal communication, the host can quickly process the packets, so there are almost no packets with a receive window of 0. Therefore, The number of TCP packets whose RWND equals 0 ($RWND_0$) is chosen as the specific feature of SlowREAD, which can distinguish attack traffic from normal traffic excellently.

The above features are selected based on the characteristics of SHD traffic, as shown in Table 1. Pck_0 , Pck_1 , Pck_SYN and $RWND_0$ appear in very

large or small proportions. Distributions of victims or attackers are very fixed or scattered. The Len_Std of SHD is decided by the attack, independent of the sample rate. Thus, even after sampling, most features can still reflect the SHD traffic characteristics well. Besides, since these features can be extracted from unidirectional flows, they are well-suited for backbone networks with asymmetric routing. Furthermore, since our features do not involve application layer specifics, they are also suitable for encryption scenarios.

Table 1. Details of proposed features

Feature	Details
Pck_0	Number of TCP packets sent without payload
Pck_1	Number of TCP packets sent with payload
Pck_SYN	Number of TCP packets whose SYN equals to 1
$RWND_0$	Number of TCP packets whose RWND equals to 0
S_Disp	Dispersion of source IP and port
D_Disp	Dispersion of destination IP and port
Pck_Spd	Sending rate of TCP packets
Len_Std	Length standard deviation of TCP packets

3.3 Slow HTTP DoS Sketch

Because these features cannot be extracted directly from individual packets, we used a fast data structure SHD-Sketch to obtain the statistics for calculating these features.

Structure of SHD-Sketch. In order to quickly extract features from sampled packets, we construct SHD-Sketch based on DCSS [8], where the counters and Bitmaps are changed according to SHD characteristics, as shown in Fig. 2. Every bucket in SHD-Sketch has multiple counters and Bitmaps to record the traffic. SHD-Sketch can extract features without composing packets to flows, which greatly reduces the required storage and computing resources and is suitable for backbone network scenarios.

SHD-Sketch is composed of a two-dimensional array of buckets whose length is L and height is H . The bucket in row i and column j is $B(i, j)$. We denote SIP , DIP , $SIPT$, and $DIPT$ as source IP, destination IP, source $\langle IP, Port \rangle$, and destination $\langle IP, Port \rangle$. We use SIP as the packet key. When a new packet arrives, we hash the key by FarmHash which can operate quickly. Then we get a 128-bit hash value called HV . After that, we divide HV into H parts, each with a length of l bits ($L = 2^l$ & $l \leq \frac{128}{H}$). Each part is denoted as p_k , where $1 \leq k \leq H$. If the decimal value corresponding to p_k is v , $B(k, v)$ should be updated.

It should be noted that dividing HV into H parts is aimed at reducing memory consumption. If not, a total of $2^{128} \times 10$ bytes of storage is needed, which is a large consumption. When divided, the required storage will drop sharply to only $L \times H \times 10$ bytes. In our experiments, $L \times H$ equals to 2^{24} , thus total storage is approximately $\frac{1}{2^{100}}$ of the original demand.

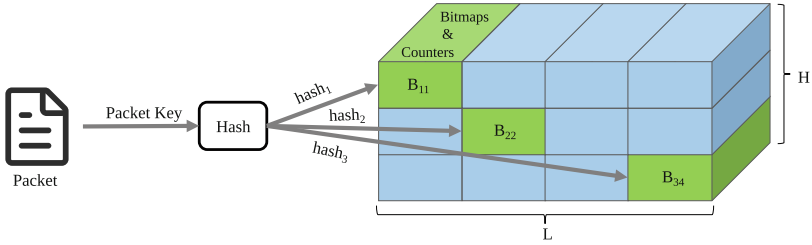


Fig. 2. The structure of SHD-Sketch

Functions of SHD-Sketch. For SHD-Sketch, there are five main functions: *Update*, *UpdateBitmap*, *Count*, *Estimate* and *Reduction*.

Update: When a new packet arrives, its key is used to find addresses of $\{B_1, \dots, B_H\}$ and then update these buckets. For counters in the bucket, we directly add one to it. For the Bitmaps, we need to perform specific operations to update them, which will be described in *UpdateBitmap*.

UpdateBitmap: For *SIPT* and *DIPT* in a packet, we compute them separately to update the *SourceBitmap* and *DestinationBitmap*. First, the FarmHash is performed on *SIPT* and *DIPT*. After that, we use the result of FarmHash to calculate the corresponding position in the Bitmap. Then the binary value of the corresponding bit in the Bitmap is updated. If the bit is 0, it is changed to 1. If it is already 1, it will not be modified. The process is shown in Fig. 3. Due to the characteristics of SHD, *SIPT* or *DIPT* in the attack traffic tends to be scattered. So the number of 1 in the *SourceBitmap* (S_Dist) and that in the *DestinationBitmap* (D_Dist) can intuitively reflect whether there is an attack.

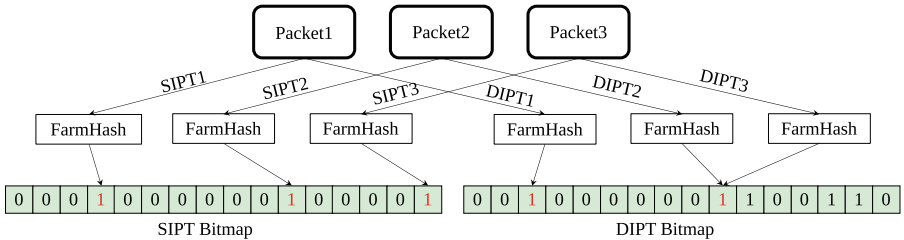


Fig. 3. Process of *UpdateBitmap*

Count: When *Update* function is finished, calculate the minimum *min_value* of the H buckets $\{B_1, \dots, B_H\}$ corresponding to the current key.

Estimate: When the *min_value* reaches a threshold α , *Estimate* function will be triggered. The current key and the data in the bucket will be recorded in a feature vector for subsequent use. In addition, the counters and Bitmaps in the $\{B_1, \dots, B_H\}$ will be cleared in *Reduction* function.

Reduction: When the *Estimate* function is triggered, the feature vector will be recorded. Then, for all counter types in H buckets, each counter is subtracted from the minimum in this counter type. For each Bitmap, we set all bits to 0.

Table 2 and Formula (1)–(3) show the details of the proposed features. Most of the statistical features we need can be obtained directly through the *Estimate* function, but for *Pck_Spd* and *Len_Std*, we need to perform specific calculations to get them. Np is the number of packets sent by SIP, R is the sample ratio, Δt is the time from the last *Estimate* call to the current one, Pck_Len_i is the length of the i th packet, and *Len_Avg* is the average value of the current counted packet length. Note that to save memory, we record the packet length by shifting each packet length to the right by 6 bits, i.e., dividing by 64, then rounding up afterward and multiplying by 64 for reduction.

Table 2. Details of contents in SHD-Sketch

Content	Source	Size
Np	Counter	1 Byte
Pck_0	Counter	1 Byte
Pck_1	Counter	1 Byte
Pck_SYN	Counter	1 Byte
$RWND_0$	Counter	1 Byte
SourceBitmap	Bitmap	2 Bytes
DestinationBitmap	Bitmap	2 Bytes

$$Pck_Spd = \frac{Np * R}{\Delta t} \quad (1)$$

$$Len_Avg = \frac{\sum_{i=1}^{\alpha} Pck_Len_i}{\alpha} \quad (2)$$

$$Len_Std = \sqrt{\frac{\sum_{i=1}^{\alpha} (Pck_Len_i - Len_Avg)^2}{\alpha - 1}} \quad (3)$$

3.4 Offline Train

In this phase, the statistical features of the traffic are obtained from SHD-Sketch. Machine learning algorithms are used to get the final classifier. Both Decision

Tree and Random Forest have a relatively small time complexity for training and prediction. They also perform well in classification, meeting the need for real-time and accuracy in backbone network scenarios. Thus they are finally chosen in our work.

3.5 Online Detection and Threat Alert

In this phase, the classifier obtained from offline training can be deployed in backbone networks. The classifier can detect the presence of SHD traffic. In addition, as mentioned above, when we record the traffic statistics by the *Estimate* function, we also record the IP. Therefore, our method can also quickly determine attackers' and victims' IPs, then generate alerts for security personnel to take subsequent defensive measures.

4 Experiments and Evaluation

4.1 Experimental Setup

In this subsection, the testbed, dataset and metrics used in the experiments will be introduced separately.

Testbed: To evaluate the proposed approach, we experimentally validated it on a high-performance host with an Intel(R) Core(TM) i9-10900K CPU @ 3.70 GHz, a 2 TB hard disk, NVIDIA GeForce RTX 3090 and 128 GB RAM.

Dataset: We use two sets of datasets. Dataset A is used for training and Dataset B for testing. Each dataset consists of normal public backbone traffic of MAWI [10] and SHD traffic. Compared to the traffic generated by simulators, the real-world traffic is more complex and the possible traffic patterns are more random, allowing for a better test of the effectiveness of the proposed method. Since there is no mature public dataset of SHD traffic, we use the public tool *slowhttptest* [11], a mainstream SHD implementation tool widely used in studies related to SHD detection.

In the MAWI public dataset, we selected data from June 10, 2020 (MAWI-202006101400) and June 03, 2020 (MAWI-202006031400). It is collected on *samplepoint-G*, an Internet exchange link of 10Gbps WIDE backbone network. We chose datasets from the two days as the background traffic for training and testing. We count the flow information of the two datasets, and it is notable that one-way traffic in these datasets reached 81.55% and 84.10% which are high percentages. This means that a large portion of the traffic on the backbone network is unidirectional, i.e., asymmetric routing is more prevalent and cannot be ignored. The details are shown in Table 3.

Table 3. Details of Datasets

Mixed datasets	# of unidirectional flows in MAWI	# of bidirectional flows in MAWI	Attack type	# of SHD flows	# of all flows	Unidirectional flows percentage
A	2228200	421145	Slowloris	2127	2651472	84.04%
			RUDY	12916	2662261	83.70%
			SlowREAD	28256	2677601	83.22%
B	1958601	443116	Slowloris	2190	2403907	81.48%
			RUDY	2182	2403899	81.48%
			SlowREAD	28233	2429950	80.60%

For the attack traffic, we use the topology in Fig. 4 in combination with the public tool slowhttptest in a real environment and obtain the attack traffic of distributed SHD. For the attack traffic in the training set, we set a series of parameters of slowhttptest that are just enough to reach denial of service. For the attack traffic in the test set, we only use the default parameters without unique settings, i.e., the most commonly used parameter in practice. After that, we mix MAWI-202006031400 with the attack traffic of custom parameters to get the training traffic Dataset A and mix MAWI-202006101400 with the attack traffic of default parameters to get the test traffic Dataset B.

Metrics: To evaluate the results, we use the following metrics commonly used in machine learning to measure the effectiveness: precision, recall, false positive rate (FPR) and F1-score (F1).

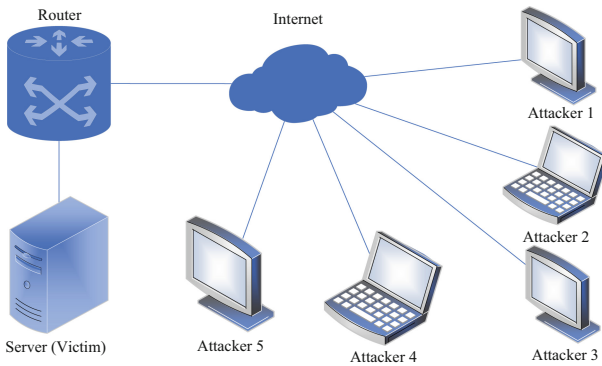


Fig. 4. Topology of experiment

4.2 Analysis of Threshold

Threshold α in SHD-Sketch decides when to record a feature vector. It has an important role in detection results and time. If α is too small, the accumulated number of packets is too small for SHD-Sketch to extract the statistical features of SHD accurately. If α is too large, it needs more packets and time to extract statistical features. Therefore, we analyze the selection of α in detail.

The selection of α is closely related to the Bitmap in SHD-Sketch. We determine the appropriate α based on Bitmap's ability to distinguish dispersion. We utilize the randomness of the hash function and conduct ten thousand Bitmap filling tests to find how many packets with different keys are needed to fill the Bitmap. The final distribution of the results is shown in Fig. 5 (a). The blue line shows the actual distribution of the simulation tests, with a mean of $\mu = 54.13496$ and a standard deviation of $\sigma = 18.6769$. The red line is the standard normal distribution, which obeys $X \sim N(\mu, \sigma^2)$. It shows that the distribution of the actual simulation test still has a certain gap with the standard normal distribution. We calculate the skewness and kurtosis of the actual sample distribution.

$$Skewness = \frac{1}{n-1} \sum_{i=1}^n \frac{(x_i - \mu)^3}{\sigma^3} = 1.2921 \geq 0 \quad (4)$$

$$Kurtosis = \frac{1}{n-1} \sum_{i=1}^n \frac{(x_i - \mu)^4}{\sigma^4} - 3 = 2.8529 \geq 0 \quad (5)$$

Formula (4) and (5) show that Skewness and Kurtosis are greater than 0, which means a positively skewed distribution. Therefore, in order to better analyze its probability distribution and thus select a suitable α , we logarithmically process the results of the Bitmap filling tests and the standard normal distribution results. We let $key_num' = \ln(key_num)$, where key_num is the number of keys that fill up the Bitmap. The mean value after processing is $\mu' = 3.9377$, and the standard deviation is $\sigma' = 0.3240$, as shown in Fig. 5 (b). The processed distribution of the simulation experiments remains almost the same as the standard normal distribution $X \sim N(\mu', \sigma'^2)$. Therefore, we can analyze the approximate distribution of the original tests through the transformed normal distribution. It is known from the Gaussian distribution that the area within the horizontal axis interval $(\mu' - 2\sigma', \mu' + 2\sigma')$ accounts for about 95.45%. The total probability of occurrence before $\mu' + 2\sigma'$ is about $95.45\% + 2.275\% = 97.725\%$. Similarly, the total probability of occurrence before $\mu' + 3\sigma'$ is 99.865%. After that, we revert it to the actual number of keys. key_num_2 represents the number of keys to fill up the Bitmap in the condition of 2σ , and key_num_3 is similar.

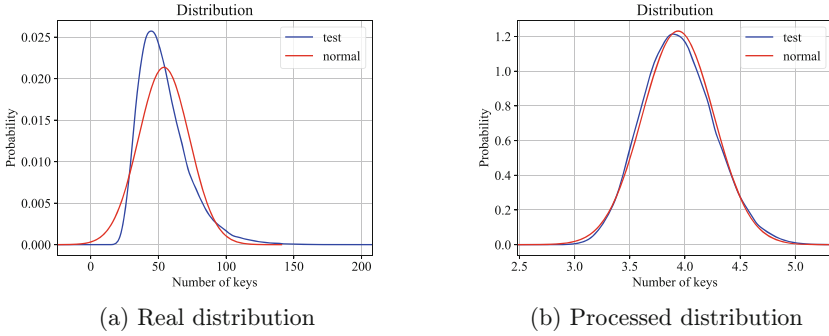


Fig. 5. Simulated test distribution before and after processing

$$key_num_2 = e^{(\mu' + 2\sigma')} = 98.076 \tag{6}$$

$$key_num_3 = e^{(\mu' + 3\sigma')} = 135.607 \tag{7}$$

Therefore, it can be approximated that if α is 99, the Bitmap is 97.725% possible to be filled up. If α is 136, the Bitmap is 99.865% possible to be filled up. Considering that covering the 2.14% possibility requires increasing α by 37, this will increase overhead and result in a longer detection time. Hence we finally choose a α of 100 for the following experiments.

4.3 Effect of Sampling Rate

Sampling network traffic is necessary to meet the real-time requirement in backbone networks and to minimize storage and computational overhead. However, sampling also has an impact on the detection. Therefore, we test the effectiveness of our method at five sampling rates of 1/8, 1/16, ..., and 1/128. Recall, precision, false positive rate (FPR) and F1-score (F1) are chosen as metrics.

Table 4 shows the detection results of our method for three different SHD. Our method achieves good results for all three attacks at different sampling rates. Precision, recall and F1 of SlowREAD achieve almost 100% at all sampling rates. As for the detection for Slowloris and RUDY, even the smallest F1 reaches almost 96% and 94% at various sampling rates. In the detection of Slowloris, the F1 increases as the sampling rate increases. The larger the sampling rate is, the shorter the sampling interval is, and the features of the potential attack traffic obtained are closer to the actual traffic. The F1 in other experiments shows a similar trend. Besides, FPR is close to 0% in all cases, which means a few negative samples are identified as positive. It is also satisfactory in attack detection.

Table 4. Detection Results in Different Sampling Rates

Sample Rate	Slowloris				RUDY				SlowREAD			
	Precision	Recall	FPR	F1	Precision	Recall	FPR	F1	Precision	Recall	FPR	F1
1/1	99.19%	97.06%	0%	98.11%	94.38%	97.15%	0.02%	95.74%	99.61%	97.74%	0.01%	98.67%
1/8	100%	94.87%	0%	97.37%	93.25%	97.44%	0.02%	95.30%	99.46%	97.02%	0.02%	98.23%
1/16	100%	94.81%	0%	97.33%	92.68%	98.70%	0.02%	95.60%	99.89%	99.37%	0%	99.63%
1/32	100%	92.11%	0%	95.89%	90.24%	97.37%	0.03%	93.67%	100%	99.58%	0%	99.79%
1/64	100%	100%	0%	100%	100%	100%	0%	100%	99.16%	100%	0.03%	99.58%
1/128	100%	100%	0%	100%	100%	100%	0%	100%	99.16%	100%	0.03%	99.58%

It should be noted that as the sampling rate decreases, less traffic information can be obtained. For tiny flows in the attack traffic, they are harder to reach α at low sampling rates. Therefore SHD-Sketch will not extract their feature vectors when the sampling rate comes to a specific smaller state. While feature vectors of other attack flows with more traffic can still be extracted at low sampling rates. And precisely because of their more traffic, they present more obvious statistical characteristics, which make them easier to be detected. Combining the above two reasons, precision and recall may reach higher values at low sampling rates. Even if this tiny part of the flow is not sampled, it is still tolerable because this part alone does not successfully result in a denial of service to the victim in practice. And fortunately, we can reduce this phenomenon by adopting suitable sampling rates.

4.4 Speed of Alarm Time

To evaluate the real-time performance of our proposed method, we use the same dataset as above to evaluate detection time. The detection time here is counted from when the first attack packet appears.

Figure 6 shows the time required to detect SHD attacks at different sampling rates. The zero of the y-axis is when the attack has just started. The figure illustrates that we can quickly detect attacks at high sampling rates. When the sampling rate decreases, the time to detect the attack also be affected. That is because, SHD-Sketch requires more packets to generate a feature vector at a lower sampling rate, i.e., it takes more time to reach the threshold α . Even though the detection time increases as the sampling rate decreases, our method can still detect all three attacks within 2 min at a sampling rate of 1/64. It further shows that our approach can meet the real-time requirements in backbone networks.

Furthermore, it should be noted that due to the slow rate of SHD, the victim is not immediately in a denial-of-service state. In our experiments, the victims of Slowloris, RUDY, and SlowREAD began to deny service at 30 s, 28 s, and 505 s, respectively. And if the sampling rate is appropriate, our method can detect attack traffic in a shorter time, which is very meaningful in practice.

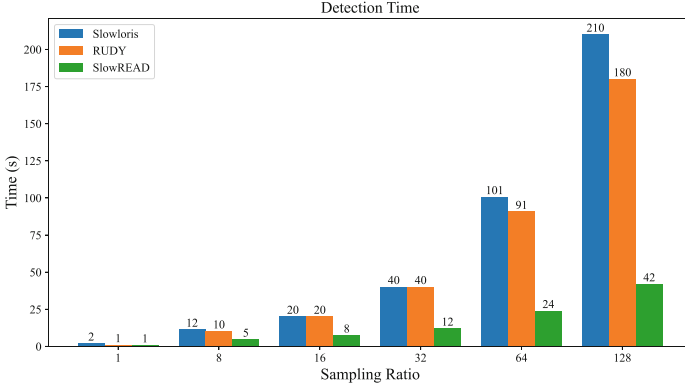


Fig. 6. Detection time under different sampling ratios

4.5 Comparison with Similar Work

Garcia et al. [4] and Lukaseder et al. [12] also used slowhttptest to conduct SHD attacks and achieved high accuracy in the complete traffic environment. Thus we compare our detection time in the complete traffic environment with their methods.

Table 5 shows that in the complete traffic condition, our method can detect three SHD attacks in 2.2s, 1.2s and 1.1s. Besides, our method can quickly and accurately detect attacks in sampled conditions, which existing works are difficult to achieve. Thus, the overall performance of our approach is better than theirs.

Table 5. Comparison with other work

Method	Slowloris	RUDY	SlowREAD
Garcia et al. [4]	16.1	10.3	9.8
Lukaseder et al. [12]	0.85	12.86	N/A
Ours	2.2	1.2	1.1

5 Conclusion and Future Work

In this paper, we propose a method for detecting SHD attacks in backbone networks and asymmetric routing scenarios. We select a set of highly representative features based on the unidirectional traffic characteristics of SHD attacks and use a data structure SHD-Sketch to obtain the features quickly. In experiments using public backbone datasets as background traffic, the results show that our approach can detect SHD attacks with high accuracy and low false positive rates at six sampling rates, even in the presence of a large number of asymmetric routings. In addition, our method can also detect the three most common types of

SHD attacks within 2 min even at a sampling rate of 1/64. In the future, we will focus on the sampling technique to make it more suitable for intrusion detection and improve SHD-Sketch to detect various SHD attacks more effectively.

Acknowledgements. This work was supported by the National Key R&D Program of China (2020YFB1807503).

References

1. Eliyan, L.F., Pietro, R.D.: DoS and DDoS attacks in software defined networks: a survey of existing solutions and research challenges. *Future Gener. Comput. Syst.* **122**, 149–171 (2021)
2. DDoS attacks reports in 2022. <https://securelist.com/ddos-attacks-in-q2-2022/107025>. Accessed 5 Mar 2023
3. Tripathi N., Hubballi N., Singh Y.: How secure are web servers? an empirical study of slow HTTP DoS attacks and detection. In: 11th International Conference on Availability, Reliability and Security (ARES), pp. 454–463. IEEE (2016). <https://doi.org/10.1109/ARES.2016.20>
4. Garcia, N., et al.: Distributed real-time SlowDoS attacks detection over encrypted traffic using Artificial Intelligence. *J. Netw. Comput. Appl.* **173**, 102871 (2021)
5. Rani, S.J., Ioannou, I., Nagaradjane, P., et al.: Detection of DDoS attacks in D2D communications using machine learning approach. *Comput. Commun.* **198**, 32–51 (2023)
6. Xu, C., Shen, J., Du, X.: Low-rate DoS attack detection method based on hybrid deep neural networks. *J. Inf. Secur. Appl.* **60**, 102879 (2021)
7. Jazi, H.H., et al.: Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling. *Comput. Netw.* **121**, 25–36 (2017)
8. Wu H., Chen T., Shao Z., et al.: Accurate and fast detection of DDoS attacks in high-speed network with asymmetric routing. In: IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2021). <https://doi.org/10.1109/GLOBECOM46510.2021.9685794>
9. Reed A., Dooley L. S., Mostefaoui S. K.: A reliable real-time slow DoS detection framework for resource-constrained IoT networks. In: 2021 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2021). <https://doi.org/10.1109/GLOBECOM46510.2021.9685612>
10. MAWI Public Traffic Datasets. <https://mawi.wide.ad.jp/mawi>. Accessed 5 Mar 2023
11. SlowHTTPTest Public Tool. <https://github.com/shekyan/slowhttpstest>. Accessed 5 Mar 2023
12. Lukaseder, T., Maile, L., Erb, B., Kargl, F.: SDN-assisted network-based mitigation of slow DDoS attacks. In: Beyah, R., Chang, B., Li, Y., Zhu, S. (eds.) *SecureComm 2018*. LNCS, vol. 255, pp. 102–121. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01704-0_6



Towards an Information Privacy Competency Model for the Usage of Mobile Applications

Aikaterini Soumelidou^(✉)  and Aggeliki Tsohou 

Ionian University, Corfu, Greece

{ksoumelidou, atsohou}@ionio.gr

Abstract. In a world where the industry of mobile applications (apps) is continuously expanding, the need for reinforcing users' protection of information privacy is urgent. Focusing on this emerging need, this study aims at highlighting the main competencies that a user of mobile apps should hold in order to protect their information privacy. The contribution of the paper is threefold; First, it proposes a framework which describes the actions that users of mobile applications make before and after the installation of the application. Second, based on conceptual analysis, this study introduces a framework for the synthesis of the Information Privacy Competency Model for Users of Mobile Apps incorporating widely known personality theories namely Protection Motivation Theory and Big Personality Theory. Finally, synthesizes the results into indicative competencies that users of mobile apps should hold so as to be competent to protect their information privacy. This study offers important implications regarding privacy protection in mobile apps not only for users, but also for privacy researchers, online service providers and educators.

Keywords: Information privacy · competencies · mobile applications

1 Introduction

Mobile applications (apps) market has undergone tremendous growth in recent years, as users daily engage in the downloading the ones that meet their needs. Statistics confirm the rise in app usage noting that in 2016, 140.7 billion apps were downloaded, whereas in 2021, 230 billion apps were downloaded, marking a 63% increase [1]. The usage of mobile apps has unleashed new possibilities to users, who have at their disposal a variety of apps to choose from, such as health apps [2], communication apps, gamification apps or educational apps [3].

Despite the fact that specific functionalities of mobile apps, such as location-based and personalized services are useful, literature highlights important issues that have arisen regarding information privacy [4]. Specifically, users state to increasingly concern about the way that mobile apps treat personal information, referring to concerns related to location tracking, large amount of required permissions and authorized selling data [5]. However, on the other hand, majority of mobile users declare lack of awareness regarding the data collected by their apps.

In changing this landscape, recent research has paid attention to the investigation of factors which urge users to disclose information when using mobile applications [6–9]. Additionally, some studies focus on investigating the way that specific apps use personal data [10] or proposing specific recommendations for mobile app developers in order to better understand the way in which privacy design principles are applied [11]. Despite the fact that aforementioned efforts are valuable, literature still lacks in providing specific directions to users for their privacy protection and mainly focus on providing directions to professionals. Nevertheless, having in mind that research in the information privacy competency domain can significantly contribute to citizens' behavioral change [12], this study aims to bridge this gap by answering the following research question “*What are the competencies that a user of mobile apps should hold in order to protect own information privacy?*”. Drawing on conceptual analysis, a framework for the design of the Information Privacy Competency Model for Users of Mobile Apps is proposed and indicative privacy competencies are presented. To the best of knowledge, this is the first attempt to handle information privacy issues of usage of mobile apps from a user point of view, providing useful insights for users, but also for privacy researchers, online service providers and educators.

The paper is structured as follows; Following this introduction, Sect. 2 presents the theoretical background on competency models. In Sect. 3, both a framework of users' actions during the usage of mobile applications and a framework for the synthesis of the Information Privacy Competency Model for Users of Mobile Apps are proposed.

Section 4 presents the preliminary results of the privacy competency model for users of mobile apps. Finally, Sect. 5 concludes the paper.

2 Literature Review: Competency and Competency Models

The term competency was first introduced in 1973, when McClelland [13] referred to it as a critical differentiator of performance. Since then, numerous definitions have been introduced and competency has been connected with the valid prediction of superior job performance in business organizations [14, 15]. However, competency is commonly referred as a measurable human ability to solve a problem in a given context and comprises three elements namely knowledge, skills and abilities [16]. Knowledge addresses the content of the necessary information that is required to perform a task, skills refer to someone's capacity to apply knowledge in order to perform the task and attitudes refer to someone's disposition to react to an idea or situation [17, 18].

A competency model is a group of key competencies which describes the combination of specific knowledge, skills and other personal characteristics that someone should have in order to efficiently perform a task [19]. Competency models are key tools in many areas of human resource management such as recruiting officials or design training. For instance, several models have been developed to define prerequisite competencies for manager's effective performance [20, 21]. One of the most known types of competency models is the Iceberg Model [22], in which the elements of competencies are represented as an iceberg. A part of the iceberg is visible on the top of the water and includes the elements of knowledge and skills which are visible characteristics of individuals,

whereas traits and motives, which consists internal characteristics of individuals, are hidden competencies under the water surface. Self-concept characteristics fall somewhere in between.

Few competency models have also been applied in the Information System (IS) literature. However, most of them, have been developed defining specific competencies that are essential for professionals, such as software requirements analysts [23], software engineers [24], software developers [25] or Information Technology specialists [26]. On the other hand, there are still limited competency models which refer to IS end-users, such as competencies for employees with higher education in Industry 4.0 [27], for competent IS users [28] or for security policy compliance behavior [29].

Even less attention has been paid to the investigation of users' competencies that relate to more cautious privacy protective behavior. Specifically, [12] introduced an information privacy competency model for citizens, including attributes that one should hold so as to be competent to protect own information privacy, such as knowledge, skills, attitudes and values. However, the remaining references to privacy competencies are fragmentary and do not constitute a comprehensive privacy competency model. [30] refer that among other competencies, teaching of digital citizenship competencies in elementary school should also include privacy. In the same vein, [31] refer to "Privacy and security" as one out of the six topics that Digital Citizenship Curriculum should try to inform young people about and [32] note that digital privacy and security competencies are necessary for students' preparation for working in the digitally enabled health sector. Moreover, [33] specify that users of Online Social Networks should have both the competency of knowledge about the recipients of their data and the metacognitive accuracy. Additionally, [34] investigated the necessary skills for the adoption of mobile health techniques and introduced five core competencies namely evidence, integration, ethics, and cultural considerations, and finally, security and privacy.

Having in mind all the aforementioned models, we conclude that there is an urgent need not only for the investigation of information privacy competencies in the mobile-apps context, but also for their composition into a comprehensive privacy competency model, which can offer significant knowledge to users' privacy protection. In order to fill this gap, we develop a privacy competency model for users of mobile applications based widely known privacy personality theories, comprising the information privacy protection competencies that users should hold in every step of app's use.

3 Methodology

3.1 A Framework for Information Privacy Competency Model for Users of Mobile Apps

For the definition of the main competencies of the privacy competency model for users of mobile apps, we mainly relied on the Information Privacy Competency Model for Citizens [12] which follows the structure of the Iceberg Model [22]. Following the above approach, our proposed competency model represents an iceberg as well, where knowledge and skills are the visible competencies on the top of it, whereas Social Role, Traits, Self-Image and Motives are hidden competencies.

The first visible competency on the top of the Iceberg Model is **knowledge**, which we define using the double definition given by [35]; “the declarative knowledge” which refers to user’s knowledge about the existence of a threat and the existence of an appropriate countermeasure and the “procedural knowledge” which refers to user’s knowledge about how to operate the security-control. Following this approach, we also define in our information privacy competency model two types of knowledge; The declarative privacy knowledge which refers to user’s knowledge about the existence of a privacy threat and the existence of an appropriate countermeasure and the procedural privacy knowledge which refers to user’s knowledge about how to operate the privacy-control.

The second type of visible competency of the Iceberg model is **skills**, which refers to one’s abilities to perform a certain task using existing knowledge. In our framework, we use the definition of “digital privacy skills” introduced by [36], defining a subset of skills which make users capable of applying strategies for their online privacy protection. Such skills are valuable tool in online privacy protection, as they enable users to participate in digital activities engaging in more privacy protective behaviors [37]. Thus, we argue that users of mobile applications should hold the competency of digital privacy skills, such as “reading a privacy policy of mobile apps, changing default security settings of mobile phone, or turning off location service enabler” [38].

Furthermore, as literature indicates, there is an inconsistency between mobile-apps users’ privacy concerns and their behavior, as they tend to use mobile apps that require access to personal information, even though they are informed about risks [39], confirming the existence of the privacy paradox phenomenon [40]. However, cautious and low-risk privacy behaviors on behalf of users is prerequisite for effective privacy protection [41]. Consequently, we added a new competency in the visible part of the Iceberg Model, namely “behavior” adopting the double definition of behavior (preventive behavior, confronting behavior) proposed by [35]. Thus, we argue that behavior is a competency that a mobile-apps’ user should hold in order to protect their privacy and it consists of two elements; Privacy Preventive behavior which includes actions that a user performs in order to mitigate the risk of being exposed to privacy violations and Privacy Confronting behavior which includes actions that a user performs while facing a privacy risk.

Following the visible competencies, on the bottom of the iceberg there are the hidden competencies which are more difficult to change; namely social role, self-image, traits, and motives. Social role refers to a person’s attitudes and values [42]. According to the literature, privacy concerns determine attitudes towards the disclosure of personal information online, namely privacy attitudes [43]. Literature shows that in the mobile apps context, privacy concerns negatively affect users’ download intention of a mobile app [4, 7]. For that reason, we argue that one main competency that mobile apps’ users should hold is strong privacy concerns.

Values, which is the next hidden competency in our competency model, are defined as “person’s preferences for certain end-states of existence” and are referred to be a determinant factor which affects one’s privacy behavior [44]. One of the most significant values, which can efficiently lead to stronger protection of information privacy, is anonymity, which describes the condition in which others cannot relate a given feature of the person to other characteristics [45]. In the smartphone context, anonymous communication applications can both act as a shield against disclosure of personal information and provide an environment where users feel free to share personal experiences without the fear of negative reputational consequences [46]. Anonymity is connected with freedom and fear-free living, which also consist values that contribute to users’ privacy protection. In more detail, when users adopt anonymity methods, they enjoy at the same time, both freedom and fear-free online, in terms that they protect their online privacy [47]. Having in mind above statements, we argue that users of mobile applications should hold the values of anonymity, freedom and fear-free as competences that lead to online cautious behavior.

The next type of competency, namely personality traits, are characteristics of constant and stable behaviors of people [17] that directly affect users’ privacy concerns. The Big Personality Theory, which describes five traits that characterize human personality, namely extraversion, agreeableness, conscientiousness, neuroticism, and openness to experience [48, 49], is widely connected to self-disclosure behavior. Table 1 below summarizes personality traits and their effect on self-disclosure behavior to mobile-apps context.

Table 1. Effect of Personality traits on self-disclosure behavior to mobile-apps context.

Trait	Description	Effect on mobile-apps context
Conscientiousness	Person self-disciplined, detailed-oriented, high-organizing. [49, 50]	Positively affects privacy concerns [8]
Agreeableness	Person with high feelings of trust, altruism, kindness. [49]	Negatively affects self-disclosure intention. [51, 52]
Neuroticism	Person characterized by sadness and emotional instability. [53]	Positively affects privacy concerns [8]
Extraversion	Person outgoing, chatty and highly characterized by optimism. [53]	Negatively affects privacy protective behaviors.[52]
Openness	Person daring, who always looks for experiencing new things. [54]	No connection found to mobile-apps context. [8]

The next competency in the hidden part of our competency model is self-image, which describes one's view or concept of oneself [55] and incorporates the concept of self-esteem. Self-esteem refers to self-evaluation as well as to an individual's descriptive conceptualization regarding himself [56]. Literature implies that in the mobile-apps context high level of self-esteem enables users to be more careful and protective about their online privacy adjusting privacy settings [57, 58]. For that reason, we claim that users of mobile apps should hold the competency of high self-esteem.

Our competency model ends up with the hidden competency namely motives; as mentioned by [59, 60] define motives as certain kinds of causes, the internal factors that arouse and direct a person's behavior. In this paper, in order to define the main competencies that a user of mobile apps should hold, we relied on the motives that the revised Protection Motivation Theory (PMT) introduces [61], namely threat appraisal and coping appraisal. Threat appraisal refers to individuals' perceived vulnerability and perceived severity of the threat, whereas coping appraisal refers to response efficacy, coping self-efficacy, and response costs associated with safe or adaptive behavior. Moreover, we included the self-representation motive as we another motive which was found to affect privacy protective behaviors. Table 2 summarizes motives and their effect on self-disclosure behavior in the mobile-apps context, whereas Table 3 presents the proposed framework for Information Privacy Competency Model for Users of Mobile-apps.

Table 2. Effects of motives on self-disclosure behavior to mobile-apps context.

Motive	Description	Effect on mobile apps context
Perceived vulnerability	Individual's belief about the likelihood of occurrence of a threat to them [62]	Positively affects privacy concerns. [63]
Perceived severity	Individual's assessment about how severe and harmful a threat can be to their life. [64]	Positively affects users' willingness to adopt protective actions. [65]
Response efficacy	Individual's belief about the effectiveness of that the recommended coping response is in alleviating the threat. [66]	Negatively affects the intention to share information. [67]
Coping self-efficacy	Individual's judgment about their ability to successfully confront challenges and apply protective behaviors. [61]	Positively affects users' confidence in restricting information sharing. [67]
Self-representation	Individual's behavior to intentionally regulate their personal image in the eyes of others. [65]	Positively affects users' intention to reveal personal data, such as social status or location [65]

Table 3. The framework for Information Privacy Competency Model for Users of Mobile apps

Element of Competency	Description
Knowledge	declarative knowledge; procedural knowledge
Digital privacy skills	Reading a privacy policy of mobile apps and turning off location service enabler
Behavior	Privacy Preventive behavior; Privacy Confronting behavior
Social Role	Self-image
Values	Anonymity; freedom; fear-free online
Self-image	Self esteem
Description	Consciousness, neuroticism, low level of agreeableness and extraversion
Motives	High perceived severity, high perceived vulnerability, high response efficacy, high self-efficacy, low response costs, low self-representation

3.2 A Proposed Framework of Actions of Users of Mobile Apps

To identify the main competencies that users should hold when using mobile apps, we first needed to clarify which actions are involved in the installation process. Given that literature lacks a framework which describes the actions taken by mobile apps users, we rely on a similar framework taken from the domain of e-commerce. Specifically, we rely on the Digital Competence Framework for Consumers of European Commission [68] which divides the online purchasing cycle into three phases; the “Pre-purchase phase”, the “Purchase” phase and the “Post-purchase” phase. Thus, we propose that the use of mobile applications is divided into two phases, namely the pre and post installation phases. Specifically, the pre-installation phase includes actions related to the downloading of the application, while the post-installation phase refers to actions taken after the installation and during the use of the application.

For the pre-installation phase, we primary relied on Google’s manual [69] for the download of apps to Android devices and as a result, we distinguished two actions included; the first action is “use of the Play Store app and find an app”, where the user is browsing to the Play Store app so looking for the desirable app; the second action refers to “checking that the app is reliable”, where the user estimates app’s reliability through checking the star ratings or the number of downloads and reading individual reviews, scrolling to “Reviews” section.

For the post-installation phase, we relied on the actions included in [68], and as a result we identified five actions related to the use of the selected app. The first action, namely “Registration” refers to the creation of user’s account as a prerequisite to join and use the app. The second action is the “Subscribing for using the app, where free use is not provided” action, which refers to user’s management of payment through digital means, where subscription is necessary for app use. The third action, namely “Establishing and managing digital identity in the app’s environment”, refers to the formation of user’s profile while using a mobile-app disclosing specific data for themselves. The

fourth action named as “Interacting in the app’s environment to take advantage of its functions” refers to targeted and specific use of app’s orders on behalf of users in order to fulfill their needs and the last action, namely “Recognizing and evaluating commercial communication and advertisement”, refers to users’ critical evaluation of different advertising methods in the app’s environment. Table 3 below summarizes the proposed framework of actions that a mobile-apps user follows, based both on google manual and the EU competence framework for consumers.

Table 4. The proposed framework of actions of mobile-apps users.

Pre-installation phase	Post-installation phase
<ol style="list-style-type: none"> 1. Using the Play Store app and finding an app 2. Checking that the app is reliable 	<ol style="list-style-type: none"> 1.Registration 2.Subscribing, where free use is not provided 3.Establishing and managing digital identity in the app’s environment 4.Interacting in the app’s environment to take advantage of its functions 5. Recognizing and evaluating commercial communication and advertisement

4 Results of the Conceptual Analysis: The Proposed Information Privacy Competency Model for Mobile Apps Users

As mentioned above, in order to develop the Information Privacy Competency Model for users of Mobile Apps, we followed the division of the use of mobile apps into two phases; the pre-installation phase and the post-installation phase (See Table 4). Moreover, we particularly identified every competence that is required in each phase with respect to information privacy based on the framework for Information Privacy Competency Model for Users of Mobile apps that is proposed above. Consequently, we propose that users of mobile apps should hold specific competencies in order to deal with information privacy issues that appear in each phase. Table 5 below represents indicative competencies that users should hold in the pre-installation phase, whereas Table 6 represents indicative competencies that users should hold in the post-installation phase. Actions 1 and 3 for the post-installation phase were selected so as to present indicative competencies not only in the initial stages of using the application (action 1 and 2), but also necessary competencies required in subsequent stages of app use.

Table 5. Indicative Privacy competencies that users of mobile-apps should hold in the pre-installation phase

Pre-installation phase		
Privacy Competencies	Action 1; Using the Play Store app and finding an app	Action 2; Checking that the app
Declarative knowledge	Users know that Play store app saves personal data which are collected with unique identifiers related to their device	Users know that privacy policy or privacy rights of each app is different
Procedural knowledge	Users know how to operate privacy control e.g. adjustment of privacy settings	Users know how to find, read and understand the privacy policy of each app
Privacy Preventive behavior	Adjustment of privacy settings/reading of privacy policies	Reading of privacy policies/reviews
Privacy Confronting behavior	Communication with the data protection officer using app's contact form	Exercise of privacy rights e.g., data deletion request
Digital Privacy Skills	Ability to accurately judge the extent of their own self-disclosure; Ability to understand privacy policy; Ability to exercise their own privacy rights	Ability to read and understand the privacy policy of each app; Ability to communicate with the communication manager about privacy issues
Social Role	High privacy concerns about the collection of information Anonymity/freedom; Users consider anonymity and freedom important; e.g., they do not use their google account	High privacy concerns about the collection of information on behalf of different apps. High confidence; Users are critical towards the promotion of apps
Traits	Conscientiousness; Focus on actions towards privacy protection (e.g., reading of privacy policy). Neuroticism; Paying attention to privacy protection adjusting privacy settings. Low Agreeableness; Express higher distrust regarding their data processing	Conscientiousness; Focus on actions towards privacy protection (e.g., reading of privacy policy). Neuroticism; Paying attention to the privacy terms of each app. Low Agreeableness; Express higher distrust in app providers about the collection of their data
Self-image	High self-esteem; Users should hold a high level of self-esteem, so as to be more protective regarding their online privacy	
Motives	Accurate perceived severity; Perceived vulnerability; Understanding of the potential privacy risks when using Play store app. Self-efficacy; Belief that they are capable of using privacy protection techniques such as adjustment of privacy settings. Response efficacy; Belief on the effectiveness of privacy protection techniques such as adjustment of privacy settings Low response costs of adopting preventing response such as the time of reading privacy policy	Accurate perceived severity; Perceived vulnerability; Understanding of the potential privacy risks involved in each app. Response efficacy; Belief on the effectiveness of reading the privacy policy of each app. Low response costs of adopting preventing response such as the time of reading privacy policy

Table 6. Indicative Privacy competencies that users of mobile-apps should hold in the post installation phase

Post-installation phase		
Privacy Competencies	Action 1; Registration	Action 3; Establishing and managing digital identity in the app's environment
Declarative knowledge	Users know that, when accepting permission app requests, specific personal information is disclosed	Users know that personal information that they provide form their digital identity, which is used for advertising
Procedural knowledge	Users know how to read and evaluate privacy policies	Users know which personal information form their digital identity
Privacy Preventive behavior	Reading of privacy policy	Adjustment of privacy settings, reading of privacy policy
Privacy Preventive behavior	Communication with app's data protection officer	Filing a complaint with the national data protection authority
Digital Privacy Skills	Ability to accurately judge the extent of their own self-disclosure when registering to an app; Ability to apply protective actions e.g., reading of privacy policies	Ability to accurately judge the extent of their own self-disclosure when using an app; Ability to apply protective actions e.g., adjustment of privacy settings
Social Role	High privacy concerns about the collection of information on behalf of the app	High privacy concerns; High confidence; Users are critical towards the use of their data (e.g., adjusting privacy settings)
Traits	Conscientiousness; Focus on protective actions (e.g., reading of privacy policy). Neuroticism; Carefully reading the privacy terms of the app. Low Agreeableness; Users are skeptical regarding the collection of their data	Conscientiousness; Focus on protective (e.g., adjustment of privacy settings). Neuroticism; Adjusting privacy settings. Low Agreeableness; Users are skeptical regarding the collection of their data. Low extraversion; Less chatty so as not to reveal too much personal information
Self-image	High self-esteem; Users should hold a high level of self-esteem, so as to be more protective regarding their online privacy	
Motives	Accurate perceived severity; Perceived vulnerability; Understanding of the potential privacy risks when registering to an app e.g., information disclosure. Response efficacy; Belief on the effectiveness of reading the privacy policy Low response costs of reading the privacy policy	Accurate perceived severity; Users implicitly understand the potential privacy risks involved in the use of app, such as third-party information sharing

5 Conclusions

This paper targets the issue of privacy protection of users of mobile applications. Conceptual analysis made by the authors resulted not only in the composition of a framework for the synthesis of the Information Privacy Competency Model for Users of Mobile Apps, but also in the application of an indicative privacy competency model for Users of Mobile Apps. Furthermore, a framework which describes the actions that users of mobile applications make before and after the installation of the application was proposed.

To the best of our knowledge, this paper is the first effort for the application of competency models to mobile-apps context. Results can prove beneficial not only for users of mobile apps, but also for their designers. Additionally, this paper is valuable tool for privacy researchers, online service providers and educators, as it offers significant insights into information privacy research.

This work has some limitations, which will be addressed in future research. First, the framework of action relies on the Google's manual for mobile apps that are available on Android devices. Further research will investigate potential refinement of the framework for iOS operating devices. Additionally, empirical investigation will allow the validation of the proposed competency model.

References

1. Amity.co homepage. <https://www.amity.co/blog/mobile-app-usage-statistics-to-grow-your-user-engagement>. Accessed 21 Feb 2023
2. Zhao, Y., Ni, Q., Zhou, R.: What factors influence the mobile health service adoption? A meta-analysis and the moderating role of age. *Int. J. Inf. Manage.* **43**, 342–350 (2017)
3. Flora, H.K., Wang, X., Chande, S.V.: An investigation on the characteristics of mobile applications: a survey study. *Int. J. Inf. Technol. Comput. Sci.* **6**, 21–27 (2014)
4. Gu, J., Xu, Y., Xu, H., Zhang, C., Ling, H.: Privacy concerns for mobile app download: an elaboration likelihood model perspective. *Decis. Support. Syst.* **94**, 19–28 (2017)
5. Nema, P., Anthonysamy, P., Taft, N., Peddinti, S.: Analyzing user perspectives on mobile app privacy at scale. In: *The 44th International Conference on Software Engineering (ICSE '22)*, pp. 112–124. Association for Computing Machinery, New York, NY, USA (2022)
6. Porter Felt, A., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions: user attention, comprehension, and behavior. In: *Proceedings of the SOUPS (2012)*
7. Wottrich, V.M., van Reijmersdal, E.A., Smit, E.G.: The privacy trade-off for mobile app downloads: the roles of app value, intrusiveness, and privacy concerns. *Decis. Support. Syst.* **106**, 44–52 (2018). <https://doi.org/10.1016/j.dss.2017.12.003>
8. Tang, J., Zhang, B., Xiao, S.: Examining the intention of authorization via apps: personality traits and expanded privacy calculus perspectives. *Behav. Sci. (Basel)* **12**, 218 (2022). <https://doi.org/10.3390/bs12070218>. PMID: 35877288. PMCID: PMC9311954
9. Chen, H., Li, W.: Mobile device users' privacy security assurance behavior: a technology threat avoidance perspective. *Inf. Comput. Secur.* **25**(3), 330–344 (2017). <https://doi.org/10.1108/ICS-04-2016-0027>
10. Brandtzaeg, P.B., Pultier, A., Harrand, N.: Privacy in mobile apps: measuring privacy risks in mobile apps. *Privacy in Mobile Apps* (2016)
11. European Union Agency for Cybersecurity, Privacy and data protection in mobile applications: a study on the app development ecosystem and the technical implementation of GDPR, European Network and Information Security Agency (2018). <https://data.europa.eu/doi/10.2824/114584>


12. Tsohou, A.: Towards an information privacy and personal data protection competency model for citizens. In: Fischer-Hübner, S., Lambrinouidakis, C., Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) *TrustBus 2021*. LNCS, vol. 12927, pp. 112–125. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86586-3_8
13. McClelland, D.C.: Testing for competence rather than for “intelligence”. *Am. Psychol.* **28**(1), 1–14 (1973)
14. Boyatzis, R.E.: Competencies in the 21st century. *J. Manage. Dev.* **27**(1), 5–12 (2008)
15. Page, C., Wilson, M.G.: Management competencies in New Zealand. On the inside looking in Wellington. Ministry of Commerce – 5 (1994)
16. Holtkamp, P., Lau, I., Pawlowski, J.M.: How do software development competences change in global settings – an explorative study. *J. Softw. Evol. Process* **27**(1), 50–72 (2014)
17. Yan, Y.J.: Problems of quality of migrant workers and countermeasures from the perspective of iceberg model. *Asian Agric. Res.* **5**, 48–50 (2012)
18. CEU (The Council of the European Union), Council Recommendation on Key Competences for Lifelong Learning. [online] [ec.europa.eu](https://education.ec.europa.eu). <https://education.ec.europa.eu/focus-topics/improving-quality/key-competences>. Accessed 22 Feb 2023
19. Staškeviča, A.: The importance of competency model development. *Acta Oeconomica Pragensia* **27**(2), 62–71 (2019)
20. Alidrisi, H.M., Mohamed, S.: Developing a personal leadership competency model for safety managers: a systems thinking approach. *Int. J. Environ. Res. Public Health* **19**(4), 2197 (2022)
21. Boyatzis, R.E.: *The Competent Manager: A Model for Effective Performance*. Wiley, New York (1982)
22. Spencer, L.M., Spencer, S.M.: *Competence at Work: Models for Superior Performance*. Wiley, New York (1993)
23. Klendauer, R., Berkovich, M., Gelvin, R., Leimeister, J., Krcmar, H.: Towards a competency model for requirements analysts. *Inf. Syst. J.* **22**(6), 475–503 (2012)
24. Moustroufas, E., Stamelos, I., Angelis, L.: Competency profiling for software engineers: literature review and a new model. In: *PCI '15: Proceedings of the 19th Panhellenic Conference on Informatics*, pp. 235–240 (2015)
25. Holtkamp, P., Jokinen, J.P., Pawlowski, M.Y.: Soft competency requirements in requirements engineering, software design, implementation, and testing. *J. Syst. Softw.* **101**, 136–146 (2015)
26. Bogoviz, A., Suglobov, A., Maloletko, A., Kaurova, O., Lobova, S.: *Frontier Information Technology and Systems Research in Cooperative Economics* (2021)
27. Prifti, L., Knigge, M., Kienegger, H., Krcmar, H.: A competency model for “Industrie 4.0” employees. In: *Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik (WI 2017)*, pp. 46–60. St. Gallen, S. (2017)
28. Eschenbrenner, B., Nah, F.F.: Information systems user competency: a conceptual foundation. *Commun. Assoc. Inf. Syst.* **34**, 80 (2014)
29. Tsohou, A., Holtkamp, P.: Are users competent to comply with information security policies? An analysis of professional competence models. *Inf. Technol. People* **31**(5), 1047–1106 (2018)
30. Lauricella, A.R., Herdzina, J., Robb, M.: Early childhood educators’ teaching of digital citizenship competencies. *Comput. Educ.* **158**, 103989 (2020)
31. James, L.T., Wallace, L., Warkentin, M., Kim, B., Collignon, S.: Exposing others’ information on online social networks (OSNs): perceived shared risk, its determinants, and its influence on OSN privacy control use. *Inf. Manage.* **54**, 851–865 (2017)
32. Cham, K., Edwards, M.-L., Kruesi, L., Celeste, T., Hennessey, T.: Digital preferences and perceptions of students in health professional courses at a leading Australian university: a baseline for improving digital skills and competencies in health graduates. *Australas. J. Educ. Technol.* **38**(1), 69–86 (2021)

33. Moll, R., Pieschl, S., Bromme, R.: Competent or clueless? Users' knowledge and misconceptions about their online privacy management. *Comput. Hum. Behav.* **41**, 212–219 (2014)
34. Schueller, S.M., Armstrong, C.M., Neary, M., Ciulla, R.: An introduction to core competencies for the use of mobile apps in cognitive and behavioral practice. *Cogn. Behav. Pract.* **29**, 69–80 (2021)
35. Bitton, R., Finkelshtein, A., Sidi, L., Puzis, R., Rokach, L., Shabtai, A.: Taxonomy of mobile users' security awareness. *Comput. Secur.* **73**, 266–293 (2018)
36. Trepte, S., et al.: Do people know about privacy and data protection strategies? Towards the "online privacy literacy scale" (OPLIS). In: Gutwirth, S., Leenes, R., de Hert, P. (eds.) *Reforming European Data Protection Law. LGTS*, vol. 20, pp. 333–365. Springer, Dordrecht (2015). https://doi.org/10.1007/978-94-017-9385-8_14
37. Büchi, M., Just, N., Latzer, M.: Caring is not enough: the importance of Internet skills for online privacy protection. *Inf. Commun. Soc.* **20**, 1261–1278 (2016). <https://doi.org/10.1080/1369118X.2016.1229001>
38. Park, Y., Jones-Jang, M.: Understanding privacy knowledge and skill in mobile communication. *Comput. Hum. Behav.* **38**, 296–303 (2014)
39. Pentina, I., Zhang, L., Bata, H., Chen, Y.: Exploring privacy paradox in information-sensitive mobile app adoption: a cross-cultural comparison. *Comput. Hum. Behav.* **65**, 409–419 (2016)
40. Kokolakis, S.: Privacy attitudes and privacy behavior: a review of current research on the privacy paradox phenomenon. *Comput. Secur.* **64**, 122–134 (2017)
41. Soumelidou, A., Tsohou, A.: Towards the creation of a profile of the information privacy aware user through a systematic literature review of information privacy awareness. *Telematics Inform.* **61**, 101592 (2021)
42. Chouhan, V.S., Sandeep, S.: Understanding competencies and competency modeling - a literature survey. *IOSR J. Bus. Manag.* **16**, 14–22 (2014)
43. Yuan, L.: Empirical studies on online information privacy concerns: literature review and an integrative framework. *Commun. Assoc. Inf. Syst.* **28**, 28 (2011). <https://doi.org/10.17705/1CAIS.02828>
44. Stone, E.F., Gueutal, H.G., Gardner, D.G., McClure, S.: A field experiment comparing information-privacy values, beliefs, and attitudes across several types of organizations. *J. Appl. Psychol.* **68**(3), 459–468 (1983)
45. Wallace, K.A.: Anonymity. *Ethics Inf. Technol.* **1**(1), 21–31 (1999)
46. Kang, J., Lan, J., Yan, H., Li, W., Shi, X.: Antecedents of information sensitivity and willingness to provide. *Mark. Intell. Plan.* **40**(6), 787–803 (2022)
47. Skalkos, A., Tsohou, A., Karyda, M., Kokolakis, S.: Identifying the values associated with users' behavior towards anonymity tools through means-end analysis. *Comput. Hum. Behav. Rep.* **2**, 100034 (2020)
48. Skrinjaric, B., Budak, J., Žokalj, M.: The effect of personality traits on online privacy concern. *Ekonomski Pregled.* **69**, 106–130 (2018). <https://doi.org/10.32910/ep.69.2.2>
49. Korzaan, L.M., Boswell, T.K.: The influence of personality traits and information privacy concerns on behavioral intentions. *J. Comput. Inf. Syst.* **48**(4), 15–24 (2008)
50. Junglas, A.I., Johnson, N.A., Spitzmüller, C.: Personality traits and concern for privacy: an empirical study in the context of location-based services. *Eur. J. Inf. Syst.* **17**(4), 387–402 (2008). <https://doi.org/10.1057/ejis.2008.29>
51. Huseynov, F.: Understanding usage behavior of different mobile application categories based on personality traits. *Interact. Comput.* **32**, 66–80 (2020)
52. van der Schyff, K., Flowerday, S., Lowry, P.B.: Information privacy behavior in the use of Facebook apps: a personality-based vulnerability assessment. *Heliyon* **6**(8), e04714 (2020). <https://doi.org/10.1016/j.heliyon.2020>. PMID: 32904276. PMCID: PMC7452521

53. Costa, P., McCrae, R.R.: A five-factor theory of personality. In: *The Five-Factor Model of Personality: Theoretical Perspectives*, vol. 2, pp. 51–87 (1999)
54. Soto, C.J., Kronauer, A., Liang, J.K.: Five-factor model of personality. In: Whitbourne, S.K. (ed.) *Encyclopedia of Adulthood and Aging*, vol. 2, pp. 506–510. Wiley, Hoboken (2016)
55. <https://dictionary.apa.org/>. Accessed 23 Feb 2023
56. Abdel-Khalek, A.: Introduction to the psychology of self-esteem (2016)
57. Wang, L., Yan, J., Lin, J., Cui, W.: Let the users tell the truth: self-disclosure intention and self-disclosure honesty in mobile social networking. *Int. J. Inf. Manage.* **37**, 1428–1440 (2017). <https://doi.org/10.1016/j.ijinfomgt.2016.10.006>
58. Nardis, Y., Panek, E.: Explaining privacy control on Instagram and Twitter: the roles of narcissism and self-esteem. *Commun. Res. Rep.* **36**(1), 24–34 (2019)
59. Kleinginna, P.R., Kleinginna, A.M.: A categorized list of motivation definitions, with a suggestion for a consensual definition. *Motiv. Emot.* **5**(3), 263–291 (1981)
60. McNeil, E.B., Rubin, Z.: *The Psychology of Being Human*, 2nd edn. Canfield, San Francisco (1977)
61. Rogers, R., Cacioppo, J., Richard P.: Cognitive and physiological processes in fear appeals and attitude change: a revised theory of protection motivation (1983)
62. Courneya, K.S., Hellsten, L.-A.M.: Cancer prevention as a source of exercise motivation: an experimental test using protection motivation theory. *Psychol. Health Med.* **6**(1), 59–64 (2001). <https://doi.org/10.1080/13548500125267>
63. Sharma, S., Singh, G., Sharma, R., Jones, P., Kraus, S., Dwivedi, Y.K.: Digital health innovation: exploring adoption of COVID-19 digital contact tracing apps. *IEEE Trans. Eng. Manage.* 1–17 (2020)
64. Chen, J.V., Widjaja, A.E., Yen, D.C.: Need for affiliation, need for popularity, self-esteem, and the moderating effect of Big Five personality traits affecting individuals' self-disclosure on Facebook. *Int. J. Hum. Comput. Interact.* **31**(11), 815–831 (2015)
65. Wang, T., Duong, T.D., Chen, C.: Intention to disclose personal information via mobile applications: a privacy calculus perspective. *Int. J. Inf. Manage.* **36**(4), 531–542 (2016)
66. Zhang, L., McDowell, W.C.: Am I really at risk? Determinants of online users' intentions to use strong passwords. *J. Internet Commer.* **8**(3), 180–197 (2009)
67. Kong, H., Anawar, S., Othman, N., Ayop, Z., Erman, H.: User privacy protection behavior and information sharing in mobile health application. *Int. J. Adv. Trends Comput. Sci. Eng.* **9**, 5250–5258 (2022)
68. Brečko, B., Ferrari, A., Vuorikari, R., Punie, Y.: The digital competence framework for consumers. Joint Research Centre Science for Policy Report; EUR 28133 EN (2016). <https://doi.org/10.2791/838886>
69. <https://support.google.com/android/answer/9457058?hl=en>



SecPassInput: Towards Secure Memory and Password Handling in Web Applications

Pascal Wichmann¹ , August See², and Hannes Federrath¹

¹ Security in Distributed Systems, Universität Hamburg, Hamburg, Germany
{pascal.wichmann,hannes.federrath}@uni-hamburg.de

² Computer Networks, Universität Hamburg, Hamburg, Germany
richard.august.see@uni-hamburg.de

Abstract. JavaScript does not provide web applications the ability to overwrite or clear variables of primitive types, such as strings, when they are no longer required. Applications instead need to rely on the garbage collector to eventually clear sensitive data from memory. When accessing input fields natively provided by the browser via JavaScript, their values are accessed through primitive type variables and thus affected by this limitation.

In this paper, we analyze how the popular browsers Chrome, Chromium, Firefox, Opera, and Edge handle input values in memory. We find that sensitive values almost always remain in memory several minutes longer than necessary.

We propose the JavaScript library SECPASSINPUT that simulates a non-native input for passwords. The library does not rely on variables of a primitive type, thereby giving web applications the ability to clear and overwrite values in memory. We evaluate the security benefits of SECPASSINPUT by measuring how long values remain in memory after they are no longer needed, finding that the on-screen keyboard of SECPASSINPUT guarantees immediate removal from memory after triggering SECPASSINPUT's clear operation.

Keywords: Secure Memory Handling · Secure Password Handling · Password Input · Web Browsers · JavaScript · Web Security

1 Introduction

Browsers and JavaScript allow web applications to address many security-sensitive tasks. Such tasks may operate on sensitive values, such as passwords and cryptographic key material. While access to these values is required during a specific task, they can usually be removed from the device and memory immediately after the task's completion.

Attackers who manage to get virtual or physical access to a victim's device can exfiltrate data stored in memory and on the disk. Previous work has shown that sensitive data may be recovered from memory [11] even after the device is

powered off [7] or rebooted [2]. Consequently, reducing the time sensitive data is stored on the device and in memory can significantly reduce the threat surface. In this paper, we analyze how web browsers treat passwords processed by web applications. We provide a JavaScript library `SECPASSINPUT` that can be used to explicitly clear and overwrite passwords in memory.

The way browsers handle variables of primitive data types such as strings do not give web applications any means to explicitly clear or overwrite such variables. For example, when assigning a new string value to a variable that already contains a string value, the value is newly allocated in memory and the reference in the variable is updated, retaining the old string value in memory for possible future reference. The web application has to rely on the garbage collector to eventually free the variable, and the respective memory page to be overwritten after reassignment by the operating system at some later time. There are no guarantees how quickly this will happen, so strings may remain in memory longer than necessary. The focus of our research lies on confidentiality, i.e., protecting sensitive data from leaking.

JavaScript provides byte array types such as the `Uint8Array` that allow direct access to byte values, including in-place overwriting values without copying. By using such data types, the web application can take care of overwriting sensitive data itself, thereby minimizing the time that it is kept in memory. However, values that are entered by the user are usually provided via the browser's native input fields, from which they are accessed via a string variable in JavaScript. To address this issue, we provide `SECPASSINPUT`, a JavaScript library that simulates an input element and directly uses byte arrays. Thus, `SECPASSINPUT` never stores the value in a string variable while handling the user input. This library allows web developers to protect the passwords of their users by reducing the time the passwords are stored in memory to the required minimum. Especially web applications with high security requirements can benefit from this, for example to securely derive cryptographic keys from a password or to implement a web-based password manager.

To summarize, we make the following contributions:

- We analyze how Chromium, Chrome, Firefox, Opera, and Edge handle values of native input fields in memory on Windows and Linux.
- We propose `SECPASSINPUT`, a JavaScript library providing a secure password input that allows applications to clear entered values from memory.
- We evaluate the security benefits of our `SECPASSINPUT` and compare it to native password inputs.
- We discuss the feasibility of a widespread adoption of our `SECPASSINPUT` by web application developers.

The remainder of this paper is structured as follows: In Sect. 2, we discuss related work on memory handling of applications and attacks on memory. Section 3 describes the methodology that we use for our analysis of the browsers' memory handling, discussing the results of our browser analysis in Sect. 4. We describe the threat model in Sect. 5, followed by a presentation of `SECPASSINPUT` in Sect. 6 which we evaluate in Sect. 7. Subsequently, we discuss our results in Sect. 8 and provide an outlook and concluding remarks in Sect. 9.

2 Related Work

In this section, we discuss related work that considers memory handling of applications and attacks that target or utilize memory.

2.1 Memory Content Recovery

Especially in the context of forensics, past research has analyzed ways to recover sensitive contents from memory. Maartmann-Moe et al. [11] describe approaches to recover cryptographic keys from memory. Their success depends on the system state, where a freshly booting system does not allow them to recover any keys, while they are able to recover keys at least partially from all other system states.

Halderman et al. [7] analyze the persistence of DRAM. They find that data can be recovered from DRAM several seconds after removing its power. This allows attackers to steal cryptographic keys and other sensitive data from memory without requiring special equipment, for example by rebooting the device into an attacker-controlled system.

Lee et al. [10] investigate how Android applications handle passwords in memory. They perform an initial preliminary analysis on a sample of 11 applications and find that all of them are vulnerable to leaking passwords through an unnecessary long retention in memory. One of the causes is the usage of Java's string data type, which is immutable and thus cannot be overwritten. They address these problems by providing a patched version of Android's TextView input, allowing developers to securely handle passwords in their applications' memory. Their solution for Android application memory security is very similar to our SECPASSINPUT for web applications.

2.2 Secure Memory Handling in Native Applications

Previous research has considered the secure handling of sensitive data in memory. Chow et al. [2] analyze the threat of data exposure through not overwritten memory values, finding that even a reboot may be insufficient to clear memory contents if the device is not fully powered off during a soft reboot. They propose a "secure deallocation" approach which overwrites memory contents with zeros at or shortly after the deallocation of corresponding heap or stack elements. Gondi et al. [5] propose a tool that transforms applications written in C so that they explicitly overwrite sensitive data in memory when it is no longer needed.

Göktas et al. [4] consider attacks that evade information hiding. In information hiding, sensitive memory content is "hidden" at random locations in a very large address space. The attack creates many new program threads to fill the address space, making it easier to locate other processes' memory locations.

Other approaches try to protect sensitive memory contents of applications through encryption [3, 6, 8]. For example, Götzfried et al. [6] provide a solution transparent to the process that encrypts all the process' memory contents with a key that is stored in a CPU register. Other related work provides access control on the hardware level through modifications to the instruction set [14].

2.3 Browser Memory Analysis and Vulnerabilities

Jensen et al. [9] propose the tool MemInsight that allows to analyze the memory handling of web applications and detect memory leaks in JavaScript applications, while also providing specific support for browser-based peculiarities such as the document object model (DOM). Others consider the memory behavior of web applications and methods to detect and debug memory leaks [12, 13, 16].

Wang et al. [17] analyze information leaks from the browser, which among others includes writing memory into swap or hibernation files. They propose a framework that can protect from several information leaks in the transport of the web application and its data to the browser as well as in the browser itself. To address the swapping of sensitive data, their framework frequently accesses sensitive JavaScript objects from within the web application, assuming that the frequent access prevents the operating system from moving them to swap.

3 Methodology

In this section, we describe our methodology that we used to evaluate the memory behavior of browsers. We have performed our evaluation on Linux (Debian “bullseye”) and Windows (Windows 10) and considered the browsers Firefox, Chromium, Chrome, Edge (only on Windows), and Opera.

We used a virtual machine (VM) to run the operating systems and the browsers. This allows us to capture all memory contents of the VM, independently of the operating system and possible access restrictions it enforces on memory regions. In addition, these captures include possible occurrences of sensitive values that are caused by the operating system running in the VM. Also, using a VM enables us to temporarily suspend its entire execution to take a consistent memory snapshot.

We performed evaluation runs in several configurations. Each configuration includes the operating system, the browser, the type of input to analyze (browser’s native password input, browser’s native plaintext input, or our SEC-PASSINPUT), and the processing mode. We investigated two processing modes: (a) entering the password and clearing it via backspace, and (b) instead pressing a button that derives a cryptographic key from the password through the Web Cryptography API and clears the input value afterwards by setting the input’s value to an empty string. In addition, the configuration includes the VM’s background activity and the amount of memory assigned to the VM. To increase the reliability, we repeated each evaluation run multiple times. Some evaluations had nearly no variations so we repeated them only two times. Other evaluations with higher variations in the results of the individual repetitions were repeated six times. A new password is randomly generated for each evaluation run.

We used a script to control the VM from the host through the automation framework PyAutoGUI [15] which can be used to control keyboard and mouse inputs. The script first launches the configured browser and opens the desired test website. In addition, a second tab is opened and navigated to a news website. Next, the script sets the focus back to the first tab and enters the randomly generated password into the designated input. After a delay of 90 s, the password

is cleared according to the configuration, i.e., through backspace or through pressing the button triggering a key derivation. After another delay, navigation within the same site is performed by pressing a link that leads to another page within the same origin. This causes a fresh page load, i.e., we do *not* use a single-page application (SPA) which would handle such a navigation through JavaScript. Next, a navigation operation to another news website outside of that origin is triggered by entering its URL into the URL bar of the tab. Afterwards, we close the first tab and after another delay close the browser.

During the evaluation runs, our script suspends the VM execution every five seconds and takes a full snapshot of its memory. In addition, after every performed action, such as clearing the password, our script takes a snapshot from the suspended VM immediately. In the background, the full memory snapshots are processed to find all matches of the password value in the encodings UTF-8 and UTF-16. The matches and any surrounding 1 MB of data are retained for further analysis. In addition, we store a copy of all memory ranges where a match was found earlier within the same evaluation run. The remainder of the memory snapshots is discarded.

To verify that the browser automation performed the intended operations, our script takes a screenshot of the VM display before every memory snapshot operation. We manually analyzed four significant screenshots of every evaluation run (password completely entered, password fully cleared, tab closed, browser closed). Some evaluation runs have been invalid, e.g., due to unexpected browser dialogs, lost focus, or similar reasons. We discarded and repeated these invalid evaluation runs.

4 Memory Handling in Browsers

In this section, we analyze how browsers treat native input values in memory. We used our methodology from Sect. 3 to measure the retention and number of occurrences in memory for values before and after they are removed.

Figure 1 shows the results for a native password input. The upper graph presents the results when clearing the password via backspace, i.e., the value is never accessed via JavaScript. In the lower graph, the password is accessed via JavaScript and used to derive a key with the Web Cryptography API, clearing the input value after this operation through JavaScript. All values are averaged across multiple evaluation runs with identical settings.

For all browsers, entering the password immediately causes several occurrences in memory, ranging from 3 with Firefox to more than 8 with Edge. When the password is cleared through backspace, the number immediately drops. For Firefox, the value is less than one, i.e., some evaluation runs did not retain any further match. However, at least for some runs, all evaluated browsers retain copies of the password in memory. For several browsers, copies of the password remain in memory even after the browser has been closed.

When processing the password via JavaScript, after triggering the processing and clearing, the number of copies decreases at most by a negligible amount. Only after navigating within the same site, copies are removed from memory, while still retaining significantly more copies than an unprocessed password.

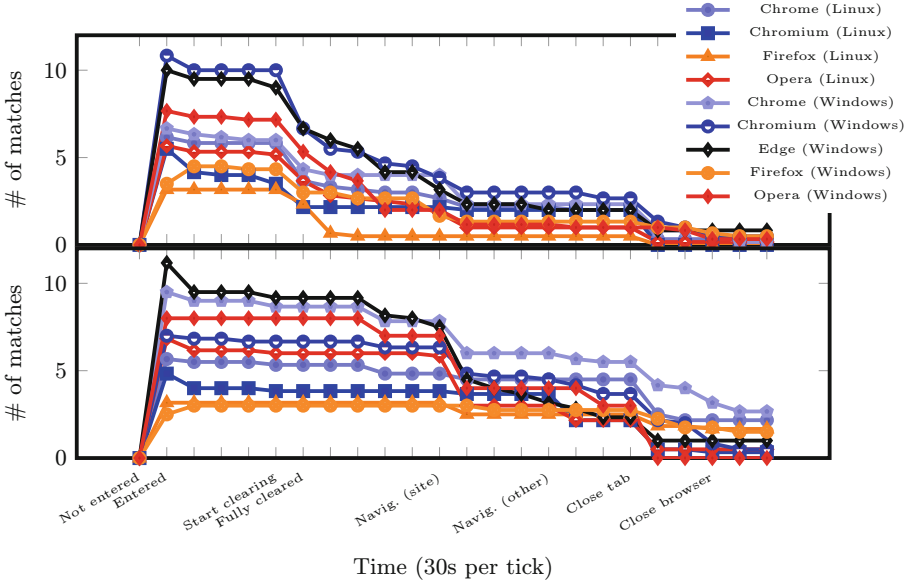


Fig. 1. Comparison of password retention in memory for native password input fields. Values are averaged across evaluation runs with identical configuration. The password is removed with backspace (upper graph) and used to derive a key using JavaScript and the Web Cryptography API (lower graph).

We also compared the browsers’ native password input (which displays dots and may indicate to the browser that its contents are particularly sensitive) and native plaintext input. Figure 2 shows the results for the plaintext input where the password is cleared via backspace and not processed with JavaScript. While the trends are similar to the password input above, all browsers produce significantly more copies in memory. This indicates that plaintext inputs provide a worse memory handling and are less suitable for sensitive inputs.

5 Threat Model

In this section, we discuss the threat model that underlies SECPASSINPUT. Within the scope of the browser, our threat model is based on the web attacker as defined by Akhawe et al. [1], i.e., an attacker that can visit web applications including those they target, host own web applications under different domains, and execute web applications in the browser of the victim. Beyond the browser, our attacker has limited local access to the victim’s devices, for example through an attacker-supplied native application that is run by the victim or through physical access. The attacker can read the memory of the browser process, which contains the variables of the interpreted JavaScript code. However, the attacker is only reading from memory at some points in time, rather than

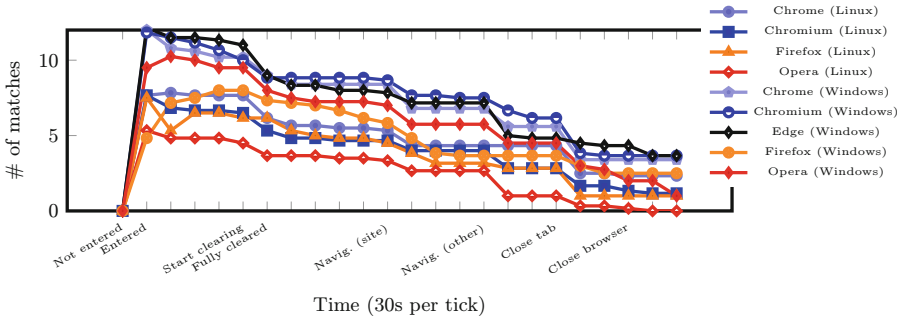


Fig. 2. Password is entered into *plaintext* input and removed via backspace.

constantly monitoring it for changes. In particular, we assume that the attacker does not read the browser’s memory after a user starts to enter a password and before it is cleared again.

While the attacker is able to *read* memory of other processes, they are not able to *modify* it. Neither can the attacker manipulate the program code or the behavior of the browser or the web application.

In the following, we describe four example scenarios to highlight the relevance of our threat model.

Cold Boot Attacks. A cold boot attack [7] can be used to extract memory contents of a running computer with physical access. If passwords remain in memory, a cold boot attack can recover such passwords when the attacker gains physical access to the still running computer later.

Malicious Application on Victim’s Device. The next scenario considers an attacker-controlled application running on the victim’s device with the goal of gaining information. Through that application, the attacker can read memory of the browser, either because the malicious application has administrative privileges or because its execution context allows to read memory of other processes of the same user. The application may for example take a memory snapshot at a regular time interval and send it to the attacker. We assume that the malicious application does not perform read operations while the user enters a password.

Swap and Hibernation Files. To extend the amount of available memory, many operating systems use an approach often called swapping, i.e., they move less-frequently accessed parts of memory contents to the disk. Unlike the physical memory, which loses its content usually a very short time after removing power, the disk is intended to persist data. Thus, when the swapped data is not encrypted and the device turned off, attackers may be able to extract sensitive information from it even a long time after the device was turned off.

The same applies to hibernation, or suspend-to-disk. There, all the contents of memory are stored to the disk, allowing to turn the device completely off



Fig. 3. Example of a SECPASSINPUT instance with activated on-screen keyboard

and restore the state of the device when it is powered on the next time. Unlike swapping approaches, suspend-to-disk stores a full memory dump that is not restricted to the less-frequently accessed parts of memory.

Crash Dumps. Operating systems may write crash dumps including parts of the browser’s memory contents into a file. The longer sensitive values are kept in memory, the more likely it gets that they are contained in such crash dumps.

6 Secure Password Input

In this section, we describe the functionality of our secure password input SECPASSINPUT. We provide a proof-of-concept implementation of SECPASSINPUT as a JavaScript library that is published on GitHub [18]. Figure 3 shows an example of a secure password input created with our library.

SECPASSINPUT relies on JavaScript’s `Uint8Array` data type, which allows explicit read and write operations for every byte contained in the array. Unlike JavaScript’s primitive string data type, which is always fully allocated in memory and immutable, the `Uint8Array` type contents can be cleared from memory by overwriting them in-place with a different value, such as zeros.

When accessing native browser input elements via JavaScript, their values are accessed through the `value` attribute, which provides a primitive `string` value that is allocated in memory and cannot be cleared by the web application. Also, besides JavaScript access, the browser stores additional copies of the inputs’ values in memory, which is outside of the control of the web application as well. Thus, we cannot rely on native input elements for SECPASSINPUT.

Instead, the SECPASSINPUT library is used to generate a secure input based on a regular `div` content element in the page’s DOM. The `div` element gets a tab index assigned, which allows user focus. Also, several child elements are dynamically added and manipulated while the user interacts with the secure input to display dots as character placeholders and a cursor that can be navigated using the arrow keys. SECPASSINPUT handles all keyboard events that are emitted while the `div` element is focused.

For each secure input, SECPASSINPUT maintains a `Uint8Array` that corresponds to the current value of the input. This array is initiated with a fixed length of 100 byte. An integer variable is used to store the actual length of the password. If the entered value exceeds the size of the array, a new array with the

size of the next multiple of 100 byte is allocated. The value is then copied into the new, larger array and the previous array securely overwritten with zeros.

A web application can access the password from `SECPASSINPUT` in two ways. Firstly, it can directly access a JavaScript object that contains the array with the password value and the integer variable containing its length. When using this method, the application itself needs to handle the length of the password, which may differ from the length of the byte array. Secondly, `SECPASSINPUT` provides a function that provides the application with a copy of the value in another array of the proper length. Using this method, it is the responsibility of the application to securely overwrite this copied array using another function provided by `SECPASSINPUT` as soon as this value copy is no longer required.

As an alternative to the preallocated array of a fixed size that is extended when needed, it would be possible to implement `SECPASSINPUT` with an array of a fixed size that is replaced by a new array on every input operation by the user. In that case, every change of the value would require a new array to be allocated, the old array value to be copied, and the old array to be securely overwritten. At the same time, that implementation would not require new copies when the application needs a properly-sized array for processing. However, we assume that the most common use case for `SECPASSINPUT` is a user entering a password once, causing a lot of change operations of the password value, followed by the processing by the web application, which requires a very low number of accesses to the password. In particular, the password is usually not updated during multiple processing steps, which means that a single copy of the value can be used in all steps. For this use case, our implemented approach of the fixed-sized array is the better trade-off.

Each keyboard event that corresponds to a printable character or a delete operation causes an update of the internal state of the secure password input as explained before. Besides printable characters, `SECPASSINPUT` maintains a cursor position to support navigation within the input. This allows users to navigate within the input field using the arrow keys, optionally combined with the `Ctrl/Command` key to move to the start or end of the input.

On-Screen Keyboard. `SECPASSINPUT` provides an on-screen keyboard to use mouse or touch inputs rather than key presses. The on-screen keyboard is displayed underneath every secure input field when activated by the user.

Developers that use `SECPASSINPUT` can enforce using the on-screen keyboard for individual secure input instances, i.e., disable handling of input from the regular keyboard. This can provide additional protection, for example from keyloggers that capture all physical keyboard input on a victims' device.

Disabling Visual Feedback. By default, `SECPASSINPUT` displays dots that correspond to the number of characters entered, including a cursor indicating the current position at which typed characters are inserted. This behavior is identical to browsers' native password input fields, which minimizes the risk of user confusions. However, to increase the security of the secure inputs in the presence

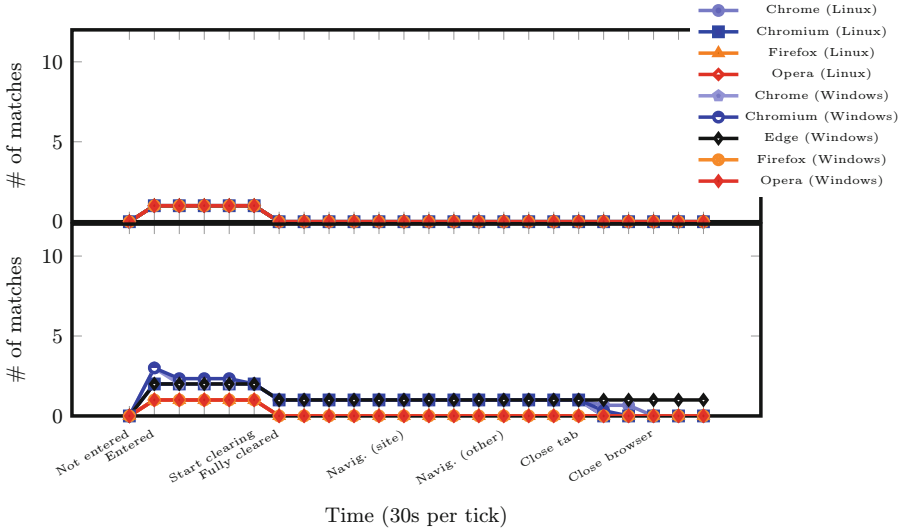


Fig. 4. Memory occurrences of passwords entered through SECPASSINPUT: via on-screen keyboard (upper graph) and virtual USB keyboard (lower graph).

of attackers that observe the user’s screen while typing a password, SECPASSINPUT offers an option that disables the displaying of dots. When activated, the input does not give any visual feedback while the user is typing, preventing the password length to be exposed through the visual state of the input.

7 Evaluation of SECPASSINPUT

To evaluate the effectiveness of our secure password input, we tested our proof-of-concept web application in Google Chrome, Chromium, Firefox, Opera, and Edge and verified that the password is immediately overwritten in memory.

Figure 4 shows the results for SECPASSINPUT when entering a password using the on-screen keyboard (upper graph) and the regular keyboard (lower graph). With the on-screen keyboard, all browsers in both operating systems cause only a single occurrence of the password in memory which is immediately gone after clearing the input. Thus, when using the on-screen keyboard, SECPASSINPUT guarantees that the password is immediately removed completely from memory as soon as it is no longer needed.

When using a regular keyboard, Chrome, Chromium, and Edge cause 2 and up to 3 memory occurrences of the password, some of them remaining after clearing the password. Only Opera and Firefox exhibit the desired behavior of causing a single memory occurrence that is immediately gone after the input is cleared. Thus, using a regular keyboard that causes key presses handled by the operating system and browser leaks the password in some of the tested browsers, implying that full protection is only possible when using the on-screen keyboard.

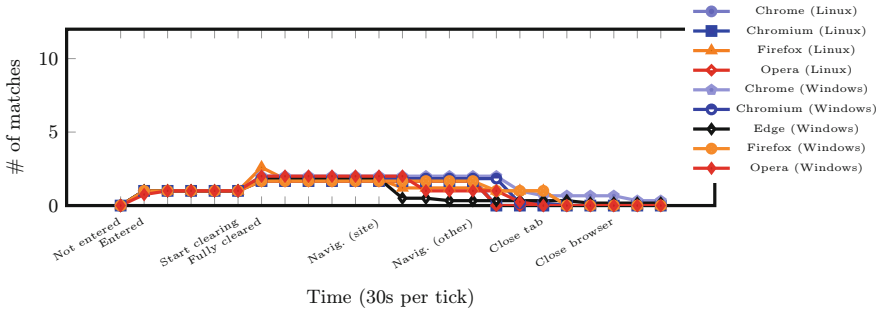


Fig. 5. Memory occurrences of password entered via SECPASSINPUT with on-screen keyboard. A key is derived via the Web Cryptography API.

In Fig. 5, we present the results for SECPASSINPUT using the on-screen keyboard and deriving a key using the Web Cryptography API. Until clearing the password, there is no difference to the evaluation without processing. After the clearing operation, which is preceded by the key generation operation, at least one and up to two additional copies are caused by the key derivation operation. These remain in memory much longer than necessary, being cleared only half a minute after navigation within the same site (Edge) or even after navigation to an external site. For Chrome on Windows, one match remained even after closing the browser in some of the evaluation runs. Thus, using the Web Cryptography API causes additional copies of the handled value which are outside of the control of SECPASSINPUT.

8 Discussion of SECPASSINPUT

In this section, we discuss SECPASSINPUT and our evaluation methodology. Also, we discuss aspects that need to be considered before deploying the library.

Password Processing. The security benefit of SECPASSINPUT depends on the purpose of the password input and the further processing by the web application. If the password is sent to the server in plain text, it needs to be written to a native JavaScript string variable, voiding most of the benefits SECPASSINPUT aims to provide. As our evaluation revealed, the derivation of a cryptographic key via the Web Cryptography API can also leak the password into long-retained copies in memory in some browsers.

One solution can be provided through incorporating cryptographic operations into SECPASSINPUT. For example, the library can be extended with functionalities to derive cryptographic keys from the password without exposing it to any APIs or functions that may cause additional, not securely cleared copies of the password.

Potential for Wrong Usage by Users. SECPASSINPUT may cause confusion for new users, as it does not support functionalities of native inputs, such as clipboard pasting or password manager filling. Consequently, users may accidentally input the password into unintended fields. This can be addressed by visible, clear warnings below SECPASSINPUT inputs.

Accessibility Software. As SECPASSINPUT relies on a simulated non-native input, it does not support text input features, including most password managers and accessibility software. Thus, in the current architecture, it may be inaccessible to some people. As a workaround, SECPASSINPUT can be extended with an insecure fallback to a native input. While this voids the security advantages of SECPASSINPUT for the affected users, it solves its accessibility issues.

Because browsers do not recognize SECPASSINPUT as an input field, touch-based devices do not display the keyboard. SECPASSINPUT's on-screen keyboard is touch-compatible, resolving this issue.

Native Browser Support for Clearing Values. SECPASSINPUT is a JavaScript library and thus limited to the capabilities the language provides within the browser. Thus, it is not always guaranteed that this approach is sufficient to remove all copies of the password from memory in every environment, especially when the value is used for further processing. As the evaluation showed, while SECPASSINPUT works reliable in some browser environments, it does not guarantee the passwords to be fully removed from memory in all browsers and scenarios.

A more reliable approach would be native browser functionalities that allow applications to securely erase values from memory and handle passwords in a secure way, or to mark values as sensitive so they are securely overwritten by the browser automatically. However, SECPASSINPUT can be deployed without further requirements on the client side, thus fills the gap and builds a foundation for further work to provide such functionality natively in the browser.

Regarding values processed via JavaScript, browsers can add a method to allow web applications to mark values such as passwords as sensitive. Such values can then be cleared by the browser in a memory-secure way.

On the operating-system level, a security mechanism can be introduced that allows processes to be marked as sensitive. The operating system can then take care of securely overwriting the processes' memory as early as possible, including immediate overwrites of deallocated memory values.

Relevance of the Threat Model. The attacker considered in our threat model has very strong capabilities, including access to most parts of memory while the password is not entered. This can be used to read any data handled by the web application, for example cryptographic keys derived from the password or session identifiers in cookies or the local storage. Consequently, protecting the password from an attacker may not be sufficient if sensitive data derived from it still leaks to the attacker.

To address this limitation, the web application can further reduce the time such sensitive data is contained in memory, for example by repeating the key derivation every time the key is required. However, this increases the number of times that the password is required, which may provide a larger threat surface for attacks on the password input.

To protect from exploitation of leaked session identifiers, IP-based access restrictions can be enforced. Thus, extracting such session identifiers from the victim's device is less critical than extracting the password, which allows to arbitrarily request new valid session identifiers.

9 Conclusion

In this paper, we analyzed the retention time of sensitive data in memory of popular web browsers. Our research revealed that passwords entered in browsers may stay in the computers' memory for a long time even after closing the browser and almost always remain in memory longer than required.

As a protective measure, we proposed SECPASSINPUT, a JavaScript library that allows web applications to explicitly clear password input values from memory. We implemented a proof of concept of SECPASSINPUT as a JavaScript library. Our evaluation showed that SECPASSINPUT with its on-screen keyboard can guarantee that passwords are immediately cleared from memory as soon as they are removed from the input in all tested browsers. Using APIs such as the Web Cryptography API in some browsers causes additional copies in memory that cannot be cleared by the web application nor SECPASSINPUT.

As future work, we intend to research more extensive protective strategies, which may be incorporated directly into the browser or operating system. Also, regarding usable security, we intend to integrate password advice and password strength validation into SECPASSINPUT.

References

1. Akhawe, D., et al.: Towards a formal foundation of web security. In: 23rd IEEE CSF 2010, pp. 290–304 (2010)
2. Chow, J., et al.: Shredding your garbage: reducing data lifetime through secure deallocation. In: 14th USENIX Security 2005 (2005)
3. Enck, W., et al.: Defending against attacks on main memory persistence. In: 24th ACSAC 2008, pp. 65–74 (2008)
4. Göktas, E., et al.: Undermining information hiding (and what to do about it). In: 25th USENIX Security 2016, pp. 105–119 (2016)
5. Gondi, K., et al.: SWIPE: eager erasure of sensitive data in large scale systems software. In: 2nd CODASPY 2012, pp. 295–306 (2012)
6. Götzfried, J., et al.: RamCrypt: kernel-based address space encryption for user-mode processes. In: 11th Asia CCS 2016, pp. 919–924 (2016)

7. Halderman, J.A., et al.: Lest we remember: cold boot attacks on encryption keys. In: 17th USENIX Security 2008, pp. 45–60 (2008)
8. Henson, M., Taylor, S.: Memory encryption: a survey of existing techniques. *ACM Comput. Surv.* **4**, 53:1–53:26 (2013)
9. Jensen, S.H., et al.: MemInsight: platform-independent memory debugging for JavaScript. In: 10th ESEC/FSE 2015, pp. 345–356 (2015)
10. Lee, J., Chen, A., Wallach, D.S.: Total recall: persistence of passwords in Android. In: 26th NDSS 2019 (2019)
11. Maartmann-Moe, C., Thorkildsen, S.E., årnes, A.: The persistence of memory: forensic identification and extraction of cryptographic keys. *Digit. Investig.* **6**, S132–S140 (2009)
12. Pienaar, J., Hundt, R.: JSWhiz: static analysis for JavaScript memory leaks. In: CGO 2013, pp. 11:1–11:11 (2013)
13. Rudafshani, M., Ward, P.A.S.: LeakSpot: detection and diagnosis of memory leaks in JavaScript applications. *Softw. Pract. Exp.* **1**, 97–123 (2017)
14. Shi, W., et al.: InfoShield: a security architecture for protecting information usage in memory. In: 12th HPCA-12 2006, pp. 222–231 (2006)
15. Sweigart, A., et al.: PyAutoGUI (2021). pypi.org/project/PyAutoGUI/
16. Vilk, J., Berger, E.D.: BLeak: automatically debugging memory leaks in web applications. In: 39th SIGPLAN 2018, pp. 15–29 (2018)
17. Wang, F., Mickens, J., Zeldovich, N.: Veil: private browsing semantics without browser-side assistance. In: 25th NDSS 2018 (2018)
18. Wichmann, P.: SecPassInput: secure password input library (2023). <https://github.com/wichmannpas/sec-pass-input>



Bl0ck: Paralyzing 802.11 Connections Through Block Ack Frames

Efstratios Chatzoglou^{1,2} , Vyron Kampourakis³ ,
and Georgios Kambourakis¹ 

¹ Department of Information and Communication Engineering,
University of the Aegean, 83200 Karlovasi, Greece

{efchatzoglou,gkamb}@aegean.gr

² TwelveSec, 15234 Athens, Greece

³ Department of Information Security and Communication Technology,
Norwegian University of Science and Technology, 2802 Gjøvik, Norway
vyron.kampourakis@ntnu.no

Abstract. Despite Wi-Fi is at the eve of its seventh generation, security concerns regarding this omnipresent technology remain in the spotlight of the research community. This work introduces two new denial of service (DoS) attacks against contemporary Wi-Fi 5 and 6 networks. Differently from similar works in the literature which focus on 802.11 management frames, the introduced assaults exploit control frames. Both these attacks target the central element of any infrastructure-based 802.11 network, i.e., the access point (AP), and result in depriving the associated stations of any service. We demonstrate that, at the very least, the attacks affect a great mass of off-the-self AP implementations by different renowned vendors, and they can be mounted with inexpensive equipment, little effort, and a low level of expertise. With reference to the latest standard, namely, 802.11-2020, we elaborate on the root cause of the respected vulnerabilities, pinpointing shortcomings. Following a coordinated vulnerability disclosure process, our findings have been promptly communicated to each affected AP vendor, already receiving positive feedback, as well as, at the time of writing, a reserved common vulnerabilities and exposures (CVE) identifier, namely CVE-2022-32666.

Keywords: Network security · IEEE 802.11 · Wi-Fi · DoS · Vulnerabilities · Attacks · CVE

1 Introduction

Over the past 26 years, the IEEE 802.11 standard, commonly referred to as Wi-Fi, continuously evolved by improving the speed, stability, and security of wireless local area network (WLAN) connections. The seventh generation (Wi-Fi 7) of this widespread technology is already on the nearby horizon, following Wi-Fi 6E, which added support for the 6 GHz spectrum to Wi-Fi 6. IEEE 802.11 networks are omnipresent, not only in public places like coffee shops, libraries,

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 250–264, 2024.

https://doi.org/10.1007/978-3-031-56326-3_18

airports, hotels, and universities but also in houses and corporate and enterprise premises. Moreover, Wi-Fi is a key enabler for smart cities; every “thing”, including lights, cameras, meters, and vehicles may be connected to the Internet through 802.11 links.

On the other hand, as with any other mainstream networking technology, Wi-Fi is an alluring target for malicious parties, thus constantly under the bombsight of threat actors. In this setting, denial of service (DoS) attacks on Wi-Fi networks may inflict a variety of real-world harms, ranging from simple annoyance or discomfort, say, due to the loss of Wi-Fi connectivity in a coffee shop, to the temporary loss of critical services, e.g., security cameras go offline. Overall, it is not to be neglected that accessing a Wi-Fi network domain does not mandate physical access to a network jack or cable. The opponent can be anywhere in the vicinity or further afield, depending on the strength/type of the wireless signal/equipment.

Excluding jamming attacks exercised on the physical layer, where malicious nodes block legitimate communication by causing intentional interference, layer 2 oriented DoS against Wi-Fi networks remains a highly interesting and timely subject. Legacy attacks of this kind are the so-called deauthentication and disassociation ones, easily exercised in Wi-Fi Protected Access (WPA) and WPA2 networks with cheap equipment, easy-to-find and use software tools, and a script-kiddie level of expertise. Such attacks basically rely on the abuse of certain types of 802.11 management frames, in the commonest case, (de)authentication and (dis)association.

Nevertheless, after the introduction of Protected Management Frames (PMF), also known as “robust”, with amendment 802.11w and its inclusion in the IEEE 802.11-2012 standard onward, this threat has received limited attention in the literature. That is, although a handful of recent studies have assessed the potential of DoS attacks through management frames in the presence of PMF [1–3], to our knowledge no study in the literature has provided facts that DoS is feasible in contemporary 802.11ac (Wi-Fi 5) and 802.11ax (Wi-Fi 6) networks through the abuse of specific control frames. Actually, potential security issues owed to certain control frames have been discussed back in 2008 in the context Wi-Fi 4, specifically in the process of compiling the IEEE 802.11n amendment [4–6], and subsequently have been addressed in newer 802.11 standards.

Our Contribution: The work at hand introduces two zero-day DoS attack against contemporary Wi-Fi networks. The attacks, jointly coined as “Bl0ck”, take advantage of specific 802.11 control frames and can be mounted against 802.11ac or 802.11ax networks with minimal effort and inexpensive equipment. The effect of the attacks is substantial, given that they quickly paralyze any active service on the targeted stations (STA); the STA will remain associated with the access point (AP), nevertheless unable to use any service. The only way to revive the STA is to manually disconnect it from the network and let it re-associate with it, always assuming that in the meantime the attack has ceased. We have evaluated the assaults against an assortment of modern off-the-shelf

APs by different renowned vendors. By following a Coordinated Vulnerability Disclosure (CVD) process, we reported the corresponding vulnerabilities to each affected vendor. At the time of writing, two vendors have reserved a common vulnerabilities and exposures (CVE) ID, namely CVE-2022-32666, to classify the vulnerability.

The rest of this paper is structured as follows. The next section provides the necessary background on the control frames of interest and formulates the problem. Sections 3 and 4 detail the testbed and the attacks, respectively. The last section concludes and provides directions for future work.

2 Background and Problem Definition

All 802.11 frames fall under one of the three types, namely management, control, or data. The frame's type is designated by the homonymous 2-bit field in the frame's header: 00, 01, 10 for management, control, or data frames, respectively. There is also a 4-bit subtype field that indicates the specific type of management, control, or data frame. Control frames, which are the focus of this work, control access to the wireless medium and provide frame acknowledgment, therefore are used to support the delivery of all the other frame types. Control frames contain no body, only a header and trailer.

This work assumes an infrastructure-based 802.11ac or 802.11ax network, namely a Basic Service Set (BSS) comprised of an AP and a number of associated STAs. We concentrate on Block Ack Request (BAR) and Block Ack (BA) frames, having a type/subtype value of 01/1000 and 01/1001, respectively. Precisely, introduced in amendment 802.11e-2005, block acknowledgment (ack) frames are used to confirm receipt of a block of Quality of Service (QoS) data frames. Recall that any data frame with a value of 1 in the QoS subfield of the header's Subtype field is referred to as QoS data. A block may be started within a polled transmission opportunity (TXOP), i.e., an interval of time during which a particular STA is authorized to start frame exchange sequences. That is, the originator, say, an STA will transmit multiple QoS data frames (a contention-free burst) followed by a BAR to the recipient, say, the AP. The latter party will respond with a BA frame, which includes a bitmap that indicates which frames were received. This means that only the frames indicated by the bitmap with a zero value were not received and should be retransmitted in the next block by the originator. Simply put, by decreasing protocol overhead, i.e., lessening the number of single ack frames that need to be sent, the network throughput is increased. Block ack frames are acknowledged or not depending on the policy, namely, delayed or immediate, respectively.

Typically, the block ack mechanism is initialized by an exchange of ADDBA Request/Response action management frames. Specifically, for establishing a block ack agreement, the originator sends an add block ack (ADDBA) request frame to the recipient. The recipient replies with an ADDBA response frame. These (always acknowledged) frames carry information about the capabilities of each participant, including the buffer size, whether frame aggregation (A-MSDU) is supported, the block ack policy to be used, etc. Every block ack

session is unidirectional. A block ack agreement is terminated by the originator sending a delete block acknowledgment (DELBA) frame. Figure 1 provides a synoptic view of the block ack mechanism, including ADDBA session setup and termination.

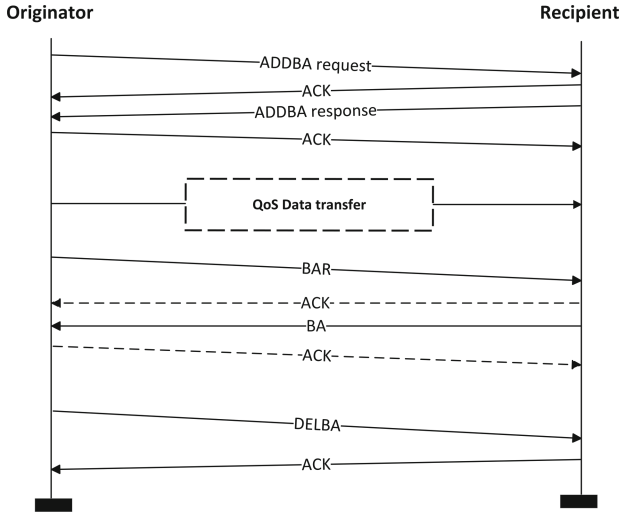


Fig. 1. Overview of the ADDBA life cycle

Both the BAR and BA frames include an Information field of variable length, which is structured as follows.

- *Information field in BAR:* It carries a 2-octet Block Ack Starting Sequence Control subfield, which comprises two subfields: (i) a 12-bit Starting Sequence Number (SSN), which contains the sequence number (SN) of the first MAC service data unit (MSDU) for which this BAR frame is sent, and (ii) a 4-bit Fragment Number (FN) subfield which is set to 0.
- *Information field in BA:* It comprises a Block Ack Starting Sequence Control subfield (as in the case of BAR above) and an 8-octet Block Ack Bitmap subfield. As already pointed out, the latter subfield is used to signal the received status of at most 64 MSDUs, aggregated or not. Precisely, every bit set to 1 in the bitmap acknowledges the reception of a single MSDU in SN ascending order. The first bit of the bitmap corresponds to the MSDU with the SN that matches the SSN subfield of the Block Ack Starting Sequence Control subfield.

It is important to note that with reference to § 10.25 of the latest standard [7], “the number of frames in the block is limited, and the amount of state that is to be kept by the recipient is bounded.” Specifically, the standard states that for each block ack agreement, a receive reordering buffer shall be kept. This buffer maintains a record, which includes the following pieces of data:

- Any buffered received MSDU, aggregated or not, that not yet been forwarded to the next MAC process.
- A WinStartB parameter signifying the value of the SN subfield of the first not yet received MSDU. This parameter is initialized to the SSN subfield value of the ADDBA request frame that matches the corresponding ADDBA response frame. Recall that the SN is part of the Sequence Control field existing in any management or data frame.
- A WinEndB parameter, denoting the highest SN awaited to be received in the current reception window. This parameter is initialized to WinStartB + WinSizeB - 1.
- A WinSizeB parameter, denoting the size of the reception window. It is set to the smaller of 64 and the value of the Buffer Size field of the ADDBA response frame that established the block ack agreement. The Buffer Size field is included in the Block Ack Parameter Set field of an ADDBA response frame.

Moreover, the recipient keeps a temporary block ack record known as Scoreboard Context Control, which includes a bitmap indexed by SN subfield. The lowest and highest SNs delineated in the bitmap are called WinStartR and WinEndR, respectively. In addition, the WinSizeR parameter designates the maximum transmission window size, which is set similar to WinSizeB.

The originator also maintains a transmit buffer with the WinStartO and WinSizeO parameters. The former parameter designates the SSN of the transmit window, while the second the number of buffers negotiated in the block ack agreement. A transmit buffer is released after receiving a BA frame from the recipient.

Potentially Exploitable Vulnerability: Given the above, and considering that neither a BAR nor BA frame is robust (protected) [7], any device may be prone to attacks where the opponent transmits spoofed BAR or BA frames with random Block Ack SSN or Bitmap or both against a target. For instance, an arbitrary SSN carried by such a spoofed BAR frame may muddle the recipient buffer or disorder the scoreboard context at the recipient. Such an assault is anticipated to require minimal effort from the attacker, namely, the injection of a few tens of spoofed BAR frames would suffice, and can be mounted with low-cost, off-the-shelf equipment and a handful of lines of code.

On the other hand, this vulnerability is dealt with by the current standard [7] if the communicating parties indicate support for protected block ack. Based on § 10.25.7 of the standard, this is done by setting the flags Management Frame Protection Capable (MFPC), Management Frame Protection Required (MFPR), and Protected Block ack Agreement Capable (PBAC) of the Robust Security Network (RSN) Capabilities field to 1; the latter field is contained in the Robust Security Network Element (RSNE). In this case, the recipient must not update the WinStartB parameter based on the SSN information conveyed by a BAR frame, specifically in the Starting Sequence Control subfield included in the BAR Information field. Instead, for advancing the window, the originator must send a robust (protected) ADDBA Request frame.

On the downside, at the time of writing, we are unaware of any existing implementations of protected block ack; note that this feature must be supported and enabled by both the initiator and recipient. This means that virtually all 802.11ac or 802.11ax capable devices, even the most recent ones, are susceptible to the above-mentioned vulnerability. Indeed, Sect. 4 assesses this attack vector and demonstrates its feasibility on modern, off-the-self devices of diverse renowned vendors. Given that the AP is the pivotal entity in any 802.11 infrastructure-based network, therefore by DoSing it the attacker can possibly deprive all STAs of receiving services, in Sect. 4 the attacks are evaluated against the AP, not the STA.

3 Testbed

To assess the potential of the DoS attack portrayed in the previous section, we set up a testbed comprising seven APs by several different vendors. For reasons of completeness, we also included the commonly accepted user space daemon for APs, namely host access point daemon (hostapd); however, it should be noted that the attacks are due to chipset implementation, namely the Wi-Fi driver, not the hostapd software per se. The key features of the employed APs are summarized in Table 1. Note that with reference to the second column of the table, we selected APs from all major chipset vendors. All the APs were tested in both Wi-Fi 5 and 6 protocol versions and the protections mandated by Wi-Fi Protected Access (WPA2) and WPA3 certifications.

Table 1. List of APs used in our experiments. Vendors who requested to remain undisclosed until all the vulnerabilities have been patched due to policy reasons related to the Bugcrowd vulnerability disclosure platform are shown with a star exhibitor.

AP	Chipset and firmware version	Wi-Fi 6
Asus RT88AXU	Broadcom v388_20518	✓
Vendor*	Intel (version undisclosed)	✓
TP-Link AX10v1	Broadcom v1.221103	✓
D-Link DIR X-1560	MediaTek v1.10WWB09	✓
Zyxel NWA50AX	Mediatek v6.25(ABYW.10)C0	✓
Huawei AX3	Huawei v11.0.5.5	✓
Linksys MR7350	Qualcomm v1.1.7.209317	✓
Hostapd v2.10	Intel AX200 v22.140.0.3	✓

Regarding the STAs, we utilized four different ones, namely a wpa_supplicant v2.10, an Intel AX200 wireless network interface controller (WNIC) on both Windows 10 and Ubuntu 20.04, a Samsung Galaxy A52s 5G, and an iPhone X. The attacker and the desktop STAs operated on a machine with 16 GB of RAM

and an eight-core CPU. Moreover, the attacker possessed an Alfa AWUS036ACH (802.11ac) WNIC for frame injection. This WNIC operated on an Ubuntu v18.04 machine with firmware version 5.2.20. Python3 and the Scapy library in v2.4.3 were used for coding the attack scripts. For reasons of reproducibility, the latter are made available at a public GitHub repository [8] and in the Appendix.

4 Attacks

This section presents two effective attack cases which take advantage of BAR or BA frames, as explained in Sect. 2. Both the attacks were performed against all APs listed in Table 1. Each time, all four available STAs were associated with the tested AP. Based on our observations, all the APs always conduct an ADDBA transaction with the associated STAs. It is important to mention that, as a first investigative step, we also relied on the well-respected WPAXFuzz fuzzing tool [9] for specifically fuzzing BAR and BA control frames against each AP.

4.1 Attack I: Blocking Any Single STA

This attack, illustrated in Fig. 2, exploits a spoofed BAR frame, i.e., the transmitter’s MAC address (address 1) of the frame is spoofed to that of the legitimate STA and the receiver’s MAC address (address 2) to that of the AP. As observed from the figure, the Block Ack Starting Sequence Control subfield of the spoofed frame carries an FN value equal to 4 and an arbitrary SSN lower than 2^{12} , given that the SN subfield provides a 12-bit space. Recall from Sect. 2 that based on the latest standard [7], for BAR and BA frames, the FN should be equal to 0. It is also to be noted that the random SSN contained in the Block Ack Starting Sequence Control subfield of the BAR frame is irrelevant to the SNs of any QoS data frames transmitted previously from the AP toward the legitimate STA. For example, the SNs of the sent QoS data frames from the AP to the STA are from 100 to 120, while the Block Ack Starting Sequence Control subfield in the spoofed BAR frame carries an SSN equal to 1175. In any case, the attack works with unsolicited BAR frames, i.e., it is not necessary for the legitimate STA and AP to have previously gone through a QoS data exchange. As shown in Fig. 2, the AP always responds to such BAR frames with a BA one carrying an all-zero Block Ack Bitmap subfield. Nevertheless, this behavior is discrepant with the current context, given that the BAR frame is unsolicited and the AP’s buffer contains no relevant information.

After sending a few tens of such BAR frames, the targeted STA was rendered incapable of communicating with the AP, i.e., it could send or receive any QoS data frame, although it remained associated with the AP. The attack was repeated several times, and in most cases, even after it was ceased, the STA remained in a state of complete QoS service paralysis. This situation can only be fixed by manually disconnecting the STA and next re-associating it with the AP. Naturally, the aggressor can relaunch the assault at any time aiming at either the same or different STA.

The exploit given in Listing 1.1 in the Appendix is drastic the same way for all the affected APs in our testbed but one. That is, as given in listing 1.3, specifically for the Zyxel AP, the attack works differently. Namely, the attacker needs to first eavesdrop on the QoS data frames exchanged between the AP and the STA to learn the SNs of the transmitted frames. After that, they use one of these (valid) SNs as the SSN in the Information field in the spoofed BAR frame. Finally yet importantly, for APs equipped with Intel or MediaTek chipsets, the attack can be also successfully carried out using unsolicited BA frames instead of BAR ones.

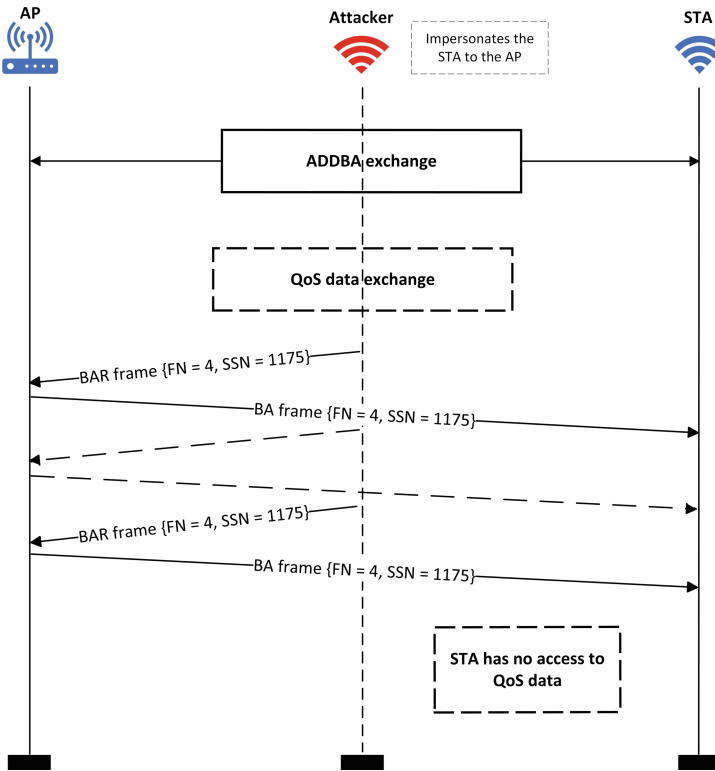


Fig. 2. Synoptic illustration of attack I. QoS exchange in the dotted-line rectangle is not a requirement.

4.2 Attack II: Blocking All STAs at Once

The outcome of this attack is that all the associated STAs with the targeted AP suffer a QoS service disruption. Precisely, as seen in Fig. 3, this case exploits a spoofed BA frame transmitted from the attacker impersonating an STA to the AP. Significantly, this assault is effective even if the transmitter’s MAC address (address 1) of the frame is set to a random, well-formed value. Simply put, the

attacker does not need to impersonate an already associated STA to the AP. With reference to the exploit in Listing 1.2 in the Appendix, the Block Ack Starting Sequence Control subfield of the frame contained an FN equal to 4 and an arbitrary SSN. The Block Ack Bitmap subfield was also set with random binary values, 1 or 0. As with the attack of Subsect. 4.1, a few tens of attack frames are enough to cripple the targeted AP, rendering it unable to serve all the associated STAs; however, the STAs remain connected to the unresponsive AP. Shortly after the attack stops, the AP manages to restore normal operation without user intervention.

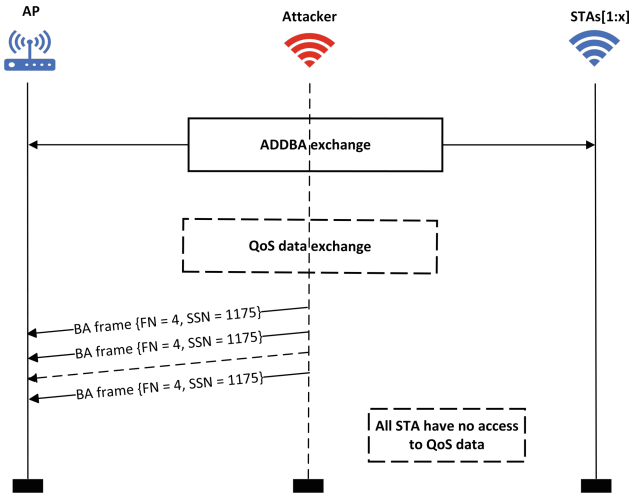


Fig. 3. A bird’s eye view on attack II. QoS exchange in the dotted-line rectangle is not a requirement, and the attacker may use a random transmitter’s MAC address.

4.3 Discussion

The two middle columns of Table 2 summarize the efficacy of each attack to each tested AP. Overall, all but two APs were found vulnerable to attack I and three of them to attack II. As already mentioned, through a CVD procedure, all the affected vendors have been informed about this vulnerability. Although at the time of writing all the vendors have acknowledged or are still investigating the vulnerability along with the corresponding exploits, only MediaTek has already reserved a CVE ID, namely CVE-2022-32666, to communicate the flaw related to the attack of Subsect. 4.1. Subsequent information, including CVE IDs and vendors’ firmware patches, will be reported through the GitHub public repository at [8].

Table 2. Outcome of each attack to the APs of our testbed. Hostapd behavior depends on the particular WNIC, in our case Intel AX200. The star exhibitor designates that this attack is effective with both BAR and BA frames. A dash in the rightmost column indicates that the vendor has not yet reserved a CVE-ID.

AP	Attack I	Attack II	CVE ID
Asus RT88AXU	✗	✓	–
Vendor	✓*	✗	–
TP-Link AX10v1	✗	✓	–
D-Link DIR X-1560	✓*	✗	CVE-2022-32666
Zyxel NWA50AX	✓	✗	CVE-2022-32666
Huawei AX3	✓	✗	–
Linksys MR7350	✓	✗	–
Hostapd	✓	✓	–

It is clear that both the attacks of Subsects. 4.1 and 4.2 are due to the Wi-Fi driver, i.e., the respective chipset’s firmware. This in turn means that any device which incorporates the same driver will be most probably vulnerable as well. This is for instance the case with Hostapd which relies on the particular WNIC. Naturally, the root cause of these latent vulnerabilities is owed to business logic flaws, that is, design and implementation defects in software applications. As explained further down, among others, this may be due to either a software bug, a misconfiguration, an unwitting supposition, or a misconception regarding the standard during the software development phase.

Specifically for attack I, and with reference to Sect. 2, the problem stems from the fact that the BAR and BA frames are unprotected, therefore an opponent can confuse or disorder the recipient buffer. This can be done either by instructing the recipient buffer to erroneously advance the `WinStartB` parameter or forcing it to an endless loop of SN checks, which ultimately leads to paralysis. Precisely, § 10.25.6.6.3 of the latest standard [7] specifying the operation for each received BAR frame, mentions as a first step that “If $WinStartB < SSN < WinStartB + 2^{11}$, then in a block ack agreement that is not a protected block ack agreement, set $WinStartB = SSN$.” Therefore, in this respect, the standard does not provide any protection, except for robust (protected) block ack agreements as detailed in Sect. 2.

Even in the latter case, however, the current standard does not specifically differentiate between an ADDBA request made to update the `WinStartB` value and another to update the parameters of the active BA agreement. Precisely, in § 10.25.2 of the standard [7] regarding the setup and modification of the block ack parameters, it is mentioned that “The originator STA may send an ADDBA Request frame in order to update block ack timeout value.”. On the other hand, in § 10.25.7 detailing the protected block ack agreement, the standard [7] defines that “Upon receipt of a valid robust ADDBA Request frame for an established

protected block ack agreement whose traffic identifier (TID) and transmitter address are the same as those of the block ack agreement, the STA shall update its WinStartR and WinStartB values based on the SSN in the robust ADDBA Request frame according to the procedures [...], while treating the SSN as though it were the SSN of a received BAR frame. Values in other fields of the ADDBA Request frame shall be ignored.” From the above-mentioned passages, and especially with reference to the last sentence of the second passage, it is not apparent how an implementation can discern between the two kinds of ADDBA requests.

On the other hand, as explained in Subsect. 4.1, the problem with the affected AP implementations is that they even accept unsolicited BAR frames. Namely, any BAR frame which is not sent in the context of a QoS data frame transmission should be outright (and silently) dropped. If not, the existing vulnerability becomes more easily exploitable. Even more, the affected implementations accept unspecified FN values, say, 4, while the current standard stipulates that this parameter for BAR and BA frames must be set to zero. Generally, any frame that carries an unspecified value in any of its fields should be ignored.

Roughly the same observations apply to attack II too. That is, this attack (as well as a specific variation of attack I detailed in Subsect. 4.1) exploits unsolicited BA frames, which are accepted and parsed at the recipient side, while they should not. Put simply, a BA frame should only arrive in response to a BAR one. From our experiments, it seems that some implementations fail to perform this check and are left exposed to DoS attacks. Similar to attack I, these implementations also neglect to ignore BA frames that carry an unspecified FN value. On top of everything else, attack II can be exercised using a random transmitter’s MAC address. Nevertheless, it is clear that an AP implementation should not accept and process BA frames that originate from a MAC address not already associated with the AP.

It is obvious that attack II has a much greater impact vis-à-vis attack I. This is not only because the attacker can paralyze all the associated STAs at once, but equally important because, if exercised as a first step, this attack allows the evildoer to escalate its malicious scheme through more dangerous methods. For instance, think of a public Wi-Fi hot spot, on which the opponent launches attack II; this will result in all the associated STAs losing Internet access. Since no STA disconnection takes place, end-users will probably look for another Wi-Fi service, which the attacker will happily provide by exercising an evil twin, or more generally, a rogue AP configured in open access mode. Once the victims connect to the rogue AP, the attacker can use phishing techniques to acquire user credentials and other private information [10]. The assailant can also perform a deauthentication attack back-to-back with attack II to force all or specific STAs to disconnect and hopefully automatically connect to the attacker’s rogue AP. We did create such a scenario, and we realized that following attack II, the attacker will need a handful of deauthentication frames to disconnect any STA, even those that operate on WPA3.

Overall, the key takeaways of the above discussion are that the introduced attacks in Subsects. 4.1 and 4.2:

- Will most probably fly under the radar. That is, both of them present a small footprint (a few tens of BA or BAR frames, and depending on the attack may stem from a random transmitter’s MAC address), therefore they would go unnoticed by the typical intrusion detection system (IDS), which typically focuses on management rather on control frames.
- (Due to their DoS nature) they do not directly threaten end-user’s privacy. Nevertheless, indirectly, they can be used as a stepping-stone for devising and performing more perilous attacks, including evil twin, and subsequently phishing. On top of that, both attacks require a low level of expertise and can be done with low-cost, off-the-shelf equipment and open-source software tools.
- Are feasible due to implementation defects in the driver of the affected devices and the fact that the BAR and BA frames afford no protection. The standard does address the latter shortcoming in terms of the protected block ack agreement, but as explained earlier, some unclear points still exist. Moreover, currently, the support of the PBAC flag even in the newest Wi-Fi devices is practically non-existent; this means that virtually all the existing and at least near-future devices cannot leverage protected block ack. The vendors’ implementation defects on the other hand are assumed to exist because of common business logic flaws and misinterpretations of the standard. In any case, however, this situation signifies that the implementation of the latest versions of the protocol by vendors is not yet mature enough.

5 Conclusions

The work at hand introduces two new DoS attacks against modern, popular Wi-Fi implementations. The attacks can be easily mounted even by a script kiddie, simply using inexpensive equipment and software available for free. Contrary to the previous literature, the presented attacks exploit specific control frames of the 802.11 standard, precisely, those used to provide block acknowledgment. We detail the way each attack can be exercised and, with reference to the current 802.11 standard, explain the reasons why. It is also highlighted that even though the attacks do not directly threaten users’ privacy, they can straightforwardly serve as a springboard for potentially orchestrating more harmful assaults against the end-user. This conclusion is strengthened by the positive feedback received so far from some of the affected vendors. We consider extending this work to other types of control frames, both through fuzz testing [9] and code review.


```

21     while QoS_found:
22         pass
23     if pkt.haslayer(Dot11):
24         if pkt.type == 2 and pkt.subtype == 8:
25             print("\n" + pkt.addr1 + " received QoS data from " + pkt.addr2 + "\n")
26             SN = pkt.SC.to_bytes(2, 'little')
27             QoS_found = True
28
29     def run(self):
30         sniff(iface='wlan0', stop_filter=self.packet_handler, filter="ether dst " + self.
               targeted_STA)
31
32     sniffer = Sniffer(targeted_STA)
33     sniffer.start()
34     sleep(10)
35     while not QoS_found:
36         pass
37
38     frame1 = MAC_header / BAR_control / SN
39
40     print('\n- - - - -')
41     print('Testing the exploit')
42     print('- - - - -')
43
44     while True:
45         sendp(frame1, count=128, iface='wlan0', verbose=0)

```







References

1. Schepers, D., Ranganathan, A., Vanhoef, M.: On the robustness of wi-fi deauthentication countermeasures. In: Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 245–256 (2022)
2. Chatzoglou, E., Kambourakis, G., Koliass, C.: How is your wi-fi connection today? dos attacks on wpa3-sae. *J. Inf. Secur. Appl.* **64**, 103058 (2022)
3. Chatzoglou, E., Kambourakis, G., Koliass, C.: Empirical evaluation of attacks against ieee 802.11 enterprise networks: the awid3 dataset. *IEEE Access* **9**, 34188–34205 (2021). <https://doi.org/10.1109/ACCESS.2021.3061609>
4. Marvell, C.: A simplified solution for critical a-mpdu dos issues. <https://mentor.ieee.org/802.11/dcn/08/11-08-1021-02-000n-a-simplified-solution-for-critical-a-mpdu-dos-issues.ppt>. Accessed 22 Aug 2022
5. Cisco: Issues and solutions to ieee 802.11n a-mpdu denial of service attacks. <https://mentor.ieee.org/802.11/file/08/11-08-0703-00-000n-11n-a-mpdu-dos-issues-and-solutions.ppt>. Accessed 22 Aug 2022
6. Cisco: Review of 802.11n a-mpdu dos issues – progress and status. <https://mentor.ieee.org/802.11/file/08/11-08-0755-01-000n-review-of-a-mpdu-dos-issues.ppt>. Accessed 22 Aug 2022
7. IEEE standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2020* (Revision of IEEE Std 802.11-2016), pp. 1–4379 (2021). <https://doi.org/10.1109/IEEESTD.2021.9363693>
8. Bl0ck attack exploits. <https://github.com/efchatz/Bl0ck>. Accessed 09 Feb 2023

9. Kampurakis, V., Chatzoglou, E., Kambourakis, G., Dolmes, A., Zaroliagis, C.: Wpaxfuzz: sniffing out vulnerabilities in wi-fi implementations. *Cryptography* **6**(4), 53 (2022)
10. Chatzoglou, E., Kambourakis, G., Koliass, C.: Wif0: all your passphrase are belong to us. *Computer* **54**(7), 82–88 (2021). <https://doi.org/10.1109/MC.2021.3074262>



Enhancing the ACME Protocol to Automate the Management of All X.509 Web Certificates

David A. Cordova Morales¹ , Ahmad Samer Wazan² ,
David W. Chadwick³ , Romain Laborde⁴ , April Rains Reyes Maramara² ,
and Kalil Cabral⁵ 

¹ LIP6 - Sorbonne Université, Paris, France
david.cordova@lip6.fr

² Zayed University, Dubai, UAE
{ahmad.wazan,M80008159}@zu.ac.ae

³ TrueTrust Ltd., Huddersfield, UK
d.w.chadwick@truetrust.co.uk

⁴ IRIT - Université Toulouse III Paul Sabatier, Toulouse, France
romain.laborde@irit.fr

⁵ Cesar School, Recife, Brazil
kbcv@cesar.school

Abstract. X.509 Public Key Infrastructures (PKIs) are widely used for managing X.509 Public Key Certificates (PKCs) to allow for secure communications and authentication on the Internet. PKCs are issued by a trusted third-party Certification Authority (CA), which is responsible for verifying the certificate requester's information. Recent developments in web PKI show a high proliferation of Domain Validated (DV) certificates but a decline in Extended Validated (EV) certificates, indicating poor authentication of the entities behind web services. The ACME protocol facilitates the deployment of Web Certificates by automating their management. However, it is only limited to DV certificates. This paper proposes an enhancement to the ACME protocol for automating all types of Web X.509 PKCs by using W3C Verifiable Credentials (VCs) to assert a requester's claims. We argue that any CA's requirements for issuing a PKC can be expressed as a set of VCs, returned in a Verifiable Presentation (VP). We propose a generic communication workflow to request and present VPs, and provide proof-of-concept of the viability of our approach.

Keywords: Public Key Certificate · Verifiable Credentials · ACME

1 Introduction

A lot of our real-world actions are now taking place on the Web. To effectively secure these online communications, we use the HTTPS protocol, which primarily aims to provide site authentication, integrity, and privacy of in-transit data

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 265–278, 2024.

https://doi.org/10.1007/978-3-031-56326-3_19

through encryption. However, the adoption of HTTPS has not always been an easy task. In its earlier days, it was an expensive, complicated, and error-prone process, which resulted in limited adoption [9, 11, 12]. More recently, the Web industry has made proactive efforts to promote the use of HTTPS, most notably through the “HTTPS Everywhere” campaign by the Electronic Frontier Foundation (EFF) [12], which announced in late 2022 that at least 82% of the Web is now using HTTPS [9]. Nevertheless, it is currently being used to provide encryption but fails to authenticate, in a trustworthy manner, the entities behind the web services. This is mainly because of various technological and administrative shortcomings in the issuance and validation processes of the applicants for X.509 Public Key Certificates (PKCs) [20, 21].

The authentication of web servers using HTTPS is based on the X.509 PKC data model, which is a data structure that associates the public key of the subject with its identity information. In the case of web servers, this might be information such as its domain name or IP address, as well as information about the owner of the server, which could be a person or a company. Certificates are issued by a Certification Authority (CA) after performing varying verification procedures. There are three standardised types of digital certificates: Domain Validated (DV), Organization Validated (OV), and Extended Validation (EV). The main difference between them is the level of verification procedures performed by the CA on the applicant (i.e., the PKC subject). These procedures are based upon a series of guidelines provided by the CA/Browsers (CA/B) forum¹. Such verification is reflected in the varying granularity of the content of their Subject Identity Information (SII). Although EV certificates provide the most granular SII, it is now the least adopted among all certificate types [9]. The main reason behind this decline is the poor cost-to-benefit ratio of this certificate. The high cost is due to the long and manual human work needed for the verification process, whilst the low benefit is due to the poor visual indicators in browsers, which makes it very difficult for users to differentiate between all types of certificate [15].

Verifiable Credentials (VCs) are a W3C standard that allows for the representation of verifiable claims issued by trusted issuers. VCs enable the expression of physical credentials (e.g., passports, company registrations, bank account details, etc.) in a machine-verifiable manner [13]. Currently, many certificate applicants and CAs manage certificate requests manually due to the heterogeneous nature of the requirements. In this regard, VCs might be used by CAs to verify a domain owner’s identity or any other information that requires a verification process with little or no-human intervention in an automated manner.

The Automated Certificate Management Environment (ACME) protocol [1] currently automates the management of DV certificates only. The main objective of this paper is to propose an enhancement to ACME that will allow the automation of the management of all types of certificate. Our work uses W3C Verifiable Credentials to automate the issuance and verification process required for EV and other X.509 certificate types. We argue that the automation of this process

¹ <https://cabforum.org/>.

can make EV certificates much more affordable and easier to deploy since there will be no need for a manual and time-consuming verification process, thereby improving the cost/benefit ratio.

The remainder of this paper is organized as follows: In Sect. 2 we discuss related work, the Verifiable Credentials data model, and ACME protocol [1]. Section 3 presents our solution, the ACME HIGH protocol. Section 4 describes our implementation. Section 5 discusses security issues and other concerns of our protocol, whilst Sect. 6 presents our conclusions and future work.

2 Related Work

2.1 Web X.509 Certificates

The certificate issuance process is regulated by the CA/B Forum in its Baseline Requirements [4], wherein minimum verification requirements are specified for each certificate type. For EV certificates, the CA/B Forum has issued a dedicated set of guidelines detailing its more extensive verification requirements [3].

DV Certificates require the least verification, wherein the CA only needs to verify the subject's control over a domain name. To do so, an applicant performs one of the recommended Domain Control Validation (DCV) methods [4]. Consequently, an RP (e.g., users, systems, web browsers) can only have reasonable assurance that the web server controls the domain name(s) but has no real-world identity information about the subject behind the domain name. This might not be a problem in the case of well-known domains such as www.microsoft.com, which is widely recognized. However, this is not the case for lesser-known domains. For example, who knows that the domain **.noam.ccctp.com* is owned by Microsoft Corporation? This is especially important for sensitive transactions, wherein we need more information about the entity to which we are handing our money to or to whom we are sharing our detailed personal information. This issue is partially solved by the higher level of assurance provided by OV certificates.

OV certificates are issued after a basic level of organization validation. This typically involves verifying that the organization requesting the certificate is a legally registered entity and that the applicant has control over the domain name. More specifically, a CA must verify the organization's operational address via a government agency, a third-party database, a site visit, or an attestation letter (as per the Baseline Requirements Version 2.0.0). Unlike a DV PKC, the SII in an OV PKC must include information, such as organization name, locality name, state or province name, and country name.

EV certificates verify the applicant's physical and operation addresses, and other information such as registration jurisdiction information (e.g., jurisdiction locality name, jurisdiction state or province name) and jurisdiction country name, in addition to the legal identity of the subject and its domain name. These are all reflected in its SII. Aside from the lengthier and more stringent requirements imposed on CAs regarding EV issuance, the CA is required to disclose all verification sources.

2.2 Verifiable Credentials

Verifiable credentials (VCs) are digital representations of information about an entity (individual or organization), that can be securely shared and easily verified. They are typically used to verify the identity, qualifications, or other attributes of the entity that have been assigned by trusted third parties, known as issuers, e.g., governments, universities, and other well-known organizations. VCs use cryptographic primitives to ensure that the information contained in the credential has not been tampered with and that it came from a trustworthy issuer. In this regard, an RP (called the verifier in the W3C VC data model) can easily verify the contents of a VC without the need for a central authority by checking that the digital signature on the VC corresponds to that expected from the trusted issuer. A Verifiable Presentation (VP) is a signed statement by the holder about a set of Verifiable Credentials passed to the verifier (that may have been selectively disclosed). The holder must possess a digital wallet for storing and managing its VCs.

In a typical VC scenario, the VC issuer issues VCs to the holder, who stores them in its wallet. When the holder wants to prove its identity to a verifier, it presents a VP. The verifier then verifies the information contained in the VP and makes a decision based on the verified information.

2.3 The ACME Protocol

DV PKCs are relatively easy to obtain because of their limited verification requirements. This has allowed them to be issued automatically for little to no cost [1]. Most DV certificates in the market are issued by *Let's Encrypt*² through the ACME protocol, which is a challenge-response protocol that simplifies certificate management by automating certificate requests, domain verification, and server installation. The ACME protocol was first created by *Let's Encrypt* and then was standardised by the IETF ACME working group and is defined in RFC 8555 [2].

The ACME protocol follows a client-server approach where the client, running on a server that requires an X.509 certificate, requests a certificate from the ACME server run by the CA. The protocol uses a set of JavaScript Object Notation (JSON) messages carried over HTTPS to facilitate the interaction. The first step is for the client to request an account with the ACME server, during which it, must generate an asymmetric key pair and optionally provide contact information such as the applicant's email address. Subsequently, all messages generated by the client must be signed with the corresponding private key using JSON Web Signatures (JWS) [8]. Once the account is created, there are 4 main tasks that the ACME client needs to undertake to obtain a DV PKC, as follows:

1. *Submit an order for a certificate*: The applicant submits a description of the requested certificate. The ACME protocol specifies a JSON data struc-

² <https://letsencrypt.org/>.

ture containing the subject's claimed *identifiers*³ (the domain names) and other information that cannot be sent with the Certificate Signing Request (CSR). Every request is tracked with an order number used to reference all the required authorizations.

2. *Prove control*: The server provides the client with a list of *authorization objects* (or requests). These represent the server's authorization requests for the applicant's account to prove control of the claimed *identifiers*. They usually contain a set of *challenge objects*, which represent a server's offer to validate a client's possession of a claimed *identifier* in a specific way. A challenge is validated by employing a *Validation Method*, which defines the conditions and protocols used to validate the claim. The ACME protocol defines three challenge types for which the applicant has to provide authorizations to the CA: (1) an HTTP challenge, where the applicant creates an object containing a random token at a specific HTTP URL of the requested domain, (2) a DNS challenge, where the applicant creates a DNS record that has a specific format and contains a random token provided by the CA, and (3) TLS-ALPN-01 that uses the TLS protocol and is only suitable for reverse proxies. Our ACME enhancements seek to automate this step for all types of certificate. Thus, we propose a new *challenge type* where the applicant must provide a Verifiable Presentation as proof of possession over the claimed *identifiers*.
3. *Submit a PKCS#10 CSR*: Once all the authorizations are validated and the CA is satisfied with the applicant's claims, the applicant generates a PKCS#10 CSR [14] and sends it to the ACME server.
4. *Download certificate*: The client only needs to wait for the availability of the certificate that will be issued by the CA based on the received CSR request.

ACME uses an HTTP-based protocol to provide the client with the resources needed for the certificate management functions. At the begging of the communications, the CA server provides a Directory Object to the client to help it configure itself with the right URL(s) for the ACME resource. Additionally, the Directory Object includes a "meta" field, which is a JSON object containing metadata relating to the service provided by the ACME server. In our enhancement, we propose a modification to this "meta" field to announce the new types of certificate supported by the CA.

The ACME protocol is also capable of dealing with certificate renewals and revocations through automation. This can be achieved by anyone in control of either the ACME account or the private key of the issued PKC. Figure 1 summarizes the different ACME client-server messages of the ACME protocol.

³ Identifiers for EV and OV certificates include also organization name, address, business category, etc.

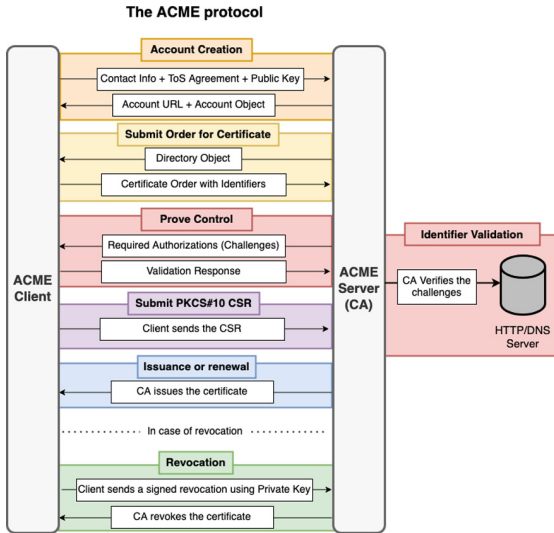


Fig. 1. The ACME Protocol

3 Proposed Solution

3.1 The ACME HIGH (ACMEH) Protocol

The ACME HIGH protocol aims to automate the issuance of EVs and other types of X.509 certificates that currently require manual human verifications. Such human verification of the Subject’s claimed Identifiers is an error-prone, time-consuming, and expensive process since it requires the CA to go through a series of administrative procedures with trusted entities capable of corroborating the claims made by the applicant. Our solution foresees that Subject Identifiers will eventually be represented as Verifiable Credentials and thus will be suitable for automatic verification. For the ACME protocol to support this model, we propose two main modifications to it: (i) Protocol-related modifications: the addition of new fields to the existing JSON data structures to support the issuance and processing of new types of PKC and (ii) Validation Method considerations: a set of considerations for implementing the new Validation Methods for requesting and presenting a VP, based on a generic communication workflow between the ACMEH client and the applicant’s VC wallet. Figure 2 illustrates the general ACMEH architecture.

3.2 Protocol-Related Modifications

In our ACMEH protocol, the CA server uses the existing “meta” object within the Directory Object to announce the new supported types of certificate in a new field called “CertTypes” (whose value is an array of strings). This informs the client of the available certificate types supported by the ACMEH protocol

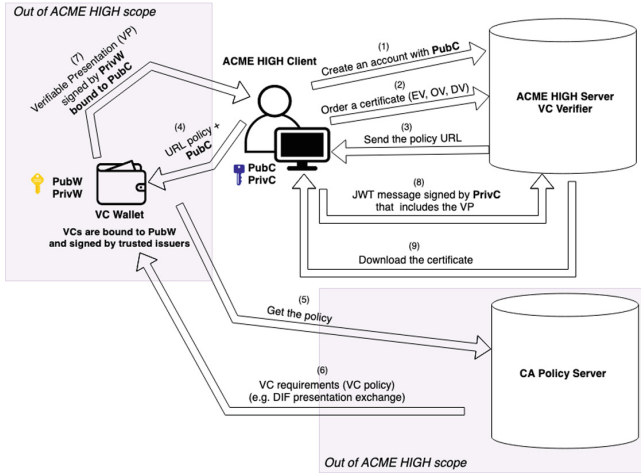


Fig. 2. The ACME HIGH Architecture. Note. Only steps 1, 2, 3, 8 and 9 comprise ACMEH. We do not propose to mandate the other protocols (4, 5, 6, 7) as these are currently being standardised by other bodies.

and the CA. In contrast, the ACME protocol assumes the client requests just one type of certificate: the DV PKC. Figure 3 illustrates the Directory Object in ACMEH and highlights the new field.

The account creation process in the ACMEH server remains unchanged from the ACME protocol. However, to submit an order for a certificate, the client begins the certificate issuance process by sending an HTTP POST request to the server’s *newOrder* resource. The POST request is composed of a set of JSON objects representing a client’s request for a certificate. This includes the *identifiers*, which is an array of *identifier objects* representing the applicant’s claims that wish to be included in the certificate, the validity time frame of the requested certificate, and a new field called *CertType*, which specifies the desired type of the requested certificate (e.g., EV, OV, QWAC, etc.) as shown in Fig. 4a.

Regarding the identifiers, the ACME protocol only defines one type of “ACME Identifier Type” to validate a DV PKC (i.e., the “dns” Identifier Type). This makes sense for the issuance of a DV PKC since the CA only needs the domain name for the requested certificate to propose a set of challenges to be completed by the client. For EV and OV PKC certificates, there is a more complex set of information requirements about the identity of the PKC applicant that needs to be verified. Therefore, we propose a new ACME HIGH Identifier Type that we call *dn* (short for Distinguished Name), which refers to a set of relative distinguished name (rdn) values that needs to be verified (e.g., organization name, organization address, legal representative name, etc.). Each of these should eventually be returned to the ACMEH server as VCs encapsulated in a VP for their verification. Figure 4a illustrates an example using our new *dn* Identifier Type for requesting a new EV certType order.

```

< HTTP/1.1 200 OK
< Content-type: application/json
{
  "newNonce": "https://example.com/acmeh/new-nonce",
  "newAccount": "https://example.com/acmeh/new-account",
  "newOrder": "https://example.com/acmeh/new-order",
  "newAuthz": "https://example.com/acmeh/new-authz",
  "revokeCert": "https://example.com/acmeh/revoke-cert",
  "keyChange": "https://example.com/acmeh/key-change",
  "meta": {
    "termsOfService": "https://example.com/acmeh/terms/2023-01-01",
    "website": "https://www.example.com/",
    "caaIdentities": ["example.com"],
    "externalAccountRequired": false,
    "certTypes": ["DV", "OV", "EV", "QWAC"]
  }
}

```

Fig. 3. The ACMEH Directory Object

Once the ACMEH server receives the client's *NewOrder* Object and the server is willing to issue the requested certificate type, it responds with a 201 (Created) response. The body of the response is the client's Order Object with any *authorizations* that the client needs to complete before the CA can issue the certificate. In the case of our *dn identifier*, the corresponding authorization object contains, within the challenge object, a new Challenge Type that we call *vp-01*, and a new *policy* field. The new Challenge Type tells the client that the server expects to receive a W3C VP as proof of the claims made by the PKC applicant, whereas the new policy field contains a reference to the CA's policy server where the applicant will find the CA's requirements for the VP that must be returned. Figure 4b illustrates an example of an authorization object for our new *dn identifier*.

Although we can specify what kind of outcome the new challenge should produce (i.e., a W3C VP), we cannot mandate a particular Validation Method for the requested VP since today different specifications are being standardised to represent VCs and VPs (e.g., LD-Proofs [19], JWT [10], VC-JWT [18]). There are also different proposed standard mechanisms for wallets to obtain and present them (e.g., OpenID4VP [5], DIDcomm [6], Verifiable Credential Handler API [16]). These variations are all handled by the CA's policy so that our ACMEH protocol is independent of them and does not need to change as the different working groups develop their VC standards.

Regarding the VP/VCs validation process by the CA, this can either be implemented directly by the CA server or delegated to a trusted service. The validation process includes checking that the VP was signed by the applicant, that the returned VP is not a replay, the embedded VCs have been signed by trusted issuers, and verifying that the claims made by the applicant are in the order payload. Thus, we consider the specifications related to VC collection from VC wallets, the VC policy query language, and the validation process of VCs as out of the scope of the ACMEH protocol (although they are equally important to standardise in due course).


```

> HTTP/1.1 200 OK
> Content-Type: application/json
> Link: <https://example.com/acmeh/directory>;rel="index"

{
  "status": "pending",
  "expires": "2023-01-01T14:09:01.13Z",
  "identifier": {
    "type": "dn",
    "value": "companyNumber = 08235199,
jurisdictionC = GB,
businessCategory = Private Organization,
streetAddress = Leamington Road,
postalCode = CV8 3EN,
C = GB,
ST = Warwickshire,
L = Kenilworth,
O = College of Policing Limited,
CN = www.police.uk"
  },
  "challenges": [
    {
      "type": "vp-01",
      "url": "https://example.com/acme/chall/prV_B7yEyA4",
      "status": "pending",
      "policy": "https://example.com/acme/policies/4NFDGDRFA"
    }
  ]
}

```

```

> POST /acmeh/new-order HTTP/1.1
> Host: example.com
> Content-Type: application/jose+json
{
  "protected": {
    "alg": "ES256",
    "kid": "https://example.com/acmeh/acct/evOfKhNU60wg",
    "nonce": "5XJlL3lEKMG7tR6pA0GcIA",
    "url": "https://example.com/acmeh/new-order"
  },
  "payload": {
    "identifiers": [
      {
        "type": "dn", "value": "serialNumber = 08235199,
jurisdictionC = GB,
businessCategory = Private Organization,
streetAddress = Leamington Road,
postalCode = CV8 3EN,
C = GB,
ST = Warwickshire,
L = Kenilworth,
O = College of Policing Limited,
CN = www.police.uk"
      },
      {
        "type": "dns", "value": "www.example.org"
      },
      {
        "type": "dns", "value": "example.org"
      }
    ],
    "certType": "EV",
    "notBefore": "2023-01-01T00:04:00+04:00",
    "notAfter": "2023-01-08T00:04:00+04:00"
  },
  "signature": "H62XtGjT2ytnPeKn....wEA4tKlBdh3e454g"
}

```

(a) Order Object

(b) Authorization Object

Fig. 4. The ACME HIGH

3.3 Validation Method Considerations

A Validation Method defines the specifications of how a client validates a challenge proposed by the ACMEH server. In the case of our *vp-01* Challenge Type, there is currently no official international agreed standard that can be used to request and present VPs. Thus, we present a generic communication workflow that ACMEH clients can integrate along with a set of considerations.

When the ACMEH client receives the new *vp-01* challenge from the server, it should request a VP from the applicant's wallet that contains verification of all the claims made by the applicant and required by the ACMEH server. The VP will act as undeniable proof of the applicant's control over the *identifiers*. It is important to notice that the ACMEH client will act as a Man in the Middle (MITM) entity between the user's VC wallet and the VC verifier (the CA). Thus, the ACMEH client acts as a wallet to the CA verifier and as a verifier to the VC wallet. To prove that this is an authorized MITM, the ACMEH client provides its public key to the VC wallet, and the VC wallet includes this in its signed VP response. The ACMEH client can now create a JSON Web Token (JWT) with the VP as its payload and sign the JWT with its corresponding private key (message 8). The ACMEH server can verify that the JWT response is signed by the ACMEH client that received the VP from the applicant's wallet.

Figure 5 illustrates the communication workflow and the entities involved in the message exchange from the ACMEH client's request for a VP (message 4) to its obtaining the VP in response (message 7). Message 4 should include the CA's policy URL and the client's public key. Once the VC wallet receives the message, it uses the VC policy URL to request the CA's policy from its policy server. The policy will state which VCs are needed, from which trusted issuers, and using which proof format. However, until there is an agreed international standard for

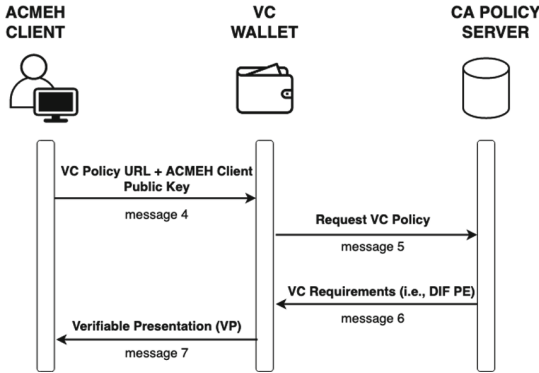


Fig. 5. Generic Communication Workflow

writing VC policies, the policy server may contain the same semantic policy in different policy languages. For example, the Decentralized Identity Foundation (DIF) has specified Presentation Exchange v1 and v2 [7], and is considering v3 as policy language, whilst the W3C Credentials Community Group has specified the Verifiable Presentation Request policy language [17]. Message 5 could include a list of policy languages supported by the VC wallet, while message 6 should return the VC policy expressed in one of the wallet-supported languages.

Once the VC wallet has the CA’s VC policy, it selects its VCs that match the policy rules imposed by the CA. It then generates a VP, embedding the matched VCs and the ACMEH client’s public key, signs the VP with its private key, and sends this to the ACMEH client (message 7). Once the ACMEH client has received the VP, the normal communication workflow between the ACMEH client and server is resumed. In this regard, the client indicates to the server that it is ready for the challenge validation by replying to the corresponding challenge URL and including the VP in a JWT format within the challenge’s payload.

The remainder of the certificate issuance process (issuance, renewal, and revocation) remains unchanged from the ACME protocol. Our considerations for implementing a new Validation Method for the *vp-01* challenge can be summarized as follows:

1. Policies should be expressed in a machine-readable manner and should be understood by the VC wallet and the ACMEH server. Various policy languages are currently being standardised.
2. VCs should be issued by authorities trusted by the CA.
3. Each VC should include the public key of the VC wallet (to prove ownership).
4. The VP should be signed with the private key of the VC wallet corresponding to the public key inside the VCs.
5. The ACMEH client’s public key should be included in the “audience” field within the VP.
6. The VP should be expressed and returned to the ACMEH server in the form of a JWT and must be signed with the private key of the ACMEH client.

4 Implementation

To demonstrate the feasibility of our proposal, we implemented a user-friendly proof-of-concept using the Identiproof⁴ VC platform, which is an easy-to-implement VC middleware facilitating the deployment of VC entities. In this regard, we have deployed a VC verifier and a policy server installed on a desktop computer using Docker containers and an Identiproof wallet installed on a mobile device. The ACMEH server is implemented using Java's Spring boot from an OpenAPI definition, and the ACMEH client uses Go and JavaScript in a web-based application. Both entities are running on the same computer in separate containers. We also assumed that the needed VCs were already in the possession of the VC wallet.

In the beginning, the user accesses our web-based ACMEH client and provides the URL of the CA's ACMEH server from which an X.509 PKC will be requested. The ACMEH client then interacts with the ACMEH server to create an account and sends the client order. To create the order, the client is required to specify the type of certificate (e.g., EV, OV, etc.). When the client receives the policy URL from the ACMEH server in response to its order, it presents this as a QR Code to the VC wallet. We expect the contents of this QR code will eventually be standardised, as is being done by the OID4VP protocol [5] from the OpenID Foundation. The QR code tells the VC wallet four things: how it can obtain the CA's policy, what nonce to use to stop replay attacks, the id of the requestor, and to which endpoint it should send the VP in response to this policy. To generate the VP, the user must scan the QR Code with its VC wallet, which dereferences the policy URL and fetches the policy from the CA's policy server. This allows the VC wallet to search its VC database for VCs that match the policy. The VC wallet then asks the user for consent to send the selected VCs to the verifier (i.e., the ACMEH client), generates a VP from the available VCs, nonce, and client ID, and submits it to the ACMEH client's stated endpoint. The ACMEH client must generate a CSR containing all the needed information for the PKC, such as the domain name and the company details. In our implementation, the user can just input the CSR in a special field of the ACMEH client interface and request the EV certificate.

The ACMEH server must perform the necessary verification of the EV certificate request. This includes verifying that the JWT is signed by the client's private key, that the VP has not been replayed and was sent to the ACMEH client, and that the VC attributes match the provided CSR. To aid all CAs in this task, we utilize the Identiproof verifier service. Given a VP, the verifier service verifies that the VP was signed by the private key of the VC wallet, whose public key is identified as the issuer of the VP. The verifier then verifies that all the embedded VCs are signed by trusted issuers and that they were issued to the public key of the VC wallet, thereby proving the rightful possession of the VCs. The verifier then returns to the CA the verified list of VC attributes and

⁴ <https://www.crosswordcybersecurity.com/identiproof>.

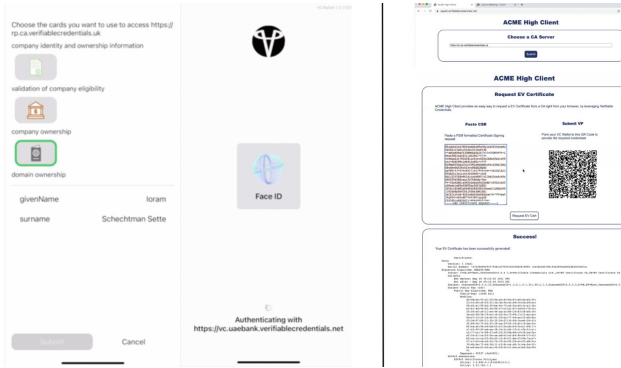


Fig. 6. Implementation. The left side of the figure shows the VC wallet, while the right side of the figure shows three different stages of the ACMEH client during the EV issuance process.

the VP’s properties. As a result, the ACMEH server will either return an error message to the ACMEH client if any verification processes fail or the expected EV certificate if all verification steps are passed (Fig. 6).

5 Security Considerations

In ACME HIGH as in the ACME protocol, all requests and responses are sent via HTTPS by all entities in the ACMEH architecture and use JWS to sign and verify the data within the JSON structure. This provides a complementary layer of security, as even if an attacker intercepts the HTTPS request, they will not be able to modify the data without the signature being invalidated. In this regard, a “nonce” field is included in message 3, which must be returned in the VP as a custom property to protect the ACMEH resources from replay attacks. This nonce is being standardised in the OID4VP protocol. By including the nonce, the recipient of the VP (i.e., the ACMEH server) can verify that the VP was generated specifically for this request and not retransmitted from a previous request. However, the VP is not signed by the ACMEH client that the ACMEH server is talking to but by the VC wallet holding the VCs. This is because the ACMEH client is acting as a genuine MITM. Thus, the request from the ACMEH client to the VC wallet contains, the public key of the ACMEH client as the ID of the verifier, which the VC wallet places in the audience field of the returned VP.

Concerning the security considerations regarding the authenticity of the applicant’s VCs to the ACMEH server, regardless of the communication protocol used for this exchange, the CA can easily authenticate the applicant’s VCs by checking the signature of the credentials corresponds to those of the expected trusted issuers. This process is also efficient against network-layer attacks such as BGP route hijack (as opposed to the ACME protocol that does not protect

against this) since even if an attacker manages to redirect the ACMEH challenges to another server, the attacker would never be able to provide authentic proof of control over the identifiers because the ACMEH servers maintain a list of trusted issuers with their respective public keys.

Another risk mentioned in ACME's RFC [2] is the misconfiguration on a web server that would allow non-administrative users to write to *.well-known* folder that should be used to store the random token. In this case, any non-administrative user can obtain a certificate for the web server. This threat does not exist for ACME HIGH because only the signed VCs are considered to validate the request of an applicant. In addition, the ACME protocol is very vulnerable to the DNS since any incorrect DNS answer can allow an attacker to validate his requests. In our case, the DNS does not constitute a risk because the CA has a list of valid public keys of VC issuers. Finally, it is important to notice that the ACME HIGH protocol is still vulnerable to DoS threats and well-known TLS vulnerabilities such as encryption algorithms downgrade.

6 Conclusions and Future Work

In this paper, we have proposed ACME HIGH, an enhanced certificate automation protocol that uses W3C Verifiable Credentials to cryptographically prove possession of Subject Identity Information (SII) dictated by the CA/B Forum. We argue that the current issuance process of certificates that allows a high level of authentication is slow, tedious, and expensive. Our protocol specifies the structural changes needed in the ACME protocol to support the issuance of all types of X.509 certificate and a generic communication workflow to request and present Verifiable Presentations that satisfy the CA's policy requirements. In this regard, we have also implemented a proof-of-concept that demonstrates the feasibility of our solution and proves that it can easily be adapted to interact with different communication protocols for the exchange and definition of VCs. Once these protocols are standardised, they can be used without necessitating any changes to our ACMEH proposal.

As a perspective of our future activities, we would like to propose a new RFC draft containing all the details of our protocol. Moreover, we are interested in contributing to the definition of policy requirements using the Decentralized Identity Foundation Presentation Definition JSON language to facilitate the request and obtention of such policies between a VC wallet and a policy server. Finally, we believe that facilitating the automation of X.509 certificate issuance can allow for the resurgence of EV certificates, which have been greatly abandoned, and reduce their cost by significantly reducing the manual process that it currently entails. This will encourage wider adoption of high-security certificates, thus contributing to a safer and more trustworthy Internet.

References

1. Aas, J., et al.: Let's encrypt: an automated certificate authority to encrypt the entire web. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 2473–2487 (2019)
2. Barnes, R., Hoffman-Andrews, J., McCarney, D., Kasten, J.: Automatic certificate management environment (ACME). Technical report (2019)
3. CA/Browser Forum: Guidelines for the issuance and management of extended validation certificates (2022). <https://cabforum.org/wp-content/uploads/CA-Browser-Forum-EV-Guidelines-1.8.0.pdf>
4. CA/Browser Forum: Baseline requirements for the issuance and management of publicly-trusted certificates (2023). <https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-v2.0.0.pdf>
5. Chadwick, K.N., Vercammen, J.: OpenID for verifiable credentials (2022)
6. Curren, S., Looker, T., Terbu, O.: DIDComm messaging v2.1 editor's draft (2022). <https://identity.foundation/didcomm-messaging/spec/v2.1>
7. Daniel, B., Brent, Z., Martin, R., Kim, H.D.: Presentation exchange (2022). <https://identity.foundation/presentation-exchange/>
8. Internet Engineering Task Force (IETF): Json web signature (JWS) (2015). <https://www.rfc-editor.org/rfc/rfc7515>
9. Internet Security Research Group (ISRG): 2022 ISRG annual report (2023). <https://www.abetterinternet.org/documents/2022-ISRG-Annual-Report.pdf>
10. Jones, M., Bradley, J., Sakimura, N.: Json web token (JWT). Technical report (2015)
11. Krombholz, K., Mayer, W., Schmiedecker, M., Weippl, E.: “i have no idea what i'm doing”-on the usability of deploying HTTPS (2017)
12. Matthew, B., Jonathan, S., Ziegler, A.C., Philip, K., Wallach, D.S., Alex, H.J.: On the usability of https deployment. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1–10 (2019)
13. Sedlmeir, J., Smethurst, R., Rieger, A., Fridgen, G.: Digital identities and verifiable credentials. *Bus. Inf. Syst. Eng.* **63**(5), 603–613 (2021)
14. The Internet Society: PKCS#10: Certification request syntax specification version 1.7 (2000). <https://www.rfc-editor.org/rfc/rfc2986>
15. Thompson, C., Shelton, M., Stark, E., Walker, M., Schechter, E., Felt, A.P.: The web's identity crisis: understanding the effectiveness of website identity indicators, pp. 1715–1732 (2019)
16. W3C Community Group: Credential handler API 1.0 (2021). <https://w3c-ccg.github.io/credential-handler-api/>
17. W3C Community Group: Verifiable presentation request v0.2 (2022). <https://w3c-ccg.github.io/vp-request-spec/>
18. W3C Working Draft: Securing verifiable credentials using JSON web tokens (2023). <https://www.w3.org/TR/vc-jwt/>
19. W3C Working Draft: Verifiable credential data integrity 1.0 (2023). <https://www.w3.org/TR/vc-data-integrity/>
20. Wazan, A.S., Laborde, R., Chadwick, D.W., Barrere, F., Benzekri, A.: TLS connection validation by web browsers: why do web browsers still not agree? In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 665–674. IEEE (2017)
21. Wazan, A.S., et al.: On the validation of web X.509 certificates by TLS interception products. *IEEE Trans. Dependable Secure Comput.* **19**(1), 227–242 (2020)



MADONNA: Browser-Based Malicious Domain Detection Through Optimized Neural Network with Feature Analysis

Janaka Senanayake¹ , Sampath Rajapaksha¹ , Naoto Yanai² ,
Chika Komiya² , and Harsha Kalutarage¹ 

¹ School of Computing, Robert Gordon University, Aberdeen, UK
{j.senanayake,s.rajapaksha,h.kalutarage}@rgu.ac.uk

² Department of Information Security Engineering, Osaka University, Suita, Japan
{yanai,c-komiya}@ist.osaka-u.ac.jp

Abstract. The detection of malicious domains often relies on machine learning (ML), and proposals for browser-based detection of malicious domains with high throughput have been put forward in recent years. However, existing methods suffer from limited accuracy. In this paper, we present MADONNA, a novel browser-based detector for malicious domains that surpasses the current state-of-the-art in both accuracy and throughput. Our technical contributions include optimized feature selection through correlation analysis, and the incorporation of various model optimization techniques like pruning and quantization, to enhance MADONNA's throughput while maintaining accuracy. We conducted extensive experiments and found that our optimized architecture, the Shallow Neural Network (SNN), achieved higher accuracy than standard architectures. Furthermore, we developed and evaluated MADONNA's Google Chrome extension, which outperformed existing methods in terms of accuracy and F1-score by six points (achieving 0.94) and four points (achieving 0.92), respectively, while maintaining a higher throughput improvement of 0.87 s. Our evaluation demonstrates that MADONNA is capable of precisely detecting malicious domains, even in real-world deployments.

Keywords: malicious domain detection · machine learning · feature engineering · browser extension

1 Introduction

The incidence of cybercrime has significantly increased in recent years, with the use of rogue domains being a common tactic employed by adversaries for various purposes such as running command and control (C&C) servers or setting up phishing websites. Shockingly, the creation of about 40,000 malicious domains per day has been reported, and this leads to an average loss of \$17,700 per minute [15]. To circumvent blacklist blocking, attackers often use short-lived malicious domains generated by domain generation algorithms (DGAs).

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 279–292, 2024.

https://doi.org/10.1007/978-3-031-56326-3_20

Therefore, the use of machine learning (ML) for detecting malicious domains has received significant attention in recent years [29].

A browser is the closest interface for a user, and hence users can be alerted to ongoing malicious attempts, such as phishing, via the browser in real-time. However, the inference throughput and accuracy of the existing work need to be increased further to build an efficient malicious domain detection model. In this paper, we aim to design an artificial intelligence-based domain detection application for web browsers with higher accuracy and lower computation overhead (throughput). We note that such a design is *non-trivial*.

Indeed, the reason for the low accuracy of previous work is caused by the use of simple neural network models or traditional ML models to improve the throughput. If an enriched ML model is trivially introduced in the application instead of a simple neural network, the throughput will be downgraded drastically [9]. It might be unsuitable for a browser-based deployment due to the high throughput. Hence, we need to address a trade-off problem between the throughput and accuracy to develop a practically deployable browser-based malicious domain detection application.

Our approach to addressing the aforementioned problems involves two standpoints. Firstly, we identify the most relevant features for detecting malicious domains. In general, choosing such features is a technical matter [23], and one potential approach is to analyze feature correlations [2, 10]. Through an in-depth examination of feature correlations, we can eliminate redundant features, thereby improving throughput without compromising accuracy. Secondly, we employ model optimization techniques, such as pruning and parameter quantization, in deep learning models. By implementing these techniques and eliminating unnecessary neurons in the models, we can significantly improve throughput while maintaining accuracy. Based on the above viewpoints, we propose *MADONNA (MALicious Domain detection through Optimized Neural Network with feature Analysis)*. We demonstrate that MADONNA outperforms other state-of-the-art models in terms of both accuracy and throughput by virtue of the analysis of feature correlations and neural network-based learning techniques while achieving 94% accuracy.

To sum up, we make the following contributions:

- We present MADONNA, an open-source browser-based extension (plug-in) that runs AI in the backend to detect malicious domains in near real-time.
- We analyze feature correlations for malicious domain detection by removing highly correlated features to improve both throughput and accuracy.
- We show that parameter quantization and pruning in a deep learning model can strikingly improve throughput by keeping the same-level accuracy for malicious domain detection.
- We conduct a real-world experiment to distinguish benign and malicious domains in the real world and show that MADONNA can detect these domains precisely.
- We demonstrate that MADONNA outperforms the benchmarked models with respect to the accuracy and throughput of malicious domain detection.

The rest of the paper is organized as follows: Sect. 2 contains preliminaries. Section 3 explains the methodology, and Sect. 4 discusses the results. Finally, the conclusions and future work directions are discussed in Sect. 5.

2 Preliminaries

This section provides background knowledge on domain names and malicious domain detection using machine learning (ML) to aid in the understanding of our work.

2.1 Domain Names

Domain names are texts correlated to network hosts and are operated via the Domain Name System (DNS). Generally, domain names are hierarchically managed under namespaces called a zone, and the highest domain is called root. The most popular domains are .com, .org, .us, .uk, and .jp, and such domains are called top-level domains (TLDs). There are multiple domains under each TLD, which are managed hierarchically and distributively through their zones. In a normal URL, protocol, subdomains (optional), domain name, TLD, and subdirectories can be linked as shown in this example: <https://ifipsec2023.psnc.pl/program/>.

2.2 Malicious Domain Detection

There are two main approaches to detect malicious domains, namely knowledge-based and machine learning-based methods [29]. The former approach is based on expertise and heuristics to distinguish benign and malicious domains. However, these methods often fail to detect novel attacks, leading to zero-day exploits. In contrast, the latter approach can effectively infer unknown domains as benign or malicious [12]. Especially, supervised learning is a common setting because of the efficiency and the selection of the most relevant features from raw data [13, 14, 16, 17, 19, 25]. It can infer new domains after the training with labeled domains with high accuracy. Hence, we focus on malicious domain detection based on ML.

Malicious domain detection based on ML provides inferences to determine whether the given domains are malicious. Informally, an ML model learns features of domains and their labels that represent the domains as benign or malicious. Afterward, the model takes features of a target domain as inputs in the inference phase and then infers its label as benign or malicious. A typical approach for domain detection in recent years is based on deep neural networks.

Problem Formulation: We formalize the problem of domain detection based on ML below. Let $\mathcal{F} = \{f_1, \dots, f_l\}$ be a set of features. Each domain $d_i \in D$ has features $F_i = \{f_{i,1}, \dots, f_{i,l}\}$, where D denotes a set of domains, and $l \in \mathbf{N}$ denotes the size of F_i , i.e., the number of features of each domain. In addition, each $d_i \in D$ has a label $L_i \in \{0, 1\} \subseteq L$, where each label denotes a benign domain by 0 and a malicious domain by 1. For the size n of D , i.e., the number of

domains, $DFL = \{(d_1, F_1, L_1), \dots, (d_n, F_n, L_n)\}$ denotes the combinations with domains, features, and labels. Let $Model = M(DFL)$ denote a trained model, where M denotes a learning algorithm. If d_t is a test domain (test case) unseen by M during its training time, our goal is then to obtain an inference result, $L_t = Model(F_t)$, by extracting features $F_t = \{f_{t,1}, \dots, f_{t,l}\}$ for the unlearned domain d_t .

2.3 Related Works

We describe related works in three aspects: feature selection, feature engineering, and browser-based applications.

For the feature selection, the major way of malicious domain detection is to utilize an enriched model only with domain names [4, 27, 28]. While domain names are dealt with text data, malicious domain detection often needs more information, such as DNS information [21, 22] and web contents [1, 3]. We follow features in [9], which include domain names, DNS information, and web content.

For feature engineering, a typical way to improve accuracy is to evaluate the feature importance. To this end, features can be analyzed by principal component analysis [31] or decision trees [26]. Redundant features can also be removed by computing zero scores [18] or the equality of data points [30]. We adopt feature correlations [10] to remove redundant features for the design of MADONNA because training prediction models with too many correlated features reduces their accuracy and increases the model's computational overhead.

For browser-based applications, a web browser for detecting a phishing site [6] was developed in recent years. However, it is not a common browser such as Firefox or Google Chrome. As plug-ins for existing browsers, several works [3] have developed phishing site detection, and there are several products according to the recent survey [23]. They utilize whitelists and blacklists of domains as a part of detection. We discuss more general malicious domains, including phishing sites, only with ML as browser-based applications. The closest work to ours is MAD-MAX [9], which is a browser-based application for detecting malicious domains and includes feature selection. Another important related work in [2] provided four feature importance evaluations. Our goal is to design a high-performing browser-based extension that outperforms the benchmarked and state-of-the-art model. Therefore, we introduce MADONNA, which will be compared to MAD-MAX in the benchmarking process.

Various libraries [11, 20] exist for implementing deep learning in web browsers, which have the potential to deliver performance equivalent to JavaScript. Additionally, MADONNA functions as a distributed platform [7]. These libraries can be used to build a browser-based system for detecting malicious domains. While these libraries focus on creating generalized ML platforms, MADONNA is designed specifically for detecting malicious domains.

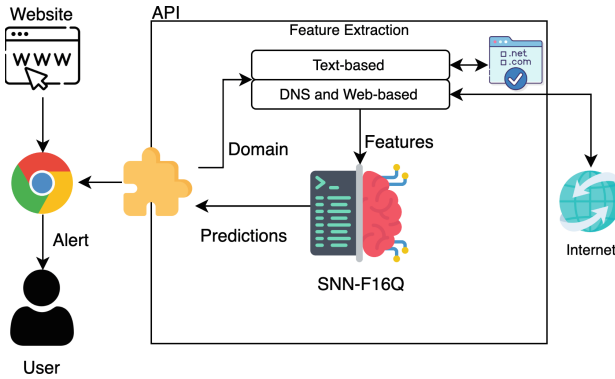


Fig. 1. The Overview of MADONNA

3 Methodology

This section outlines the specific methodology used for detecting malicious domains, which is divided into three subsections: Feature Extraction, Model Training and Optimization, and Browser-based Deployment. Figure 1 provides an overview of the MADONNA system and explains each of the steps involved. When a user clicks on the MADONNA extension to check the malicious status of a domain, the system’s Application Programming Interface (API) is called. The API extracts the required features and uses the trained SNN model to generate prediction results, which are then displayed to the user through the extension.

3.1 Feature Extraction

This work utilized the dataset introduced in [5,9]. This dataset consists of 25 features, including text-based features, DNS-based features, and web-based features. Text-based features represent information obtained from strings of domain names and discuss whether malicious domains can be detected from the domain names. DNS-based features represent information obtained from DNS records of their corresponding domains and discuss the difference of DNS records between malicious domains and benign domains. Web-based features represent information obtained from contents on domains and discuss characteristics of the contents provided by malicious domains.

Throughput is one critical criterion of the proposed model, as real-time or near-real-time malicious domain detection is highly important. Therefore, a minimum number of features should be selected whilst achieving the highest level of detection rate. Figure 2 shows the feature correlation metrics for the 25 features.

Label feature includes the ground truth 1 and 0 for malicious and benign domains, respectively. Based on this, features n_ns and ns_similarity have the highest Pearson correlations with the label, whereas mean_TTL and stdev_TTL have the lowest Pearson correlations. Some features have higher correlations with

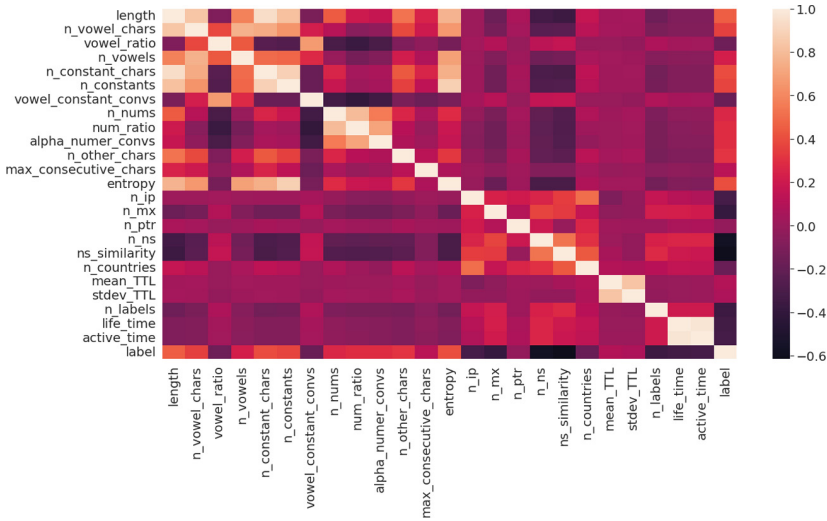


Fig. 2. Feature Correlations

some other features. For example, active_time and life_time have a correlation of 0.97. Therefore, we can remove one of these highly correlated features from the ML model.

In their study [9] the authors employed the permutation importance algorithm to choose seven features. The backward selection was utilized to take into account feature correlation and distribution. The chosen features include: length, n_ns, n_vowels, n_vowel_chars, life_time, n_constant_chars, n_nums, ns_similarity, n_other_chars, entropy, n_countries, n_mx, and n_labels. Table 1 provides a summary of the feature description and their behavior in both benign and malicious scenarios based on our analysis.

Feature distributions for a sample of features are depicted in Fig. 3. X-axis represents malicious and benign class labels, while the y-axis represents value ranges for the respective variable. None of the features create 100% class separability. For example, the highest correlated variable ns_similarity, which is shown in Fig. 3c, has shared values between 0.71 and 1.00 for both benign and malicious domains. Further, this clearly shows that outliers are available in the benign dataset. Therefore, these outliers are removed using Z-score to achieve higher generalization capability. Z-score greater than +3 or less than -3 is considered as the threshold to identify the outliers.

3.2 Model Training and Optimization

A supervised learning model was trained by utilizing the dataset proposed in [5] and selecting the optimized features only. To evaluate the performance of the models, a variety of experiments were conducted, which involved training them with different machine learning algorithms, such as Logistic Regression (LR) and

Table 1. Selected Features

Feature Name	Description
length	The length of the domain. The average length of malicious domains is about two times that of benign domains
n_ns	The number of distinct name servers. n_ns values tend to be low for malicious domains
n_vowels	The number of vowels in the domain. These values tend to be high for malicious domains
life_time	The difference of expiration date and creation date of WHOIS data, in days. Generally, life_time is low for malicious domains
n_vowel_chars	The number of vowel characters in the domain. n_vowel_chars has similar characteristics as n_vowels
n_constant_chars	The number of constant characters in the domain. Malicious domains include more constant characters
n_nums	The number of numeric characters in the domain. This is typically high in malicious domains
n_other_chars	Number of characters other than digits and alphabets in the domain. This is comparatively high in malicious domains
entropy	The entropy of the domain. High values can be observed for malicious domains
ns_similarity	The similarity between name servers. This is significantly low for malicious domains
n_countries	The number of countries obtained from GeoLite2 service queried using each of the distinct IP addresses. This tends to be greater than 1 for malicious domains
n_mx	The number of distinct mail exchange records. Low values can be observed for malicious domains
n_labels	The number of HTML elements of the content. This is significantly low in malicious domains

Random Forest (RF), boosting algorithms like Gradient Boosting (GB), eXtreme Gradient Boosting (XGB), Light Gradient Boosting (LGB), and Extreme Learning Machine (ELM). In addition, Multilayer Perceptron (MLP) and SNN were trained and assessed the performance in terms of the accuracy, F1-Score, and throughput of the models, and it was found that the SNN delivered a good performance. Therefore, the optimal artificial neural network model was selected for further experiments. A simple model architecture was selected considering the detection latency of the model. To this end, grid search was used to select the optimum hyperparameters of the SNN model.

Pruning and Quantization. The throughput of a Neural Network can be improved by eliminating the least significant weight parameters. This aims to keep the model’s accuracy while improving its efficiency. Magnitude-based pruning [24] is a simple but effective approach that eliminates the weights whilst keeping the same level of accuracy. Magnitude-based pruning gradually removes the insignificant weights by assigning value zeros during the model training process. Model accuracy depends on the level of sparsity and therefore, sparsity level should be selected carefully to achieve the same level of accuracy. The TensorFlow model optimization toolkit was used to apply the magnitude-based model

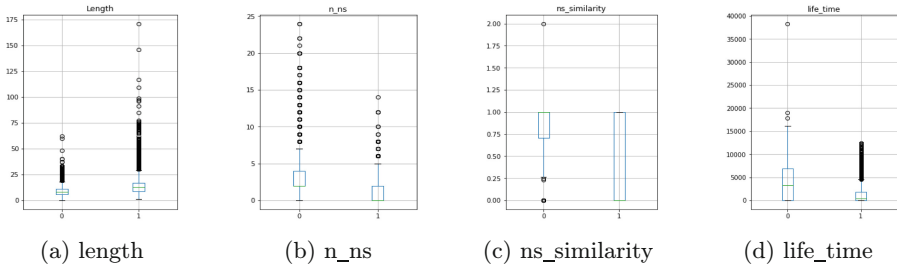


Fig. 3. Feature Selection

pruning. First, the model was trained with all parameters and then applied pruning to achieve 50% of parameter sparsity starting from 0% sparsity. The pruned model is referred to as SNN-P in this work.

Quantization [8] is another optimization technique that reduces the precision of the numbers used for model parameters. Typically, TensorFlow uses 32-bit floating point numbers. Quantization leads to achieving a better throughput by moving 32-bit numbers into 16 or 8-bit numbers. However, this might reduce the model accuracy slightly due to the loss of precision. The TensorFlow optimization toolkit provides different quantization options. Accordingly, we used non-optimized quantization (SNN-NOQ), dynamic range quantization (SNN-DRQ), float16 quantization (SNN-F16Q), and int8 quantization (SNN-I8Q). The TensorFlow quantization also converts the model into a more lightweight TFLite version. Therefore, SNN-NOQ, SNN-DRQ, SNN-F16Q, and SNN-I8Q models are TFLite versions.

3.3 Browser Deployment

The browser extension named MADONNA, which was produced in this work, can be used to detect malicious domains near real-time when visiting websites. Due to the popularity of web browsers, Google Chrome was selected as the target browser to develop the extension. The extension can be downloaded from GitHub¹ and can be easily installed in Google Chrome.

The MADONNA extension has been connected with a Python Flask web API. The API can be started by executing the `start_api.bat` file (for Windows) or `start_api.sh` file (for Linux). To execute the API, Python 3 runtime should be available. Once the API is running in the backend, when a user uses the browser extension, it sends the URL user attempts to visit, to the API, and the API extracts the required text-based, DNS-based, and web-based features of the given URL. These features are passed to the trained SNN model. Then the model predicts whether the domain is malicious or benign. Based on the results user will be prompted by the browser extension whether the URL is safe to visit.

¹ <https://github.com/softwaresec-labs/MADONNA>.

4 Results and Discussion

This section presents an evaluation of the performance of MADONNA using an existing dataset [5]. The analysis includes several key metrics, such as accuracy, model size, and throughput. Additionally, we benchmark the MADONNA extension in Google Chrome against MADMAX [9] to evaluate its performance.

4.1 Experimental Setting

The dataset introduced in [5], consisting of 48,252 domains (24,126 benign and 24,126 malicious), was utilized to train the model using the 13 identified important features. The SNN model was trained with 28 nodes in the hidden layer for 50 epochs and a batch size of 128, using the Adam optimizer with a learning rate of 0.001. To prevent overfitting, early stopping was applied. ReLU was used as the activation function for the hidden layer, while softmax was used for the classification layer. The model architecture was simple, with only 533 trainable parameters, which helped to achieve low latency. TensorFlow and Keras libraries were used to implement the SNN model, and Google Colab standard environment with 12GB of RAM was utilized for model training and inference.

4.2 Accuracy and Throughput of the Model

To evaluate the accuracy and F1-score of MADONNA's SNN model, traditional machine learning algorithms and boosting algorithms were trained using the same dataset and set of features. 5-fold cross-validation was performed for all models. The SNN model was further optimized for throughput by implementing pruning and quantization techniques. As the initialization time of the API is dependent on the model size, it is crucial to have a smaller model size to achieve faster performance. This is particularly important for detecting malicious domains since web users prefer fast browsing without additional delays. Therefore, it is necessary to ensure that domain analysis is performed within a reasonable timeframe to predict whether a domain is malicious or not. The accuracy, precision, recall and F1-Score, model size, and inference time of these models are compared in Table 2.

According to Table 2, it was identified that when applying boosting algorithms (XGB, GB, LGB), higher accuracies and F1-Scores can be obtained compared with the other ML algorithms (LR, RF, ELM, and MLP). However, the SNN model outperformed all the traditional machine learning models with 94% accuracy and 0.92 F1-Score. It was also identified that the optimized SNN variants achieve the same level of performance except for a slight performance drop in the SNN-I8Q model.

The best-performed ML-based model was XGB. It required the largest model size, whereas the ELM model showed the longest inference time. This is because, even if ELM requires a small training time, it still requires a large number of model parameters to learn the benign and malicious domain patterns. Therefore, it takes much time for the inference. On the other hand, due to the simple

Table 2. Comparison of Accuracy, Precision, Recall, F1-Score, Throughput

Model	Accuracy	Precision	Recall	F1-Score	Model size(KB)	Inference time(μ s)
LR	87%	0.87	0.86	0.87	443	151
RF	88%	0.92	0.83	0.87	480	41
GB	89%	0.91	0.89	0.89	422	112
MLP	83%	0.88	0.78	0.82	78	69
LGB	89%	0.91	0.89	0.89	482	98
XGB	90%	0.92	0.89	0.90	526	27
ELM	87%	0.88	0.86	0.87	312	198
SNN	94%	0.96	0.89	0.92	33	64
SNN-P	94%	0.96	0.90	0.92	19	36
SNN-NOQ	94%	0.96	0.90	0.92	4	19
SNN-DRQ	94%	0.96	0.90	0.92	4	16
SNN-F16Q	94%	0.96	0.90	0.92	3	10
SNN-I8Q	93%	0.95	0.89	0.91	3	12

model architecture of the SNN model, it only takes 33KB of memory and 64 μ s inference time. Pruned model (SNN-P) optimized these values due to the removal of insignificant weights. As expected, quantized models further optimized both model size and inference time. However, despite low precision, SNN-I8Q took small additional time compared to the SNN-F16Q model. This is likely because quantized int requires an arm device such as Raspberry Pi to get the optimum inference.

By considering all the evaluation matrices, including the accuracy, precision, recall, F1-Score, model size, and inference time, the SNN-F16Q model was selected to integrate with the browser extension to detect malicious domains.

4.3 Performance of Browser Extension

Users receive notifications reflecting predictions of the malicious status of a domain upon clicking on the MADONNA extension icon in Google Chrome. Examples of these notifications are shown in Fig. 4. The extension displays a notification similar to Fig. 4b if a domain is benign (e.g., <https://www.google.com>). On the other hand, if the checked URL contains a malicious domain (e.g., <https://chromnius.download/browser2/?mrddp=1&mrddz=2353135>), a notification resembling Fig. 4c appears. Figure 4a illustrates the notification displayed when checking a domain's malicious status.

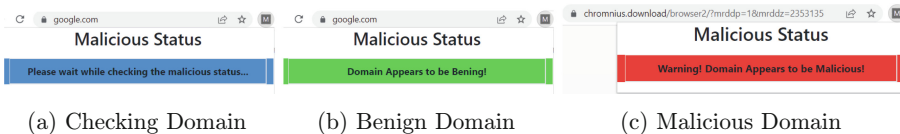
**Fig. 4.** Chrome Browser Extension Notifications

Table 3. Comparison of MADMAX and MADONNA

Aspect	MADMAX	MADONNA
Used Text-based features	length, n_constant_chars, n_vowel_chars, num_ratio	length, n_vowels, n_vowel_chars, n_constant_chars, n_nums, entropy, n_other_chars
Used DNS-based Features	n_ns	n_ns, ns_similarity, n_mx, n_countries
Used Web-based Features	life_time, n_labels	life_time, n_labels
Model Inference Time	198 μ s	10 μ s
Supported Browser	Firefox	Chrome
Avg. Prediction time in Browser	3.3 s	2.43 s
Accuracy	88%	94%
F1-Score	0.88	0.92
Precision	0.90	0.96
Recall	0.86	0.90
Connectivity	Externally-hosted Sever	Internally-hosted API

4.4 Comparison with Existing Works

A comparison between MADONNA and MADMAX was conducted as MADMAX is the closest high-accuracy AI-based malicious domain detection work. The comparison results are presented in Table 3, indicating that MADONNA includes more text-based and DNS-based features than MADMAX, which were selected based on their significance in feature analysis. By combining these features with an optimized and quantized SNN model, MADONNA achieved better accuracy, F1-Score, precision, and recall than MADMAX. Specifically, MADONNA outperformed MADMAX by 6% in accuracy and 4% in F1-Score, representing a significant improvement.

MADONNA supports the widely used Google Chrome browser and its backend model achieves significantly faster inference times (10 μ s compared to MADMAX’s 198 μ s) which should be considered as notable advantages. Additionally, the MADONNA browser extension predicts the malicious status of domains more quickly, with an average prediction time of 2.43 s compared to MADMAX’s 3.3 s. MADONNA’s extension connects to an internally hosted web API, prioritizing user privacy, while MADMAX’s extension relies on an external server for predictions. In summary, MADONNA outperforms MADMAX in all aspects.

We also conducted real-world experiments to evaluate the performance of MADONNA, validating the performance of the MADONNA Chrome extension by visiting well-known benign sites and malicious sites listed in CyberCrime, PhishTank, and Tranco websites [9]. The experimental machine used had a Core

i5 processor with 16 GB RAM and 66.6 Mbps fiber broadband internet connectivity. The MADONNA extension can predict whether the URL is malicious or benign on average in 2.43s, which is reasonable for practical use. Although the SNN model can predict the malicious status of a given feature set in just 10 μ s, the MADONNA extension takes more time to extract some of the DNS-based and web-based features, which is why it takes 2.43s on average to provide a notification. The detailed experiment results are available in MADONNA's GitHub repository², although they are not presented in the paper due to space constraints.

4.5 Limitations

The misclassification results suggest that certain malicious domains exhibit benign feature values while some benign domains exhibit malicious feature values for the selected features. This observation implies the need for more sophisticated and distinguishable web-based features to further minimize misclassifications. Features such as pop-up messages, alert boxes, a high percentage of advertisements, and site redirection are examples of malicious web-based features that can be taken into account during the feature analysis stage. However, these features must be extracted after the page has been loaded in the browser, which could potentially reduce the throughput and real-time usability of the solution.

The MADONNA model has a very fast inference time of just 10 μ s, but the prediction time through the browser extension takes on average 2.43s. This delay is mainly due to internet connectivity and cannot be easily solved with existing web-engineering techniques. The authors explored the possibility of using Pyscript³ to remove the API execution step and convert it to a fully browser-based model, but this was not feasible due to limited library support for extracting web and DNS-based features. Therefore, MADONNA still requires a connection with a hosted API on the local machine, which adds computational overhead to the overall process.

5 Conclusion and Future Work

Overall, MADONNA is a promising approach to detecting malicious domains and demonstrates the potential of leveraging machine learning and browser-based applications for malicious domain detection. The authors' contributions in optimizing feature extraction, applying ML methods and optimization techniques, and introducing the SNN architecture are significant and demonstrate the effectiveness of MADONNA compared to state-of-the-art methods. However, there are also some limitations to consider, such as the computational overhead of connecting with a hosted API on the local machine and the dependence on internet connectivity for timely predictions. Further research could explore ways

² <https://github.com/softwaresec-labs/MADONNA>.

³ <https://pyscript.net/>.

to minimize prediction time and improve accuracy while integrating web-based features to enhance MADONNA's capabilities. Nonetheless, MADONNA represents a significant step forward in the field of cybersecurity (malicious domain detection) and shows promise for future development and improvement.

References

1. Abdelnabi, S., Krombholz, K., Fritz, M.: VisualPhishNet: zero-day phishing website detection by visual similarity. In: Proceedings of CCS 2020, pp. 1681–1698. ACM (2020)
2. Alhogail, A.A., Al-Turaiki, I.: Improved detection of malicious domain names using gradient boosted machines and feature engineering. *Inf. Technol. Control* **51**(2), 313–331 (2022)
3. Ariyadasa, S., Fernando, S., Fernando, S.: Combining long-term recurrent convolutional and graph convolutional networks to detect phishing sites using URL and HTML. *IEEE Access* **10**, 82355–82375 (2022). <https://doi.org/10.1109/ACCESS.2022.3196018>
4. Berman, D.S.: DGA CapsNet: 1D application of capsule networks to DGA detection. *Information* **10**(5), 157 (2019)
5. Chien, C.J., Yanai, N., Okamura, S.: Design of malicious domain detection dataset for network security (2021). <http://www-infosec.ist.osaka-u.ac.jp/~yanai/dataset.pdf>
6. Mohith Gowda, H.R., Adithya, M.V., Gunesh Prasad, S., Vinay, S.: Development of anti-phishing browser based on random forest and rule of extraction framework. *Cybersecurity* **3**(1), 1–20 (2020)
7. Huang, Y., Qiao, X., Dustdar, S., Li, Y.: AoDNN: an auto-offloading approach to optimize deep inference for fostering mobile web. In: Proceedings of INFOCOM 2022, pp. 2198–2207 (2022)
8. Idelbayev, Y., Carreira-Perpinan, M.A.: An empirical comparison of quantization, pruning and low-rank neural network compression using the LC toolkit. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2021). <https://doi.org/10.1109/IJCNN52387.2021.9533730>
9. Iwahana, K., et al.: MADMAX: browser-based malicious domain detection through extreme learning machine. *IEEE Access* **9**, 78293–78314 (2021)
10. Li, T., Kou, G., Peng, Y.: Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. *Inf. Syst.* **91**, 101494 (2020)
11. Morell, J.A., Camero, A., Alba, E.: JSDoop and TensorFlow.js: volunteer distributed web browser-based neural network training. *IEEE Access* **7**, 158671–158684 (2019)
12. Palaniappan, G., Sangeetha, S., Rajendran, B., Sanjay, Goyal, S., Bindhumadhava, B.S.: Malicious domain detection using machine learning on domain name features, host-based features and web-based features. *Procedia Comput. Sci.* **171**, 654–661 (2020)
13. Rajapaksha, S., Kalutarage, H., Al-Kadri, M.O., Petrovski, A., Madzudzo, G., Cheah, M.: AI-based intrusion detection systems for in-vehicle networks: a survey. *ACM Comput. Surv.* **55**(11), 1–40 (2023). <https://doi.org/10.1145/3570954>
14. Rupa, C., Srivastava, G., Bhattacharya, S., Reddy, P., Gadekallu, T.R.: A machine learning driven threat intelligence system for malicious URL detection. In: Proceedings of ARES 2021, pp. 1–7. ACM (2021)

15. Saleem Raja, A., Vinodini, R., Kavitha, A.: Lexical features based malicious URL detection using machine learning techniques. *Mater. Today Proc.* **47**, 163–166 (2021)
16. Senanayake, J., Kalutarage, H., Al-Kadri, M.O.: Android mobile malware detection using machine learning: a systematic review. *Electronics* **10**(13), 1606 (2021). <https://doi.org/10.3390/electronics10131606>. <https://www.mdpi.com/2079-9292/10/13/1606>
17. Senanayake, J., Kalutarage, H., Al-Kadri, M.O., Petrovski, A., Piras, L.: Android source code vulnerability detection: a systematic literature review. *ACM Comput. Surv.* **55**(9), 1–37 (2023). <https://doi.org/10.1145/3556974>
18. Shabudin, S., Sani, N.S., Ariffin, K.A.Z., Aliff, M.: Feature selection for phishing website classification. *Int. J. Adv. Comput. Sci. Appl.* **11**(4), 587–595 (2020)
19. Shi, Y., Chen, G., Li, J.: Malicious domain name detection based on extreme machine learning. *Neural Process. Lett.* **48**(3), 1347–1357 (2018)
20. Smilkov, D., et al.: TensorFlow.js: machine learning for the web and beyond (2019). <https://doi.org/10.48550/ARXIV.1901.05350>. <https://arxiv.org/abs/1901.05350>
21. Sun, X., Tong, M., Yang, J., Xinran, L., Heng, L.: HinDom: a robust malicious domain detection system based on heterogeneous information network with transductive classification. In: *Proceedings of RAID 2019*, pp. 399–412. USENIX Association (2019)
22. Sun, X., Yang, J., Wang, Z., Liu, H.: HGDom: heterogeneous graph convolutional networks for malicious domain detection. In: *Proceedings of NOMS 2020*, pp. 1–9. IEEE (2020)
23. Tang, L., Mahmoud, Q.H.: A survey of machine learning-based solutions for phishing website detection. *Mach. Learn. Knowl. Extr.* **3**(3), 672–694 (2021)
24. Vadera, S., Ameen, S.: Methods for pruning deep neural networks. *IEEE Access* **10**, 63280–63300 (2022). <https://doi.org/10.1109/ACCESS.2022.3182659>
25. Vinayakumar, R., Soman, K., Poornachandran, P.: Detecting malicious domain names using deep learning approaches at scale. *J. Intell. Fuzzy Syst.* **34**(3), 1355–1367 (2018)
26. Yahya, F., et al.: Detection of phishing websites using machine learning approaches. In: *Proceedings of ICoDSA 2021*, pp. 40–47. IEEE (2021)
27. Yang, L., Liu, G., Dai, Y., Wang, J., Zhai, J.: Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework. *IEEE Access* **8**, 82876–82889 (2020)
28. Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M.: Character level based detection of DGA domain names. In: *Proceedings of IJCNN 2018*, pp. 1–8. IEEE (2018)
29. Yu, T., Zhauniarovich, Y., Khalil, I., Dacier, M.: A survey on malicious domains detection through DNS data analysis. *ACM Comput. Surv.* **51**(4), 1–36 (2018)
30. Zabihimayvan, M., Doran, D.: Fuzzy rough set feature selection to enhance phishing attack detection. In: *Proceedings of FUZZ-IEEE 2019*, pp. 1–6. IEEE (2019). <https://doi.org/10.1109/FUZZ-IEEE.2019.8858884>
31. Zamir, A., et al.: Phishing web site detection using diverse machine learning algorithms. *Electron. Libr.* **38**(1), 65–80 (2020)



Cyber Key Terrain Identification Using Adjusted PageRank Centrality

Lukáš Sadlek^{1,2}  and Pavel Čeleda^{1,2} 

¹ Institute of Computer Science, Masaryk University, Brno, Czech Republic

² Faculty of Informatics, Masaryk University, Brno, Czech Republic

sadlek@mail.muni.cz, celeda@fi.muni.cz

Abstract. The cyber terrain contains devices, network services, cyber personas, and other network entities involved in network operations. Designing a method that automatically identifies key network entities to network operations is challenging. However, such a method is essential for determining which cyber assets should the cyber defense focus on. In this paper, we propose an approach for the classification of IP addresses belonging to cyber key terrain according to their network position using the PageRank centrality computation adjusted by machine learning. We used hill climbing and random walk algorithms to distinguish PageRank's damping factors based on source and destination ports captured in IP flows. The one-time learning phase on a static data sample allows near-real-time stream-based classification of key hosts from IP flow data in operational conditions without maintaining a complete network graph. We evaluated the approach on a dataset from a cyber defense exercise and on data from the campus network. The results show that cyber key terrain identification using the adjusted computation of centrality is more precise than its original version.

Keywords: cyber key terrain · network centrality · host criticality · hill climbing · random walk

1 Introduction

The current need in cybersecurity is to interpret security as a business risk, according to Gartner [4]. Therefore, organizations focus on achieving the resilience of their missions that define objectives fulfilled by services and supporting assets, e.g., technologies [3]. Cybersecurity entities essential for mission execution are classified as *cyber key terrain* from the perspective of the organization's operations [6]. However, research often deals with a not specified mission in practice, when an organization requires only to maintain its operations, and these entities are called *critical assets*. An example of a critical asset from an implicit mission is a local domain name server (DNS) that allows for locating network resources.

Identification of *cyber key terrain* determines network devices, network services, cyber personas, and other network entities that provide an advantage

for attackers and defenders [6]. It is used for security assessment of cyber assets [1], probabilistic mission impact assessment [11, 23], mission-centric risk assessment [21], and mission-centric decision support [12]. It is of utmost importance for achieving security because cyber defense can become meaningless without knowing which cyber assets to protect. Solutions for asset discovery can determine the types of devices (e.g., [16]) and provide an asset inventory [13], but they often require tagging the critical devices manually. Therefore, automated approaches that determine critical cyber assets from raw network monitoring data can verify the content of populated asset inventory and its tags of critical devices.

An indispensable position in the asset criticality classification belongs to network centrality measures [10], identifying the most influential network entities according to the position in a graph, i.e., centrality. For example, the PageRank measure by *Brin and Page* [2] estimates the importance of web pages based on references by other web pages and the importance of these referencing web pages. Researchers also claim that the most critical network paths contain assets with high centrality [22], and high vertex centrality indicates asset criticality [9]. However, these methods can only be used to determine criticality plausibly if they consider computer network specifics, i.e., differences among vertices and edges from the network [10].

In this paper, we deal with two research questions. The first is: *How to determine which IP addresses from cyber terrain are the key according to the network communication?* The second question is related to the evaluation of the approach: *Does adjusting the PageRank centrality lead to better correctness of determining the cyber key terrain, and can it process IP flows from the real-world network?* Our focus is on the essential properties of IP flows [7], i.e., IP addresses, ports, and timestamps.

We contributed to the current state by studying PageRank-based cyber key terrain identification adjusted by optimization methods called hill climbing [8] and random walk [20]. These machine learning methods used static data to learn different damping factors for different port pairs to optimize output PageRank values. The learning success was measured using the harmonic mean of the criticality classification's precision and recall (F1 score). Estimated damping factors were then used for dynamic stream-based computation of PageRank centrality on IP flow data. The approach was evaluated using a public dataset from a cyber defense exercise [24] and a campus network.

This paper is organized as follows. Section 2 describes the current state of cyber key terrain mapping, network centrality measures, hill climbing and random walk methods, and IP flow. Section 3 proposes the approach for the identification of key IP addresses using the adjusted PageRank centrality. Consequently, Sect. 4 provides the evaluation and discusses the results, their importance, and the method's limitations. Last, Sect. 5 concludes the paper.

2 Related Work

The related work consists of four parts. The first part describes cyber key terrain mapping in general. The second part explains how centrality measures are used

to estimate the criticality of network entities. The third part is dedicated to hill climbing and random walk methods. The last part describes IP flow, its types, and its properties.

2.1 Cyber Key Terrain Mapping

Cyber key terrain mapping identifies entities essential for attackers and defenders from five cyberspace planes – the supervisory, the cyber persona, the logical, the physical, and the geographic plane [17]. For example, the logical plane, which is the most relevant plane for this paper, describes the software, network services, IP addresses, and domain names. Usual mapping approaches are based on crown jewels analysis, impact dependency graphs, and ontologies [6]. The crown jewels analysis determines cyber assets (i.e., crown jewels) that perform mission-critical functions [12], access other crown jewels, protect them, or enable them to work correctly. The impact dependency graphs contain assets, services, missions, mission steps, and their dependencies [6]. Ontologies represent the mission domain using data entities, their properties, and their relationships.

Mentioned approaches use various data sources to identify cyber terrain. For example, *Goodall et al.* [5] used LDAP queries, NetFlow traffic, Unix logs, and FTP logs. *Sun et al.* [23] used system call logs containing operations on files, processes, and sockets. *Motzek and Möller* [11] analyzed captured network traffic using Wireshark. As a result, created mission models contain various entities. *Musman et al.* [12] applied business process modeling during crown jewels analysis to express the mission using its activities and necessary IT assets. Cyber-ARGUS [1] uses an impact graph depicting dependencies among tasks, services, and cyber assets. A dependency graph by *Motzek and Möller* [11] contains business resources, functions, processes, and companies as vertices. *Silva et al.* [21] created a metamodel expressing mission, business processes, network services, infrastructure nodes, software, and their relationships.

The dynamic nature of cyber key terrain complicates its identification, which requires aging out old data related to mission-critical components [14] and modeling information that flows between them [6]. In our opinion, research works often focus on high-level models while not considering all the necessary details of filling the model with information extracted from raw data. Besides, manual modeling of missions is infeasible for large networks. These aspects must be addressed by methods that identify critical network assets.

2.2 Network Centrality Measures

Centrality measures form a significant group of methods for determining asset criticality [10] based on position in a graph. The most used variants are degree, betweenness, closeness, and eigenvector centralities. Degree centrality uses degrees of vertices, and the betweenness assigns higher criticality to vertices located between clusters, i.e., bridges. Closeness favors vertices close to others due to the easy transmission of resources in the network. Finally, eigenvector centrality considers the importance of neighbors connected by edges [10].

The PageRank centrality belongs to eigenvector centralities but redistributes the vertex importance only among vertices linked by its outgoing edges.

PageRank centrality can be computed iteratively based on equation:

$$PR_v = \frac{1-d}{n} + d \cdot \sum_{(u,v) \in E} \frac{PR_u}{deg^{out}(u)} \quad (1)$$

where u and v denote vertices, n is the count of vertices in a graph, E denotes a set of edges, and d is a damping factor. PR_v and PR_u are the PageRank centralities of vertex v in the current iteration and vertex u from the previous iteration, and $deg^{out}(u)$ is equal to the number of outgoing edges from vertex u . The damping factor with a default value of 0.85 represents the probability that the random surfer on web page v will continue with a random web page [2].

The most relevant variant of the original PageRank algorithm for our paper is Temporal PageRank [18], processing a stream of timestamped edges representing interactions between vertices. It uses the same damping factor as the static PageRank. However, it adds another transition probability that influences how fast the temporal PageRank converges to the static one because it describes the probability that the random surfer will choose the next edge from the stream to continue the random walk that follows links between vertices [18].

Centrality measures cannot be applied to network data directly if they consider all vertices and relationships equal [10]. Therefore, researchers often adjust the input graph for centrality measures. For example, *Stergiopoulos et al.* [22] used dependency risk paths consisting of asset dependencies for risk mitigation. Another centrality-based method was proposed by *Kay et al.* [9] to identify critical assets using a PageRank-based criticality score in the infrastructure network representing resource supplies between assets. On the contrary, *Oliva et al.* [15] applied different perspectives from the degree, betweenness, eigenvector, and PageRank centrality measures to identify the most influential vertices in a network. However, in our opinion, it is worth adjusting the computation of centrality measures instead of adjusting their input or combining them.

2.3 Hill Climbing and Random Walk

Hill climbing and random walk are machine learning methods used for finding optimal solutions to optimization problems, e.g., problems related to the boolean satisfiability problem (SAT). In general, the hill climbing method is usable for local search of solution space consisting of considered variables with respect to an objective function that defines the best solution. Its iterative algorithm slightly modifies the current solution to achieve a better neighboring solution [8], which often differs in the value of one variable.

The hill climbing algorithm can find a local optimum different from the global one. Therefore, it is often combined with the random restart, which assigns random values to all variables after finding the local optimum [8]. However, we can also add noise to the search algorithm using the random walk method instead of waiting until the local optimum is reached [20]. The random walk

method differs in assigning a random value to one conflict variable with some probability. For example, the conflict variable in the SAT problem is the one that appears in an unsatisfied clause [20]. Other approaches from this area include, e.g., simulated annealing and modified versions of the mentioned methods.

2.4 IP Flow

The paper focuses on two types of IP flows – unidirectional and bidirectional. A unidirectional IP flow is a set of IP packets with common properties (e.g., source and destination IP address, source and destination transport port) observed by the network observation point during a specific time window [7]. On the contrary, the bidirectional flow contains packets sent between two endpoints in both directions. The source of bidirectional flow is determined by the initiator of the first observed packet, by network perimeter, or arbitrarily [26].

Many applications using IP flows require their ordering. The start timestamp of IP flow in the IP Flow Information eXport (IPFIX) protocol is determined according to the timestamp of the first observed packet from the flow. In contrast, the end timestamp is the timestamp of the flow's last packet or packet before timeout will cause the flow to be exported [7].

3 Method for Cyber Key Terrain Identification

Our method for cyber key terrain identification at the logical cyberspace layer determines key IP addresses necessary for network operations. The innovation of the PageRank centrality measure consists of considering communication-specific damping factors based on IP flows. We use hill climbing and random walk methods to learn the damping factors that numerically encode critical IP addresses inside a previously unknown network environment. Further, the computation phase quickly processes IP flow data using a stream-based PageRank version.

3.1 Learning Phase

The learning phase is based on the iterative PageRank algorithm. The previous Eq. 1 implies that all PageRank values sum to one during one iteration when all PageRank values from the previous iteration summed to one because

$$\sum_{v \in V} PR_v = (1 - d) + d \cdot \sum_{v \in V} PR'_v = 1 - d + d = 1 \quad (2)$$

where PR'_v denotes the PageRank value of the vertex v in the previous iteration that appears exactly $deg^{out}(u)$ times in the sum. We modified Eq. 1 to consider the specifics of network communication by using damping factors adjusted to source and destination port pairs. We obtained

$$PR_v = \frac{1}{n} - \frac{PR'_v}{deg^{out}(v)} \cdot \sum_{w \in out(v)} d_{vw} + \sum_{u \in in(v)} \left(d_{uv} \cdot \frac{PR'_u}{deg^{out}(u)} \right) \quad (3)$$

where d_{uv} denotes the damping factor for edge (u, v) and $out(v)$ denotes the set of vertices w such that edge (v, w) exists in the graph. Similarly, $in(v)$ denotes a set of vertices from which an edge leads to the vertex v . Finally, the sum of all n PageRank values using adjusted damping factors is

$$\sum_{v \in V} PR_v = n \cdot \frac{1}{n} + \sum_{(v,u) \in E} \left(d_{vu} \cdot \frac{PR'_v}{deg^{out}(v)} - d_{vu} \cdot \frac{PR'_v}{deg^{out}(v)} \right) = 1 \quad (4)$$

because $d_{vu} \cdot \frac{PR'_v}{deg^{out}(v)}$ appears in the sum with positive and negative signs.

Iterative Algorithm 1 uses the hill climbing and the random walk methods to estimate the best values of damping factors. It starts with preprocessing the input graph containing port pairs that appear in more than some small fraction of IP flows, e.g., 0.1% (see also Sect. 4). Each node is assigned the initial PageRank value of $\frac{1}{n}$ and other necessary attributes, e.g., its IP address and predecessors with their port pairs. Initial damping factors for all processed source and destination port pairs are equal to default values of 0.85.

Algorithm 1: Learning phase

Input : graph, max_iterations, probability, results, heuristic

Output: best F1 score, best damping factors

```

1 preprocessing()
2 one_iteration_of_pagerank(graph, factors, results)
3 iterations ← 0
4 while  $F1\_score \neq 1$  and  $iterations \leq max\_iterations$  do
5   | assign_best_F1_score()
6   | port_pair ← choose_random_conflict_port_pair()
7   | if  $random\_experiment > 1 - probability$  then
8     | | factors[port_pair] ← random(0, 1)
9   | end
10  | else hill_climbing(heuristic) iterations ← iterations + 1
11  | one_iteration_of_pagerank(graph, factors, results)
12 end
13 assign_best_F1_score()

```

The first PageRank iteration on line 2 processes the current damping factors for port pairs and updates centrality values for all nodes using Eq. 3. It returns the F1 score and the list of misclassified nodes based on the ground-truth labels for individual IP addresses. Nodes are considered critical if their centrality is above the threshold of $\frac{1}{n}$. The F1 score is a suitable measure of classification's correctness for imbalanced datasets where most IP addresses are noncritical.

The while condition checks whether the computation is completed, i.e., all nodes are classified correctly or the maximum number of iterations is achieved. Then we update the best F1 score on line 5 if necessary and randomly choose a port pair located on the edge leading to a misclassified (conflict) node. We

continue with a random walk or hill climbing based on the probability from a random experiment. The solution space of optimization consists of damping factors and their values, while the objective function is the F1 score.

The random walk on line 8 chooses a random damping factor for the current port pair from 0 to 1 and allows escaping from the local minimum found by hill climbing. On the contrary, the hill climbing on line 10 gradually assigns values from 0 to 1 by 0.05 steps for the damping factor to find one that improves the F1 score for PageRank iteration. We use four heuristics to decide whether to change a damping factor for the port pair, if it has the same F1 score as before hill climbing. The first one, the *maximum* heuristic, always rewrites the current damping factor and achieves the maximum of changes. On the contrary, the *minimum* heuristic never rewrites the current value and achieves the minimum of changes. The third one uses an *average* of consecutive allowable values. Last, the *smallest difference* heuristic uses default values of 0.85 when possible to differ as least as possible from the default PageRank factors. When the loop is executed, the highest F1 score achieved during all iterations and damping factors valid during the best iteration form the result.

3.2 Computation Phase

Learned values of damping factors for individual port pairs are then used in the dynamic streaming PageRank computation using [18] since a computer network can produce such a large amount of data that it cannot be processed as a static graph. We adjusted the streaming algorithm by replacing each appearance of one common damping factor with a specific damping factor d_{uv} valid for the edge (u, v) . We also assume that one edge corresponds to one IP flow. The algorithm passes through the list of edges only once and has linear memory complexity with respect to the count of vertices. All port pairs not present during the learning phase are assigned the default damping factors equal to 0.85.

The stream-based algorithm allows near real-time processing of IP flows in practice with respect to time constraints. It is expected that the PageRank values for individual nodes may vary throughout the computation because the dynamic data causes some noncritical devices to obtain high centrality for a short time. Therefore, we advise focusing on high centrality in a longer time window because the approximate algorithm needs to converge to the static PageRank.

4 Evaluation

The method was evaluated on data captured in a network infrastructure emulated during a cyber defense exercise [24] and in the campus network of Masaryk University. In both cases, we used 70% of the data for learning damping factors on static graphs. The static graphs contained IP addresses and port pairs with a number of occurrences above 0.5% of total IP flows for data from the cyber defense exercise and 0.1% for university network data. These graphs also did not

Table 1. F1 scores for heuristics in the learning phase according to the networks of individual teams from the cyber defense exercise (T1 – T6). F1 scores for ten-minute-long (U10m) and one-hour-long capture from the university network (U1h).

Heuristic	T1	T2	T3	T4	T5	T6	U10m	U1h
Default PageRank	0.03	0.02	0.04	0.04	0.05	0.09	0.65	0.47
Minimum	0.75	0.77	0.72	0.74	0.77	0.60	0.84	0.75
Maximum	0.65	0.72	0.64	0.67	0.74	0.58	0.84	0.74
Average	0.66	0.67	0.65	0.65	0.76	0.58	0.83	0.75
The smallest difference	0.71	0.77	0.69	0.71	0.76	0.59	0.83	0.74

contain duplicate edges between IP addresses and timestamps. Consequently, the whole datasets were used to evaluate the stream-based PageRank variant.

We used 1,000 iterations of the while loop from Algorithm 1, which should provide enough time to improve damping factors demonstrably. The probability of applying random walk was set to 0.1, which implies that computation usually applied hill climbing. Moreover, we accomplished several runs for each heuristic. Averaged results are compared with the performance of the PageRank algorithm with the default damping factors of 0.85.

The stream-based variant used transition probability $\beta = 0.5$, expressing that the random surfer can follow the next edge with equal probability as ignoring it in the random walk. Since PageRank values of nodes dynamically change during stream-based computation, we regularly evaluated samples after some count of IP flows. The evaluation was accomplished on a personal computer with 16 GB RAM, four CPU cores, and a processor’s clock speed of 3.3 GHz. The method was implemented in Python.

4.1 Dataset from Cyber Defense Exercise

The first IP flow dataset contains data captured during a two-day cyber defense exercise in 2019 and ordered according to the observation time [24, 25], which also corresponds to sorting by the end timestamp. The network topology consisted of six equal networks (blue teams 1 – 6) and one global network providing services for all teams. Despite being provided with the same network topology, all teams behaved differently during the exercise, which allows for evaluating the method on six partial datasets. The flow capture interface captured bidirectional flows and was located between the networks of the teams and the global network. We consider only the forward direction of bidirectional flow from the initiator of the connection because it corresponds to the dependency the centrality measure should consider, and the backward direction contains a reply.

We labeled critical hosts from team networks and the global network manually. Table 1 contains averaged best F1 scores of learning on the static graphs for individual heuristics, while the default PageRank contains a converged value for comparison. The adjusted method was much better during the whole phase due

Table 2. The size of the processed graph for learning and measured time. Results are divided according to the networks of individual teams.

		T1	T2	T3	T4	T5	T6
Data	Nodes	554	1,380	542	884	503	219
	Edges	1,468	3,064	1,631	2,418	1,361	584
	IP flows	66,499	116,897	63,400	88,734	78,254	30,781
Time	Preprocessing	0.47 s	1.35 s	0.44 s	0.91 s	0.39 s	0.09 s
	Learning time	175.04 s	17.49 min	168.33 s	7.41 min	142.85 s	30.20 s
	Computation time	1.34 s	2.06 s	1.22 s	1.81 s	1.56 s	0.58 s

to estimating better damping factors for criticality classification. Results from Table 2 also show that the method’s execution time depends on the graph size in the learning phase and the total number of IP flows. Both phases are feasible in practice because the learning phase will be run only once, or we can combine factors learned on several partial graphs. The method will require learning the factors again if the probability distribution of the most frequent IP addresses and port pairs will considerably change.

Figure 1 contains average F1 scores for the stream-based computation phase obtained from five measurements per each heuristic, i.e., 20 measurements for each team. In each iteration, we used newly learned damping factors. We observed the classification’s progress after 4,000 loaded IP flows for the second team and 2,000 IP flows for other teams. In general, all heuristics provide almost always better F1 scores than the default PageRank, focusing only on source and destination ports, and follow a similar pattern. The comparison should be considered from a long-term perspective due to fluctuating activity of hosts.

4.2 Dataset from the Campus Network

The university network is assigned class B address space with/16 CIDR prefix. Network probes in the campus network were situated at the network edge. In our evaluation, we used IP flows from a ten-minute window captured during working hours on one Tuesday in March 2022 and a one-hour window captured during working hours on one Wednesday in February 2023. We executed the method ten times for each heuristic and accomplished five or eight samples of results during each execution. Data from the ten-minute-long window were unordered, but we sorted them according to the start timestamps, while in the second case, we used ordering according to the end timestamps.

The learning phase used a static graph that contained 90 vertices and 1,569 edges (ten-minute capture) and 78 vertices and 3,605 edges (one-hour capture), which could be preprocessed in 0.18 and 0.58 s on average. F1 scores from the learning phase are listed in Table 1, where heuristics achieved better F1 scores compared to the PageRank with default damping factors. The hill climbing and random walk with 1,000 iterations took 15.25 and 28.92 s on average. Finally, stream-based processing of 8.56 million flows representing ten-minute-long IP

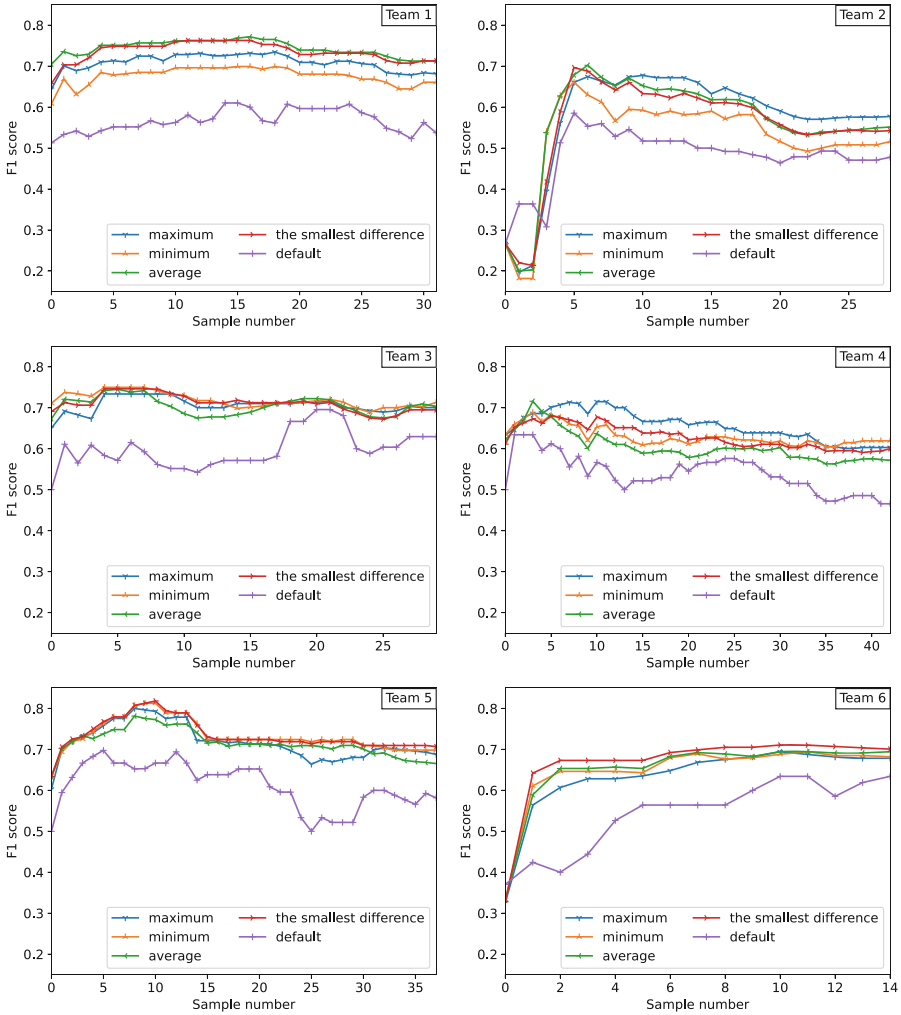


Fig. 1. Line graphs containing F1 scores for heuristics and the PageRank with default damping factor divided according to six team networks.

flow capture took only 81.66 s on average (including sorting). On the contrary, the one-hour-long IP flow capture contained 89.28 million flows and was processed in less than nine minutes. Results show that the method can quickly process large amounts of IP flow data.

A hard problem was obtaining result labels since we could not explicitly enumerate all critical IP addresses. Therefore, we focused on an overview of network subnets maintained by network administrators. It contains an organization unit (e.g., faculty) and a short description for each subnet. Critical IP addresses belong to subnets that can contain a lot of critical devices, according to the

Table 3. The number of true positives in the top 100 results according to samples (S1–S5), the number of hosts from the university network (N1–N5), and the average variance of true positives during the ten-minute-long window.

Heuristic	S1	S2	S3	S4	S5	N1	N2	N3	N4	N5	Var
Default PageRank	24	24	23	23	23	35	32	30	28	29	–
Minimum	26.4	28.3	27.6	27.9	27.8	41.4	37.8	37.8	37.0	35.7	18.9
Maximum	25.2	25.8	24.5	24.2	24.2	38.1	34.7	33.9	32.4	31.4	13.6
Average	22.8	23.3	22.3	22.6	22.4	33.6	30.2	29.3	27.8	27.6	10.3
The smallest difference	24.0	24.5	23.5	23.4	23.4	36.9	33.4	32.6	30.8	30.2	7.72

description. However, the overview does not provide exact but consistent labels. After evaluating the results, we manually added false positives that were critical to the labels and re-executed the whole method in a way that did not devalue the results of the default PageRank.

False positives confirmed the correctness of criticality classification. External false positives were usually Google servers (1e100.net), Facebook servers, domains for downloading updates to user devices, and generally important services, such as Google DNS. They are often influenced by the activity of students. False positives from the internal network are typically devices from the wireless network with predetermined IP ranges, staff devices, VPN addresses, and hosts providing less essential network services. These false positives were often quickly replaced by another in the ten-minute-long dataset.

Table 3 shows that the minimum, the maximum, and the smallest difference heuristics increase the count of recommended critical devices and devices from the campus network in the sorted ten-minute-long time window. The most significant improvement was achieved by the minimum heuristic that recommends approximately two to five more critical devices compared to the default damping factors. The variance describes an opportunity to tune damping factors using heuristics in practice. When executed on unordered data, the PageRank with default values achieved approximately the same results differing at most in only one device. The average heuristic achieved better results on non-sorted data, but the other heuristics did not.

In the case of one-hour-long capture sorted according to the end timestamps, the top 100 IP addresses contained more critical IP addresses from the university network, even for default PageRank (see Table 4). Therefore, the improvement is not visible among the top 100 results because the top results form only a small part of all results. The best heuristics, in this case, are alternately maximum, minimum, and the smallest difference, while the average heuristic demonstrates the worst criticality classification.

4.3 Limitations

Several possible limitations of the proposed method and the performed evaluation need to be discussed. First, samples of results may not be accomplished at the right moments, and necessary progress could remain hidden. Second, the

Table 4. The number of true positives in the top 100 results according to samples (S1 – S8), the average number of hosts from the university network (N), and the average variance of true positives in these samples during the one-hour-long window.

Heuristic	S1	S2	S3	S4	S5	S6	S7	S8	N	Var
Default PageRank	41	41	43	42	41	41	41	41	59.1	–
Minimum	40.3	41.5	41.1	40.7	40.1	40.1	40.3	39.9	59.2	2.7
Maximum	40.5	41.7	41.2	40.8	40.4	39.9	39.9	39.3	58.3	11.2
Average	38.6	39.8	39.7	39.1	38.7	38.0	38.6	38.3	56.9	7.8
The smallest difference	40.4	41.4	41.1	40.6	40.5	40.3	40.2	39.9	58.4	3.8

static graph for learning can only contain the most influential vertices and port pairs. The learning phase should be accomplished on the same type of flows (unidirectional or bidirectional) to estimate consistent damping factors. Moreover, damping factors should be tuned to their best values in practice, and other IP flow attributes could be considered, e.g., length of communication.

Third, the streaming algorithm [18] was designed for short interactions with only one timestamp. However, IP flows contain start and end timestamps and may not be optimally sorted, e.g., because of timeouts that will flush incompleting flows from a collector. Furthermore, the wrong forward direction of bidirectional flow, e.g., because of the not captured first packet, can also influence the results. Last, a large amount of IP flows will not fit into the main memory. However, maintaining a sliding window of IP addresses with the highest centrality and the total value of centrality for the removed vertices throughout the algorithm is feasible. This value would be equally divided among the retained IP addresses.

5 Conclusion

In this paper, we proposed a method for determining which IP addresses from cyber terrain are the key by adjusting PageRank centrality. We used two machine-learning methods – hill climbing and random walk – to distinguish damping factors according to the source and destination ports. Despite using only essential properties of IP flows, we showed that this approach leads to better correctness of classifying critical hosts compared to the default damping factors, except for the natural temporal fluctuation of the stream-based PageRank variant. The evaluation also proved that the PageRank centrality is suitable for determining IP addresses related to critical network’s and organization’s services.

When using only the top 100 results, the precision between using default and estimated damping factors was almost equal with the increased count of flows. The method can process IP flows from the real-world network using a stream-based PageRank algorithm with the estimated damping factors in practice. Supplementary materials at [19] contain a proof-of-concept implementation of the learning and computation phases with the ground-truth labels for IP addresses from the cyber defense exercise data published in [25].

Acknowledgement. This research was supported by ERDF project “CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence” (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

References

1. Barreto, A.B., Costa, P.C.: Cyber-ARGUS - a mission assurance framework. *J. Netw. Comput. Appl.* **133**, 86–108 (2019). <https://doi.org/10.1016/j.jnca.2019.02.001>
2. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* **30**(1), 107–117 (1998). [https://doi.org/10.1016/S0169-7552\(98\)00110-X](https://doi.org/10.1016/S0169-7552(98)00110-X)
3. Caralli, R.A., Allen, J.H., White, D.W.: *CERT Resilience Management Model - CERT-RMM*. Addison-Wesley Educational Publishers Inc. (2016)
4. Gartner Unveils the Top Eight Cybersecurity Predictions for 2022-23. Gartner, Inc. <https://www.gartner.com/en/newsroom/press-releases/2022-06-21-gartner-unveils-the-top-eight-cybersecurity-predictio>. Accessed 3 Feb 2023
5. Goodall, J.R., D’Amico, A., Kopylec, J.K.: Camus: automatically mapping cyber assets to missions and users. In: MILCOM 2009-2009 IEEE Military Communications Conference, pp. 1–7. IEEE (2009). <https://doi.org/10.1109/MILCOM.2009.5380096>
6. Guion, J., Reith, M.: Cyber terrain mission mapping: tools and methodologies. In: 2017 International Conference on Cyber Conflict (CyCon US), pp. 105–111. IEEE (2017). <https://doi.org/10.1109/CYCONUS.2017.8167504>
7. Hofstede, R., et al.: Flow monitoring explained: from packet capture to data analysis with NetFlow and IPFIX. *IEEE Commun. Surv. Tutor.* **16**(4), 2037–2064 (2014). <https://doi.org/10.1109/COMST.2014.2321898>
8. Jacobson, S.H., Yücesan, E.: Analyzing the performance of generalized hill climbing algorithms. *J. Heuristics* **10**, 387–405 (2004). <https://doi.org/10.1023/B:HEUR.0000034712.48917.a9>
9. Kay, B., Lu, H., Devineni, P., Tabassum, A., Chintavali, S., Lee, S.M.: Identification of critical infrastructure via PageRank. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 3685–3690 (2021). <https://doi.org/10.1109/BigData52589.2021.9671620>
10. Kim, A., Kang, M.H.: Determining asset criticality for cyber defense. Technical report, Naval Research Laboratory (2011). <https://apps.dtic.mil/sti/pdfs/ADA550373.pdf>
11. Motzek, A., Möller, R.: Context- and bias-free probabilistic mission impact assessment. *Comput. Secur.* **65**, 166–186 (2017). <https://doi.org/10.1016/j.cose.2016.11.005>
12. Musman, S., Tanner, M., Temin, A., Elsaesser, E., Loren, L.: A systems engineering approach for crown jewels estimation and mission assurance decision making. In: 2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), pp. 210–216. IEEE (2011). <https://doi.org/10.1109/CICYBS.2011.5949403>
13. Netbox documentation (2022). <https://netbox.readthedocs.io/en/stable/>. Accessed 15 Dec 2022
14. Noel, S., Dudman, T., Trepagnier, P., Badesha, S.: Mission models for cyber-resilient military operations. Technical report, MIT Lincoln Laboratory Lexington United States (2018). <https://apps.dtic.mil/sti/pdfs/AD1091410.pdf>

15. Oliva, G., Esposito Amideo, A., Starita, S., Setola, R., Scaparra, M.P.: Aggregating centrality rankings: a novel approach to detect critical infrastructure vulnerabilities. In: Nadjm-Tehrani, S. (ed.) CRITIS 2019. LNCS, vol. 11777, pp. 57–68. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-37670-3_5
16. Orion Platform – Scalable IT Monitoring. SolarWinds (2022). <https://www.solarwinds.com/solutions/orion>. Accessed 15 Dec 2022
17. Raymond, D., Cross, T., Conti, G., Nowatkowski, M.: Key terrain in cyberspace: seeking the high ground. In: 2014 6th International Conference on Cyber Conflict (CyCon 2014), pp. 287–300 (2014). <https://doi.org/10.1109/CYCON.2014.6916409>
18. Rozenshtein, P., Gionis, A.: Temporal PageRank. In: Frasconi, P., Landwehr, N., Manco, G., Vreeken, J. (eds.) ECML PKDD 2016. LNCS, vol. 9852, pp. 674–689. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46227-1_42
19. Sadlek, L., Čeleda, P.: Supplementary materials: cyber key terrain identification using adjusted PageRank centrality. Zenodo (2023). <https://doi.org/10.5281/zenodo.7884228>. Accessed 2 May 2023
20. Selman, B., Kautz, H.A., Cohen, B., et al.: Noise strategies for improving local search. In: AAAI, vol. 94, pp. 337–343 (1994). <https://cdn.aaai.org/AAAI/1994/AAAI94-051.pdf>
21. Silva, F.R.L., Jacob, P.: Mission-centric risk assessment to improve cyber situational awareness. In: Proceedings of the 13th International Conference on Availability, Reliability and Security. ARES 2018. Association for Computing Machinery, New York (2018). <https://doi.org/10.1145/3230833.3233281>
22. Stergiopoulos, G., Theocharidou, M., Kotzanikolaou, P., Gritzalis, D.: Using centrality measures in dependency risk graphs for efficient risk mitigation. In: Rice, M., Sheno, S. (eds.) ICCIP 2015. IFIPAICT, vol. 466, pp. 299–314. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26567-4_18
23. Sun, X., Singhal, A., Liu, P.: Who touched my mission: towards probabilistic mission impact assessment. In: Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense. SafeConfig 2015, pp. 21–26. Association for Computing Machinery, New York (2015). <https://doi.org/10.1145/2809826.2809834>
24. Tovarňák, D., Špaček, S., Vykopal, J.: Traffic and log data captured during a cyber defense exercise. Data Brief **31**, 105784 (2020). <https://doi.org/10.1016/j.dib.2020.105784>
25. Tovarňák, D., Špaček, S., Vykopal, J.: Traffic and log data captured during a cyber defense exercise. Zenodo (2020). <https://doi.org/10.5281/zenodo.3746129>. Accessed 9 Mar 2023
26. Trammell, B., Boschi, E.: Bidirectional flow export using IP Flow Information Export (IPFIX). RFC 5103, Internet Engineering Task Force (2008). <http://www.ietf.org/rfc/rfc5103.txt>. Accessed 5 Mar 2023



Machine Learning Metrics for Network Datasets Evaluation

Dominik Soukup¹ , Daniel Uhříček² , Daniel Vašata¹ ,
and Tomáš Čejka³ 

¹ Faculty of Information Technology, CTU in Prague, Prague, Czech Republic
{soukudom, vasatdan}@fit.cvut.cz

² Faculty of Information Technology, Brno University of Technology,
Brno, Czech Republic
poliakov@fit.vut.cz

³ CESNET a.l.e., Prague, Czech Republic
cejkat@cesnet.cz

Abstract. High-quality datasets are an essential requirement for leveraging machine learning (ML) in data processing and recently in network security as well. However, the quality of datasets is overlooked or underestimated very often. Having reliable metrics to measure and describe the input dataset enables the feasibility assessment of a dataset. Imperfect datasets may require optimization or updating, e.g., by including more data and merging class labels. Applying ML algorithms will not bring practical value if a dataset does not contain enough information. This work addresses the neglected topics of dataset evaluation and missing metrics. We propose three novel metrics to estimate the quality of an input dataset and help with its improvement or building a new dataset. This paper describes experiments performed on public datasets to show the benefits of the proposed metrics and theoretical definitions for more straightforward interpretation. Additionally, we have implemented and published Python code so that the metrics can be adopted by the worldwide scientific community.

1 Introduction

Network traffic analysis is a key area to ensure the security of our networks and protection against attacks. Due to the growing amount of traffic and increasing portion of encrypted communication, it is challenging to use traditional methods based on deep packet inspection (DPI) and application data analysis [17]. Machine learning (ML) has promising results in detecting security events even in such a challenging environment and helps to increase the level of network security. Many researchers have already published their solutions to classify network traffic or identify malware in encrypted traffic [1, 13, 16]. However, used datasets can include errors such as mislabeled, duplicated, and missing data or do not include enough information to provide reliable classification [10, 20]. This approach also brings the question of how to ensure that the results of ML solutions

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 307–320, 2024.

https://doi.org/10.1007/978-3-031-56326-3_22

will be consistent over time and applicable among different environments. Due to this, it is important to enhance ML automation, e.g., with Active learning (AL) and data drift detection methods. AL benefits from query strategy algorithms to select the most valuable samples for the dataset. Data drift detection can identify changes in ML performance to trigger training. Unfortunately, most of the community is focused on ML performance metrics instead of the dataset, which is necessary for good results [21]. The wrong or bad quality dataset can lead to false positive results that can have a negative impact on network security. Research regarding dataset quality is in the early stage, but the first prototypes are being delivered [12,21].

In this paper, we bring novel metrics that assess dataset quality and suitability for identified network traffic classification tasks. We react to the current state, where the quality of many publicly available datasets is assessed only by the total amount of flows, which can be insufficient. Large datasets are mainly useful for the intensive evaluation of developed ML algorithms, but applying them in training is more computationally demanding. Additionally, even a large stable dataset can become obsolete sooner or later in an evolving production network, and the precision level of ML models is affected, as it is discussed by Brabec et al. [2]. Network traffic is dynamic and requires an up-to-date dataset for reliable results. This requirement is specific to the network domain, and other domains, such as image classification, are not necessarily sensitive to such dynamic behaviour over time. We investigate the current state of the art and identify its limitation. Metrics from Wasielewska et al. [21], Maillor et al. [14], and Lorena et al. [12] bring promising results; however, our evaluation of public networking datasets shows limitations in the metric calculation, normalization, and robustness.

To evaluate datasets, we use three novel metrics that are capable of universally assessing linear and non-linear multi-class tasks. The identified metrics can be further used by AL or data drift methods to more accurately assess the dataset's quality over time to build a relevant dataset for the target use case. The main contributions of this paper are:

- We propose three novel metrics that enhance the current state of the art in evaluating network traffic datasets.
- We provide theoretical definitions and interpretation of the proposed metrics.
- We evaluate the proposed metrics in detail on publicly available datasets to demonstrate described benefits.
- We share our implemented code publicly for further use by the community.

The rest of the article is organized as follows. Section 2 introduces related work in the literature. Section 3 describes in detail the proposed metric and provides theoretical consideration. Section 4 contains experimental setup used datasets and software package used for evaluation. Section 5 present findings from experiments carried out on publicly available datasets. Finally, Sect. 6 contains conclusions and discusses future work.

2 Related Work

Usually, the quality of ML-enabled systems is assessed based on the used ML algorithm and its performance metrics. Celdrán et al. [3] propose the RITUAL platform to quantify the trustworthiness of supervised ML algorithms. A similar approach based on ML performance is researched by Koh et al. [8], who suggest a solution based on benchmarking ML models using a collection of predefined datasets. The goal is to test the generalization of ML models over a variety of unseen data from different domains.

These methods are beneficial for ML model verification if we have already known datasets. For unknown datasets, this approach provides limited value. We must analyze if the input dataset provides reasonable structure and information to apply ML techniques. This area is rarely explored, especially in the network traffic datasets domain. Without this, we can have volatile results, leading to many unwanted false positive defections.

Another approach in the literature is leveraging normalization to improve overall dataset quality. The effect of data normalization and dimensional reduction was studied by Obaid et al. [15] in the intrusion detection NSL-KDD dataset. Gonzalez [5] proposed a method to assess the influence of specific data preparation steps on the model performance. Yoon et al. [22] proposed a novel meta-learning framework for data evaluation that is jointly optimized with the target task predictor model. These approaches focus on dataset optimization that modifies the input dataset to improve ML model operation. However, the impact or correlation with dataset quality is not considered.

Some papers examine data quality and what prerequisites should be considered for a good dataset. Chen et al. [4] describe data quality attributes: comprehensiveness, correctness, and variety used to enhance data quality to improve ML results. The list of attributes for data quality assessment varies among authors who use different segmentation. For example, Lee et al. [11] use categories: Intrinsic, Accessibility, Contextual, and Representational. The need to extend data quality to the dataset quality is introduced by Soukup et al. [20]. He suggests definitions of several categories (good/minimal/better dataset) to consider the complete view of the whole dataset. Wasielewska et al. [21] introduces a dataset quality assessment metric based on permutation tests with different permutation levels. This work provides promising results of binary classification.

The area of dataset quality metrics is also analyzed by researchers from other domains. Lorena et al. [12] published a survey of data complexity measures. Based on the realized survey, the author group the data complexity measure into the following groups: (i) feature-based, (ii) linearity, (iii) neighborhood, (iv) network, (v) dimensionality, and (vi) class imbalance. These groups include 22 metrics. The main aim of these metrics is to characterize datasets to help researchers to select learning and preprocessing techniques on an unknown domain; however, evaluation on existing datasets is missing. Also, some measures are defined for binary classification problems only, and any multiclass problem must be first decomposed into multiple binary sub-problems. Several metrics are limited to linear tasks only. While evaluating the implementation of the proposed metrics

created by Komorniczak et al. [9], we encountered memory errors for bigger datasets due to the high memory requirements of selected metrics. Maillor et al. [14] aimed to deal with informative metrics for big datasets. The author implements metrics from Lorena et al. in Apache Spark to allow big data processing. Also, the paper proposes two novel metrics focused on dataset density. These metrics are based on fixed classifiers – the nearest neighbors (1NN) and decision tree (DT) which can lead to non-optimal results and are less universal in different classification scenarios.

3 Proposed Dataset Metrics

This section introduces suggested new metrics with their benefits and expected behavior. To the best of our knowledge, there are no metrics that evaluate the quality of network traffic datasets from the same perspective as ours. The defined metrics are evaluated in detail in Sect. 5.

3.1 Metric 1 (M_1) - Dataset Redundancy

The first metric is focused on the level of dataset size redundancy. Using this measure, one can estimate what portion of the original dataset can be randomly removed while keeping the classification performance drop below the certain controlled level. Zero redundancy indicates not enough data for the classification task. For evaluation, we use the pool of classifiers and acceptance level α to generally assess the level of redundancy with a certain probability.

Dataset redundancy was mentioned as one of the requirements by Soukup et al. [20] since many available datasets are unnecessarily large to provide the same quality of results. Maillor et al. [14] introduced a dataset redundancy method based on 1NN density comparison between the original dataset and reduced dataset by 50%. The analysis is done for redundancy levels of 25%, 50%, and 75% without more detailed steps.

Our metric defines redundancy based on the performance metric, e.g., F1 score, from the pool of selected classifiers. Let D denote the dataset and $\alpha \in (0, 1)$ be the relative acceptance level of the performance. Moreover, let $\mathcal{C} = \{c_1, \dots, c_m\}$ be a pool of m classifier models. For $\varphi \in (0, 1)$ and for $i = 1, \dots, k$, where k is some positive number, we denote by $D_{\varphi,i}$ the i th dataset of relative size φ sampled randomly (without replacement) from D . Hence, $D_{\varphi,i}$ is a subset of D that contains $\lfloor \varphi |D| \rfloor$ data points, where $|D|$ is the number of data points in D . $D_{\varphi,i}$ is used for training of the models from the pool \mathcal{P} . Moreover, by $T_{\varphi,i}$ we denote the remaining part of D , i.e.

$$T_{\varphi,i} = D \setminus D_{\varphi,i},$$

that is used for testing of the performance of each trained model.

For a given $\varphi \in (0, 1)$ each $i = 1, \dots, k$ and $j = 1, \dots, m$ one takes the model $c_j \in \mathcal{C}$, trains it using dataset $D_{\varphi,i}$ and evaluates it using dataset $T_{\varphi,i}$. Let us denote the obtained testing performance by $\tau(c_j, D_{\varphi,i})$.

By taking $\min_i \tau(c_j, D_{\varphi,i})$ we observe the minimal performance of a j th model over all random splits of the dataset at a level φ . Now we will compare this value with the lowest acceptable performance level that is given by $\tau_\alpha = \alpha \max_{i,j} \tau(c_j, D_{0.99,i})$, i.e. is relative according to maximal performance measured over all dataset splits and all models from the pool for $\varphi = 0.99$. If there is at least one model from the pool of classifiers \mathcal{C} for which its minimal performance $\min_i \tau(c_j, D_{\varphi,i})$ is above this lowest acceptable performance level, the redundancy of the dataset should be larger than $1 - \varphi$.

This means that the redundancy can be defined as maximum of all possible $1 - \varphi$ satisfying the above condition $\max_j \min_i \tau(c_j, D_{\varphi,i}) \geq \tau_\alpha$. Formally,

$$M_1 = 1 - \inf_{\{\varphi | \max_j \min_i \tau(c_j, D_{\varphi,i}) \geq \tau_\alpha\}} \varphi. \quad (1)$$

To implement the search for the infimum we used the half splitting search.

Note that there can be more policies to set the redundancy level where we, for example, do not require to fit above τ_α for all runs of at least one model. For stability purposes, we take a strict policy that insists on acceptable results for all runs from a specific percentage level. The evaluation process of single ML model is depicted in Fig. 1. The metric domain is $[0, 1]$, and it describes the percentage size of the dataset that is redundant.

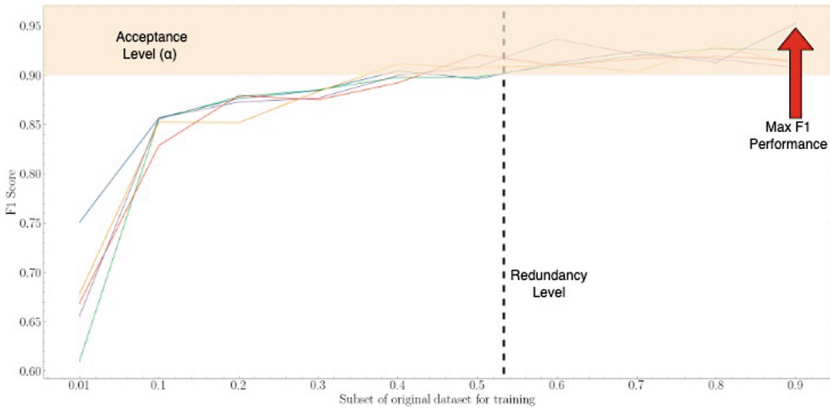
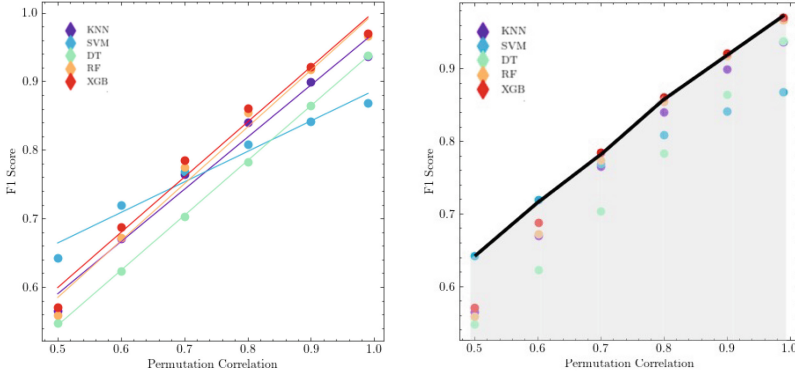


Fig. 1. Visualization of M_1 calculation workflow. The figure depicts single model with five independent runs.

3.2 Metric 2 (M_2) - Dataset Association Quality

The second metric evaluates the level of association between labels and respective data. Especially for the public datasets, we don't know how the dataset was collected and if it's meaningful to apply ML algorithm on such a dataset. The level of association is estimated based on permutation tests which are interpreted by this novel metric. As a result, we get an estimate of how strong is the connection between data and related labels.



(a) Original permutation slope visualization which selects DT as the top performing ML model, however, XGB is more suitable
 (b) Maximized Area Under Curve (AUC) across the whole pool of classifiers

Fig. 2. Comparison of original permutation slope metric with the proposed M_2 metric

Permutation tests [18] are known area in state of the art. Wasielewska et al. [21] introduced permutation tests using different permutation percentage levels to estimate the relationship between labels and data with a certain sensitivity. During our evaluation on public datasets, we identified incorrect behavior of permutation slope that is used for the interpretation of results. In several cases, it selected the wrong ML model as a top performing, as depicted in Fig. 2a, and the normalization of received values is not defined. We follow up on this work and enhance this limitation. For each permutation level from 0.5 to 0.99 the best performing ML classifier from the pool of classifiers is selected, which yields a piecewise-linear curve f in the domain $[0.5, 0.99]$. Then the Area Under Curve (AUC) is calculated and normalized.

This gives us the best possible association of labels and data among all selected models in the pool of classifiers. Moreover, we extended this metric to support both binary and multiclass classifications. A comparison of permutation slope and the proposed metric calculation is depicted in Fig. 2.

To define the metric formally, let us assume that γ indicates the permutation level and for $i = 1, \dots, k$ we denote by $P_{\gamma,i}$ the i th sample given as the original dataset D where the fraction of γ of labels was randomly permuted. On this sample dataset one uses cross-validation with ℓ folds to evaluate each model c_j from the pool \mathcal{C} of m classification models. Let us denote by $\bar{\tau}(c_j, P_{\gamma,i})$ the obtained performance.

The function f at permutation level γ is defined by

$$f(\gamma) = \max_j \frac{1}{k} \sum_{i=1}^k \bar{\tau}(c_j, P_{\gamma,i}).$$

The area under curve (AUC) is then given by

$$\text{AUC} = \int_{0.5}^1 f(\gamma) d\gamma.$$

Finally, the metric M_2 is given by the normalized value of the AUC:

$$M_2 = \frac{0.5 - \frac{\text{AUC}}{\max_{\gamma} f(\gamma)}}{0.25} \quad (2)$$

In calculations the function f is taken as a piecewise linear function evaluated only at breaking points $\gamma = 0.5, 0.6, \dots, 0.9, 0.99$. The metric is shown in Fig. 2b.

The normalization is done by subtracting the AUC from 0.5 which effectively means taking the Area Above Curve. Then it is divided by the highest score of the best ML model, which effectively changes the range of possible values of f to $[0, 1]$. Then the division by 0.25 is to make it relative to the largest value possible after the previous transformations.

The metric domain is $[0, 1]$ and it corresponds to the level of association between data and labels in the dataset. Generally, the permutation slope and its sensitivity are dependent on the imbalanced ratio and selected performance metric [21]. Therefore, this metric cannot be used to easily compare the results of different datasets. However, the focus of this paper is on a single dataset.

3.3 Metric 3 (M_3) - Dataset Class Similarity

The third metric looks at the dataset classes. We propose a method to estimate how instances of different classes are similar to each other. In other words, how complex the classification task generally is on the input dataset. The metric measures relative class similarity using autoencoders and their respective reconstruction error. Calculated relative similarity lays out direct indicators of how prone are other machine learning models to misclassifications.

Methods of representation learning, and specifically autoencoders, have already been successfully applied to network data. For example, Zhang et al. [23] proposed a framework for network data feature extraction using autoencoders and successfully evaluated extracted features based on their clustering performance. Hwang et al. [6] applied autoencoders to image datasets comparison and examined how reconstruction error can be used to predict inter-dataset similarity. Their solution introduces multiple autoencoders, each trained on a separate dataset and used to calculate reconstruction error on rest. Moreover, they state that the method has the potential for inter-class similarity within the same dataset. Based on their results, authors discussed that such or similar approach might be beneficial for inter-class similarity, but, in their words, more experiments are needed. Our metric follows up on their work, showing how a similar approach would be applied to network and security datasets. We leverage the fact that security datasets are often imbalanced and have some majority (or benign class), and instead of training n autoencoders for each class, we train only one autoencoder on the benign class giving us information about how much other

classes could be confused with the majority class. Later in our experiments, we see that this method gives consistent results and corresponds well both with our prior knowledge about tested datasets and with other evaluated metrics.

The metric is calculated in two phases – initialization phase and the evaluation phase. In the initialization phase, an autoencoder model is trained on one of the classes as a base class (most commonly a majority class or a benign class). The instances of the base class are split in a ratio of 8:2 to training and validation datasets. During our experiments, we encountered that the relative similarity (calculated relative to the base class) is consistent even when the underlying model is randomly reinitialized differently and has different numbers of hidden layers and their sizes.

In the evaluation phase, the trained autoencoder is then used to calculate the reconstruction error for each of the classes – both in absolute values and relative values to the class used for training. In our reports, we either state all values (absolute and relative) for non-base classes or the weighted average of the relative error to describe the dataset as a whole. Formally, the described process can be stated as follows. We denote individual classes as C_i and base class as B . Reconstruction error for instance x is denoted as $\text{err}_{\mathcal{A}}(x)$. We define mean absolute error (MAE) for each of the classes based on the reconstruction error from the trained autoencoder (Eq. 3). MAE is used to define mean relative error (MRE) for each of the classes (Eq. 4). Finally, metric M_3 is defined as a weighted average over MRE values of non-base classes (Eq. 5).

$$\text{MAE}_{C_i} = \frac{1}{|C_i|} \sum_{x \in C_i} \text{err}_{\mathcal{A}}(x) \quad (3)$$

$$\text{MRE}_{C_i} = \frac{\text{MAE}_{C_i}}{\text{MAE}_B} \quad (4)$$

$$M_3 = \frac{1}{\sum_{C_i, C_i \neq B} |C_i|} \sum_{C_i, C_i \neq B} |C_i| \cdot \text{MRE}_{C_i} \quad (5)$$

Since metric M_3 represents average relative reconstruction error over all instances of non-base classes, the domain are positive numbers – multiples of reconstruction error on base class. Corresponding with our experiments in Sect. 4, $M_3 \leq 1$ means that autoencoder performs similarly for all the classes and from this point of view, classes are similar, or even some classes might seem as a subset of the base class in the feature space. On the other hand, if $M_3 > 1 + K$, classes are different in the feature space and might be easier to separate from the base class.

Table 1. Summary of selected datasets description for defined experiments

Dataset	Samples	Features	Classes
DoH [7]	20,000	24	2
TLS [13]	125,000	44	5
CICIDS2017 [19]	71,599	78	6
CICIDS2017 Fixed [10]	63,321	86	6

4 Experiments Setup

This paper introduces novel metrics to evaluate dataset suitability for the target use case. We are considering binary and multi-classification tasks that are involved in selected experiments. In this section, we describe our experiments and selected datasets for evaluation of proposed metrics.

4.1 Datasets

The applicability of the proposed solution is demonstrated on three publicly available network datasets – DoH – Real-World [7], CESNET-TLS22 [13], and CICIDS2017 [19]. DNS over HTTPS (DoH) is an encrypted communication protocol with a domain name resolver that increases the privacy of internet users. Jerabek et al. [7] published this dataset from a large ISP network with labels for binary classification. Luxemburk et al. [13] recently collected a dataset from the national network CESNET2. It includes 191 labels of network services using TLS traffic. Sharafaldin et al. [19] published the CICIDS2017 dataset that included malware samples with classes of benign and seven common attacks. This dataset was investigated by Lanvin et al. [10], who identified several errors in this dataset and published a new version with the necessary fixes.

To make easier development and proper evaluation of our metric, we selected subsamples of input datasets. The description of datasets for the provided experiments is summarized in Table 1. Analysis of findings from experiments is described on Sect. 5.

4.2 Experiments

After the introduction of the proposed metrics, we provide a list of experiments to demonstrate the value on real datasets. The results of these experiments are described in Sect. 5.

Case Study 1: Evaluation for Binary Classification. In this case study, we take the DoH dataset for several tests. In the first test, we test sensitivity to a reduced size of the dataset. We started with the original dataset, which set the baseline, and then we randomly removed the defined portion. The second test

investigates the impact of mislabels, which we created by randomly switching specific percentages of labels in a balanced way. The last test analyses the sensitivity of dataset imbalance. The imbalance is created by setting the different ratio between negative (non-doh) and positive (doh) classes, however, the total size is always the same.

Case Study 2: Evaluation for Multi-class Classification. The second case study contains a multi-class CESNET-TLS22 dataset. The main aim is to validate metrics application on a multi-class dataset that has not been considered in several related papers and related metrics. Moreover, the classification of TLS encrypted traffic is more complex than the previous binary classification for DoH.

Case Study 3: Evaluation of Bad and Corrected Dataset. The final case study is focused on the CICIDS2017 dataset that contains known errors identified by Lanvin et al. [19]. His paper describes each error with respective evaluation in detail. However, a comparison between completely fixed and original datasets is missing. This case study evaluates the CICIDS2017 dataset before and after the application of suggested fixes to verify findings from other researchers who used this dataset.

4.3 Software Package

All introduced metrics are implemented in the public Github repository¹. The implementation is done in Python and in Jupyter Notebook, which is popular in the community and allows easy usage of proposed methods. Part of the notebook is an enhanced visualization of the proposed metric for further analysis of their behavior. Together with the novel metric, we included other known dataset metrics to allow a complete valuation of the input dataset.

5 Evaluation of Introduced Metrics

In this section, we describe the received results from defined experiments. For all metric, we use consistent configuration to provide reliable verification. Parameters of input datasets are described in each experiment.

5.1 Case Study 1: Evaluation for Binary Classification

In the first case study, we analyze the proposed metrics on binary classification tasks for DoH traffic. For the first test, we take 10,000 samples from DoH—Real-World and calculate all metrics to set the baseline for the reduction of samples. We reduced 10, 30, and 50% of the samples to see the impact of the introduced metric. The results are summarized in Table 2a. The original input dataset gets

¹ <https://github.com/soukudom/NDVM>.

Table 2. (a) (Test 1) Consistency on the reduced dataset. The percentage represents a removed part of the dataset. Removed records are balanced over both classes. The last column represents F1 score for XGB classifier. (b) (Test 2) Sensitivity to mislabels. The percentage represents amount of created mislabels in the whole dataset in a balanced way. The last column represents F1 score for XGB classifier.

	M_1	M_2	M_3	XGB F1		M_1	M_2	M_3	XGB F1
Dataset 0%	0.308	0.437	0.732	0.975	Mis. 0%	0.308	0.437	0.732	0.975
Dataset 10%	0.257	0.436	0.769	0.974	Mis. 10%	0.0	0.397	0.729	0.865
Dataset 30%	0.0	0.447	0.767	0.972	Mis. 20%	0.0	0.324	0.806	0.764
Dataset 50%	0.0	0.443	0.799	0.965	Mis. 30%	0.0	0.132	0.810	0.673

Table 3. (Test 3) Dependency on imbalanced ratio. The percentage represents ratio between negative (non-DoH) and positive (DoH) class. The last column represents F1 score for XGB classifier.

	M_1	M_2	M_3	XGB F1
Imbl. 50/50%	0.492	0.429	0.731	0.982
Imbl. 60/40%	0.446	0.413	0.744	0.982
Imbl. 70/30%	0.140	0.371	0.759	0.982

M_1 score of 30.8% with sensitivity parameter α equal to 1%. When we remove defined portions of samples, we can see a decreasing trend for this metric. For a reduction 30%, we can see M_1 score of 0, and other metrics, including the F1 score, are consistently at the same level with the original dataset since this is the threshold value for M_1 . For a reduction of 50%, we can see the continuous trend for M_3 and F1 score. However, the change is not significant since there is still value in the dataset which is reflected by M_2 .

The second test is based on the same input dataset as in the first test. Table 2b summarizes the impact of mislabels we added to each class in a balanced way. We can see a clear trend in all metrics. Especially, M_2 shows the decrease of relevant association between class labels and data.

In the last test, we increased the dataset size to 20,000 samples so that we have enough samples in each class for different imbalanced ratios, as we know the minimal size from Test 1. Results are included in Table 3. Since the dataset is twice bigger as in previous experiments, we can see an increase of M_1 . The F1 score is consistent over all versions, but the dataset quality is consistently lower for all metrics. This dependency is aligned with work from Brabec et al. [2].

5.2 Case Study 2: Evaluation for Multi-class Classification

In the second case study, we take the CESNET-TLS22 dataset, which contains two types of labels - Category and TLS_SNI. TLS_SNI is a subset of the higher-level Category label. From the original dataset, we created two datasets for this experiment with different label type. We take 5 classes from Category label

Table 4. Summary of results for multi-class classification. The last column represents F1 score for XGB classifier.

	M_1	M_2	M_3	XGB F1
TLS (Category)	0.614	0.558	1.489	0.969
TLS (TLS_SNI)	0.967	0.526	4.093	0.998

Table 5. Summary of comparison between original and fixed version of CICIDS2017 dataset. The last column represents F1 score for XGB classifier.

	M_1	M_2	M_3	XGB F1
CICIDS2017	0.767	0.520	9.956	0.946
CICIDS2017 Fixed	0.783	0.492	6.296	0.975

(Antivirus, Videoconferencing, Streaming media, Analytics & Telemetry, File sharing) and 5 classes with TLS_SNI labels², each with 25,000 records.

The results of the evaluation are shown in Table 4. Even though we have the same dataset used for different use cases, we can see different behavior. Based on the F1 score and M_2 metric, we can see good classification performance and high accuracy of assigned labels. M_3 shows higher classification similarity for TLS_SNI since 4 classes are related to Microsoft and are very similar in comparison with class `api.github.com`. M_1 indicates higher redundancy for TLS_SNI classification.

5.3 Case Study 3: Evaluation of Error and Corrected Dataset

The last case study evaluates recent findings in the CICIDS2017 dataset from Lanvin et al. [10]. The author identifies several errors in terms of duplicated, mis-ordered data and incoherent timestamps. We compare the original and fixed versions of the CICIDS2017 dataset. We analyze Wednesday traffic with all available labels (BENIGN, DoS Hulk, DoS GoldenEye, DoS slowloris, DoS Slowhttptest, Heartbleed) and limit the size of each class to 5000 samples. Results are summarized in Table 5. Even though the fixed version of the dataset contains less amount of samples due to duplicated data, the M_1 and M_2 metrics are almost consistent due to valid corrections of the original CICIDS2017 dataset. M_3 is influenced by the sensitivity of Heartbleed class which has only 11 samples.

As the author briefly said, this dataset has a similar ML classification score. The main reason is due to the wrong collection process, which does not necessarily must cause classification failure. Since our metrics are working with a single dataset, we can confirm findings from Lanvin et al. [10] that datasets are providing similar results. The difference and relationship between two datasets or correct dataset development is a separate challenge that should be investigated.

² login.microsoftonline.com, settings-win.data.microsoft.com, outlook.office365.com, api.github.com, v10.events.data.microsoft.com.

6 Conclusion

In this paper, we have proposed three novel metrics to estimate the quality of datasets that are usually used to train and evaluate ML models used in network security. The introduced metrics are robust, since we combine several ML models with statistical methods. Contrary to other existing published works, our proposed metrics can be used as a universal assessment procedure to evaluate linear and non-linear tasks over datasets for binary or even multi-class classification.

Experiments on three different case studies tested all the metrics and explained their added value, sensitivity, and interpretation. According to the results, we can see high redundancy and reliability of the labels in all tested datasets, which can be considered good for target classification.

After intentional modifications of the datasets for test purposes (mislabels, samples reduction and imbalance), we showed successful identification of such events/corruptions by our metrics. Surprisingly and most importantly, these evident flaws in the modified datasets were not observable by any change in F1 score, which is commonly used in literature to evaluate ML models. We believe this highlights the urgent need to include additional dataset evaluation techniques as a useful practice in scientific research as well as production deployment of ML technologies. Especially, this is essential during the work with datasets, i.e., creation or optimization either manually or with some automation technology such as Active Learning. It is worth repeating that imperfect datasets used for training can lead to poor performance of the machine learning models that are becoming popular for network security area, such as traffic recognition, detection of security threats, or detection of suspicious behavior of devices.

The main focus of this paper was to evaluate a single dataset. However, during our experiments, we identified dataset relationship comparison as a challenge that should be researched in future work.

Acknowledgement. This research was funded by the Ministry of the Interior of the Czech Republic, grant No. VJ02010024: Flow-Based Encrypted Traffic Analysis, and by the MEYS of the Czech Republic, grant No. LM2023054: e-Infrastructure, and by internal grant of CTU in Prague, grant No. SGS23/207/OHK3/3T/18.

References

1. Anderson, B., McGrew, D.: Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In: 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2017)
2. Brabec, J., et al.: On model evaluation under non-constant class imbalance. In: Computational Science (ICCS) (2020)
3. Celdrán, A.H., et al.: RITUAL: a platform quantifying the trustworthiness of supervised machine learning. In: 18th International Conference on Network and Service Management (CNSM) (2022)
4. Chen, H., et al.: Data curation and quality assurance for machine learning-based cyber intrusion detection (2021)

5. Zelaya, C.V.G.: Towards explaining the effects of data preprocessing on machine learning. In: 35th International Conference on Data Engineering (2019)
6. Hwang, I., et al.: SimEX: express prediction of inter-dataset similarity by a fleet of autoencoders. arXiv preprint [arXiv:2001.04893](https://arxiv.org/abs/2001.04893) (2020)
7. Jeřábek, K., Hynek, K., Čejka, T., Ryšavý, O.: Collection of datasets with DNS over https traffic. *Data Brief* **42**, 108310 (2022)
8. Koh, P.W., et al.: WILDS: a benchmark of in-the-wild distribution shifts. In: Proceedings of the 38th International Conference on Machine Learning (2021)
9. Komorniczak, J., Ksieniewicz, P.: Proplexity - an open-source python library for binary classification problem complexity assessment. arXiv preprint [arXiv:2207.06709](https://arxiv.org/abs/2207.06709) (2022)
10. Lanvin, M., et al.: Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes (2023). <https://hal.science/hal-03775466>
11. Lee, Y.W., et al.: AIMQ: a methodology for information quality assessment. *Inf. Manag.* (2002)
12. Lorena, A.C., Garcia, L.P.F., Lehmann, J., Souto, M.C.P., Ho, T.K.: How complex is your classification problem? A survey on measuring classification complexity. *ACM Comput. Surv.* **52**(5) (2019)
13. Luxemburk, J., Čejka, T.: Fine-grained TLS services classification with reject option. *Comput. Netw.* **220**, 109467 (2023)
14. Maillo, J., Triguero, I., Herrera, F.: Redundancy and complexity metrics for big data classification: towards smart data. *IEEE Access* **8**, 87918–87928 (2020)
15. Obaid, H.S., et al.: The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning. In: 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON) (2019)
16. Papadogiannaki, E., Ioannidis, S.: A survey on encrypted network traffic analysis applications, techniques, and countermeasures. *ACM Comput. Surv.* **54**(6) (2021)
17. Pendlebury, F., et al.: TESSERACT: eliminating experimental bias in malware classification across space and time. In: Proceedings of the 28th USENIX Conference on Security Symposium, USA (2019)
18. Pesarin, F., Salmaso, L.: A review and some new results on permutation testing for multivariate problems. *Stat. Comput.* **22**(2), 639–646 (2012)
19. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: International Conference on Information Systems Security and Privacy (2018)
20. Soukup, D., et al.: Towards evaluating quality of datasets for network traffic domain. In: 17th International Conference on Network and Service Management (CNSM) (2021)
21. Wasielewska, K., et al.: Dataset quality assessment with permutation testing showcased on network traffic datasets (2022). <http://dx.doi.org/10.36227/techrxiv.20145539.v1>
22. Yoon, J., Arik, S., Pfister, T.: Data valuation using reinforcement learning. In: Daumé, H., III., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 10842–10851. PMLR (2020)
23. Zhang, Y., Zhao, S., Sang, Y.: Towards unknown traffic identification using deep auto-encoder and constrained clustering. In: Computational Science – ICCS (2019)



Factors of Intention to Use a Photo Tool: Comparison Between Privacy-Enhancing and Non-privacy-enhancing Tools

Vanessa Bracamonte^{1(✉)}, Sebastian Pape², and Sascha Löbner²

¹ KDDI Research, Inc., Saitama, Japan
va-bracamonte@kddi-research.jp

² Goethe University Frankfurt, Frankfurt, Germany
{sebastian.pape, sascha.loebner}@m-chair.de

Abstract. Tools that detect and transform privacy sensitive information in user content have been proposed to enhance privacy in contexts such as social media. However, previous research has found that privacy-related concerns can be higher in these types of tools compared to similar non-privacy tools. In this paper, we focus on adoption of these tools and investigate how the knowledge that a data-processing tool has a privacy purpose affects privacy-related factors of intention to use such a tool, when compared with a similar tool with a non-privacy-related purpose. We conducted a user study where we described a privacy-enhancing and a non-privacy-enhancing photo manipulation app to two groups of participants. The results show that general and context-specific privacy-related perception has different effects for the two types of apps. In particular, although participants perceived the same level of privacy risk towards both types of apps, this risk only had a significant negative effect on intention to use in the case of the privacy-enhancing app. Furthermore, disposition to value privacy increased both perceived risk and intention to use the privacy-enhancing app. We discuss these findings in the context of the diffusion of privacy-enhancing tools for user content.

Keywords: Privacy-enhancing tools · Privacy · Risk · User perception

1 Introduction

Users increasingly reveal great amounts of personal information, related to themselves or others, on social media. The consequences of sharing this information can be negative and result in regret from users [28, 32]. Automated analysis of images has been proposed as a way of protecting peoples' privacy in a social media context [20]. In general terms, these proposals work by analyzing the content of peoples photos to detect whether the content reveals private or sensitive information and potentially transforming that content to anonymize it [12, 21]. However, research on perception towards privacy tools has identified that users have privacy-related concerns towards these types of tools. In the evaluation of

third-party tracking blockers, Schaub et al. [27] reported that some participants distrusted the tools because of the perception that the tools themselves would collect their personal data, even though the trackers could do the same. In a study on privacy add-ons, Corner et al. [3] also found that some users distrusted the tools because they thought the tool itself would be used to access their data. Although people understand and agree with the beneficial purposes of a privacy-enhancing tool that analyses their data, they also worry about surveillance and having their privacy intruded upon by these tools [1,2]. In principle, there is not much to distinguish a privacy-enhancing tool and a non-privacy enhancing tool besides the purpose of protecting privacy. For both types of tools, users would have to provide their photos in order to receive the service. However, previous research has found that for privacy-enhancing tools that process user content, the level of privacy concern can be higher than for similar tools with a non-privacy related purpose [1].

Although there is evidence of a different level of privacy-related concern towards privacy-enhancing tools, research has not examined whether this difference also applies to the mechanism of intention to adopt such a tool. The objective of this study is to investigate if the perception towards privacy-enhancing tools might be different from tools that process data for a non-privacy-related purpose. We conducted a user experiment to examine how the knowledge of the privacy-enhancing purpose of the tool influenced the effect of privacy-related factors on intention to use the tool, compared to a non-privacy-enhancing tool. The results indicate that there is a difference in the relationship between factors for these two types of tools. In particular, risk perception negatively influences the intention to use the privacy-enhancing app, but does not significantly affect the non-privacy-enhancing app. In addition, for the privacy-enhancing app, disposition to value privacy had a significant positive influence on intention to use the app, but for the non-privacy-enhancing app, there is no significant effect. The contribution of this research is a clarification of the mechanism through which privacy-related factors have contrary effects on intention to use a privacy-enhancing tool.

2 Methodology

In this section we describe the methodology used for the study, including the research questions, experiment design and ethical considerations.

2.1 Research Objectives

The study focuses on examining any differences in user perception towards a privacy-enhancing and non-privacy-enhancing tool, in light of the fact that both types of apps have the same potential for privacy risk. In order to do so, we use a privacy-focused model of intention to use a technology, adapted from [6,23]. The research model is presented in Fig. 1. The model establishes that context-specific privacy-related constructs (Perceived risk, Perceived benefit and Trust) influence Intention to use the tool. In addition, it also establishes

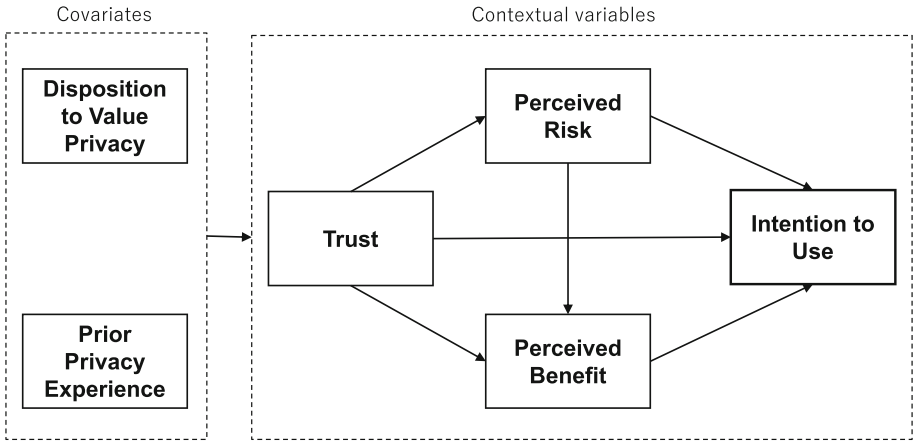


Fig. 1. Research model.

that privacy-related dispositions and experience (Disposition to value privacy and Prior privacy experience) influence all context-specific constructs. The relationships between the constructs in the model have been proposed and validated in previous research [5,6,23]. As mentioned, the focus in this study is in the differences that may arise due to the type of tool priming. More specifically, we seek to answer the following questions:

- Are there differences in the relationships between privacy-related factors and intention to use for privacy-enhancing and non-privacy-enhancing tools?
- Are there differences in the relationships between general privacy attitudes and experiences, and privacy-related factors of intention to use, for privacy-enhancing and non-privacy-enhancing tools?

2.2 Experiment Design and Task

In order to answer the research questions, we designed an experiment which consisted of a task for participants to read and give their opinion about an hypothetical app that would be used to transform photos for uploading on social media. We manipulated the purpose of app: privacy-enhancing vs. non-privacy-enhancing. The objective of this study was to evaluate differences in perception that resulted from the manipulation (priming), therefore, the privacy-enhancing app was explicitly described as such. Participants viewed the description and a mockup of only one type of app (between-subjects design). After reading about the app, the participants answered a questionnaire.

We described to participants an hypothetical free, third-party app for social media photos. For the privacy-enhancing app, the purpose was described as protecting privacy; for the non-privacy-enhancing app, the purpose was described as enhancing the content for fun. The app would hypothetically work by

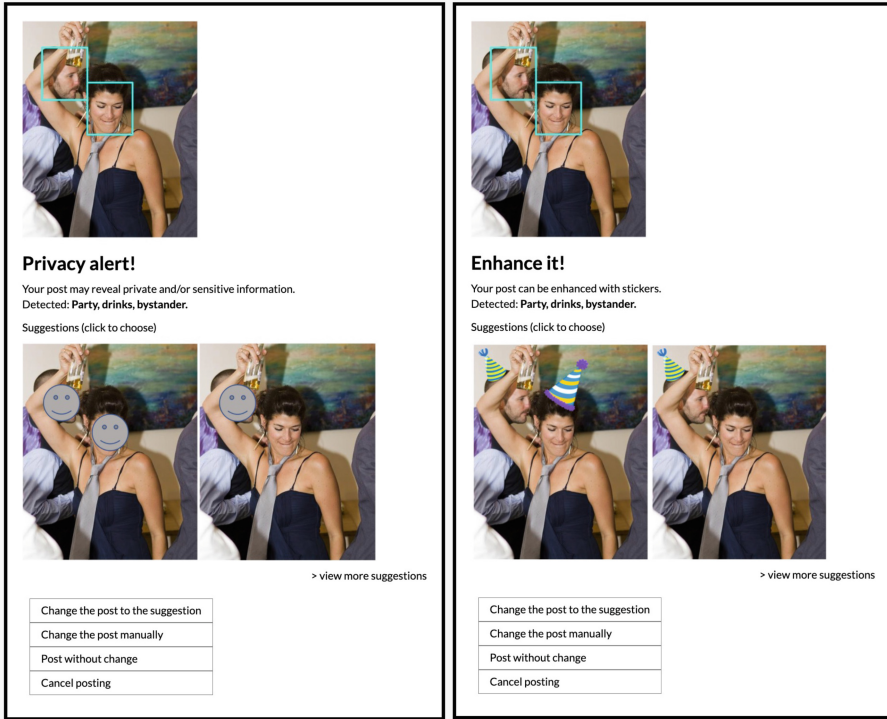


Fig. 2. Experiment app mockups. Left: Privacy-enhancing app. Right: Non-privacy-enhancing app.

analyzing and detecting the content in the users’ photos. We described the type of information the app would detect from the photos: private information for the privacy-enhancing app, and information that could be enhanced with stickers for the non-privacy-enhancing app. We then presented a non-interactive mockup of the app which showed how it would work. The mockups for each group had the same general design, and only differed in their message and the transformation performed on the photo (privacy-enhancing vs non-privacy enhancing) The detail of the app mockup is shown in Fig. 2. After the mockup, we showed five additional photo examples to the participants. Photos were sourced from the COCO dataset [22].

We measured the constructs of interest with scales adapted from previous research: Intention to use [24], Perceived benefit [6], Perceived risk [23], Trust [15], Disposition to value privacy [33] and Prior privacy experience [29]. The responses used a 7-point scale, ranging from *Strongly disagree* to *Strongly agree*, except for Prior privacy experience which was measured on a 7-point scale, ranging from *Never* to *Very frequently*. The detail of the measurement items is shown on Table 1. The questionnaire also included age, gender (as an open text box [30]), frequency of social media posting and attention check questions.

We validated the questionnaire with pretests conducted with Amazon Mechanical Turk workers. The pretest workers were compensated with US\$1.7 and we rewarded an additional US\$1 to participants who provided detailed feedback. In the pretests, we also validated that it was clear to the participants what was the purpose of the app (privacy-related or non-privacy-related).

Table 1. Measurement items

Intention to use	Given the chance, I intend to use this app Given the chance, I predict that I would use this app in the future It is likely that I would use with this app in the future
Perceived risk	In general, it would be risky to give my photos to this app In general, it would be risky to give personal information to this app There would be high potential for privacy loss associated with giving personal information to this app There would be too much uncertainty associated with giving personal information to this app Providing this app with personal information would involve many unexpected problems (reverse scale)
Perceived benefit	Revealing my personal information on this app will help me obtain the result I want I need to provide my personal information so I can get exactly what I want from this app I believe that as a result of my personal information disclosure, I will benefit from a better, customized result
Trust	I would feel safe giving personal information to this app This app would tell the truth and fulfill promises related to the information provided by me I trust that this app would keep my best interests in mind when dealing with my personal information
Disposition to value privacy	Compared to others, I am more sensitive about the way my personal information is handled Keeping my information private is the most important thing to me Compared to others, I tend to be more concerned about threats to my information privacy
Prior privacy experience	How often have you personally experienced incidents whereby your personal information was used by some company or e-commerce web site without your authorization? How much have you heard or read during the last year about the use and potential misuse of the information collected from the Internet? How often have you personally been the victim of what you felt was an improper invasion of privacy?

2.3 Participant Recruitment and Ethical Considerations

We recruited participants on Amazon Mechanical Turk with the following qualifications: workers from the USA, who had at least a 99% acceptance rate for their tasks, and who had worked on at least 5000 tasks. We set the participant reward at US\$2.5 (US\$11.5/hour rate for a 11 min survey). We obtained 400 responses in total and we identified 20 responses which were duplicated submissions or had answered the attention questions with unrelated content. These responses were rejected and the rest of participants (380) were rewarded.

This study was exempt from review according to our institution's criteria for research of this type. Nevertheless, we provided a notice to inform potential participants about the characteristics of the study. The notice included a description of the purpose of the survey, the approximate time to finish it and the task participants were expected to do (read a description and answer questions). The notice also explained that the survey included attention questions, but that we would not reject the participants answers based only on these questions. However, we clarified that we would reject duplicated answers or answers unrelated to the question asked. We indicated that the survey was completely voluntary and that participants were free to decline to participate, that we would not collect identifying information such as name, email or IP address, and that the results would be used for academic purposes only. We also indicated that the survey was limited to adults who lived in the United States. Finally we provided the principal researchers' name and email address in case of any questions about the study. Participants were asked to access the link to the survey itself if they accepted to participate.

3 Results

In this section, we describe the sample obtained and the results of the data analysis.

3.1 Sample

We first identified 22 multivariate outliers using the Mahalanobis distance ($\alpha = 0.001$). We removed these cases from analysis, resulting in a sample size of 358 participants (exactly 179 in each group). This sample size is over the minimum sample required for finding path coefficients of 0.11 - 0.2, with a significance level of 5% and a power of 80% [7], based on the inverse square root method for minimum sample size estimation [18]. The age mean was 41 for both groups, with a median of 37 years-old in the Privacy app group and 38 years-old in the Non-privacy app group. The gender distribution was 93 (52%) female/86 (48%) male participants in the Privacy app group and 101 (56%) female/78 (44%) male participants in the Non-privacy app group.

We compared the characteristics of participants between groups using non-parametrical tests. Mann-Whitney U tests indicated that there were no significant differences in age ($p = 0.94$), gender ($p = 0.4$) or frequency of social media

posting ($p = 0.11$) between groups. The sample results indicate that the participant groups are comparable.

3.2 Group Comparison

We used a partial least squares structural equation modeling (PLS-SEM) method to evaluate the hypotheses of the study. This method accounts for interrelationships between the constructs of interest and for data which may not be normally distributed [8]. We conducted the PLS-SEM analyses using Smart-PLS (v.3.3.9) [25]. The focus of this study is to investigate differences between the experiment groups. In other words, we evaluate if participant perception is influenced by the privacy purpose priming. In order to do this, we conducted a PLS-SEM analysis, in particular, a multigroup analysis (PLS-MGA [13]).

First, we evaluated the reliability and validity of the measurement model. We examined indicator reliability by inspecting the items loading on their respective constructs, that is, the correlation weights between the construct and its indicators (measurement items). All loadings had a value over the threshold of 0.708 [8], ranging from 0.821 to 0.992. To evaluate internal consistency reliability, which is the association between indicators of the same construct, we examined the rhoA reliability coefficient [4]. For all constructs, rhoA values were higher than the satisfactory minimum of 0.7 [7], ranging from 0.836 to 0.99. The rhoA values were higher than the ideal upper limit of 0.9, but this was likely due to the use of established scales from previous research. Convergent validity, which is how much the construct converges to explain indicator variance, was examined using the average variance extracted (AVE). The AVE for all constructs had a value above the minimum level of 0.5 [8], ranging from 0.74 to 0.981. We examined discriminant validity, which is how much a construct is distinct from other constructs, using the heterotrait-monotrait (HTMT) ratio of correlations criterion. As required, all values were significantly lower than the threshold value of 0.9 [7], ranging from 0.04 to 0.775.

Before the multigroup comparison analysis, we conducted a measurement invariance of composite models (MICOM) procedure [14] to validate that the models can be compared. The procedure consists of three steps to test for configural invariance, compositional invariance, and composite mean values and variances equality. The first two steps are necessary to establish partial measure invariance, which is required to be able to meaningfully compare the structural model between groups. The first configural invariance step requires that the data has the same handling, that all constructs have the same measurement items, and the same estimation settings are used across groups. We ensured that these requirements were met for our analysis. The second compositional invariance step requires that the constructs are formed equally in the groups. We conducted a permutation test to evaluate compositional invariance. The results show that no correlations were significantly different ($p > 0.05$) and all fall within the 95% confidence interval, which indicates compositional invariance of the models for all constructs. We then tested the composite mean values and variances equality. The results show that Intention to use (difference in mean = 0.229, $p = 0.033$)

and Perceived benefit (difference in mean = 0.332, $p=0.002$) were significantly different between groups in terms of composite mean value. All other constructs had equal mean values and variances. The positive difference in mean indicates that Intention to use and Perceived benefit were significantly higher for the privacy app. Although this is not the focus of the analysis, we interpret this result to simply reflect a difference in the benefit of the hypothetical apps, of enhancing privacy in comparison to enhancing enjoyment.

The results of the MICOM procedure indicate that there was partial measurement invariance, due to the significant differences in the mean of Intention to use and Perceived benefit. Therefore we also examined the difference in structural models in terms of the standardized path coefficients between groups. We first validated the quality of the structural models, by examining collinearity issues and the coefficient of determination (R-squared). Collinearity (too high correlation) issues in the structural models was examined by calculating the variance inflation factor (VIF) values for the constructs. All VIF values were below 5, that is, below threshold for critical collinearity [8]. The values ranged from 1.111 to 3.373. R-squared values measure the variance in a construct explained by the predictors; values of 0.25 are considered weak [8]. The values ranged from 0.1 to 0.704. Trust was the only construct with a value lower than 0.25 (0.1 in the Privacy app group and 0.163 in the Non-privacy group), but this was due the covariates being its only predictors in the model.

We then examined the standardized path coefficients for each group. A bootstrapping procedure with 10,000 samples [26] was used to calculate the path significance. Statistical significance criteria for the path coefficients is determined by the bootstrapped standardized t statistic [7]: >3.291 , significant at 0.1% ($\alpha = 0.001$) probability of error; >2.576 , significant at 1% ($\alpha = 0.01$), >1.96 , significant at 5% ($\alpha = 0.05$) (two-tailed). The results of the analysis are shown in Table 2 for the privacy-enhancing app group and Table 3 for the non-privacy-enhancing app group.

The results for the privacy-enhancing app group show that only two relationships were not statistically significant: Trust did not have an effect on Intention to use and Disposition to value privacy did not have an effect on Perceived benefit of the app. In addition, Disposition to value privacy increased both Perceived risk and Intention to use the privacy-enhancing app, and it reduced Trust in the app. The same was true for Prior privacy experience, which in addition also significantly increased Perceived benefit of the privacy-enhancing app. On the other hand, for the non-privacy-enhancing app group the results show that Perceived risk did not have a significant effect on Intention to use the app. In addition, the covariates had a reduced influence. Disposition to value privacy and Prior privacy experience only significantly influenced Perceived risk and Trust. The result models are shown in Fig. 3. With regards to indirect effects, Perceived risk and benefit both significantly mediated the effect of Trust on Intention to use the privacy-enhancing app. In addition, Trust mediated a negative effect of Disposition to value privacy on Perceived benefit. For the non-privacy-enhancing app, Trust mediated the effect of the covariates on Perceived benefit and Intention to use.

Table 2. Path coefficients - Privacy-enhancing app group.

	Original Sample	Sample Mean	Std. Dev.	95% CI	T Statist.	p-value
Risk → Intention	-0.282	-0.279	0.114	[-0.497, -0.047]	2.460	0.014
Benefit → Intention	0.397	0.394	0.098	[0.194, 0.579]	4.042	0.000
Benefit → Risk	-0.174	-0.174	0.058	[-0.293, -0.062]	2.985	0.003
Trust → Intention	0.176	0.181	0.119	[-0.047, 0.421]	1.472	0.141
Trust → Benefit	0.704	0.704	0.047	[0.600, 0.787]	14.967	0.000
Trust → Risk	-0.494	-0.495	0.075	[-0.635, -0.341]	6.594	0.000
DispPriv → Intention	0.179	0.179	0.063	[0.055, 0.304]	2.817	0.005
DispPriv → Benefit	-0.077	-0.078	0.061	[-0.195, 0.040]	1.273	0.203
DispPriv → Risk	0.238	0.237	0.049	[0.145, 0.339]	4.880	0.000
DispPriv → Trust	-0.160	-0.158	0.078	[-0.311, -0.006]	2.041	0.041
ExpPriv → Intention	0.124	0.127	0.062	[0.005, 0.250]	1.998	0.046
ExpPriv → Benefit	0.153	0.153	0.067	[0.025, 0.289]	2.294	0.022
ExpPriv → Risk	0.222	0.222	0.061	[0.105, 0.343]	3.658	0.000
ExpPriv → Trust	-0.207	-0.212	0.092	[-0.380, -0.020]	2.250	0.025

Table 3. Path coefficients - Non-privacy-enhancing app group.

	Original Sample	Sample Mean	Std. Dev.	95% CI	T Statist.	p-value
Risk → Intention	0.021	0.021	0.127	[-0.239, 0.254]	0.170	0.865
Benefit → Intention	0.360	0.362	0.089	[0.184, 0.530]	4.054	0.000
Benefit → Risk	-0.144	-0.144	0.062	[-0.265, -0.019]	2.317	0.021
Trust → Intention	0.365	0.364	0.119	[0.122, 0.590]	3.078	0.002
Trust → Benefit	0.665	0.663	0.054	[0.549, 0.763]	12.242	0.000
Trust → Risk	-0.512	-0.511	0.066	[-0.635, -0.374]	7.719	0.000
DispPriv → Intention	-0.026	-0.025	0.083	[-0.183, 0.142]	0.310	0.757
DispPriv → Benefit	-0.123	-0.125	0.075	[-0.268, 0.022]	1.645	0.100
DispPriv → Risk	0.316	0.317	0.060	[0.204, 0.438]	5.294	0.000
DispPriv → Trust	-0.274	-0.272	0.082	[-0.425, -0.105]	3.333	0.001
ExpPriv → Intention	0.049	0.048	0.067	[-0.081, 0.181]	0.724	0.469
ExpPriv → Benefit	0.098	0.097	0.071	[-0.039, 0.235]	1.390	0.164
ExpPriv → Risk	0.103	0.103	0.052	[0.002, 0.207]	1.983	0.047
ExpPriv → Trust	-0.203	-0.208	0.080	[-0.348, -0.032]	2.554	0.011

Finally, we conducted the multigroup analysis procedure (PLS-MGA), which can test moderation by the group variable across the model, to evaluate the differences in the strength of the variables’ relationships between experiment groups. The results show that the relationship between Disposition to value privacy and Intention to use (difference in path coefficient = -0.303, p=0.08), and Perceived risk and Intention of use (difference in path coefficient = 0.204,

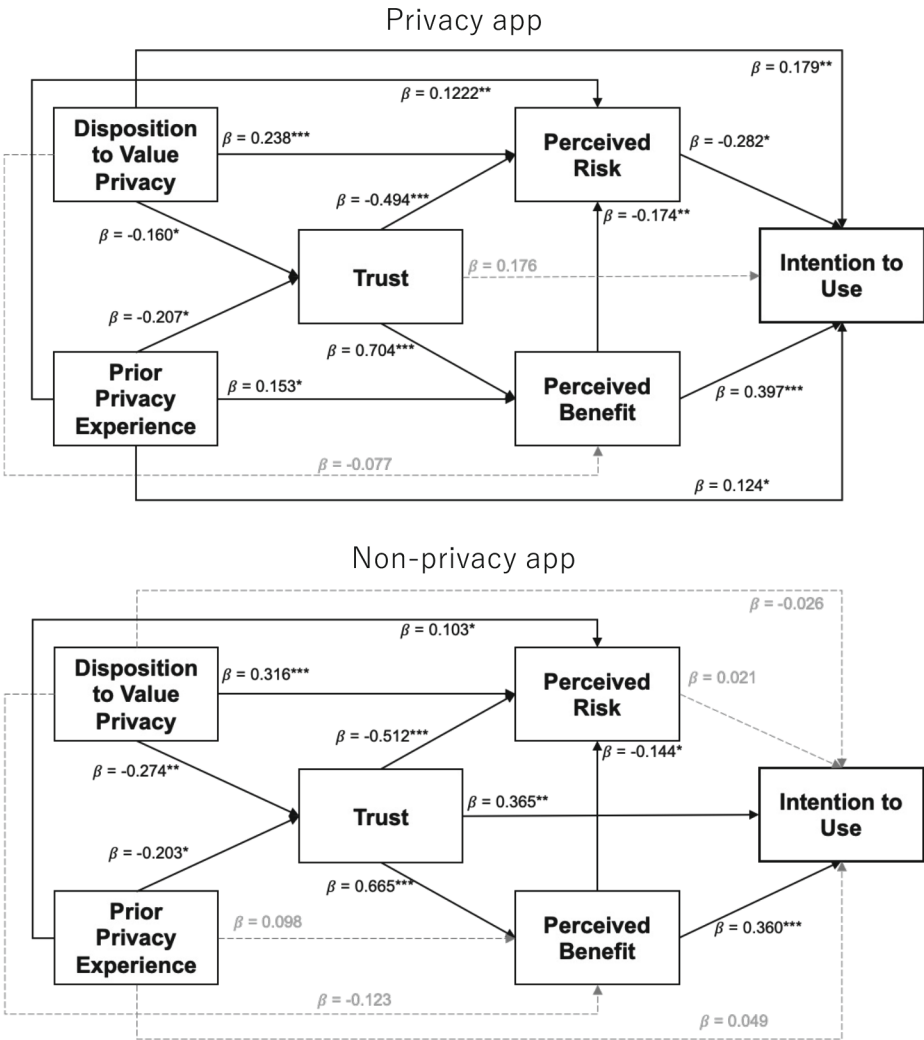


Fig. 3. Result models. Top: Privacy-enhancing app group. Bottom: Non-privacy-enhancing app group.

$p=0.051$) had the largest differences in path coefficient strength with $p < 0.1$. However, the differences were not significant at $p < 0.05$. This may be result of insufficient power for PLS-MGA: although the sample size per group is adequate for the separate analyses, the sample size is lower than the recommended sample size per group of 200 for multigroup analysis [17].

4 Discussion

As already known from previous research, one of the main factors of the intention to use an app are the perceived benefits (named performance expectancy in UTAUT2) [31]. Thus, it is not surprising to identify that also in our experiment. If the tool is not perceived as beneficial, one wouldn't expect the (potential) users to use it. However, for those users who have an interest in using such a tool, there will be other factors contributing to and obstacles reducing the intention to use.

In our model, those factors are the perceived risk of the app and the trust in the app. Although participants perceive the same level of privacy-related risk towards both apps, an increase of perceived risk is not associated with a decrease of intention to use the non-privacy-enhancing app. In contrast, for the privacy-enhancing app, higher perceived privacy risk does have a significant negative influence on intention to use. In other words, the findings suggest that for the non-privacy-enhancing app, participants opinions follow along the privacy paradox, which states that privacy issues are considered important to users but that it does not affect their subsequent choices. Instead, other considerations, such as benefit, are more important to their actual behavior [19]. The results show that this does not happen for the privacy-enhancing tool. Considering that the hypothetical apps work in the same way and are almost identical except for their stated purpose, we can say this is a result of the experiment priming since the participants' are aware of the privacy-related purpose of the app. Regarding trust in the app, interestingly there is no significant direct influence from trust to intention to use for the privacy-enhancing app. While previous findings showed a direct effect [9–11], it seems that in our experiment the effect of trust on intention to use for the privacy-enhancing app was to a large degree mediated by perceived risk and perceived benefits. Similarly to perceived risks, disposition to value privacy increases both intention to use and perceived risk towards the privacy-enhancing app. This is a logical relationship, but the findings suggest a challenge for the adoption of this type of privacy-enhancing apps, which rely on user data processing, by people who are concerned about their privacy.

The result of our experiment shows that privacy-enhancing tools seem to prime users simply by stating their purpose, even though they would not fundamentally work differently than apps for other purposes. Naturally, it is difficult for privacy-enhancing tools to avoid priming users since tools need to have a proper name, often including privacy-related terms, to be found and of course described properly to let the (potential) users know what the tools are good for. If privacy terms were avoided, then those who are more disposed to think of privacy as important could fail to find these tools. On the other hand, as the findings show, the disposition to think that privacy as important has contrary effects: these users might want to use such tools to protect their privacy but at the same time feel increased privacy risk regarding the tool. Summing up, providers offering a privacy enhancing-tool should emphasize the privacy aspects. However, they should not only focus on explaining the benefits of their tool, but also trying to explain how their tool addresses potential risks of abusing user data to build trust. We encourage further research on how emphasizing

these benefits and the risk mitigation, which are likely to be privacy-related, have additional impact on users' privacy risk perception.

4.1 Limitations

This study has a number of limitations. First, we are considering a parsimonious research model, and it is possible that other constructs may also affect intention to use, and that those relationships could be affected by the type of app. Second, we relied on the responses from Amazon Mechanical Turk workers. Research has shown that these workers have a higher sensitivity to privacy issues [16], and we also limited participation to experienced workers with a minimum of 5k tasks. Therefore, the results might not be generalizable to other populations. Third, we used a non-interactive app mockup for the experiment. This decreases the realism of the situation for participants, who are not risking their private information. Future research should consider validating these results in a more realistic situation, i.e. by investigating user perception of a real privacy-enhancing app.

5 Conclusions

In this study, we investigated whether user perception of a privacy-enhancing photo app is different from perception of a similar app that does not have a privacy purpose. The results show that there are differences in the relationships between privacy-related factors and intention to use the two types of app. Specifically, perceived risk does not have a significant influence on intention to use the non-privacy-enhancing app, which is congruent with the privacy paradox. However, for the privacy-enhancing app, perceived risk significantly negatively influences intention. In addition, although participants' disposition to value privacy positively influences how much risk they perceive towards the two types of apps, for the privacy-enhancing app it also has a positive influence on intention. That is, that the same disposition has a contrary effect of participants wanting to use the privacy-enhancing app, which would protect their privacy, and increasing how much privacy risk they feel from potentially using the app. In future research, we plan to experimentally investigate if the manipulation of the levels of these constructs might alter the balance between them in a way that results in increased or decreased intention to use a privacy-enhancing tool.

References

1. Bracamonte, V., Pape, S., Loebner, S.: "All apps do this": comparing privacy concerns towards privacy tools and non-privacy tools for social media content. *Proc. Priv. Enhancing Technol.* **2022**(3) (2022)
2. Bracamonte, V., Tesfay, W.B., Kiyomoto, S.: Towards exploring user perception of a privacy sensitive information detection tool. In: 7th International Conference on Information Systems Security and Privacy (2021)

3. Corner, M., Dogan, H., Mylonas, A., Djabri, F.: A usability evaluation of privacy add-ons for web browsers. In: Marcus, A., Wang, W. (eds.) HCII 2019. LNCS, vol. 11586, pp. 442–458. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23535-2_33
4. Dijkstra, T.K., Henseler, J.: Consistent partial least squares path modeling. *MIS Q.* **39**(2), 297–316 (2015)
5. Dinev, T., McConnell, A.R., Smith, H.J.: Research commentary—informing privacy research through information systems, psychology, and behavioral economics: thinking outside the “APCO” box. *Inf. Syst. Res.* **26**(4), 639–655 (2015). <https://doi.org/10.1287/isre.2015.0600>
6. Dinev, T., Xu, H., Smith, J.H., Hart, P.: Information privacy and correlates: an empirical attempt to bridge and distinguish privacy-related concepts. *Eur. J. Inf. Syst.* **22**(3), 295–316 (2013). <https://doi.org/10.1057/ejis.2012.23>
7. Hair, J., Hult, G.T.M., Ringle, C., Sarstedt, M., Danks, N., Ray, S.: *Partial Least Squares Structural Equation Modeling (PLS-SEM) Using R: A Workbook* (2021)
8. Hair, J.F., Risher, J.J., Sarstedt, M., Ringle, C.M.: When to use and how to report the results of PLS-SEM. *Eur. Bus. Rev.* (2019)
9. Harborth, D., Pape, S.: How privacy concerns and trust and risk beliefs influence users’ intentions to use privacy-enhancing technologies – the case of tor. In: 52nd Hawaii International Conference on System Sciences (HICSS) 2019, pp. 4851–4860 (2019). <https://scholarspace.manoa.hawaii.edu/handle/10125/59923>
10. Harborth, D., Pape, S.: How privacy concerns, trust and risk beliefs and privacy literacy influence users’ intentions to use privacy-enhancing technologies - the case of tor. *ACM SIGMIS Database: DATABASE Adv. Inf. Syst.* **51**(1), 51–69 (2020). <https://doi.org/10.1145/3380799.3380805>. <https://dl.acm.org/doi/abs/10.1145/3380799.3380805>
11. Harborth, D., Pape, S., Rannenber, K.: Explaining the technology use behavior of privacy-enhancing technologies: the case of tor and Jonym. *Proc. Priv. Enhancing Technol. (PoPETs)* **2020**(2), 111–128 (2020). <https://doi.org/10.2478/popets-2020-0020>. <https://content.sciendo.com/view/journals/popets/2020/2/article-p111.xml>
12. Hasan, R., Crandall, D., Fritz, M., Kapadia, A.: Automatically detecting bystanders in photos to reduce privacy risks. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 318–335 (2020). <https://doi.org/10.1109/SP40000.2020.00097>
13. Henseler, J.: PLS-MGA: a non-parametric approach to partial least squares-based multi-group analysis. In: Gaul, W., Geyer-Schulz, A., Schmidt-Thieme, L., Kunze, J. (eds.) *Challenges at the Interface of Data Analysis, Computer Science, and Optimization*. STUDIES CLASS, pp. 495–501. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24466-7_50
14. Henseler, J., Ringle, C.M., Sarstedt, M.: Testing measurement invariance of composites using partial least squares. *Int. Mark. Rev.* (2016)
15. Jarvenpaa, S.L., Tractinsky, N., Saarinen, L.: Consumer trust in an internet store: a cross-cultural validation. *J. Comput.-Mediated Commun.* **5**(2), JCMC526 (1999). <https://doi.org/10.1111/j.1083-6101.1999.tb00337.x>
16. Kang, R., Brown, S., Dabbish, L., Kiesler, S.: Privacy attitudes of mechanical Turk workers and the U.S. public. In: 10th Symposium on Usable Privacy and Security (SOUPS 2014), pp. 37–49 (2014)
17. Klesel, M., Schuberth, F., Henseler, J., Niehaves, B.: A test for multigroup comparison using partial least squares path modeling. *Internet Res.* **29**(3), 464–477 (2019). <https://doi.org/10.1108/IntR-11-2017-0418>

18. Kock, N., Hadaya, P.: Minimum sample size estimation in PLS-SEM: the inverse square root and gamma-exponential methods. *Inf. Syst. J.* **28**(1), 227–261 (2018)
19. Kokolakis, S.: Privacy attitudes and privacy behaviour: a review of current research on the privacy paradox phenomenon. *Comput. Secur.* **64**, 122–134 (2017)
20. Korayem, M., Templeman, R., Chen, D., Crandall, D., Kapadia, A.: Enhancing lifelogging privacy by detecting screens. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, CHI 2016, pp. 4309–4314. Association for Computing Machinery (2016). <https://doi.org/10.1145/2858036.2858417>
21. Li, Y., Vishwamitra, N., Knijnenburg, B.P., Hu, H., Caine, K.: Effectiveness and users' experience of obfuscation as a privacy-enhancing technology for sharing photos. *Proc. ACM Hum.-Comput. Interact.* **1**(CSCW), 67:1–67:24 (2017). <https://doi.org/10.1145/3134702>
22. Lin, T.Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
23. Malhotra, N.K., Kim, S.S., Agarwal, J.: Internet users' information privacy concerns (IUIPC): the construct, the scale, and a causal model. *Inf. Syst. Res.* **15**(4), 336–355 (2004). <https://doi.org/10.1287/isre.1040.0032>
24. Pavlou, P.A.: Consumer acceptance of electronic commerce: integrating trust and risk with the technology acceptance model. *Int. J. Electron. Commer.* **7**(3), 101–134 (2003)
25. Ringle, C.M., Wende, S., Becker, J.M.: SmartPLS 3 (2015)
26. Sarstedt, M., Hair, J.F., Ringle, C.M., Thiele, K.O., Gudergan, S.P.: Estimation issues with PLS and CBSEM: where the bias lies! *J. Bus. Res.* **69**(10), 3998–4010 (2016)
27. Schaub, F., et al.: Watching them watching me: browser extensions impact on user privacy awareness and concern. In: Proceedings 2016 Workshop on Usable Security. Internet Society (2016). <https://doi.org/10.14722/usec.2016.23017>
28. Sleeper, M., et al.: “I read my Twitter the next morning and was astonished”: a conversational perspective on Twitter regrets. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2013, pp. 3277–3286. Association for Computing Machinery (2013). <https://doi.org/10.1145/2470654.2466448>
29. Smith, H.J., Milberg, S.J., Burke, S.J.: Information privacy: measuring individuals' concerns about organizational practices. *MIS Q.* **20**(2), 167–196 (1996). <https://doi.org/10.2307/249477>
30. Spiel, K., Haimson, O.L., Lottridge, D.: How to do better with gender on surveys: a guide for HCI researchers. *Interactions* **26**(4), 62–65 (2019). <https://doi.org/10.1145/3338283>
31. Venkatesh, V., Thong, J.Y., Xu, X.: Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology. *MIS Q.* 157–178 (2012)
32. Wang, Y., Norcie, G., Komanduri, S., Acquisti, A., Leon, P.G., Cranor, L.F.: “I regretted the minute I pressed share”: a qualitative study of regrets on Facebook. In: Proceedings of the Seventh Symposium on Usable Privacy and Security, SOUPS 2011, pp. 1–16. Association for Computing Machinery (2011). <https://doi.org/10.1145/2078827.2078841>
33. Xu, H., Dinev, T., Smith, J., Hart, P.: Information privacy concerns: linking individual perceptions with institutional privacy assurances. *J. Assoc. Inf. Syst.* **12**(12) (2011). <https://doi.org/10.17705/1jais.00281>



Real-Time Platform Identification of VPN Video Streaming Based on Side-Channel Attack

Anting Lu¹, Hua Wu^{1,2}(✉), Hao Luo¹, Guang Cheng^{1,2}, and Xiaoyan Hu^{1,2}

¹ School of Cyber Science and Engineering, Southeast University, Nanjing, China
{atlu,hwu,hluo,chengguang,xyhu}@seu.edu.cn

² Jiangsu Province Engineering Research Center of Security for Ubiquitous Network, Nanjing, China

Abstract. The video platforms that users watch leak the privacy of their preferences. More and more video streaming is being encrypted to protect users' privacy. In addition, many users use VPN to enhance their privacy protection further. VPN makes video platform identification challenging because it poses traffic obfuscation and further data encryption. Although the segment-based transmission mechanism and Variable Bit-Rate encoding in HAS make network video traffic show still identifiable patterns, most existing work cannot distinguish different platforms due to the similarity of video streaming. Therefore, we propose a traffic-based side-channel attack method to identify VPN video streaming platforms in real time. The aggregated feature sequence of the unidirectional video streaming is extracted to significantly retain the characteristics of different video platforms. Experiments on 10Gbps backbone background traffic show that the F1-score of the method exceeds 97% and can be processed in real time. In addition, we verify the method's robustness on datasets with different path features and encryption techniques. A comparison with similar methods shows that our method only requires 1/1260 of the storage and 1/60 of the processing time to identify accurately.

Keywords: Video Streaming · Side-channel attack · VPN · Privacy

1 Introduction

Video streaming is becoming increasingly popular as multimedia technologies evolve and network bandwidth increases. Cisco's IP Traffic Report [6] showed that video streaming accounted for 75% of total IP traffic. The prevalence of video streaming causes more and more users to worry about privacy breaches, such as the video platforms that users watch leaking the privacy of their preferences. Therefore, More and more video streaming is being encrypted to protect users' privacy. In addition, many users use VPN (Virtual Private Network) [16] to enhance privacy protection further.

Many studies have been proposed for video streaming. However, most of these studies focused on video QoE metrics [12], video titles [11], and other aspects.

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 335–349, 2024.

https://doi.org/10.1007/978-3-031-56326-3_24

Few studies focused on video platform identification, especially in VPN encryption scenarios. VPN makes video platform identification challenging because it poses traffic obfuscation, further data encryption, masking of the actual communication address, and others [15]. Nevertheless, platform identification of VPN video streaming can still be achieved by traffic-based side-channel attacks. Most video service providers rely on HAS (HTTP Adaptive Streaming) [9] technology to support their video services. The segment-based data transmission mechanism and VBR (Variable Bit Rate) encoding in HAS causes video streaming to display distinct traffic patterns, thus posing the risk of traffic-based side-channel attacks.

Identifying traffic types by traffic-based side-channel attacks has been brought into focus for years. Although the identification objects of some studies include VPN video streaming, most of these studies suffer from the following problems.

- **Limited application scenarios:** Asymmetric routing scenarios are common in wide-area networks, where only unidirectional flow data can typically be collected. However, many methods [8, 13] rely on extracting features from bidirectional flows.
- **Coarse granularity of identification:** Most existing studies [10, 18] use service types as attack granularity, such as video, email, file, and others. Traffic belonging to different service types is relatively easy to distinguish, as different services have different traffic patterns. However, video streaming from different platforms is often supported by the same type of protocols, such as HAS. As a result, their traffic pattern is similar, which makes it very challenging to identify the platform further.
- **Insufficient practicality evaluation:** Most existing studies [14, 15] only evaluated the accuracy of their methods. However, most methods are characterized by complex features, which causes them to be time-consuming. Therefore, these methods may not apply to high-speed networks and real-time scenarios. Moreover, these studies lack consideration of the reality that real-world network environments contain massive complex background traffic, and specific types of traffic usually account for a small proportion. Therefore, these methods will likely suffer from failure in real-world network environments. In addition, these studies lack the additional evaluation of using new samples, so the robustness is not guaranteed.

To solve the above issues, we propose a traffic-based side-channel attack method to identify the platform of VPN video streaming. Firstly, the method uses unidirectional flows as feature sequence extraction objects, and thus, it applies to common asymmetric routing scenarios. Secondly, we construct feature sequences by aggregation to significantly preserve the traffic characteristics of different video platforms. With a lightweight 1D-CNN model that learns the recognizable feature patterns from the extracted feature sequences, we accurately identify multiple video platforms within VPN channels. Finally, we evaluate the practicality of our method on different datasets. Our method achieves real-time

recognition in a high-speed network environment at 10Gbps with an average accuracy of over 97%. In the testing set with the addition of new samples, the accuracy of our method decreases by no more than 2%.

The rest of the paper is organized as follows: Sect. 2 discusses the relevant work, and Sect. 3 describes the method proposed. The evaluation results of the method are given in Sect. 4. Finally, Sect. 5 gives a summary of the entire paper.

2 Related Works

This section focuses on reviewing studies using side-channel attacks to identify traffic types. Most of these studies have considered VPN encryption scenarios, and the identified objects include video streaming, voice streaming, and file streaming.

Threats to user privacy via traffic-based side-channel attacks have attracted much attention for years. Chen et al. [4] stated that this attack is severe and general because it exploits basic traffic features such as the number of packets. In the following, we focus on the studies that use these basic features to train ML (Machine Learning) or DL (Deep Learning) models that can identify traffic types and thus reveal user behavior, such as watching videos, chatting, and browsing.

Bidirectional flows contain rich feature information, so many methods [8, 13] extract features from them, including statistical features of interval time and packet size. However, asymmetric routing in the backbone network results in upstream and downstream traffic passing through different paths, capturing only unidirectional traffic at the collection point. Therefore, these methods may not be suitable for the backbone network. While there are also methods [10, 18] based on unidirectional flows, they typically focus on classifying traffic by service types, such as video, email, and files. In addition, some methods focus on identifying VPN traffic [1] and multimedia protocol tunnel traffic [3]. This type of identification is coarser in granularity and does not reveal the user's preferences.

Some approaches identify traffic more finely by application type rather than service type. For example, some studies [14, 17] processed traffic as images for deep learning to identify applications such as YouTube and Skype that VPN encrypts. Some studies [7, 15] specifically identify video applications. Since [15] also identifies VPN video streaming, we specifically compared our work to it in Sect. 4. Although these studies use application types as identification granularity close to our goal, they only evaluated the accuracy of their methods. The practicality of their methods, such as real-time performance, and their effectiveness in real networks containing massive background traffic cannot be guaranteed.

Therefore, this paper proposes a traffic-based side-channel attack method to identify VPN video streaming platforms and comprehensively evaluates the method's practicality on different datasets.

3 Methodology

In this section, we describe the traffic pattern of video streaming, the feature extraction process, and the construction of the 1D-CNN model. Figure 1 shows

the overall workflow of our method. We extract feature sequences from the captured background and video traffic to construct the training set. Using the training set, we trained a 1D-CNN model that can identify the video streaming platforms in the presence of massive background traffic. This 1D-CNN model is applied to eavesdrop on network traffic. The application set obtained after the feature extraction of the network traffic is fed into the 1D-CNN model, and then the 1D-CNN model outputs its identity directly.

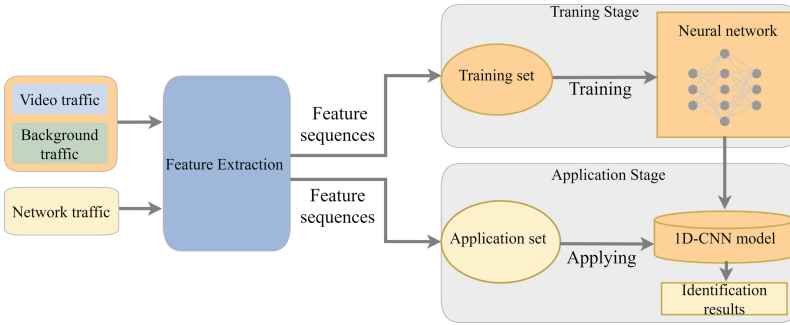


Fig. 1. Overall workflow

3.1 Analyzing the Traffic Patterns of Video Streaming

Segment-based data transmission mechanism and VBR encoding in HAS result in video streaming displaying distinct traffic patterns. Figure 2(a) and Fig. 2(b) show 100 s traffic traces for VPN video streaming from Vimeo and Dailymotion, respectively. They both show distinctive “ON-OFF” pulse patterns. The “ON” (blue bumps in Fig. 2) refers to short periods of high-intensity data transmission. The “OFF” (the part between the two blue bumps in Fig. 2) refers to long periods of low-intensity data transmission. The interval between two adjacent $\langle \text{ON}, \text{OFF} \rangle$ is a period.

According to [2], video streaming goes through two successive stages during playback: the buffering and steady stages. Figure 2 shows that the traffic pattern indeed behaves in two stages. In the early stages of video playback, the traffic pattern shows a continuous pattern of high-intensity data transmission. After some time, the traffic pattern shows an “ON-OFF” pattern with some periodicity. Compared to other types of flows, this staged traffic pattern makes video streaming more distinctive. This characteristic makes it possible to identify video streaming in the presence of many background flows.

Different video platforms use different development techniques, resulting in different characteristics of their traffic patterns in these two phases, although they are similar in appearance. These differences mainly manifest in two aspects: the different duration of the buffering phase and the different pulse periods of the steady stage. We can see these differences in Fig. 2(a) and Fig. 2(b). Inspired by this, in 3.2, we construct feature sequences that preserve the phase and

periodicity characteristics of the video streaming traffic patterns. These feature sequences make it possible to identify different video platforms with similar traffic patterns.

3.2 Extracting Feature Sequences

Inspired by the characteristics of video streaming traffic patterns, we construct feature sequences that can quickly identify the video streaming platforms from the traffic. Video streaming usually have a long lifetime. Therefore, we introduce the concept of a time window to represent the shortest period of eavesdropping traffic to achieve accurate identification. We define slidable time cells within a time window, and the aggregated statistical features extracted from these time cells form the feature sequence with temporal information. These feature sequences describe the traffic data over time, thus preserving the phased and periodic character of the video streaming traffic pattern. In addition, the amount of feature data can be significantly reduced by aggregation while making the features more stable.

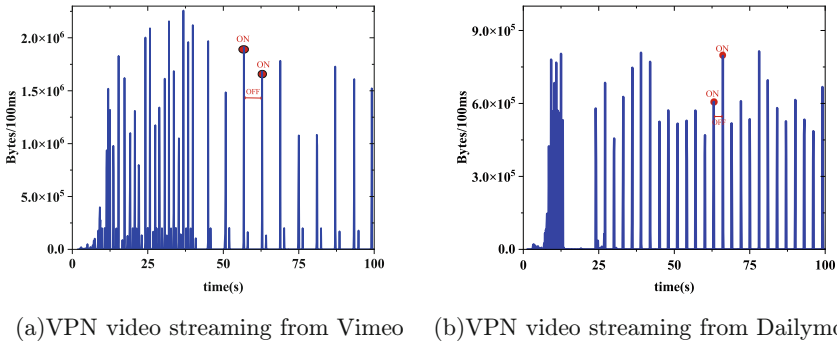


Fig. 2. The traffic patterns of video streaming.

Specifically, for a flow of duration S , we only eavesdrop on traffic within a time window ($S > T_w$, T_w represents the time window size). Within the time window, the time cells slide backward in steps of T_c (the time cell size) from the earliest transmitted packet to the last packet. Our aggregation method is Packets Summation within a Time Cell (PSTC). Equation (1) shows the i -th feature value extracted from each flow is obtained by summing the sizes of all packets within the i -th time cell in the time window, where p_1 and p_n denote the first and last packet within this time cell, respectively. Multiple such feature values can be obtained in a time window, and these feature values are arranged linearly in time order to form the feature sequence we want to obtain, as shown in (2).

$$pstc_i = pstc_i = \sum_1^n Length(p_1, p_2, p_3, \dots, p_m, \dots, p_n) \tag{1}$$

$$Feature_sequence = (pstc_1, pstc_2, pstc_3, \dots, pstc_m, \dots, pstc_n), n = \frac{T_w}{T_c} \tag{2}$$

In the following, we give the process of extracting the above feature sequence from the raw traffic.

- (1) **Extracting flows:** We extract the unidirectional flows based on the five-tuple from the raw traffic. To quickly locate the corresponding flows, we use a chained hash table to store the records for each flow.
- (2) **Filtering out efficient flows:** Flows carrying efficient information are generally long flows with a relatively large number of packets and a long duration. To avoid short flows taking up computational resources, we filter out efficient flows using a minimum packet number threshold and a minimum duration threshold.
- (3) **Aggregating packets:** We use a vector with a dynamic array function to store packets. Aggregation is achieved by adding up the sizes of all packets in the same storage position.

3.3 1D-CNN Model

In recent years, 1D-CNN has been successfully applied in sequential signal processing and NLP (Natural Language Processing), such as sentiment analysis and text classification. Inspired by this, we use the 1D-CNN model and train it by supervised learning.

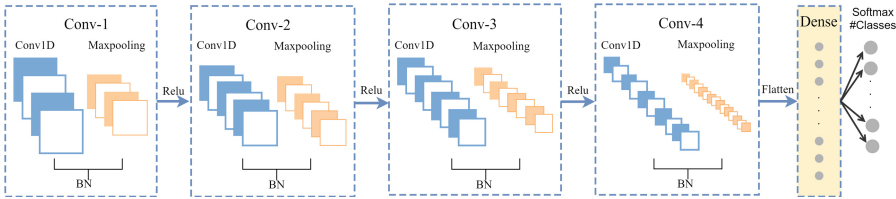


Fig. 3. Lightweight 1-DCNN model structure.

We design a 1D-CNN model consisting of four layers of Conv units and a Dense layer. The model is lightweight, and its structure is shown in Fig. 3. Each layer of Conv units consists of a Convolutional layer (Conv1D), a BatchNormalization (BN), and a MaxPooling1D (MP). BN allows the data fed into the convolution layer to maintain a standard normal distribution with a mean of 0 and a variance of 1. This treatment allows the feature values to fall in the sensitive region of the nonlinear activation function, avoiding gradient disappearance and preventing overfitting. In addition, adding the MaxPooling1D reduces the number of parameters in the model, speeding up the training process and improving model generalization. We use the Relu activation function to connect each layer of Conv units. As we deal with a multi-classification task, we use the softmax activation function to transform the output of the Dense layer into probability values.

The parameters for each layer of units are set as shown in Table 1. Conv1D(x, y, z) denotes a 1-dimensional convolutional layer with x filters and

a convolutional kernel of size y , where z denotes the feature form of the input. $\text{MaxPooling1D}(x)$ denotes the maximum pooling layer with a stride length of x . $\text{Dense}(n)$ denotes that n neurons are used to output the recognition results of n classes.

Table 1. Parameter details of the 1D-CNN model

Layer	Parameter Settings Description
Conv-1	Conv1D (256, 3, (600, 1))-BatchNormalization()-MaxPooling1D(2)
Conv-2	Conv1D (128, 3)-BatchNormalization()-MaxPooling1D(2)
Conv-3	Conv1D (64, 3)-BatchNormalization()-MaxPooling1D(2)
Conv-4	Conv1D (32, 3)-BatchNormalization()-MaxPooling1D(2)
Dense	Dense(n)

4 Experimental Evaluation

In this section, we first introduce the dataset and the experimental environment. Secondly, we evaluate our method’s effectiveness and real-time performance in a 10Gbps high-speed network. Furthermore, we also evaluate the robustness in complex situations. We chose Precision (Pre), Recall (Rec), F1-score (F1), and Accuracy (Acc) as evaluation metrics.

4.1 Experimental Datasets

Throughout the experiments, we use three types of datasets: VPN video traffic, regular encrypted video traffic, and 10Gbps backbone traffic.

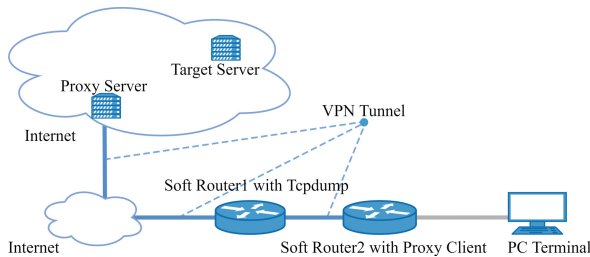


Fig. 4. Topology of the VPN video streaming capture environment.

Datasets of the VPN Video Traffic: To obtain the latest VPN video traffic, we set up a traffic collection environment, as shown in Fig. 4. Given the diversity of VPN tunnel encryption technologies, we focus on two popular VPN tunnel encryption technologies, Shadowsocks(Ss) and Vmess. Soft Router 1 deploys a Tcpcdump application for data collection, and Soft Router 2 deploys a VPN

client node while the PC plays the video. In addition, to complicate the data, our data are collected in three different network environments: campus, home, and corporate networks.

As the data to be collected is relatively large, a scripting program is designed on the PC to collect the data automatically. The scripting program includes three main functions: automatically crawling video URLs, playing video, and automatically capturing and saving traffic tracking during video playback. The software used is Uibot, Xshell, Tcpdump, and Browser.

Finally, we obtain the Shadowsocks encrypted dataset (Ss-video) and the Vmess encrypted dataset (Vmess-video). These two datasets contain various video content from 2 to 10 min, including sports, animation, movies, news, etc.

Table 2. Dataset description

dataset name	type	number of flow samples	total number of flow samples
Ss-video	vimeo	3764	18450
	twitter	3898	
	facebook	3445	
	youtube	3476	
	dailymotion	3867	
Vmess-video	vimeo	3901	18020
	twitter	2842	
	facebook	3543	
	youtube	3743	
	dailymotion	3991	
NonVPN-video	vimeo	2221	6653
	twitter	2519	
	facebook	1917	
10Gbps-background	10Gbps backbone traffic	375307	375307

Dataset of the Regular Encrypted Video Traffic: Regular encrypted video streaming is similar to VPN video streaming, so dataset NonVPN-video is used to evaluate the robustness of our model further. This dataset contains traffic traces of regular encrypted videos from Vimeo, YouTube, and Facebook collected by our lab in 2022.

Datasets of the Background Traffic: Dataset 10Gbps-background is a public dataset provided by the MAWI Working Group [5] and consists of weekly 900 s traffic traces collected on the 10 Gbps main IX link from WIDE to DIX-IE in April, May, and June 2020.

The final number of flow samples from the above dataset is shown in Table 2. By dividing the samples by a ratio of 9:1, we obtain a training set and a testing set. The training set is used to build models that meet different recognition requirements, and the testing set is used to evaluate the model’s performance.

4.2 Experimental Environment

We perform the task of extracting feature sequences from raw traffic on a device with a 12th generation Intel (R) Core(TM) i7-12700K (20 CPUs), 3.6 GHz, and 131072 MB RAM. In addition, we train and test our model on a PowerEdgeR740 with an Intel(R) Xeon(R) Gold 5220R CPU @ 2.20 GHz. The Tensorflow and Keras libraries are used as the software environment.

Since we perform a multiclassification task, categorical_crossentropy, which applies to the single-label multiclassification problem, is used as the loss function for model training. Regarding the model optimization, we chose SGD (Stochastic Gradient Descent) as the optimizer. The parameters of SGD set as : (lr = 0.01, nesterov = true, decay = 1e-6, momentum = 0.9). The min-batch is set to 256, and the training will stop after running 30 epochs.

Setting the size of time windows and time cells is crucial for constructing feature sequences. Through experimentation with different T_w and T_c , we thoroughly analyze the model's sensitivity to the feature form. In the experimental setting with T_w at 30, 40, 60, and 120 s while keeping T_u at 0.1 s, the difference in average accuracy does not exceed 1.17%, which is insignificant. Considering the high accuracy, strong stability, and less resource consumption, our later experiments are conducted with T_w at 60 s and T_u at 0.1 s.

Table 3. Identification results in the 10Gbps backbone network

VPN	Labels	Acc	Pre	Rec	F1	percentage*
Ss	vimeo	0.94	0.92	0.94	0.93	0.94%
	twitter	0.98	1.00	0.98	0.99	0.95%
	facebook	0.97	0.93	0.97	0.95	0.84%
	youtube	0.95	0.98	0.95	0.94	0.84%
	dailymotion	0.95	0.98	0.95	0.97	0.95%
	background	1.00	1.00	1.00	1.00	95.47%
Vmess	vimeo	0.97	1.00	0.97	0.98	1.03%
	twitter	0.98	0.98	0.98	0.99	0.77%
	facebook	0.98	0.95	0.98	0.98	0.94%
	youtube	0.98	0.99	0.98	0.97	0.95%
	dailymotion	0.99	0.99	0.99	0.99	0.95%
	background	1.00	1.00	1.00	1.00	95.35%
Ss+Vmess	vimeo	0.97	0.96	0.98	0.97	1.95%
	twitter	0.98	1.00	0.96	0.98	1.59%
	facebook	0.94	0.97	0.95	0.96	1.65%
	youtube	0.98	0.95	0.98	0.96	1.79%
	dailymotion	0.98	0.97	0.98	0.97	1.75%
	background	1.00	1.00	1.00	1.00	91.26%

* The percentage indicates the proportion of each type of flow sample in the testing set.

4.3 Effectiveness Evaluation in Real Backbone Networks

The real-world network environment contains a large amount of complex background traffic, while specific types of traffic usually account for a small percentage. These background flows will likely interfere with the identification process. We use real 10Gbps backbone traffic as background traffic to evaluate the effectiveness of our method in real network environments. These backbone flows do not include packet payload content to protect sensitive private data. However, our method identifies traffic through a traffic-based side-channel attack. Therefore, this backbone traffic can still be used to evaluate our method.

Table 3 shows the results of our method for identifying the VPN video streaming platforms in the 10Gbps high-speed network. For the identification results of video streaming encrypted by Shadowsocks, the average precision, average accuracy, average recall, and average F1-score are all 97%. For the identification results of video streaming encrypted by Vmess, the average accuracy, average recall, and average F1-score are all 98%. The average precision is 97%. For the identification results of mixed video streaming encrypted by Shadowsocks and Vmess, the average accuracy, average precision, and average F1-score are all 98%. The average recall is 97%.

The results of Table 3 show that our model can accurately identify VPN video streaming platforms in high-speed network environments, even when the proportion of VPN video streaming is small. Furthermore, our model achieves accurate identification even in scenarios where video streaming encrypted by different VPN tunnel encryption techniques is mixed. These experimental results demonstrate the effectiveness of our method in a real network environment.

4.4 Real-Time Evaluation in High-Speed Networks

We evaluate the real-time performance of our model in a high-speed backbone network environment at 10Gbps. We analyze the time performance in terms of model training time and recognition time.

Model Training Time: For the training set containing 354382 samples, our model training time is 1179.32 s. The average training time per epoch is 39.31 s, and the average training time per sample is 3.33 ms. This result indicates that we do not need to spend much time training the model because our model is lightweight, and the input feature sequence is not complex.

Recognition Flow Time: For the test set of size 83.62 GB, we extracted 39376 feature samples from it. The total processing time is 130.35 s, and the total prediction time is 6.06 s, so the total recognition time is 136.41 s. Our method's average recognition speed is 4.9Gbps, and the maximum throughput of high-speed traffic provided by the MAWI working group [5] does not exceed 3.4Gbps. This result indicates that our recognition speed is faster than the data transfer speed, thus validating that our method can achieve real-time identification in a high-speed network environment.

4.5 Robustness Evaluation in Complex Scenarios

We evaluate the robustness of our method in complex cases through two experiments. One is the effect of the MTU (Maximum Transmission Unit) for different paths on the method’s accuracy, and the other is the effect of different encryption techniques on the method’s accuracy.

Complex Scenarios with Different MTU Values: In a real network environment, packet payload size is limited by the device’s MTU, and excessively long packets are split into multiple smaller sizes than MTU slices. In fact, due to distance and other factors, data in the network is often transmitted over multiple paths, and devices on different paths are likely to have different MTU values. However, changes in packet size and number due to variations in MTU values are likely to harm methods that rely on the statistical characteristics of packets.

Although a well-trained model performs well on a testing set, it may fail with new samples. By modifying the router’s MTU to 1000, we collect additional video streaming encrypted by Shdowsocks and Vmess from Vimeo, Twitter, and Dailymotion. These new samples with an MTU of 1000 are added to the original testing set (MTU = 1500) to form a new testing set. We evaluate the robustness of our trained model on this new testing set.

Figure 5(a) and Fig. 5(b) show that new samples with different MTU values have little effect on our trained model’s identification accuracy. This result suggests that our model is robust enough to identify new samples.

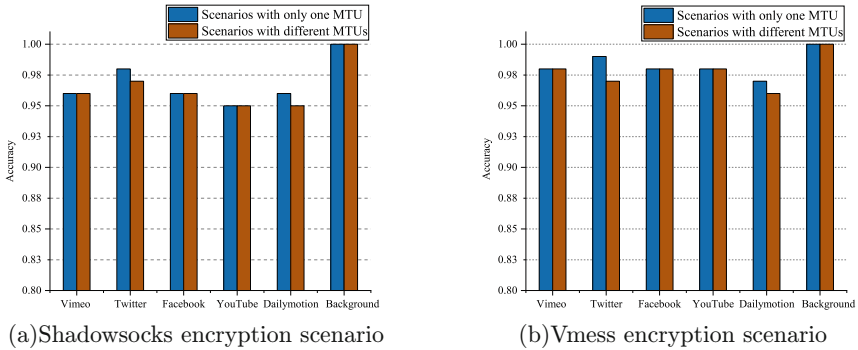


Fig. 5. Accuracy comparison of scenarios with only one MTU and scenarios with different MTUs.

Complex Scenarios with Different Encryption Techniques: We combine the datasets Ss-video, Vmess-video, and NonVPN-video into a new dataset. We evaluate our model on this dataset with three types of encryption techniques (HTTPS, Shdowsocks, and Vmess).

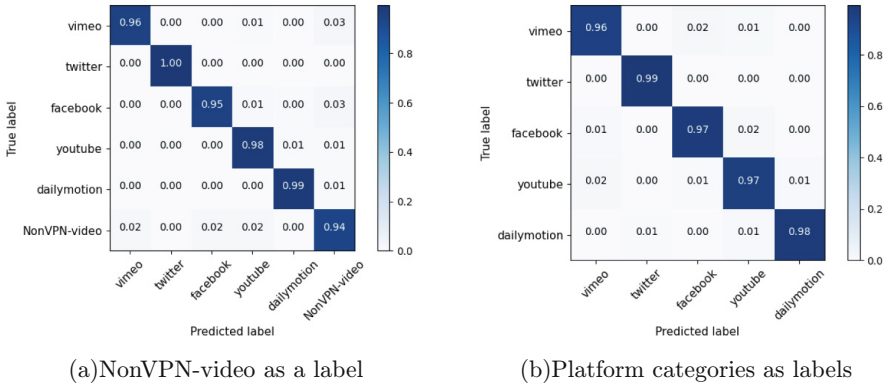


Fig. 6. Classification results in the complex scenarios.

Figure 6(a) shows the classification results for regular encrypted video streaming as a class with an average identification accuracy of 97%. Analysis of Fig. 6(a) shows that confusion occurs mainly when VPV video streaming from Vimeo, Facebook, and YouTube is predicted as NonVPN-video, and NonVPN-video is predicted as Vimeo or Facebook, or YouTube. This result is acceptable because the NonVPN-video dataset consists of traffic traces from YouTube, Vimeo, and Facebook. Figure 6(b) shows the classification results for further classifying regular encrypted video streaming into platform categories, showing an accuracy of over 96% for each platform category.

The above results show that even though VPN video streaming is similar to regular encrypted video streaming, our model can also classify them. This is because VPN transmits more negotiation information than regular encryption due to the requirement of establishing tunnels. Moreover, we can identify video streaming platforms in complex scenarios with different types of encryption techniques, further validating our model’s robustness.

4.6 Comparison with Existing Methods

We conduct comparative experiments on the dataset Vmess-video with two works (Approach1 [14] and Approach2 [15]) of similar identification granularity to ours. Approach1 transformed the raw traffic into 1500*1500 images, which were fed into a CNN for learning. With this method, they achieved application-level traffic identification. Approach2 used a Random Forest model to achieve video streaming source identification.

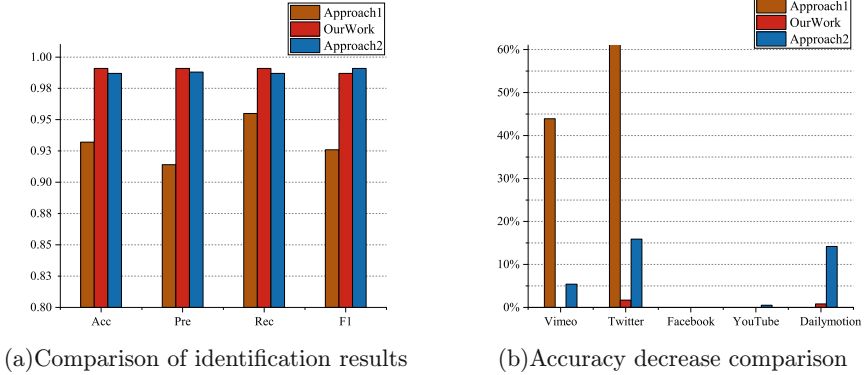


Fig. 7. Comparison results of the three methods.

Figure 7(a) shows the results of comparing the average values of different evaluation metrics for the three methods. It shows that our method performs better than the existing methods in accuracy. Although the identification accuracy of Approach 2 is close to ours, Fig. 7(b) shows that it is not as robust as ours when facing new samples.

Figure 7(b) shows each method’s decrease in identification accuracy after adding a new sample with $MTU = 1000$ to the testing set. Figure 7(b) shows that the average decrease of our method is only 0.6%, while the average decrease of Approach1 and Approach2 is 7.2% and 21.2%, respectively. Among them, Approach1 significantly decreases the identification accuracy by more than 60% in some classes. The reason for this is that our method is based on packet aggregation, while Approach1 and Approach2 are based on individual packets, so they are more sensitive to changes in MTU values than ours. The comparison results show that our method is more robust.

We further compare the temporal and spatial performance, and Table 4 shows that our method occupies less memory and time than the FlowPic method. The memory occupation and processing time of the FlowPic method are 1260 and 61 times than our method, respectively. This result is because the object processed by Approach1 are images. Although both Approach1 and our method are based on CNN, Table 4 shows that Approach1 does not converge as fast as ours.

Table 4. Comparison of the three methods in time and space performance

Compare items	OurWork	Approach1	Approach2
The average size of each sample after feature characterization	0.0017 MB	2.1421 MB	0.0009 MB
Average processing time(including feature extraction and recognition time) per sample	3.65 ms	222.12 ms	3.31 ms
Average convergence time of the model	30.4 s	68.61 s	-

5 Conclusion

Although VPN protects video streaming, traffic-based side-channel attacks can still threaten users' privacy. This paper proposes a traffic-based side-channel attack method to identify VPN video streaming platforms. The method constructs aggregated feature sequences from unidirectional flows and feeds these feature sequences into a 1D-CNN model to obtain depth features automatically. Experiments in a high-speed network environment with massive background traffic show that the method can achieve real-time identification with F1-score over 97%. Furthermore, experiments on datasets with different path features and encryption techniques verify that the method is robust. Finally, the method has superior performance in time and space compared to similar methods.

Considering the diversity of encryption technologies and traffic types, we will cover more types of encrypted traffic in our future work.

Acknowledgements. This work was supported by the National Key R&D Program of China (2021YFB3101403).


References

1. Afuwape, A.A., Xu, Y., Anajemba, J.H., Srivastava, G.: Performance evaluation of secured network traffic classification using a machine learning approach. *Comput. Stand. Interfaces* **78**, 103545 (2021)
2. Akhshabi, S., Anantkrishnan, L., Begen, A.C., Dovrolis, C.: What happens when http adaptive streaming players compete for bandwidth? In: *Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 9–14 (2012)
3. Barradas, D., Santos, N., Rodrigues, L.: Effective detection of multimedia protocol tunneling using machine learning. In: *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 169–185 (2018)
4. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-channel leaks in web applications: a reality today, a challenge tomorrow. In: *2010 IEEE Symposium on Security and Privacy*, pp. 191–206. IEEE (2010)
5. Cho, K.: Km, and kato, a. traffic data repository at the wide project. *USENIX ATC, Freenix track* (2000)
6. Cisco.: Cisco visual networking index: Forecast and trends, 20172022 white paper. <https://davidellis.ca/wp-content/uploads/2019/05/cisco-vni-feb2019.pdf>
7. Dias, K.L., Pongelupe, M.A., Caminhas, W.M., de Errico, L.: An innovative approach for real-time network traffic classification. *Comput. Netw.* **158**, 143–157 (2019)
8. Draper-Gil, G., Lashkari, A.H., Mamun, M.S.I., Ghorbani, A.A.: Characterization of encrypted and VPN traffic using time-related. In: *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 407–414 (2016)
9. Esteban, J., Benno, S.A., Beck, A., Guo, Y., Hilt, V., Rimac, I.: Interactions between http adaptive streaming and TCP. In: *Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 21–26 (2012)

10. Izadi, S., Ahmadi, M., Rajabzadeh, A.: Network traffic classification using deep learning networks and Bayesian data fusion. *J. Netw. Syst. Manage.* **30**(2), 25 (2022)
11. Khan, M.U., Bukhari, S.M., Maqsood, T., Fayyaz, M.A., Dancey, D., Nawaz, R.: SCNN-attack: a side-channel attack to identify youtube videos in a VPN and non-VPN network traffic. *Electronics* **11**(3), 350 (2022)
12. Mangla, T., Halepovic, E., Ammar, M., Zegura, E.: Using session modeling to estimate HTTP-based video QoE metrics from encrypted network traffic. *IEEE Trans. Netw. Serv. Manage.* **16**(3), 1086–1099 (2019)
13. Pacheco, F., Exposito, E., Gineste, M.: A framework to classify heterogeneous internet traffic with machine learning and deep learning techniques for satellite communications. *Comput. Netw.* **173**, 107213 (2020)
14. Shapira, T., Shavitt, Y.: FlowPic: a generic representation for encrypted traffic classification and applications identification. *IEEE Trans. Netw. Serv. Manage.* **18**(2), 1218–1232 (2021)
15. Shi, Y., Ross, A., Biswas, S.: Source identification of encrypted video traffic in the presence of heterogeneous network traffic. *Comput. Commun.* **129**, 101–110 (2018)
16. Stanton, R.: Securing VPNs: comparing SSL and IPSEC. *Comput. Fraud Secur.* **2005**(9), 17–19 (2005)
17. Tang, J., et al.: Caps-LSTM: a novel hierarchical encrypted VPN network traffic identification using CapsNet and LSTM. In: Lu, W., Sun, K., Yung, M., Liu, F. (eds.) *SciSec 2021*. LNCS, vol. 13005, pp. 139–153. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-89137-4_10
18. Tang, P., Dong, Y., Mao, S.: Online traffic classification using granules. In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1135–1140. IEEE (2020)



Toward the Establishment of Evaluating URL Embedding Methods Using Intrinsic Evaluator via Malicious URLs Detection

Qisheng Chen^(✉) and Kazumasa Omote

University of Tsukuba, 1-1-1, Tennodai, Tsukuba, Ibaraki 305-8577, Japan
s2230146@s.tsukuba.ac.jp

Abstract. In order to compare the performance of the malicious URLs detection method, researches used the F-score or other detection accuracy to evaluate, but there are some difficulties in evaluating the URL embedding method used in malicious URLs detection because the detection accuracy is also effect by machine learning or deep learning models and data sets. An evaluation method of URL embedding method that is not affected by other factors is particularly important. In this paper, we proposed an intrinsic evaluation method for URL embedding method that is not affected by machine learning models or deep learning models and data sets. Besides, We analyse some URL embedding methods according to intrinsic and extrinsic methods and offer a guidance in selecting suitable embedding methods in URL by analysing the results.

Keywords: Malicious URLs detection · URL Embedding Evaluation · Network security

1 Introduction

With the development of machine learning and deep learning, machine learning and deep learning are also used to detect malicious URLs. In these methods, in order to be able to convert the URL as a string into a number column that can be recognized by machine learning or deep learning, like the natural language processing, it will segment the URL and embed the URLs into the feature vectors. Chen's research shows most malicious URLs detection methods use embedding, segmentation methods, and machine learning algorithms, which means either segmentation method, embedding method, or machine learning model will affect the performance of malicious URL detection method.

The feature of the method of using machine learning to detect malicious URLs is that it can detect malicious URLs efficiently under the premise of a low false detection rate. In this case, the accuracy of malicious URLs detection method is an important evaluation index. For this reason, researches on malicious URLs detection method based on machine learning focus on increasing the accuracy of detection.

As an important part of malicious URLs detection methods, the method of turning URLs into feature vectors which call URL embedding method will also significantly affect the performance of malicious URL detection methods. However, the only way to evaluate the performance of URL embedding method is the accuracy result after training the machine learning model in related research. The accuracy of malicious URLs detection is not only based on the performance of the detection methods, but is also related to training sets and test sets. In other words, the accuracy of malicious URLs detection methods will change due to different test sets, so it is not comprehensive to evaluate URL embedding methods only from the detection accuracy of single test sets.

To solve this problem, the evaluation of another aspect, in addition to accuracy, becomes particularly important. The evaluation method focus on the embedded feature vectors called intrinsic evaluating method. Unlike extrinsic evaluating method, it does not depend on the other part of detection and the training test set. Because the embedding method is the only variable, the advantage is not to worry about the impact of other variables.

In this paper, we proposed an intrinsic evaluation method for URL embedding method based on cosine similarity. The intrinsic evaluation method can evaluate URL embedding method without the effect of machine learning models and data sets. Besides, we evaluated several URL embedding methods with intrinsic and extrinsic method and found that the traditional extrinsic evaluation methods have some difficulties in evaluating URL embedding methods and proved the intrinsic method's usefulness. At last, we offered guidance in selecting suitable embedding method in malicious URLs detection according to the results of the evaluation.

2 Preliminary

In this section, we will introduce F-1 score and cosine similarity, which will be used as the indicator of extrinsic evaluation method and the evaluation algorithm of intrinsic evaluation method. Besides, we will introduce the URL embedding methods used in our test.

2.1 F-1 Score

F-1 score is the harmonic mean of the precision: attempts to answer the question that what proportion of positive identifications was actually correct; and recall: attempts to answer the question that what proportion of actual positives was identified correctly:

$$Precision = \frac{tp}{tp + fp} \quad (1)$$

$$Recall = \frac{tp}{tp + fn} \quad (2)$$

$$F = 2 \frac{Precision \cdot Recall}{Precision + Recall} \quad (3)$$

2.2 Cosine Similarity

We used cosine similarity as an indicator of measuring how much information is retained. Specifically, the URLs are embedded as vectors in an inner product space and the cosine similarity is defined as the cosine of the angle between two vectors, that is, the dot product of the vectors divided by the product of their lengths.

$$S = \frac{v_x \cdot v_y}{\|v_x\| \|v_y\|} \quad (4)$$

As shown in Eq. 4, v_x and v_y are two feature vectors and $\|v_x\|$ and $\|v_y\|$ mean their L2 norm. The advantage of cosine similarity is its low complexity, especially for sparse vectors: only the non-zero coordinates must be considered. Cosine similarity represents the relationship between two Tokens; when cosine similarity is close to 1, it means that the two Tokens are very similar in the meaning of embedding method. On the contrary, if cosine similarity is close to 0, the two Tokens are not similar in the sense of embedding method. On the contrary, if cosine similarity is close to 0, the two Tokens are not similar in the sense of embedding method.

2.3 URL Embedding Methods

The method that turns the URL into feature vectors that can be trained is called embedding method. In this section, we will introduce several famous embedding methods by dividing them into context-considering embedding methods and context-agnostic embedding methods.

Context-Considering Embedding Methods. Context-considering embedding methods means the generation of word vectors takes into account the context of the corpus. Like the algorithms CBOW and Skip-gram in Word2Vec, they can predict a word based on context or predict the context based on a word. When they change words into word vectors, they will consider their context, which will increase the accuracy of prediction. In this paper, we used Word2Vec [4], FastText [5], GloVe [8] as the target context-considering URL embedding methods.

Context-Agnostic Embedding Methods. Context-agnostic embedding methods like One-hot Code and TF-IDF [9] are the basic embedding methods. They turn words into vectors easily, and they embed words only by using the word's quantity or physical order.

3 Related Work

3.1 A Three-Step Framework for Detecting Malicious URLs

Chen [3] proposed a three-steps framework to review 14 methods of detecting malicious URLs. They divided the method of malicious URL detection

using machine learning into three parts: Segmentation, Embedding, and Machine learning. They evaluated some machine learning models and context-considering methods by three-step framework, and they verified the importance of considering context and found that context-considering embedding methods are more important and the malicious URLs detection accuracy improved by about 6% with context-considering methods. Chen’s research uses F-1 score to evaluate the suitability of each embedding method and malicious URL detection methods according to the specific malicious URL detection task. However, once the training set and test set of malicious URL detection task change, F-1 score will also change, which will affect the evaluation results. In this case, their evaluation of embedding methods is incomplete.

3.2 The Extrinsic and Intrinsic Evaluating Method in NLP

Wang [11] categorizes the NLP evaluators into intrinsic and extrinsic two types. Intrinsic evaluators test the quality of a representation independent of specific natural language processing tasks, while extrinsic evaluators use word embeddings as input features to a downstream task and measure changes in performance metrics specific to that task. Although the Token split by URL embedding in the Segmentation step is different from natural language processing, and the Token is not a Word in the language sense, because the process and method of URL embedding and Word embedding are similar, we can refer to the evaluation method of Word embedding.

3.3 URL2Vec

The method proposed by Yuan et al. [12] named URL2Vec: “URL Modeling with Character Embeddings for Fast and Accurate Phishing Website Detection” is a typical research that uses machine learning to detect malicious URLs. In the segmentation step, they divided the URLs by the structure of URL protocol, sub-domain name, domain name, domain suffix, and URL path 5 parts. In the embedding step, each part of URL were embedded by using Skip-Gram as feature vectors. In the machine learning step, they trained many machine learning algorithms, including LR, GBDT, DT, KNN, RF and XGB.

3.4 Token Segmentation Method

The method proposed by Kaneko et al. [6] named “Detecting Malicious Websites by Query Templates” used the machine learning algorithm DBSCAN to cluster malicious URLs and benign URLs. In the segmentation step, they chose a different way to divide URLs is that use all delimiters into URLs. Each part of the split URL were called a Token and we call this method as Token segmentation method. Obviously, this method maintains the information of words in the URL, and we used the method to split URL in this paper.

4 Intrinsic Evaluation Method

Intrinsic evaluation methods focus on the embedding performance of URL embedding methods. They test the quality of a representation independent of specific malicious URLs detection tasks and they measure the relationships among domains in the URL directly. In other words, the embedded feature vector contains the relative information of the URL Token, and the accuracy of the amount of information retained after changing the URL string into a digital string reflects the performance of URL embedding method.

4.1 Intrinsic Score

With the premise in Sect. 2.2, we can know that if two Tokens have similar meanings in the URL, and their cosine similarity is close to 1, it means that the two Tokens are well embedded. The method to evaluate a group of Token’s similarity is to calculate their average value, as shown in Eq. 5, which means the Tokens in a group calculate cosine similarity with each other and take their average value. Because the Tokens in the group are similar to each other, the closer $S_{Similar}$ is to 1, the better they are embedded.

$$S_{Similar} = \frac{1}{n(n-1)} \sum_{v_x \in A, v_y \in A, v_x \neq v_y} S(v_x, v_y) \quad (5)$$

On the other hand, if two Tokens are not similar in URL meaning, and their cosine similarity is close to 0, it also means that the two Tokens are well embedded. The method is similar to Eq. 5 but it needs two groups of Tokens and group A is not similar to group B in the meaning of URL. As shown in Eq. 6, it calculates the average of cosine similarity of group A with group B because the Tokens in group A are not similar to the Tokens in group B, so the closer $S_{Dissimilar}$ is to 0, the better they are embedded.

$$S_{Dissimilar} = \frac{1}{n(n-1)} \sum_{v_x \in A, v_y \in B} S(v_x, v_y) \quad (6)$$

More expansion, there are three characteristics of embedded well: $S_{Similar}$ close to 1, and $S_{Dissimilar}$ close to 0, and the difference between $S_{Similar}$ and $S_{Dissimilar}$ is large, so we propose the following algorithm to evaluate the performance of URL embedding method, the larger the score, the better performance of embedding:

$$Score = (100 \cdot S_{Similar} - 100 \cdot S_{Dissimilar})^2 + 100 \cdot S_{Similar} \quad (7)$$

4.2 URL Token Pair

In order to verify the relationship between two feature vectors, we need a pair of URL Tokens that already know their relationship. Likes Token “amazon” and “google”, they usually play the role of the domain name in URL

“www.amazon.com” or “www.google.com” so they should be similar in either URL or vector. We collected the top 50 domains in AlexaTop and looking forward to selecting 15 of them to form a similar Token set. We calculated the cosine similarity of these Tokens with the embedding methods Word2Vec, Fast-Text and GloVe to ensure they are not only similar in the meaning of domain but also similar in the meaning of embedding. After manually selecting and verifying by different embedding methods, we form a similar Token set shown in Table 1. Besides, we were also looking forward to selecting 15 Tokens which dissimilar from the Tokens in a similar Token set like the domain suffix part in URL, such as “www” or “com”. After selection and verification, the dissimilar Token set is shown in Table 2.

Table 1. Similar Token Set

amazon	google	reddit	youtube	facebook
taobao	yahoo	twitter	microsoft	sina
weibo	adobe	zoom	xinhuanet	ebay

Table 2. Dissimilar Token Set

com	co	net	htm	html
sports	finance	news	blog	www
shtml	search	exe	edu	index

5 Evaluation Process

5.1 Process of Extrinsic Evaluation

Extrinsic evaluating method uses URL embedding method as input features to a downstream task and measures changes in performance metrics specific to that task, which means we set up a specific malicious URLs detection task as a downstream task, and we used several indicators for evaluating the performance of malicious URL detection methods.

Figure 1 shows the outline of the process of extrinsic evaluation. The segmentation module will segment the URL first, and then the URL will be embedded into the feature vectors according to different embedding methods. The machine learning model will be trained with the feature vectors, after training the output model can predict the URL used for testing. In order to evaluate different URL embedding methods, we changed several methods in the embedding step and machine learning step, including Random Forest [10] and LightGBM [7]. Besides, the dimension also be set as a variable.

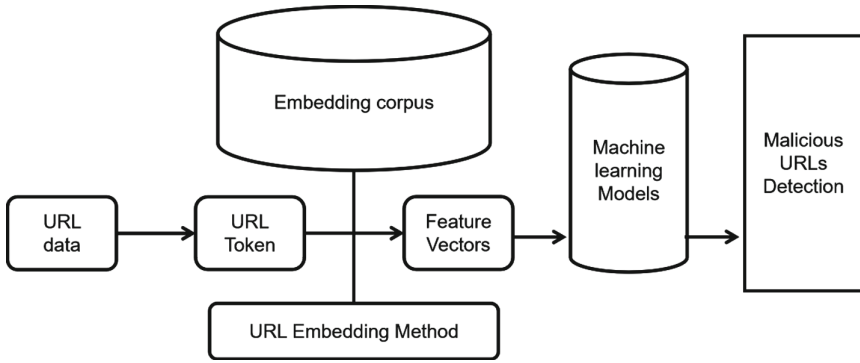


Fig. 1. Process of Extrinsic Evaluation

5.2 Process of Intrinsic Evaluation

Figure 2 shows the outline of the process of intrinsic evaluation. We split the URL in the corpus by all delimiters into URL, and set up the embedding method by using the corpus. Then the embedding method to be evaluated will calculate $S_{Similar}$ and $S_{Dissimilar}$ with a similar Token set and a dissimilar Token set.

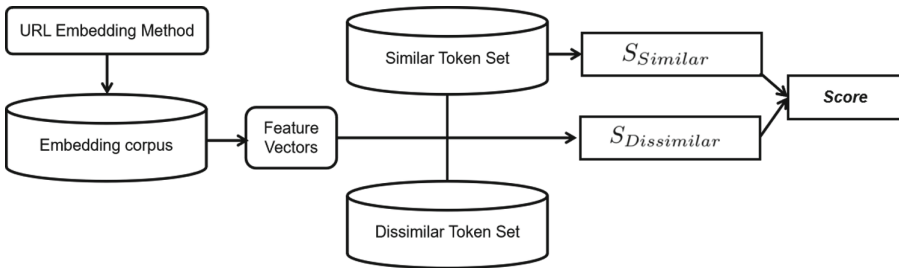


Fig. 2. Process of Intrinsic Evaluation

6 Evaluation

In this section, we will show and analyze the results obtained according to the evaluation process described in the Sect. 5.

6.1 Data Set

The extrinsic evaluating method requires a complete set of malicious URL detection tasks, so we have prepared an URL set for use as a corpus and the training test set for training and detection. We set up a crawling program to crawl 140 thousands URLs from AlexaTop [1], a website that counts the most used domain

names made by Amazon. We selected 5 thousands malicious URLs with classic URL structure in URLhaus [2], a manually maintained malicious URL database as malicious URLs set and we selected 5 thousands benign URLs in the crawl results with classic URL structure. The training set and test set will be produced by the random seed of cross-validation from the malicious URLs and benign URLs mentioned above.

6.2 Experiment Results

We took several URL embedding methods as variables and tested them with extrinsic and intrinsic evaluation methods. Table 3 shows the results of the extrinsic evaluation mentioned in Sect. 5.1, and Table 4 shows the results of the extrinsic evaluation mentioned in Sect. 5.2. The Table 5 shows the 64D $S_{Similar}$ and $S_{Dissimilar}$ of each URL embedding methods.

Table 3. F-1 Score Comparison of URL Embedding Method

	Machine Learning Model	64D	2D
Word2Vec/Skip-gram	Random Forest	0.99675	0.98979
	LightGBM	0.99065	0.98818
Word2Vec/CBOW	Random Forest	0.98965	0.99110
	LightGBM	0.98877	0.98790
FastText/Skip-gram	Random Forest	0.98792	0.98501
	LightGBM	0.98857	0.98444
FastText/CBOW	Random Forest	0.98799	0.98753
	LightGBM	0.98512	0.98746
GloVe	Random Forest	0.99473	0.99378
	LightGBM	0.99675	0.99591
TF-IDF	Random Forest	0.97811	0.97591
	LightGBM	0.95019	0.96047
One-hot Code	Random Forest	0.93805	0.92917
	LightGBM	0.92492	0.92386

Table 4. Score Comparison of URL Embedding Methods

	64D	2D
Word2Vec/Skip-gram	564	121
Word2Vec/CBOW	152	156
FastText/Skip-gram	450	148
FastText/CBOW	109	118
GloVe	438	157
TF-IDF	89	152
One-hot Code	97	96

6.3 Results Analyse

Intrinsic Method Solve the Disadvantages of Extrinsic Method. As shown in Table 3, even if the dimension of the word vector is reduced from 64 to 2, the F-1 score results are very close under the test of different machine learning models. In this case, it is difficult for us to compare the performance of each URL embedding methods and hard to select the right URL embedding method. With the help of the intrinsic evaluation method, we can know the embed situation like Table 4, 64D vectors are obviously better than 2D vectors.

Context-Considering Embedding Methods Are Better. As shown in Table 3 and Table 4, both extrinsic evaluation results and intrinsic evaluation results, the context-considering embedding methods are better than context-agnostic embedding methods, means not only in NLP but also in URL embedding, considering context is essential.

7 Discussion

Even though most URL embedding methods in extrinsic tests have achieved good detection accuracy, the specific cosine similarity of each URL embedding methods shown in Table 5 shows that this URL embedding methods are not the most suitable for malicious URLs detection. They usually get high similarity in $S_{Similar}$, but $S_{Dissimilar}$ is not too low, which means different URL Tokens are not well distinguished by existing URL embedding methods. The most special example is the GloVe method, it got the highest $S_{Similar}$, but the difference between $S_{Similar}$ and $S_{Dissimilar}$ is low. In general, Word2Vec is more suitable for malicious URLs detection, and the Skip-gram algorithm is more suitable for URL embedding.

However, the common problem with the existing URL embedding methods is these embedding methods are originally used for NLP. They identify similar words and related words from the perspective of natural language, which is base on the relative position of words in the corpus. These algorithms are not the most

Table 5. $S_{Similar}$ and $S_{Dissimilar}$ of URL Embedding Methods

	$S_{Similar}$	$S_{Dissimilar}$
Word2Vec/Skip-gram	0.88807	0.67008
Word2Vec/CBOW	0.95354	0.87822
FastText/Skip-gram	0.89620	0.70631
FastText/CBOW	0.98454	0.95189
GloVe	0.92190	0.73583
TF-IDF	0.81473	0.78621
One-hot Code	0.70278	0.65098

suitable for URL embedding because the relative position of Tokens in URL is different from natural language. Besides, the treatment of polysemous Tokens is also unsatisfactory, like the Token ‘zoom’, when it is used as the domain name ‘zoom’, its meaning is different from that of other domain names as part of the path, which makes the cosine similarity of Tokens related to ‘zoom’ very poor. In conclusion, URL embedding methods need to solve the above problems to obtain a better embedding performance.

8 Conclusion

In this paper, we proposed an intrinsic evaluation method for URL embedding method, and it can evaluate URL embedding method without the effect of machine learning models and data sets. Besides, we evaluated several URL embedding methods with intrinsic and extrinsic method and found that the results of traditional extrinsic evaluation methods are hard to compare in evaluating URL embedding methods, and the results of intrinsic evaluation method proved intrinsic evaluation method plays its role in URL embedding methods evaluation. At last, we found that Word2Vec embedding method and Skip-gram algorithm are suitable for URL embedding according to the results of the evaluation. s

Acknowledgements. This work was supported by JSPS KAKENHI Grant Number JP22H03588.

References

1. Top sites - alexa. <https://www.alexacom/topsites>
2. Urlhaus — malware url exchange. <https://urlhaus.abuse.ch/>
3. Chen, Q., Omote, K.: A three-step framework for detecting malicious URLs. In: 2022 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6. IEEE (2022)
4. Goldberg, Y., Levy, O.: word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method. CoRR **abs/1402.3722** (2014). <http://arxiv.org/abs/1402.3722>
5. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: Fasttext. zip: compressing text classification models. arXiv preprint [arXiv:1612.03651](https://arxiv.org/abs/1612.03651) (2016)
6. Kaneko, S., Yamada, A., Sawaya, Y., Thao, T.P., Kubota, A., Omote, K.: Detecting malicious websites by query templates. In: Simion, E., Géraud-Stewart, R. (eds.) Innovative Security Solutions for Information Technology and Communications. LNCS, vol. 12001, pp. 65–77. Springer, Cham (2020)
7. Ke, G., et al.: Lightgbm: a highly efficient gradient boosting decision tree. Adv. Neural. Inf. Process. Syst. **30**, 3146–3154 (2017)
8. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
9. Rajaraman, A., Ullman, J.D.: Data Mining, pp. 1–17. Cambridge University Press, Cambridge (2011). <https://doi.org/10.1017/CBO9781139058452.002>

10. Ho, T.K.: Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, pp. 278–282 (1995). <https://doi.org/10.1109/ICDAR.1995.598994>
11. Wang, B., Wang, A., Chen, F., Wang, Y., Kuo, C.C.J.: Evaluating word embedding models: methods and experimental results. *APSIPA Trans. Sig. Inf. Process.* **8**, e19 (2019)
12. Yuan, H., Yang, Z., Chen, X., Li, Y., Liu, W.: Url2vec: URL modeling with character embeddings for fast and accurate phishing website detection. In: 2018 ISPA/IUCC/BDCLOUD/SocialCom/SustainCom, pp. 265–272 (2018). <https://doi.org/10.1109/BDCLOUD.2018.00050>



Key Management Based on Ownership of Multiple Authenticators in Public Key Authentication

Kodai Hatakeyama, Daisuke Kotani^(✉), and Yasuo Okabe

Kyoto University, Sakyo, Kyoto 6068501, Japan
hatakeyama@net.ist.i.kyoto-u.ac.jp, kotani@media.kyoto-u.ac.jp,
okabe@i.kyoto-u.ac.jp

Abstract. Public key authentication (PKA) has been deployed in various services to provide stronger authentication to users. In PKA, a user manages private keys on her devices called authenticators, and registers public keys to services. To protect private keys, authenticators are usually designed not to export private keys outside. Nowadays, a user has multiple authenticators like PCs and smartphones, and struggles to manage multiple public keys in many services every time she starts to use new services and replaces some of her authenticators. To ease the burden, we propose a mechanism where users and services manage public keys based on the owner of authenticators and users can access services with PKA using any of their authenticators. We introduce a key pair called an Ownership Verification Key (OVK) consisting of the private key (OVSK) and the public key (OVPK). All authenticators owned by a user derive the same OVSK from the pre-shared secret. Services verify the ownership of the authenticators using the OVPK to decide whether binding the requested public key to her account. To protect user privacy, authenticators generate a unique OVK for each service. We implement the Proof-of-Concept, show that our proposal achieves some security goals, and discuss how to mitigate threats not completely handled.

Keywords: Public Key Authentication · Key Management · Authenticator

1 Introduction

Public key authentication (PKA) is attracted as an alternative or complement way of password authentication. PKA assumes that only a user has a private key, and a service has the corresponding public key. The service authenticates the user by asking the user to sign a random string called a challenge and verifying the signature sent from the user. PKA is regarded as stronger than password authentication because: (1) only the user has a key to make a signature so it is resistant to data breaches of services. (2) PKA with proper mechanism such as Webauthn [11] can systematically check a service that requests authentication is the same as the

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 361–375, 2024.

https://doi.org/10.1007/978-3-031-56326-3_26

service the user has already registered. (3) the keys are automatically generated so the users can avoid using weak keys.

Services have to bind public keys to a user's account. Services verify the correctness of this binding based on several models, where relying on trusted third parties (e.g., WebPKI [2]) and where being provided directly via trusted channels (e.g., FIDO [7]). This study focuses on the latter.

Users have to securely store private keys, and authenticators are often used for this purpose. Authenticators, such as Yubikey [24] and Keychain [1], store key pairs in secure storage so that private keys cannot be exported nor easily accessed by the outside of the authenticator.

It is important for services to verify that private keys are securely stored, and attestation is a way for an authenticator to prove that it has processed with a key inside its secure storage [20]. Attestations are signed by the attestation key embedded in an authenticator by its manufacturer, so services can evaluate the trustworthiness of the received public key and the authenticator that stores the corresponding private key by verifying the attestation. Some mechanisms like Webauthn [11] require authenticators to generate different key pairs for each service to avoid collusion of accounts by collusion with multiple services.

Users should use authenticators storing private keys corresponding to registered public keys when accessing services. However, users have multiple authenticators such as smartphones and PCs, and manage multiple public keys in each service, which annoy users [10]. Passkeys [8] tries to solve this problem by making private keys exportable, but this invalidates the attestation.

To ease the burden on users, we aim to enable users to use PKA with any owned authenticators without explicitly registering each public key. To realize this, we propose the mechanism where users and services manage public keys based on the owner of authenticators storing the corresponding private keys. We introduce a key pair called an Ownership Verification Key (OVK) derived from a pre-shared seed in all authenticators owned by a user. A user proves the ownership of authenticators by the private key of an OVK (Ownership Verification Secret Key; OVSK). A service verifies the possession of the authenticators by the public key of the OVK (Ownership Verification Public Key; OVPK), and binds a public key for authentication with OVPK. For pseudonymity, authenticators generate a unique OVK for each service. When updating a set of authenticators, a user updates an OVSK, and services update an OVPK bound to her accounts.

We implemented the Proof of Concept, and evaluated what measures our proposal takes against the threats found with threat modeling. We confirmed that our proposal achieves some security goals such as preventing correlation of accounts and correctly binding public keys to accounts.

2 Related Work

Nishimura [18] proposes sharing private keys among authenticators that users own. This approach weakens the security of private keys because authenticators export private keys from secure storage, thus services cannot trust attestations by the authenticators. James [3] introduces certificate chains to FIDO public

keys so that users can register multiple authenticators and recover accounts with FIDO. Services verify the owner of the public key by certificates issued by the trusted third party. This approach relies on an trusted third party, so the authentication becomes untrusted when the trusted third party is compromised.

Oogami [19] proposes the mechanism in which users register a new FIDO public key of an authenticator via authenticated sessions established by the previously registered public key. This requires users to keep multiple authenticators at the same time when registering a new public key, thus reduces usability. Frymann [9] and Lundberg [14] propose a mechanism for account recovery when losing registered authenticators that users use daily. The main authenticator (for daily use) receives the seed for deriving public keys from the backup authenticator in advance, generates a different public key of the backup authenticator for each service, and registers the public key whose corresponding private key the backup authenticator can only derive. Services cannot verify the attestation of the public key of the backup authenticator during registration.

3 Key Management with an Ownership Verification Key

3.1 Overview

We propose the mechanism where a user and a service manage keys for authentication based on a public key cryptographic key pair called an Ownership Verification Key (OVK). An OVK is derived by all authenticators of a user to prove that the private key corresponding to the public key to be registered is stored in her owned authenticator. The public key of the OVK (Ownership Verification Public Key; OVPK) is registered with the service via the trusted channel when registering a new account. The service binds the OVPK to her account. The private key of the OVK (Ownership Verification Secret Key; OVSK) is used for signing the public key to be registered. The service binds the public key verified by the OVPK to her account.

Figure 1 illustrates how a user registers a public key when she has two authenticators (AuthA and AuthB). She shares an OVSK among AuthA and AuthB in advance. When she registers a new account using AuthA, she sends a public key in AuthA and an OVPK to the service. Then she registers a new public key for AuthB by signing the public key with the OVSK whose corresponding OVPK has been already registered with AuthA. The service verifies the signature by the registered OVPK and, if succeeded, binds the public key to her account.



Fig. 1. Registering Public Keys Using An OVK

Users and services update OVKS according to the lifecycles of users’ authenticators. When a user changes a set of her authenticators, she updates an OVK in her all authenticators and notifies services of updating the OVPK. To make an update message, registered authenticators sign the new OVPK by the previous OVSK whose corresponding OVPK has been registered with services. Services update an OVPK bound to her account based on the most trustworthy update message, re-bind public keys to her account based on the new OVPK, and revoke the public keys not bounded to the new OVPK.



Fig. 2. Updating an OVK

Figure 2 shows how a user updates an OVK when she has registered two authenticators (AuthA and AuthB) with a service and replaces AuthA with Authenticator C (AuthC) because of losing AuthA. First, she shares a new OVSK between AuthB and AuthC. Then, AuthB derives the new OVK (OVK^2), signs the new OVPK ($OVPK^2$) by the previous OVSK ($OVSK^1$) to make an update message, and notifies the service of $OVPK^2$ by sending the update message. The service evaluates the message received from AuthB as the most trustworthy, and binds $OVPK^2$ and the public key for AuthB to her account. It also revokes the public key for AuthA because AuthA has sent no messages. Then she can register a new public key for AuthC by signing the public key with the $OVSK^2$.

3.2 Deriving an OVK from a Shared Secret

This section describes how to derive an OVK from the pre-shared secret, called the seed, and how to register public keys using an OVK. We assume that the seed has been shared among all authenticators owned by the same user.

Requirement. Our proposal does not interfere with what PKA described in Sect. 1 can achieve during public key registration [6, 21], as follows.

First, our proposal should not rely on trusted third parties for proving the owner of authenticators except for verifying attestations and establishing secure channels. Users can register public keys via a trusted channel established when registering a new account or established by registered authenticators.

Second, our proposal must prevent services from correlating their accounts by using the proof of the owner of authenticators. Users can register different public keys with each service to protect user privacy against services seeking to correlate their accounts based on registered public keys.

Third, services should verify the attestation of the public key requested to be registered to confirm that the authenticator has the corresponding private key. Services calculate the trustworthiness of the public key by this verification.

Finally, our proposal should minimize the number of times a user operates multiple authenticators at the same time for convenience.

Deriving an OVK. Figure 3 shows how to derive and register an OVK when a user has two authenticators, Authenticator A (AuthA) and B (AuthB). A user registers a new account with service α using AuthA and then she accesses the service with AuthB. Note that we assume that messages between authenticators and the service have protected by trusted channels (e.g., via TLS).

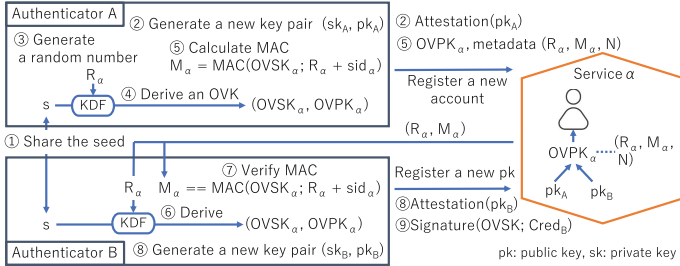


Fig. 3. Deriving an OVK from the shared seed

The two authenticators agree in advance on the following parameters: s (the seed value shared among authenticators (1)), N (the number of authenticators sharing the same seed), the public key cryptographic algorithm that accepts any random value as a private key for OVK, KDF (the key derivation function that takes a seed and a random value as inputs and outputs pseudorandom numbers for an OVSK), MAC (the message authentication code function taking an OVSK as a key), and the identifier of service α (sid_α).

First, the user registers a new account with service α using AuthA. AuthA generates a new key pair (sk_A, pk_A) and an attestation of the public key (2 in Fig. 3). At the same time, AuthA derives an OVK and the corresponding metadata and registers them in addition to the public key (pk_A) with service α . The derivation consists of the following three steps.

- 3 Generate a random number (R_α) .
- 4 Calculate $OVSK_\alpha = \text{KDF}(s, R_\alpha)$ and the corresponding $OVPK_\alpha$. If the authenticator cannot derive valid OVSK using R_α , start over from 3.
- 5 Register $OVPK_\alpha$ and the corresponding metadata consisting of $R_\alpha, M_\alpha = \text{MAC}(OVSK_\alpha, R_\alpha + sid_\alpha)$ and N .

The service receives and binds the public key (pk_A), $OVPK_\alpha$, and the corresponding metadata (R_α, M_α, N) to the new account.

Second, the user access service α using unregistered AuthB. The service returns a challenge for PKA and the metadata $(R_\alpha$ and $M_\alpha)$ in reply to an authentication request. AuthB tries to register a new public key because it has no public key for signing in to service α . AuthB generates a new key pair (sk_B, pk_B) and the attestation of the public key (8 in Fig. 3), then signs the public key

(pk_B) by $OVS K_\alpha$ so that service α verifies the owner of AuthB storing the corresponding private key (pk_B) is the same as the owner of registered authenticators. AuthB derive $OVS K_\alpha$ from the received metadata by the following two steps.

- ⑥ Derive $OVS K_\alpha$ and $OV P K_\alpha$ using R_α in the same way as ④.
- ⑦ Verify the received metadata M_α using $OVS K_\alpha$. If the verification failed, the derived OVS K or the received metadata is not for service α .

Service α binds the public key (pk_B) if the attestation (⑧) and the signature (⑨) is valid and the number of the registered public keys is not more than N .

Unique OV Ks per Service. Authenticators can derive different OV Ks per service by generating different random numbers (R) per service. Note that it is impossible to compute the seed value of an OVSK because of the properties of a key derivation function (KDF). So, authenticators can register unlinkable OV PKs with different services. Random numbers are stored to services in a verifiable format, so authenticators only need to remember the seed.

3.3 Sharing a Seed Among Authenticators

Requirement. A user operates multiple authenticators and makes them communicate to share a seed. There are various short-range communication protocols (e.g., Bluetooth, generating and reading QR codes), each of which has its different security features. So, we assume no security features of communication channels. This requires that attackers cannot calculate a seed using data obtained by eavesdropping. Authenticators also need to verify the received data is generated by the legitimate authenticator for resistance to tampering.

Method. Figure 4 shows the case where a user has two authenticators, Authenticator A (AuthA) and B (AuthB). AuthA and AuthB agree on the same seed based on the Diffie-Hellman key exchange protocol. They encrypt DH public keys using an authenticated encryption based on a password set by the user to ensure the confidentiality of DH public keys and verify the authenticity.

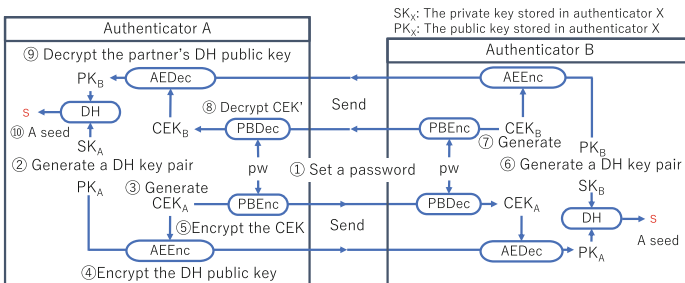


Fig. 4. Sharing a seed between two authenticators

AuthA and AuthB agree on the following parameters in advance: pw (password set by the user (①)), DH (a Diffie-Hellman key exchange protocol), a list of

authenticated encryption algorithms (and identifiers) for encrypting DH public keys, and a list of password-based encryption algorithms (and identifiers).

First, the user operates AuthA as follows, and AuthA sends the generated ciphertexts and the algorithm identifiers (at ④ and ⑤) to AuthB.

- ② Generate a DH key pair (SK_A, PK_A) .
- ③ Generate a random number called a Content Encryption Key (CEK_A) .
- ④ Encrypt the DH public key (PK_A) using CEK_A with the authenticated encryption algorithm in the list.
- ⑤ Encrypt CEK_A using pw with the password-based algorithm in the list.

In the same way, AuthB generates a DH key pair $((SK_B, PK_B)$ at ⑥) and a random number $(CEK_B$ at ⑦), encrypts the DH public key (PK_B) using CEK_B , and encrypts CEK_B using pw .

AuthA receives the ciphertexts from AuthB, then

- ⑧ Decrypt a received CEK_B using pw with the password-based algorithm selected by AuthB.
- ⑨ Decrypt a received PK_B using the decrypted CEK_B with the authenticated encryption algorithm selected by AuthB.
- ⑩ Agree the same seed using the Diffie-Hellman key exchange protocol.

Even if the user has three or more authenticators, authenticators can share a seed in a similar way with n-party Diffie-Hellman key exchange protocol [23].

3.4 Verifying the Trustworthiness of an OVK

Since service binds public keys to an account by an OVK, the trustworthiness of public keys can never be higher than the trustworthiness of the OVK. A service can evaluate the trustworthiness of an OVK using the following two criteria.

Criterion1. An OVK is derived as described in Sect. 3.2

Criterion2. A seed is securely stored in all authenticators

The evaluation mechanism depends on the attestation mechanism that authenticators already have. Authenticators send an OVPK as well as the attestation of the OVPK at ⑤ on Fig. 3.

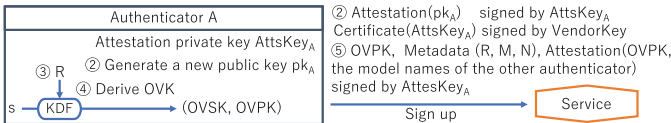


Fig. 5. Sending the attestation of an OVPK

Figure 5 details attestations of public keys in the registration of an OVPK in Fig. 3. The manufacturer embeds an attestation private key $AttsKey_A$ in an authenticator, and issues a certificate for the attestation public key $Certificate(AttsKey_A)$. The authenticator signs the public key generated at ② and the information about the public key by $AttsKey_A$ and sends them to a service.

The authenticator also signs a derived OVPK by $AttsKey_A$ to notify the service that the OVPK is derived from the seed stored in the authenticator as described in Sect. 3.2. The service verifies the attestation of the OVPK based on the trusted policy about what authenticators comply with Sect. 3.2. In this way, the service can validate Criterion1.

An attestation of an OVPK contains the OVPK itself and model names of the other authenticators sharing the same seed. The model names are shared while sharing the seed at ② on Fig. 4. A service verifies whether they store the seed securely using the trusted policy about what authenticator model stores the seed in the secure storage. In this way, the service can validate Criterion2.

3.5 Re-sharing a New Seed and Updating an OVK

This section describes how a user revokes an OVPK registered with a service and updates a new OVPK in the service in the case of losing authenticators.

We assume that attackers can operate the seed and the private keys corresponding to registered public keys stored in a stolen authenticator. However, attackers cannot immediately use the stolen authenticator because authenticators protect the seed and private keys by local authentication like PIN or biometric.

Overview. Authenticators share a new seed as described in Sect. 3.3. They hold both the previous seed and a new seed. They notify a service of updating an OVK by sending an update message when a user signs in for the first time after re-sharing the new seed. The service waits for some period (OVK migration period) and accepts the new OVPK from the most trustworthy update message. A service calculates the trustworthiness of each update message, and re-binds public keys to the user’s account by verifying with the new OVPK and revokes the public key bound only to the previous OVPK.

An Update Message for a New OVK. Figure 6 shows how an authenticator generates an update message for a new OVK.

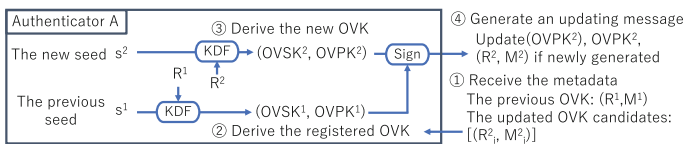


Fig. 6. Generating an update message for a new OVK

① The authenticator receives two kinds of metadata from the service to derive OVKs from seeds, one is (R^1, M^1) for the previously registered OVK^1 , and the other is $([(R_i^2, M_i^2)])$ for OVK^2 candidates that other authenticators have registered as new OVKs. The second one is a list because attackers can

also generate a malicious update message derived from a seed by using a stolen authenticator.

- ② The authenticator verifies OVK^1 using (R^1, M^1) and the previous seed (s^1) as described in Sect. 3.2. If not verified, abort this update process.
- ③ The authenticator derives OVK^2 from the received metadata $([(R_i^2, M_i^2)])$ and the new seed (s^2). If the metadata list is not empty, the authenticator derives the new OVK from the legitimate metadata $((R^2, M^2) = (R_i^2, M_i^2))$ with which the authenticator can verify the MAC value $(M_i^2 == MAC(s^2, R_i^2 + sid))$. If the metadata list is empty or the metadata has no legitimate metadata, the authenticator generates a new random value (R^2) and then derives a new OVK.
- ④ The authenticator signs $OVPK^2$ by $OVSK^1$ corresponding to the previously registered $OVPK^1$, and sends the signature as an update message.

We assume that authenticators have two shared seeds. However, authenticators may have more than two seeds because users change their authenticators many times. Even in this case, authenticators can also generate the correct update message. As the seed corresponds to registered OVPK, authenticators can select the correct seed by validating the MAC value of the received metadata.

Evaluating the Trustworthiness of an Update Message. When a service receives an update message from a registered authenticator, it enters the OVK migration period. In this period, the service ignores a new public key by the registered OVK. If the same update message comes from more than half of the registered authenticators, the service trusts the message. Otherwise, the update message sent from the most registered authenticators, or the earliest received message if two or more such update message has been received is trusted at the end of the period.

4 Proof of Concept Implementation

We implement the Proof of Concept (PoC)¹ using JavaScript. One browser window is treated as one authenticator in order to emulate multiple authenticators on one device. Note that the PoC stores seed, private keys, and the attestation key in not secure storage. Figure 7 is an data flow diagram for the PoC.

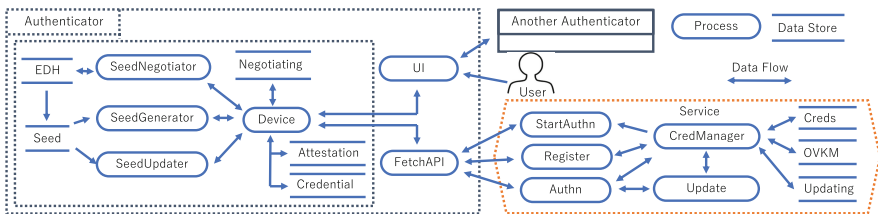


Fig. 7. Data Flow Diagram in PoC

¹ The code is available on <https://github.com/hatake5051/ovk-poc>.

Authenticator: `SeedGenerator` implements Sect. 3.2. `Seed` stores shared seeds that are 256 bits long. The algorithm of an OVK is elliptic curve cryptography [4] with `secp256r1` [5]. An OVK is calculated using a KDF output as pseudorandomly selected an integer of Sec.3.2.1 in [4]. The key derivation function (KDF) and the MAC function (MAC) are HMAC [13] using SHA-256. A service identifier is the origin of the service URL.

`SeedNegotiator` implements Sect. 3.3 except for encrypting and decrypting a DH public key by a CEK and a CEK by a password, and sending and receiving ciphertexts. `Device` implements these exceptions. `SeedNegotiator` stores an ephemeral private key for the DH key exchange protocol in `EDH`.

`EDH` stores the seed calculated as a result of the key exchange in `Seed`. The key exchange protocol (DH) is ECDH key exchange protocol [4] with `secp256r1` [5]. The authenticated encryption algorithm is AES-128-GCM [16] [17]. The password-based encryption algorithm is PBES2 [15] using AES-KW [12] and SHA-256. `SeedUpdater` implements Sect. 3.5.

`Device` generates key pairs and attestations and manages them based on an OVK. `Credential` stores generated key pairs. `Attestation` stores the attestation private key and the certificate of the corresponding attestation public key. `Negotiating` stores data used for Sect. 3.3 like a password. `Device` generates an attestation for public keys stored in `Credential`, OVPKs generated by `SeedGenerator`, and DH public keys calculated by `SeedNegotiator`.

UI transmits ciphertexts generated by `Device` with other authenticators and interacts with a user. We use QR codes as the communication channel among authenticators for Sect. 3.3. `FetchAPI` communicates with a service by TLS.

Service: `StartAuthn` accepts authentication requests from a user. It receives the account name and responds with a challenge bound to the account and, if registered, public keys, an OVPK, and the metadata of OVPK. `Register` accepts requests to register a new account and a new public key bound to an account as described in Sect. 3.2. `Authn` accepts challenge responses for authentication and update messages as described in Sect. 3.5. We use ECDSA [4] with `secp256r1` [5]. `CredManager` manages the bindings of public keys and OVPKs to accounts. `Creds` stores public keys bound to OVPKs. `OVMK` stores OVPKs and the corresponding metadata bound to accounts. `Update` handles updating OVKs. `Updating` stores update messages.

5 Evaluation with Threat Analysis

We evaluate our proposal by analyzing the PoC in Sect. 4 using threat modeling [21,22].

5.1 Security Requirement

The following assets should be protected in addition to assets for establishing a secure channel between authenticators and services.

1. Private keys stored in authenticators
2. Public keys managed by services
3. Attestation key stored in each authenticator
4. Certificate for attestation keys managed by each authenticator manufacturer
5. Trusted root certificates and policy for services to validate attestations
6. Seed stored in authenticators
7. OVPK and corresponding metadata stored in services
8. Ephemeral DH private key generated by authenticators for sharing a seed
9. Temporary password stored in authenticators for sharing a seed

The following security goals should be achieved.

SG-1 Services can authenticate users based on PKA.

SG-2 Services cannot correlate their accounts.

SG-3 Services can bind public keys to legitimate accounts.

SG-4 Services and authenticators can validate public keys by attestations.

SG-5 Be resilient from attempting to modify intercepted communications to masquerade as legitimate users.

We make the following assumptions where our proposal works.

SA-1 The processes and data stores surrounded by the trusted boundary are isolated from other processes on the authenticator. The authenticator requires local authentication before accessing these processes and data stores.

SA-2 The cryptographic algorithms achieves the objectives of each algorithm.

SA-3 A service correctly validates the certificate chain of attestations.

SA-4 A service and an authenticator establish a secure channel for service authentication, confidentiality and integrity for messages (like TLS).

5.2 Threat Analysis

We list the threats on the data flow diagram in Fig. 7 focusing on data flows across trusted boundaries shown by dotted lines, and explain goals listed in Sect. 5.1 each threat violates and measures our proposal takes.

Authenticator. Threats between a user and UI include the following.

Homograph Mis-Registration. A malicious service pretends a legitimate service. It prompts the user to register a new public key, and sends metadata stolen from other services. The malicious service correlates OVPKs by whether the user requests a new public key registration or not. This violates SG-2. Our proposal mitigates this threat because authenticators verify the MAC value of the received metadata including the identifier of the service (trusted by SA-4) that the authenticator communicates.

User Verification By-pass. An attacker can operate the authenticator, or an attacker can bypass the local authentication of the authenticator to operate it.

This threat is against SA-1, so we do not have to consider it. In Sect. 6.2, we discuss the case where SA-1 is not satisfied during the OVK update.

Threats between a service and `FetchAPI` include the following.

Service Verification Error. The authenticator cannot properly authenticate services, thus it cannot correctly identify services. As a result, an attacker can eavesdrop and tamper with the communication channel. This threat is against SA-4, so we do not have to consider it.

Threats between another authenticator and UI include the following.

Malicious Authenticator Linking. An attacker's authenticator participates in sharing a seed. As a result, the attacker gets the seed value itself, and can use this authenticator to register a new public key with any service that a legitimate user has registered. This violates SG-3. Our proposal addresses this threat because a user protects sharing a seed with a password.

Weak Authenticator. A user allows a weak authenticator to participate in sharing a seed. The weak authenticator does not securely protect a seed, so, when an attacker compromises the weak authenticator, the seed may be leaked. This violates SG-3 because the attacker can register a new public key of the attacker's authenticator by generating the OVK with the compromised seed. Our proposal addresses this threat because each authenticator validates the security properties of other authenticators through attestations when sharing a seed.

Threats between `Device` and UI include the following.

Malicious Authenticator. A user uses a malicious authenticator. Because a user cannot rely on the malicious authenticators, this threat violates any goals. Our proposal addresses this threat because services maintain a list of attestation certificates of trusted manufacturers.

Threats to the authenticator include the following.

Side Channel Attack. Access to the data store that is not described in Fig. 7 compromises assets to be protected. This threat is against SA-1, so we do not have to consider it.

Bad Cryptography Primitives. An authenticator uses a compromised cryptographic algorithm or a weak pseudo-random number generator in the process. This threat is against SA-2, so we do not have to consider it.

Service. Threats between an authenticator and `Authn` include the following.

Updating Malicious OVPK. An attacker updates to an OVPK derived from the seed held by his authenticator. This violates SG-3 because the attacker can register his public keys, and SG-1 because the attacker can revoke the user's public keys. Our proposal addresses this threat because an attacker cannot know the seed corresponding to the registered OVPK. Section 6.2 discuss the case where this SA-1 assumption is not satisfied during updating an OVK.

Threats between an authenticator and `Register` include the following.

Malicious Authenticator Registration. An attacker registers a new public key of his authenticator to a legitimate user account. This violates SG-3. Our proposal addresses this threat because an attacker cannot get the OVSK corresponding to the OVPK bound to the account, and the seed corresponding to the OVPK. The service can verify that trusted authenticators store a seed and OVSKs by verifying OVPK attestations. Even if a seed is compromised, the number of authenticators that can be registered is limited, so an attacker cannot register his public keys after a user registers public keys of her all authenticators.

Threats between an authenticator and `StartAuthn` include the following.

Linking OVPK. An attacker receives OVPKs and metadata from many services, and derives corresponding OVSKs or check OVPKs are derived from the same seed. This threat is against SA-2, so we do not have to consider it.

6 Discussion

6.1 Sharing a Seed

A user is assumed to enter a password directly into each authenticator so that the password does not flow on the communication channel where authenticators share a seed. A long password allows an attacker to take an extremely long time to decrypt ciphertexts and a secure encryption algorithm prevents him from compromising ciphertexts, so it is difficult for the attacker to participate in sharing a seed before completed. Besides, a DH key exchange protocol is secure against eavesdropping, so an attacker cannot compromise a seed.

6.2 Updating an OVK

When updating the OVK, a service treats the number of registered authenticators sending the same update message as an indicator of the trustworthiness of the update message. We regard that the trustworthiness of all registered authenticators before the OVK migration period is equal, because it is difficult for a service to know whether an authenticator is stolen or held by a legitimate user. Due to the assumption that an attacker needs time to gain control of a stolen authenticator (in Sect. 3.5), a service selects the earlier sent message when two or more update messages have the same and most trustworthiness.

We discuss what attacks the proposed method prevents when an attacker can operate the seed and the private key with a stolen authenticator. When more than two authenticators has been registered, the legitimate user can prevent an attacker from updating an OVK only if the legitimate user still has more than half of the registered authenticators. When a user and an attacker has registered the same number of authenticators, the service trust the update message received earlier, so it is important for the user to send the update message immediately. In other cases, an attacker may successfully update the OVK, so the service needs to provide alternative way to recover the OVK.

7 Conclusion

We introduce a key pair called an Ownership Verification Key (OVK) and propose the mechanism where users and services manage public keys based on the owner of authenticators storing the corresponding private keys. The mechanism allows users to access services with any of their authenticators without registering each of their public keys explicitly. A user can derive the private key of an OVK (OVSK) on her authenticators from the seed sharing among the authenticators. A service binds the public key of OVK (OVPK) and public keys signed by an OVSK to the user's account bound to the corresponding OVPK. When a user changes a set of her authenticators, she updates an OVSK, and a service updates an OVPK binding to her accounts based on the most trustworthy update message. We implemented the Proof of Concept and evaluated measures our proposal takes against the threats through threat modeling. We confirmed that our proposal achieves some security goals, such as that services cannot correlate accounts and can correctly bind public keys to accounts. We discussed how our proposal mitigates threats for which measures are not sufficient.

Future work includes formal verification of cryptographic operations and improvement of calculating trustworthiness of update messages.

References

1. Apple: Keychain data protection (2021). <https://support.apple.com/guide/security/keychain-data-protection-secb0694df1a/web>
2. Boeyen, S., Santesson, S., et al.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (2008)
3. Conners, J.S., Zappala, D.: Let's authenticate: automated cryptographic authentication for the web with simple account recovery. In: WAY 2019, pp. 1–6 (2019)
4. Daniel, R.L.: Brown: SEC 1: elliptic curve cryptography. Technical report, Standards for Efficient Cryptography Group (2009)
5. Daniel, R.L.: Brown: SEC 2: recommended elliptic curve domain parameters. Technical report, Standards for Efficient Cryptography Group (2010)
6. FIDO Alliance: FIDO: Fast Identity Online Alliance Privacy Principles Policy (2021). <https://media.fidoalliance.org/wp-content/uploads/2021/02/FIDO-Privacy-Principles.pdf>
7. FIDO Alliance: How FIDO Works (2021). <https://fidoalliance.org/how-fido-works/>
8. FIDO Alliance: Passkeys (2022). <https://fidoalliance.org/passkeys/>
9. Frymann, N., Gardham, D., et al.: Asynchronous remote key generation: an analysis of Yubico's proposal for W3C WebAuthn. In: ACM CCS 2020, pp. 939–954 (2020)
10. Ghorbani Lyastani, S., Schilling, M., et al.: Is FIDO2 the Kingslayer of user authentication? A comparative usability study of FIDO2 passwordless authentication. In: IEEE S&P 2020, pp. 268–285 (2020)
11. Hodges, J., Jones, J., et al.: Web Authentication: An API for accessing Public Key Credentials Level 2 (2021). <https://www.w3.org/TR/webauthn/>
12. Housley, R., Schaad, J.: Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394 (2002)

13. Krawczyk, D.H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (1997)
14. Lundberg, E., Nilsson, D.: Asynchronous delegated key generation without shared secrets (DRAFT) (2021). <https://github.com/Yubico/webauthn-recovery-extension>
15. Moriarty, K., Kaliski, B., Rusch, A.: PKCS #5: Password-Based Cryptography Specification Version 2.1. RFC 8018 (2017)
16. National Institute of Standards and Technology: Advanced Encryption Standard (AES). Technical report (2001)
17. National Institute of Standards and Technology: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. Technical report (2007)
18. Nishimura, H., et al.: Secure authentication key sharing between personal mobile devices based on owner identity. *J. Inf. Process.* **28** (2020)
19. Oogami, W., et al.: Multiple Authenticators for Reducing Account-Recovery Needs for FIDO-Enabled Consumer Accounts (2020). <https://fidoalliance.org/white-paper-multiple-authenticators-for-reducing-account-recovery-needs-for-fido-enabled-consumer-accounts/>
20. Powers, A.: FIDO TechNotes: The Truth about Attestation (2018). <https://fidoalliance.org/fido-technotes-the-truth-about-attestation/>
21. Rolf Lindemann: FIDO Security Reference (2018). <https://fidoalliance.org/specs/fido-v2.0-id-20180227/fido-security-ref-v2.0-id-20180227.html>
22. Shostack, A.: *Threat Modeling: Designing for Security*, 1st edn. Wiley, Hoboken (2014)
23. Steiner, M., Tsudik, G., Waidner, M.: Diffie-Hellman key distribution extended to group communication. In: *ACM CCS 1996*, pp. 31–37 (1996)
24. Yubico: Yubico Product Documentation (2021). <https://docs.yubico.com/>

Author Index

B

Berki, Georgia 163
Boudko, Svetlana 102
Bracamonte, Vanessa 321
Bruckschen, Lilli 1

C

Cabral, Kalil 265
Čejka, Tomáš 307
Čeleda, Pavel 293
Chadwick, David W. 265
Chatzoglou, Efstratios 250
Chen, Jinfeng 207
Chen, Qisheng 350
Chen, Zhenxiang 87
Cheng, Guang 207, 335
Cuppens, Frédéric 59
Cuppens, Nora 59

D

Delot, Thierry 191

E

Eskeland, Sigurd 102

F

Fang, Liming 87
Federrath, Hannes 236
Fischer, Mathias 73

G

Gallais, Antoine 191
Garcia-Alfaro, Joaquin 59
Giraud, Robin 191
Grammatikakis, Miltos 163

H

H. Nguyen, Nhung 30
Hamm, Peter 45
Hatakeyama, Kodai 361
Holm, Hannes 177
Hu, Xiaoyan 207, 335

I

Imine, Youcef 191

J

Jin, Yan 117
Jungebloud, Tino 30

K

Kalutarage, Harsha 279
Kambourakis, Georgios 250
Kampourakis, Vyron 250
Karapapas, Christos 149
Kesdoğan, Doğan 131
Kikuchi, Hiroaki 15
Komiya, Chika 279
Kornaros, George 163
Kotani, Daisuke 361

L

Laborde, Romain 265
Liu, Yuanmu 117
Liu, Zhe 87
Löbner, Sascha 321
Lu, Anting 335
Luo, Hao 335

M

Maramara, April Rains Reyes 265
Morales, David A. Cordova 265

O

Okabe, Yasuo 361
Omote, Kazumasa 350

P

Pape, Sebastian 45, 321
Patsakis, Constantinos 149
Polyzos, George C. 149

R

Radloff, Matz 73
Rajapaksha, Sampath 279
Rannenber, Kai 45
Reuben, Jenni 177

S

Sadlek, Lukáš 293
Saint-Hilaire, Kéren 59
See, August 73, 236
Semnont, Alexis 191
Senanayake, Janaka 279
Seong Kim, Dong 30
Soukup, Dominik 307
Soumelidou, Aikaterini 222

T

Tauchert, Sebastian 1
Touré, Almamy 191
Tsohou, Aggeliki 222

U

Ufer, Max Jens 1
Uhřiček, Daniel 307

V

Vašata, Daniel 307

W

Wang, Suyue 207
Wazan, Ahmad Samer 265
Wichmann, Pascal 236
Wingarz, Tatjana 73
Wittig, Maximilian 131
Wu, Hua 207, 335

X

Xu, Jun 87

Y

Yanai, Naoto 279
Yang, Mengqing 117
Ye, Chunming 117
Ye, Chunxiao 117

Z

Zemanek, Sven 1
Zhao, Chuan 87
Zhao, Shengnan 87
Zimmermann, Armin 30