Formal Verification of Security Protocols: ProVerif and Extensions

Jiangyuan Yao[1], Chunxiang Xu[1], Deshun Li[1(✉)], Shengjun Lin[1], and Xingcan Cao[2]

[1] Hainan University, Haikou 570228, China
lideshunlily@qq.com
[2] University of British Columbia, Vancouver, BC V6T 1Z1, Canada

Abstract. Secure protocols are built on cryptographic algorithms, which provide a variety of secure services to realize secure communications in a network environment. To improve the quality of security protocols and ensure their reliability, sufficient verification and testing are required. ProVerif is a classic formal verification tool for security protocols, and we describe its working mechanism and verification process in detail. In this paper, we focus on ProVerif and extensions in the verification of security protocols. We introduce some representative solutions to illustrate verification with ProVerif. And we also introduce its extension tools for protocols with stateful properties, protocols with algebraic properties, and protocol implementations, then summarize the general method of ProVerif extension tools. Finally, we discuss possible future research points, including the extension of ProVerif for protocols that combined stateful and algebraic properties, verification of security applications in SDN networks, and building models from protocol implementations without source code.

Keywords: ProVerif · Security protocols · Security properties · Verification · Pi calculus

1 Introduction

The security protocol [2] is a message exchange protocol based on cryptography, which aims to provide various security services in the network environment. It is an important part of network security, we need to use security protocol to authenticate among entities, securely key distribution or other secrets among entities. Vulnerabilities in security protocols may cause malicious attacks or information disclosure. To improve the service performance of security protocols, sufficient verification and testing for security protocols are required before deployment. Formal methods are an important method for discovering design defects of security protocols. Formal methods in security protocols have been made a lot of progress [7,9,13–15,17,29] in the field of protocol verification and testing.

To verify security protocols, researchers propose many verification methods and develop verification tools, such as ProVerif [8,10]. ProVerif is a formal tool for

X. Sun et al. (Eds.): ICAIS 2022, LNCS 13339, pp. 500–512, 2022.
https://doi.org/10.1007/978-3-031-06788-4_42

automatically analyzing security protocols, which supports vulnerability detection and security verification. It can handle many different cryptographic primitives defined by rewrite rules or equations. ProVerif has made great achievements in the verification of security protocols using formal methods.

In this paper, we mainly focus on two areas of works: security protocols verification with ProVerif and extensions of ProVerif for more security protocols. In recent years, works that directly use ProVerif verification protocol can be roughly divided into two categories: communication protocol verification [7,9,13,14,29] and Web application verification [15,17,24]. The communication protocol refers to the rules and agreements that must be followed by two entities to complete the communications and realize services. Web applications are programs that are stored on a remote server and delivered over the Internet through a browser interface. In these cases, the general process of verification is: (1) building the security model and determining the security properties, (2) then writing ProVerif input format code, such as applied pi calculus, (3) verifying the security protocol with ProVerif.

For extensions of ProVerif, we mainly introduce three aspects: protocols with stateful properties [3,11,25,26], protocols with algebraic properties [21–23,27], and protocol implementations [1,4–6,16,20]. Although ProVerif can handle a wide range of primitives and complex protocols, ProVerif still suffers from some shortcomings. For example, when considering verifying stateful protocols, the mechanism of ProVerif does not take into account the state reachable set, which leads to false attacks when verifying protocols. Moreover, there have been problems with Horn's theory in dealing with the algebraic properties of operators, leading to the inability of ProVerif to effectively analyze such type protocols. To solve these problems, researchers have extended ProVerif to efficiently verify these properties.

The protocol developers hope that the model can be automatically constructed through source codes of protocol implementations. While some protocol implementations in programming languages like Java are far from the input language of ProVerif, some researchers propose automatic model transformation tools. These tools translate protocol implementations into verifiable languages of ProVerif and subsequently invoke ProVerif to verify the security properties of the protocol.

The rest of this paper is organized as follows. In Sect. 2, we introduce the architecture of ProVerif and some basic security properties. In Sect. 3, we present several representative solutions that use ProVerif to verify security protocols. In Sect. 4, we present extension tools based on ProVerif to verify protocols. In Sect. 5, we present the future work. And finally, we conclude the paper in Sect. 6.

2 Overview of ProVerif

ProVerif is a protocol verifier developed in 2001, which was designed to automatically analyze the security of cryptographic protocols. ProVerif was modeled on functions or equations to verify security protocols, which was based on the Dolev-Yao mode [12]. ProVerif supports a variety of cryptographic primitives defined

by rewriting rules or by equations, and attackers can only use these primitives for computation. In the field of computer security, these features are particularly important which allow the analysis of security properties. This section introduces the architecture of ProVerif and its main properties.

2.1 Architecture

The architecture of ProVerif is shown in Fig. 1. ProVerif takes a protocol model and the security properties as input, and output results of verification. The protocol model describes a protocol by an extended language of pi calculus, which is converted automatically into a set of Horn clauses in ProVerif. For these clauses, the security properties to be proven are converted into derivability queries. There are three possible results of the verification: the properties are true, false, or cannot be proved. The security property is true unless the existence of an attack can be deduced from the Horn clauses. The security property is false when there is an attack deduced from the Horn clauses and where is an attack path can be reconstructed from pi calculus. In the above case, if the attack path cannot be reconstructed from pi calculus, the security property is cannot be proven. This is a false attack, which is inevitable because of the protocol abstraction of the Horn clause.

2.2 Basic Properties

We summarize several basic security properties from the user manual of ProVerif [10], including reachability property, correspondence assertion, injective correspondence, authentication, and observational equivalence.

Reachability property: items or properties that are available to the attacker, which is the most basic feature of ProVerif.

Correspondence assertion: it is used to capture relationships between events, such that if an event is executed, another event must be executed before it.

Injective correspondence: it is used to capture authentication in the case of a one-to-one relationship between the number of protocols that each participant needs to perform, which is used to make up for the lack of correspondence assertions.

Authentication: it is used to convert the authentication target into the corresponding relationship between events. For example, if a client triggers an event, the server must trigger a matching event.

Observational equivalence: two processes are observational equivalent unless the attacker can tell difference between them.

These properties are provable by ProVerif, and verification of these properties may often be involved in common security protocols. Therefore, ProVerif is widely used in the verification of security protocols and is very useful in the field of computer security.

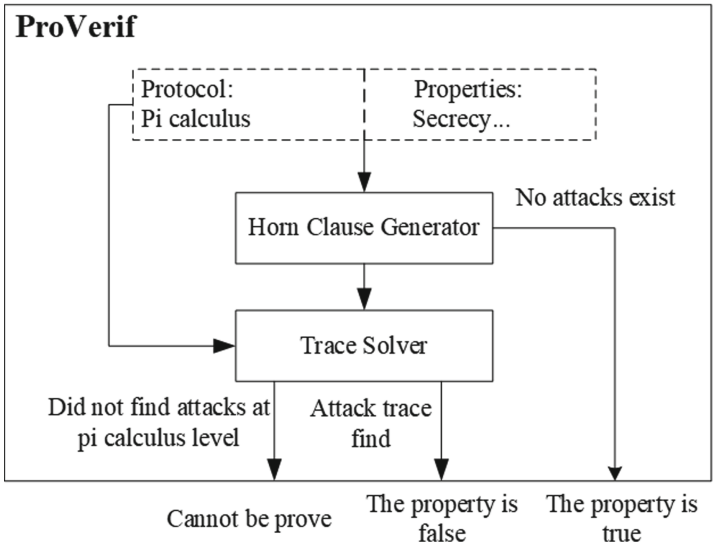

Fig. 1. The architecture of ProVerif [8].

3 Verification in ProVerif

ProVerif covers a wide range of cryptographic primitives and various protocol structures, which supports the analysis of various encoding formats and security properties of protocols. We take the tool to verify various types of security protocols, such as transport layer security (TLS), fast identity online (FIDO). We will describe two types of protocols that use ProVerif to verify security properties in the following.

3.1 Verification of Communication Protocols

A communication protocol refers to the rules and agreements that the two entities must follow to complete the communication or service. A communication protocol is mainly composed of syntax, semantics, and timing rules, which is hierarchical, reliable, and effective, as that in TCP/IP, TLS, etc. We will describe several communication protocols that have been verified by ProVerif in recent years.

Bruno Blanchet verifies the public-key and shared-key protocols [9] of the ARINC823 Avionic Protocols by ProVerif and CryptoVerif which relies on the computational model instead of the Dolev-Yao model. Some security flaws were also identified as that there is a computational replay attack in the shared-key protocol which breaks confidentiality. ProVerif finds most of the attacks easily, while it cannot find the secrecy attack in the shared-key protocol. The secrecy attack is discovered by a verification failure in CryptoVerif, which shows that ProVerif has some shortcomings in analyzing the protocol.

Karthikeyan Bhargavan [7] proposes an approach used to develop a symbolic and computational model for TLS 1.3. The authors model and analyze the draft protocol version 18 of TLS 1.3 by ProVerif and CryptoVerif respectively, which reveals some vulnerabilities in the draft protocol and proposes a reference implementation RefTLS for TLS 1.3. The authors develop models for different versions of TLS 1.3 and evaluate them to reconstruct known and previously undisclosed vulnerabilities.

To solve problems of security connection between the end user and user equipments (UEs) and sharing issues between Device-to-Device (D2D) users in the 5G network, Ed Kamya Kiyemba Edris et al. [13] propose a D2D level security solution that comprises the DDSec and DDACap protocol, which provides security for access, caching and sharing data in D2D communications. The authors model the proposed protocols and then verify the model by ProVerif. The authors firstly model and verify the DDSec protocol, and then found some security flaws. The DDACap protocol is formed by modifying the security flaws in the previous protocol. The DDACap protocol is verified by ProVerif, where the results show that all the security properties are true.

EAP-TLS is a standard defined in 5G communication technology, which is particularly important to ensure the security of the EAP-TLS protocol. Jingjing Zhang et al. [29] constructe a symbolic model about EAP-TLS, and then they use ProVerif to verify the model formally. Several weaknesses and design defects in the current protocol are obtained by analyzing the verification results.

Eman Elemam et al. [14] propose a security protocol PMQTT to protect message queue telemetry transport, which is derived from the message queue telemetry transport (MQTT) protocol. PMQTT adds cryptographic primitives from the MQTT, which is verified by ProVerif to check whether the proposed protocol satisfies the expected security properties.

This section briefly describes the application of ProVerif in several communication protocols. These researches show that ProVerif can effectively find some defects in the verification, while there are still some attacks that cannot be found. For example, ProVerif cannot detect the attack mentioned in [9] in ARINC823, because the attacks are not at the symbol level. This shows that ProVerif still has some deficiencies which need to be expanded. We will introduce some specific aspects of ProVerif extensions in Sect. 4.

3.2 Verification of Web Application

This section describes verification of web application, where protocols used by Web applications include OAuth, Web Authentication, FIDO [15], and so on.

Iness Ben Guirat and Harry Halpin [17] use ProVerif to analyze W3C Web Authentication API. They model W3C Web Authentication with the formal methods, which is proved security by analyzing the security properties of the protocol.

FIDO is a public key encryption-based authentication framework that allows users to log in to remote online services and websites by verifying their identity in a local trusted authenticator, e.g., a fingerprint scanner on a smartphone. Haonan

Feng et al. [15] analysis of the FIDO protocol suite of the universal authentication framework (UAF). By constructing protocol models for different scenarios, such as malicious entity modeling, unlinkable scenarios modeling, authors analytically determine the minimum-security assumptions required in each security objective and identify flaws of the UAF protocol.

Michael J. May and Kevin D. Lux [24] design a set of Web services to improve WSEmail. The architecture of WSEmail can be extended in a variety of ways to provide flexibility. While the increased flexibility often reduces security and performance. So they take security performance issues into consideration and verify the security of the protocol formally by TulaFale and ProVerif.

This section mainly presents several solutions of verification in the web application by ProVerif, and the researches show that ProVerif can effectively verify web applications and can detect some attacks.

3.3 Summary

In this section, we mainly present the related work of verification by ProVerif in recent years, which are mainly divided into two categories. One is the verification of communication protocols and the other is the verification of the protocol on the web application. In the verification of two categories, ProVerif can effectively detect a certain part of attacks and verify the security of protocols. However, ProVerif fails to verify some specific security properties or attacks, which indicates that ProVerif has some shortcomings needed to be further expanded. We will present the extension of ProVerif in the next section.

4 Extensions of ProVerif

Although ProVerif can handle a wide range of primitives and complex protocols, ProVerif still suffers from some shortcomings. There are possible false attacks because of the abstraction of protocols, and there are specific protocols without result in verification. As mentioned above, there are still some defects where researchers made corresponding extensions to solve them. We will present the extension of ProVerif in the three aspects as follows.

4.1 Extensions for Stateful Protocol

ProVerif builds models with pi calculus by the abstraction of protocols and converts them to Horn clauses. ProVerif approximates the behavior of each protocol by a set of first-order Horn clauses, which can be applied without limited number of times. In the analysis of protocols with global states, there may be some false attacks. Although ProVerif provides some built-in features like tables and phases, which provides a limited approach to model the state. To address the challenge of the deficiency of ProVerif in verification, several methods have been proposed to verify stateful protocols effectively.

StatVerif [3] is an extension of ProVerif with explicit state constructs, which can analyse protocols with the global state. StatVerif extends the compiler of ProVerif by extending the applied pi calculus, where the proposed compiler can avoid many false attacks. Protocol of small hardware security devices and protocol of contract signing agreements are used to verify the correctness of the compiler in the paper. StatVerif extends applied pi calculus to global cells that can be accessed and modified by processes. The extension language contains locked content to correctly write sequences of state operations. Due to the finite number of cells in the model, the number of states that can be represented is limited.

Vincent Cheval proposes a universal state converter, which is implemented as GSVerif [11]. GSVerif solves the problem of that ProVerif cannot verify global state and natural numbers, which is realized by converting the query the state and natural number properties into verifiable properties. For certain security properties, GSVerif cannot directly and automatically verify the protocol. In the case, one can design some formulas suitable for the protocol and manually verify the formulas to verify the security properties indirectly. This approach provides a method to improve tools that can handle global states, which also adds interactivity to ProVerif. Compared with StatVerif, GSVerif does not change the input language, but there are significant improvements in efficiency and coverage of protocols. However, GSVerif suffers from the deficiency that the ARINC823 avionics protocol cannot be properly verified.

The abstraction method is very effective in protocol verification, which suffers from shortcomings in dealing with stateful protocols. AIF generates a set of Horn clauses, where ProVerif or SPASS are used to verify these clauses. Although AIF adopts an infinite state approach, it can only analyze a fixed number of sets. To address this problem, Mödersheim et al. [25] proposes a new abstraction method based on a subset to verify protocols with non-monotonic and stateful properties. The authors develop AIF-ω which is an extension of AIF, where the protocol model composes with infinite sets.

To varying degrees, the above tools provide methods for verification of stateful protocols. StatVerif extends the applied pi calculus with global states, while it supports a limited number of sessions and cannot automatically verify complex protocols. GSVerif supports multiple protocols of infinite session which enriches proof strategy of ProVerif. The AIF-ω extends AIF and then invokes ProVerif to verify stateful protocols. As shown in Fig. 2, the above tools are extended from two aspects. StatVerif belong to the first situation, which build the model by extending the pi calculus. AIF-ω and GSVerif belong to other situations, where the security properties that cannot be verified in ProVerif are transformed into security properties that can be verified.

4.2 Extensions for Protocols with Algebraic Properties

ProVerif usually relies on Dolev-Yao model to verify security protocols, which ignores protocol attacks with algebraic properties in the protocol. Therefore, ProVerif cannot verify attacks that contains algebraic properties such as that

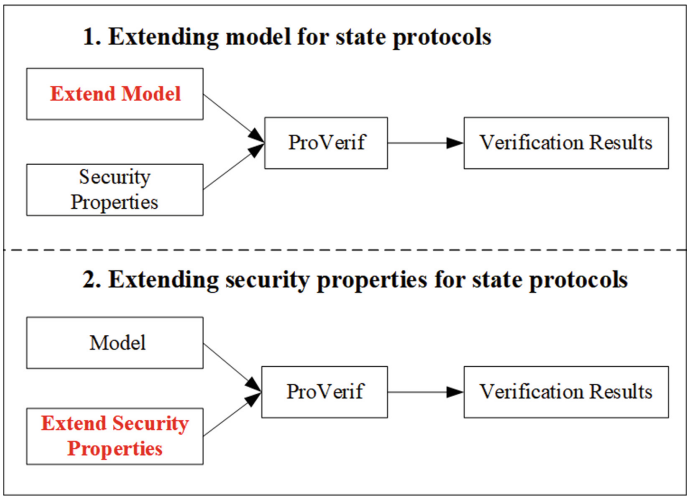

Fig. 2. Two types of extensions for stateful protocols.

in Diffie-Hellman and Exclusive-Or (XOR). To solve the problem, researchers extend ProVerif to verify protocols with algebraic properties effectively.

Dilong Li [27] proposes a new tool, ProVerif-ATP, which is a combination of ProVerif and the first-order automatic theorem prover ATP. The integrated tool allows the user to specify algebraic property in the model as an equation theory. The authors evaluate the effectiveness of ProVerif-ATP on many protocols, such as protocols with XOR, and the evaluation shows that ProVerif-ATP can discover some attacks that ProVerif cannot discover.

The security analysis of protocols by ProVerif essentially boils down to derivation of Horn theory, i.e., whether a given fact can be deduced from Horn theory. Following this essence, Ralf Küsters and Tomasz Truderung [21,22] propose two new tools, XOR-ProVerif and DH-ProVerif to verify protocols with the Exclusive-Or and Diffie-Hellma algebraic properties, respectively. For protocols with Exclusive-Or, Ralf Küsters et al. reduce the problem of deriving Horn theory with XOR to a syntactic derivation problem, where the algebraic properties of Exclusive-Or can be ignored. For protocols with Diffie-Hellman, DH-ProVerif takes the similar simplification operation, with different reduction operations and completeness of XOR-ProVerif. The technique is more effective in the reduction of Diffie-Hellman indices than XOR.

Pascal Lafourcade [23] compared the time spent by XOR-ProVerif and DH-ProVerif to verify different cryptographic protocols with Exclusive-Or and Diffie-Hellman, respectively. In [23], the authors point out that the number of Exclusive-Or affects the verification time and XOR-ProVerif cannot verify the protocols with the sufficiently large number of Exclusive-Or. In terms of complexity, it is exponential in Exclusive-Or, and it is lower in Diffie-Hellman than Exclusive-Or.

In terms of the verification of algebraic properties, the above tool extends ProVerif in two ways as shown in Fig. 3. On the one hand, ProVerif is inadequate in analyzing algebraic properties. Researchers integrate other tools to verify protocols with algebraic properties. On the other hand, security analysis of the protocols in ProVerif boils down to the derivation of Horn theory. Therefore, the derivation of Horn theory with algebra is simplified into the case without algebra, as the second case in Fig. 3, which reduces the unverifiable properties to verifiable properties in extension tools.

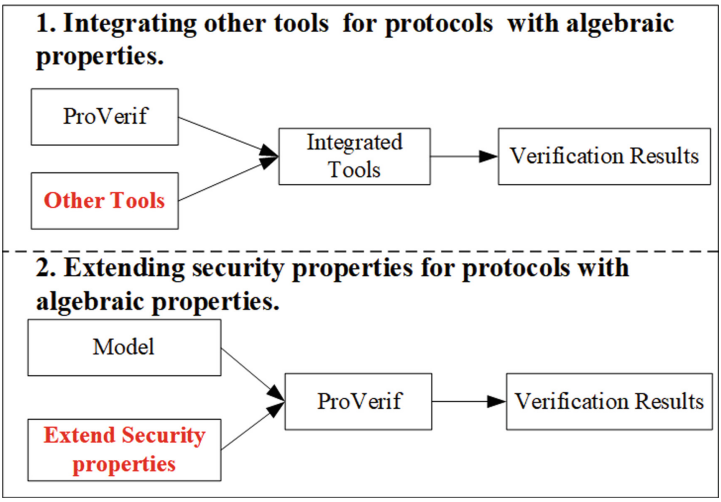

Fig. 3. Two types of extensions for protocols with algebraic properties.

4.3 Extensions for Protocol Implementations

In recent years, a large progress has been made in the area of formal verification for cryptographic protocols with powerful tools like ProVerif, where a large number of researchers choose ProVerif to verify security protocol. However, there is still a great gap between formal description languages of models and existing protocol implementation languages, as that between applied pi calculus and C, Java, f#, etc. Therefore, researchers propose some improved methods to bridge this gap. In these methods, ProVerif is used as a back-end to verify security properties by transforming the protocol implementation into an input format of ProVerif.

Bhargavan [6] propose an architecture to verify security protocol implementations. In this architecture, a prototype model extractor fs2pv is used to extract implementations of protocols, which is written in f# to obtain a subset of f# that can represent security protocols. Then the subset is compiled to specification of ProVerif, which is used to verify the security properties of the protocols finally.

Avalle [4] proposes a model-driven framework JavaSPI to develop security protocols. The framework provides the Java-ProVerif, which translates the model into the corresponding ProVerif model, and then the framework uses ProVerif to verify the security properties of the protocol. Different with [6], the ProVerif translates the model represented in Java syntax into ProVerif syntax instead of extracting it from Java code.

Bansal et al. [5] build the WebSpi library, which defines the basic components that are needed to model web applications and their security policies. This library can be used to model on standard web browsers, servers, and HTTP protocols, and it can analyze implementations of security protocols. To simplify the task of writing formal models, the authors propose a model extraction tool. The tool automatically extracts programs written in subsets of PHP and JavaScript into pi calculus model, and verifies them by ProVerif.

To verify the security properties of protocol implementation encoded in C, Aizatulinet et al. [1] extract the model of ProVerif from the implementation. However, the shortcoming of this technique is that it can only analyze code on a single execution path. Goubault-Larrecq [16] propose another method to analyze the protocol implementation in C. The difference is that they translate the protocol into clauses of the $\mathcal{H}_1$ class, and use the $\mathcal{H}_1$ prover instead of ProVerif to verify the secrecy of the protocol.

Nadim Kobeissi [20] propose a formal method for translating Noise Handshake Patterns to the processes of pi calculus. This transformation is implemented in a new tool Noise Explorer, which takes the Noise Handshake Patterns as input, and generates ProVerif scripts as output. The tool can encode the security targets in the specification of Noise Protocol Framework as security queries in the ProVerif syntax. Finally, the protocol is verified by ProVerif as the back-end.

The above multiple methods and tools share a common feature as shown in Fig. 4. By designing a translation method, the pre-existing protocol implementations are translated into an input format of ProVerif, and then ProVerif is used to verify the security properties of protocol. These methods expand the scope of application of ProVerif.

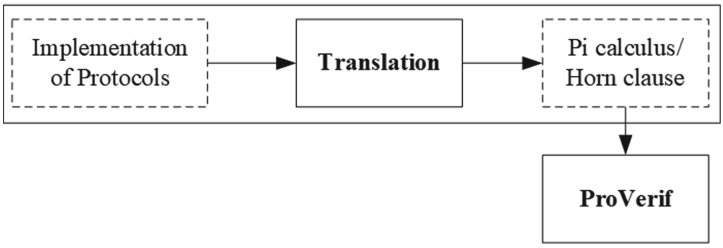

Fig. 4. General translation architecture for protocol implementations.

4.4 Summary

In this section, we describe three aspects of ProVerif extension, including protocols with stateful properties, protocols with algebraic properties, and protocol implementations. We analyze each extension tool and give the general structure diagram for each aspect of the extension.

5 Future Work

This article describes two types of extensions for protocols with stateful property and protocols with algebraic property. For a protocol with both properties, it is an interesting research point to verify the security properties. Protocols with the combination of stateful and algebraic properties can be verified by extending ProVerif or ProVerif integrated other verification tools, which is an indication of our future research direction.

The verification of network protocol has been a great success in traditional networks, while it is less involved in SDN networks. In addition, the protocols are often open-source in SDN networks, which is far from pi calculus. By translating the format of SDN code into an input format that ProVerif can recognize, we can to verify the protocols of SDN in ProVerif. In addition, security protocols need to consider issues such as logging [18], and we can combine logging with protocol verification. This is one of our future research directions.

As a third-party tester, if the source code is not obtained, or the protocol implementation is not publicly available, we can build a security model through protocol specifications and capture packet to obtain network traffic, which is refer to traffic in data center [19,28], and then we can use ProVerif to verify the corresponding security properties. This is also a significant research direction.

6 Conclusion

In this paper, we focus on ProVerif and extensions in the verification of security protocols. To illustrate verification with ProVerif, we describe some representative solutions in communication protocols and web applications. In addition, we present three aspects of ProVerif extensions, such as protocols with stateful properties, protocols with algebraic properties, and protocol implementations, and then summarize the general method of each extension tools. Finally, we discuss some possible research points for ProVerif extensions in the future.

Acknowledgement. This work was supported by the Hainan Provincial Natural Science Foundation of China (620RC562, 2019RC096, 620RC560), the Scientific Research Setup Fund of Hainan University (KYQD(ZR)1877), the Program of Hainan Association for Science and Technology Plans to Youth R&D Innovation (QCXM201910), and the National Natural Science Foundation of China (61802092, 62162021).

References

1. Aizatulin, M., Gordon, A.D., Jürjens, J.: Extracting and verifying cryptographic models from c protocol code by symbolic execution. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 331–340 (2011)
2. Akhter, A., Shah, A., Ahmed, M., Moustafa, N., Cavusoglu, U., Zengin, A.: A secured message transmission protocol for vehicular ad hoc networks. Comput. Mater. Contin. **68**(1), 229–246 (2021)
3. Arapinis, M., Ritter, E., Ryan, M.D.: StatVerif: verification of stateful processes. In: 2011 IEEE 24th Computer Security Foundations Symposium, pp. 33–47 (2011). https://doi.org/10.1109/CSF.2011.10
4. Avalle, M., Pironti, A., Sisto, R., Pozza, D.: The Java SPI framework for security protocol implementation. In: 2011 Sixth International Conference on Availability, Reliability and Security, pp. 746–751 (2011). https://doi.org/10.1109/ARES.2011.117
5. Bansal, C., Bhargavan, K., Maffeis, S.: Discovering concrete attacks on website authorization by formal analysis. In: 2012 IEEE 25th Computer Security Foundations Symposium, pp. 247–262 (2012). https://doi.org/10.1109/CSF.2012.27
6. Bhargavan, K., Fournet, C., Gordon, A., Tse, S.: Verified interoperable implementations of security protocols. In: 19th IEEE Computer Security Foundations Workshop (CSFW 2006), pp. 14–152 (2006). https://doi.org/10.1109/CSFW.2006.32
7. Bhargavan, K., Blanchet, B., Kobeissi, N.: Verified models and reference implementations for the TLS 1.3 standard candidate. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 483–502. IEEE (2017)
8. Blanchet, B.: Modeling and verifying security protocols with the applied pi calculus and ProVerif. Found. Trends® Priv. Secur. **1**(1–2), 1–135 (2016)
9. Blanchet, B.: Symbolic and computational mechanized verification of the arinc823 avionic protocols. In: 2017 IEEE 30th Computer Security Foundations Symposium (CSF), pp. 68–82 (2017). https://doi.org/10.1109/CSF.2017.7
10. Blanchet, B., Smyth, B., Cheval, V., Sylvestre, M.: ProVerif 2.02 pl1: automatic cryptographic protocol verifier, user manual and tutorial (2020)
11. Cheval, V., Cortier, V., Turuani, M.: A little more conversation, a little less action, a lot more satisfaction: global states in ProVerif. In: 2018 IEEE 31st Computer Security Foundations Symposium (CSF), pp. 344–358 (2018). https://doi.org/10.1109/CSF.2018.00032
12. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)
13. Edris, E.K.K., Aiash, M., Loo, J.: Formal verification of authentication and service authorization protocols in 5G-enabled device-to-device communications using ProVerif. Electronics **10**(13), 1608 (2021)
14. Elemam, E., Bahaa-Eldin, A.M., Shaker, N.H., Sobh, M.: Formal verification for a PMQTT protocol. Egypt. Inform. J. **21**(3), 169–182 (2020)
15. Feng, H., Li, H., Pan, X., Zhao, Z., Cactilab, T.: A formal analysis of the FIDO UAF protocol. In: Proceedings of the Network and Distributed Systems Security (NDSS) Symposium, pp. 1–15 (2021)
16. Goubault-Larrecq, J., Parrennes, F.: Cryptographic protocol analysis on real C code. In: Cousot, R. (ed.) VMCAI 2005. LNCS, vol. 3385, pp. 363–379. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30579-8_24

17. Guirat, I.B., Halpin, H.: Formal verification of the W3C web authentication protocol. In: 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security, HoTSoS 2018, pp. 1–10. ACM (2018)
18. Han, S., et al.: Log-based anomaly detection with robust feature extraction and online learning. IEEE Trans. Inf. Forensics Secur. **16**, 2300–2311 (2021). https://doi.org/10.1109/TIFS.2021.3053371
19. Jayamala, R., Valarmathi, A.: An enhanced decentralized virtual machine migration approach for energy-aware cloud data centers. Intell. Autom. Soft Comput. **27**(2), 347–358 (2021)
20. Kobeissi, N., Nicolas, G., Bhargavan, K.: Noise explorer: fully automated modeling and verification for arbitrary noise protocols. In: 2019 IEEE European Symposium on Security and Privacy (EuroS P), pp. 356–370 (2019). https://doi.org/10.1109/EuroSP.2019.00034
21. Küsters, R., Truderung, T.: Reducing protocol analysis with XOR to the XOR-free case in the horn theory based approach. J. Autom. Reason. **46**(3–4), 325–352 (2011)
22. Küsters, R., Truderung, T.: Using ProVerif to analyze protocols with Diffie-Hellman exponentiation. In: 2009 22nd IEEE Computer Security Foundations Symposium, pp. 157–171 (2009). https://doi.org/10.1109/CSF.2009.17
23. Lafourcade, P., Terrade, V., Vigier, S.: Comparison of cryptographic verification tools dealing with algebraic properties. In: Degano, P., Guttman, J.D. (eds.) FAST 2009. LNCS, vol. 5983, pp. 173–185. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12459-4_13
24. May, M.J., Lux, K.D., Gunter, C.A.: WSEmail: a retrospective on a system for secure internet messaging based on web services. arXiv preprint arXiv:1908.02108 (2019)
25. Mödersheim, S., Bruni, A.: AIF-ω: set-based protocol abstraction with countable families. In: Piessens, F., Viganò, L. (eds.) POST 2016. LNCS, vol. 9635, pp. 233–253. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49635-0_12
26. Qu, Z., Wu, S., Liu, W., Wang, X.: Analysis and improvement of steganography protocol based on bell states in noise environment. Comput. Mater. Contin. **59**(2), 607–624 (2019)
27. Li, D.L., Tiu, A.: Combining ProVerif and automated theorem provers for security protocol verification. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 354–365. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_21
28. Zhang, H., et al.: Da&fd-deadline-aware and flow duration-based rate control for mixed flows in DCNs. IEEE/ACM Trans. Netw. **27**(6), 2458–2471 (2019). https://doi.org/10.1109/TNET.2019.2951925
29. Zhang, J., Yang, L., Cao, W., Wang, Q.: Formal analysis of 5G EAP-TLS authentication protocol using ProVerif. IEEE Access **8**, 23674–23688 (2020). https://doi.org/10.1109/ACCESS.2020.2969474