A Formal Security Analysis of the $p \equiv p$ Authentication Protocol for Decentralized Key Distribution and End-to-End Encrypted Email

Itzel Vazquez Sandoval(✉) and Gabriele Lenzini

University of Luxembourg, Luxembourg City, Luxembourg
{itzel.vazquezsandoval,gabriele.lenzini}@uni.lu

Abstract. To send encrypted emails, users typically need to create and exchange keys which later should be manually authenticated, for instance, by comparing long strings of characters. These tasks are cumbersome for the average user. To make more accessible the use of encrypted email, a secure email application named $p \equiv p$ automates the key management operations; $p \equiv p$ still requires the users to carry out the verification, however, the authentication process is simple: users have to compare familiar words instead of strings of random characters, then the application shows the users what level of trust they have achieved via colored visual indicators. Yet, users may not execute the authentication ceremony as intended, $p \equiv p$'s trust rating may be wrongly assigned, or both. To learn whether $p \equiv p$'s trust ratings (and the corresponding visual indicators) are assigned consistently, we present a formal security analysis of $p \equiv p$'s authentication ceremony. From the software implementation in C, we derive the specifications of an abstract protocol for public key distribution, encryption and trust establishment; then, we model the protocol in a variant of the applied pi calculus and later formally verify and validate specific privacy and authentication properties. We also discuss alternative research directions that could enrich the analysis.

Keywords: Formal verification · Authentication protocols · Software security analysis · Privacy-by-default · Secure email · End-to-end encryption

1 Introduction

Despite the success of *instant messaging* (IM) applications, email prevails as the principal means for written communication [24]; yet, communication over email remains largely insecure nowadays [11]. Solutions for securing email have however been proposed. For instance, OpenPGP [1] is arguably the most widely used email encryption standard. Derived from the PGP software, it proposes

A. Saracino and P. Mori (Eds.): ETAA 2019, LNCS 11967, pp. 171–187, 2020.
https://doi.org/10.1007/978-3-030-39749-4_11

the use of symmetric and asymmetric cryptography plus data compression to encrypt communication, and digital signatures for message authentication and integrity.

Unfortunately, severe usability drawbacks have been identified and highlighted in the standard (e.g. [27]). Along with the need for users to understand at least general cryptographic concepts regarding encryption—which inevitably narrows down the scope of the audience—the principal issue is the need for verifying the ownership of public keys, i.e., that a public key claimed to be of an entity A does indeed belong to A exclusively. Various approaches tackle this problem, e.g., fingerprint comparisons, public key infrastructure, certificate authorities, and the notion of web of trust, which involves individuals signing each other's public keys, thus forming a chain of certifications [28]. However, these approaches have encountered limited adoption mostly due to usability or scalability issues [11].

Attempting to overcome OpenPGP's usability issues related to trust establishment, an open source commercial software, called $p \equiv p$ (Sect. 3), proposes the use of so called *trustwords* (detailed in Sect. 3.1) to carry out peer-to-peer entity authentication via an out-of-band channel—e.g., in-person, video-call. This approach argues to introduce an improvement to usability and security of the PGP word list.

In this work we present a formal security analysis of the core protocols implemented in $p \equiv p$, focusing particularly in authentication and privacy goals.

1.1 Contributions

First, we derive from the open source code the specifications of $p \equiv p$'s abstract protocols for key distribution and trust establishment, and present them as Message Sequence Charts (MSC). From now on, we will refer to this abstraction as the $p \equiv p$ *protocol*. This is the first detailed technical documentation of such protocol.

Second, we provide a symbolic formal security analysis of the $p \equiv p$ protocol with respect to authentication and privacy goals, under a Dolev-Yao threat model. The analysis validates the security claims of $p \equiv p$ and the correct assignment of privacy ratings to messages.

2 Context and Approach

The application of formal methods for verifying that specific security properties hold in cryptographic protocols in the presence of a certain adversary is a well-established research area. Both the detection of flaws in a protocol (or, contrariwise, the proof of security) and the nature of those flaws depend on different factors, such as the verification approach and the phase of the system in which it takes place (e.g., design, implementation, compilation). An introductory reference for the topic is [21].

A variety of tools and formalizations have been used to successfully analyze, amongst others, authentication scenarios in real world and authentication standards (e.g., [6,7,13]). Important flaws have been discovered even in well-established protocols years after their publication and while being used (e.g., [19]). Therefore and because the design of protocols is by default an error-prone task, to effectively protect a system, security protocols need to be not only carefully designed and rigorously implemented but also strictly verified.

Here, we carry out a symbolic formal analysis of the $p \equiv p$ protocol specification. The symbolic approach assumes cryptographic primitives to work as perfect black boxes and focuses on the description of the logic of the protocol, the interaction among participants and the exchange of messages [10]. The resulting models allow to seek for attacks that rely on logical flaws in the protocol while taking advantage of mature automated tools for protocol analysis (e.g., ProVerif [9], Tamarin [5]).

Our work concerns remote human-to-human authentication, where human A wants to be sure that human B is who he claims to be and vice versa—in $p \equiv p$, the owner of a specific public key—, in a global communication scenario where A and B might not know each other.

2.1 Methodology

At the time when we started studying the $p \equiv p$ protocol there was not substantial documentation regarding neither the protocol specifications nor the source code. In consequence, the work presented here relies on the open source code of $p \equiv p$ [22], together with online documentation mainly for users [23]. Recently some internet drafts have been released [17,18], which has helped clarifying our models.

Our security analysis consists of the following steps, which we detail in the rest of the paper:

1. Extract the specifications of the key distribution and handshake protocols from the available sources [22,23].
2. Describe the protocol in MSC notation.
3. Formalize in the *applied pi calculus* the MSC specifications of the previous step, along with the attacker model.
4. Specify and formalize in the *applied pi calculus* the properties to be verified.
5. Verify the satisfiability of the properties formalized in 4, in the model resulting from step 3.
6. Analyze and interpret the results of the verification.

We start by introducing the $p \equiv p$ software and its relevant features in Sect. 3. Then, steps 1 and 2, which deal with specifying the $p \equiv p$ protocol, are presented in Sect. 4. In Sect. 5, we define the security properties related to privacy and authentication that concern our analysis. Section 6 covers steps 3 and 4 of the methodology, i.e., the formalization of the protocol and of the security properties introduced informally in Sect. 5. The results of the execution of step 5 and the

analysis in step 6 are discussed in Sect. 6.4; we also discuss limitations of the analysis in Sect. 6.5. Further directions and conclusions are presented in the last section.

3 Background: Pretty Easy Privacy ($p \equiv p$)

Pretty Easy Privacy ($p \equiv p$)[1] is a software that claims to provide privacy-by-default in email communications via end-to-end opportunistic encryption. Roughly, this means that the software encrypts outgoing email messages without any intervention from the user, whenever a secure or trusted public key of the intended receiver is available.

$p \equiv p$ attempts to automate tasks that would otherwise require specialized-knowledge from non-expert users, while informing the user of the privacy rating assigned to messages in an intuitive way. Hence, its more relevant features are: (1) a fully automated process for the generation and management of encryption keys and for the encryption of emails; (2) an algorithm to determine the strongest privacy level that can be assigned to a message for a specific partner—this level is further communicated to the user by colored visual icons; (3) a fully decentralized architecture for key storage—this design decision eludes relying on possibly untrusted central authorities by having the users perform the trust establishment task via out-of-band channels.

$p \equiv p$ is distributed as a standalone application for Android and as plugins for desktop installations of some existing email clients, e.g., Outlook, Enigmail. In this work we consider a general abstraction of the $p \equiv p$ protocols that represent improvements to PGP by means of the features described above. Comparing and discussing specific implementations is out of the scope of this paper.

3.1 $p \equiv p$ Trustwords

Manual key-fingerprint comparison is a well-established method for entity authentication in messaging protocols; yet, the approach has been shown to perform poorly for the intended goal (e.g., [14]). As a solution, in addition to hexadecimal numbers, PGP allows fingerprints to appear as a series of so-called "biometric words", which are phonetically different English words that intend to ease the comparison for humans and to make it less prone to misunderstandings [2].

Trustwords in $p \equiv p$ follow the same idea; they are natural language words mapping hexadecimal strings that are used to authenticate a peer after having exchanged public keys in an opportunistic manner. In short, such hexadecimal strings represent a combined fingerprint obtained by applying an XOR operation to the fingerprints associated to the public keys being authenticated. Each block of 4 hex characters of the combined fingerprint is mapped to a word in a predefined trustwords dictionary. For instance, `F482 E952 2F48 618B`

[1] https://www.pep.security.

`01BC 31DC 5428 D7FA` could be mapped to `kite house brother town juice school dice broken`.

The main difference with the "biometric words" is the availability of trustwords in different languages, which improves the security for non-English speakers, and the use of longer words, which presumably increases the entropy as the dictionary is larger and therefore the likelihood for phonetic collision is decreased [17]. Considerations regarding the number of words in the dictionaries and the length of the words themselves are discussed also in [17].

3.2 Trust Rating and Visual Indicators

In agreement with the privacy-by-default principle, $p \equiv p$ assigns a specific privacy rating to each email exchange. Such a rating is determined per message and per identity depending on certain criteria and is shown to the users by colored icons in the message. The ratings are:

- MISTRUSTED: the system has evidence that the communication partner is not who (s)he claims to be, e.g., when the user explicitly mistrusts a peer.
- UNKNOWN/UNSECURE/UNRELIABLE (UNSECURE): encryption/decryption of a message cannot be properly executed, e.g., when the recipient does not use any secure email solution. The message is sent in plain text.
- SECURE: the user has a valid public key for the recipient, however it has not been personally confirmed. The message is encrypted/decrypted.
- TRUSTED: the user has the recipient's public key and it has been validated with the peer. The message is encrypted/decrypted and authenticated.

3.3 Technical Specifications of $p \equiv p$

The core component of $p \equiv p$ is `pEpEngine`, a library developed in C99 where the automation of cryptographic functionalities (e.g., key generation) is implemented relying on existing standards and tools for secure end-to-end encrypted communications (PGP, GnuPG). The $p \equiv p$ protocols are built upon those functionalities, therefore `pEpEngine` is the component from which we extracted the specifications hereby presented.

Each installation of $p \equiv p$ creates a local database of $p \equiv p$ peers, their corresponding keys and privacy ratings. Additionally, it creates a database from which the trustwords for mutual authentication are retrieved; the trustwords database contains the exact same data in all the distributions. To securely store private and public keys in the devices, $p \equiv p$ uses GnuPG[2]. A more detailed description of $p \equiv p$ can be found in [18].

4 The $p \equiv p$ Protocol

In order to carry out a security analysis it is essential to clearly understand the logic of the protocol, to know the cryptographic primitives used, the parties

[2] https://www.gnupg.org/.

involved and the messages exchanged between them. Our case study required us to obtain this information mainly from the source code of $p \equiv p$.

Following the approach in [26], we executed the first step of the methodology proposed here in Sect. 2.1 by reverse engineering a fragment of the source code files. We then represented the output of such a process by means of MSC diagrams (step 2) which $p \equiv p$ confirmed to be accurately representing their protocol.

Here, we present and describe such diagrams which correspond to our abstracted version of the key distribution and authentication protocols used by $p \equiv p$ to engage in end-to-end private and authenticated communications.

In the rest of the paper, we will use sk_x and pk_x to refer to secret and public keys owned by agent x, respectively. As well, we use $\mathcal{A}$ and $\mathcal{B}$ to refer to honest participants and $\mathcal{M}$ for the malicious agent trying to prevent the honest parties from achieving the security goals.

4.1 Public Key Distribution and Encrypted Communication

Let $\mathcal{A}$ and $\mathcal{B}$ be two partners that do not know each other's public key. $\mathcal{A}$ installs $p \equiv p$ from scratch without having any cryptographic keys. She wants to privately communicate with $\mathcal{B}$ who is already a $p \equiv p$ user owning a pair of keys (sk_B, pk_B). We denote the $p \equiv p$ instances running in $\mathcal{A}$'s and $\mathcal{B}$'s devices as $\texttt{pEp}_\texttt{A}$ and $\texttt{pEp}_\texttt{B}$ respectively.

So that the key distribution protocol (Fig. 1) can take place, when $p \equiv p$ is installed, $\texttt{pEp}_\texttt{A}$ generates a pair of keys (sk_A, pk_A) for $\mathcal{A}$ (step 1). The protocol starts when $\mathcal{A}$ sends a message m to $\mathcal{B}$; $\texttt{pEp}_\texttt{A}$ creates an identity for $\mathcal{B}$ (2) and stores his contact details (3); then, $\texttt{pEp}_\texttt{A}$ sends m as plain text along with pk_A (4). When $\texttt{pEp}_\texttt{B}$ receives the message, it displays m to $\mathcal{B}$ with the privacy rating UNSECURE (5); additionally, $\texttt{pEp}_\texttt{B}$ creates an identity for $\mathcal{A}$ (6) and stores her email address and pk_A (7); finally $\texttt{pEp}_\texttt{B}$ assigns the privacy rating SECURE to $\mathcal{A}$'s identity (8). When $\mathcal{B}$ replies to $\mathcal{A}$, $\texttt{pEp}_\texttt{B}$ attaches pk_B to his response *resp*; this message is then signed with $\mathcal{B}$'s secret key sk_B (9) and encrypted using pk_A (10). The signed and encrypted message is sent to $\mathcal{A}$ (11); $\texttt{pEp}_\texttt{B}$ shows to $\mathcal{B}$ his message as SECURE. At reception, $\texttt{pEp}_\texttt{A}$ decrypts $\mathcal{B}$'s message using sk_A (12); then it stores pk_B as the public key of $\mathcal{B}$ (13) and assigns to his identity the SECURE rating (14). $\mathcal{B}$'s response is finally shown as SECURE to $\mathcal{A}$.

Note that the identifiers created for $\mathcal{A}$ and $\mathcal{B}$ do not need to coincide in $\texttt{pEp}_\texttt{A}$ and $\texttt{pEp}_\texttt{B}$, since they are only used by the corresponding $p \equiv p$ instance. Also, pk_A and pk_B sent in steps (4) and (11) are only attached to the first communication between $\mathcal{A}$ and $\mathcal{B}$ or whenever they are updated.

The key distribution protocol allows making the communication secret to everyone but the receiver, however, it does not guarantee that the receiver is the intended person. Man-in-the-middle attacks are still possible, as we will discuss in Sect. 6.4.

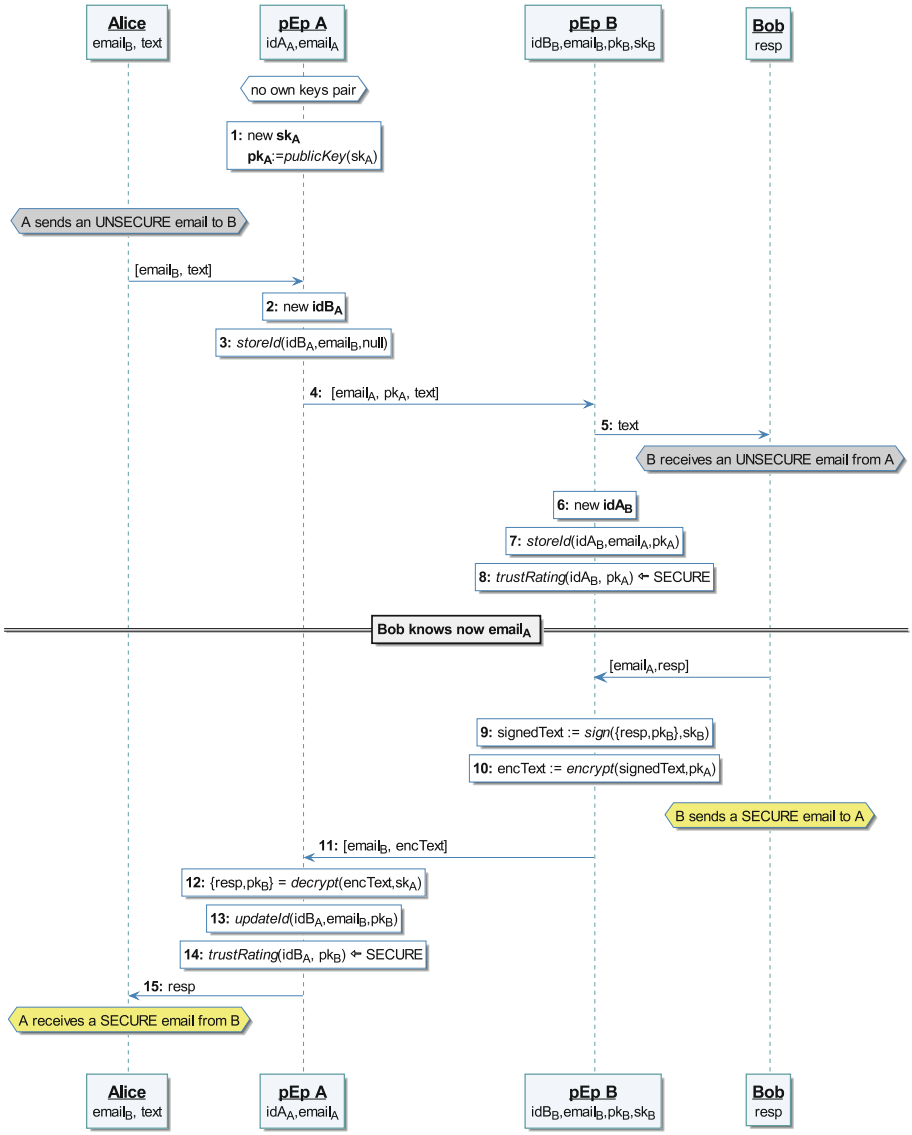

Fig. 1. $p \equiv p$ key distribution protocol

4.2 Authentication and $p \equiv p$ Privacy Rating Assignment

Trust establishment is achieved via the $p \equiv p$ Handshake protocol (Fig. 2), which consists in $\mathcal{A}$ and $\mathcal{B}$ comparing a list of trustwords via a communication channel assumed to be secure and that needs to be used only once.

When $\mathcal{A}$ selects the option to perform a handshake with $\mathcal{B}$ (1), $\mathtt{pEp_A}$ generates a combined fingerprint based on applying an *xor* function to the fingerprints of

$\mathcal{A}$ and $\mathcal{B}$ (2). The resulting hexadecimal string is mapped onto words in the selected language from the trustwords database (3) and displayed to $\mathcal{A}$ (4). The analogous actions occur in $\texttt{pEp}_\texttt{B}$ when $\mathcal{B}$ selects the handshake option. Given that the trustwords database is the same in all $p \equiv p$ distributions, if $\texttt{pEp}_\texttt{A}$ and $\texttt{pEp}_\texttt{B}$ use the same input parameters, i.e., the same public keys and thus the same fingerprints, the list of trustwords generated by each $p \equiv p$ instance must be the same.

The next step is the authentication, where $\mathcal{A}$ and $\mathcal{B}$ contact each other in a way that they are sure to be talking with the real person, and compare the list of trustwords displayed for each (5). If $\mathcal{B}$ confirms that the list of trustwords given by $\mathcal{A}$ matches exactly the one shown in his device, $\mathcal{A}$'s privacy rating is set to TRUSTED (6); we call this case a *successful handshake*. Conversely, in an *unsuccessful handshake* $\mathcal{A}$'s rating is downgraded from SECURE to MISTRUSTED (7). The analogous occurs in $\mathcal{A}$'s device with respect to $\mathcal{B}$. The privacy rating assigned after a handshake remains for all future exchanges with the communication partner.

After a successful handshake, the communication between the identities that performed the handshake is always encrypted and authenticated (8–12).

Remark that $p \equiv p$ does not force users to perform the handshake protocol. The email messages are always sent regardless of the security level, which is decided per message and per recipient according to the recipient's data available.

5 Security Properties

Our requirements for authentication match the definition of *full agreement* given by Lowe in [20]. This definition subsumes aliveness, weak agreement, non-injective agreement and injective agreement as defined in the same reference; broadly, it requires the two participants to agree on all the essential data involved in the protocol run, in our case, the public keys pk_A and pk_B and the email addresses.

Definition 1 (Full agreement, from [20]). *A protocol guarantees to an initiator A full agreement with a responder B on a set of data items ds if, whenever A completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the terms in ds, and each such run of A corresponds to a unique run of B. Additionally, ds contains all the atomic data items used in the protocol run.*

Here we redefine this property in terms of $p \equiv p$ and introduce informally other properties in which we are interested.

Property 1 (Full agreement). A full agreement between $\mathcal{A}$ and $\mathcal{B}$ holds on pk_A, pk_B, $email_A$ and $email_B$ if, whenever $\mathcal{A}$ completes a successful handshake with $\mathcal{B}$, then: $\mathcal{B}$ has previously been running the protocol with $\mathcal{A}$, the identity data of $\mathcal{A}$ is ($email_A$, pk_A) and the identity data of $\mathcal{B}$ is ($email_B$, pk_B).

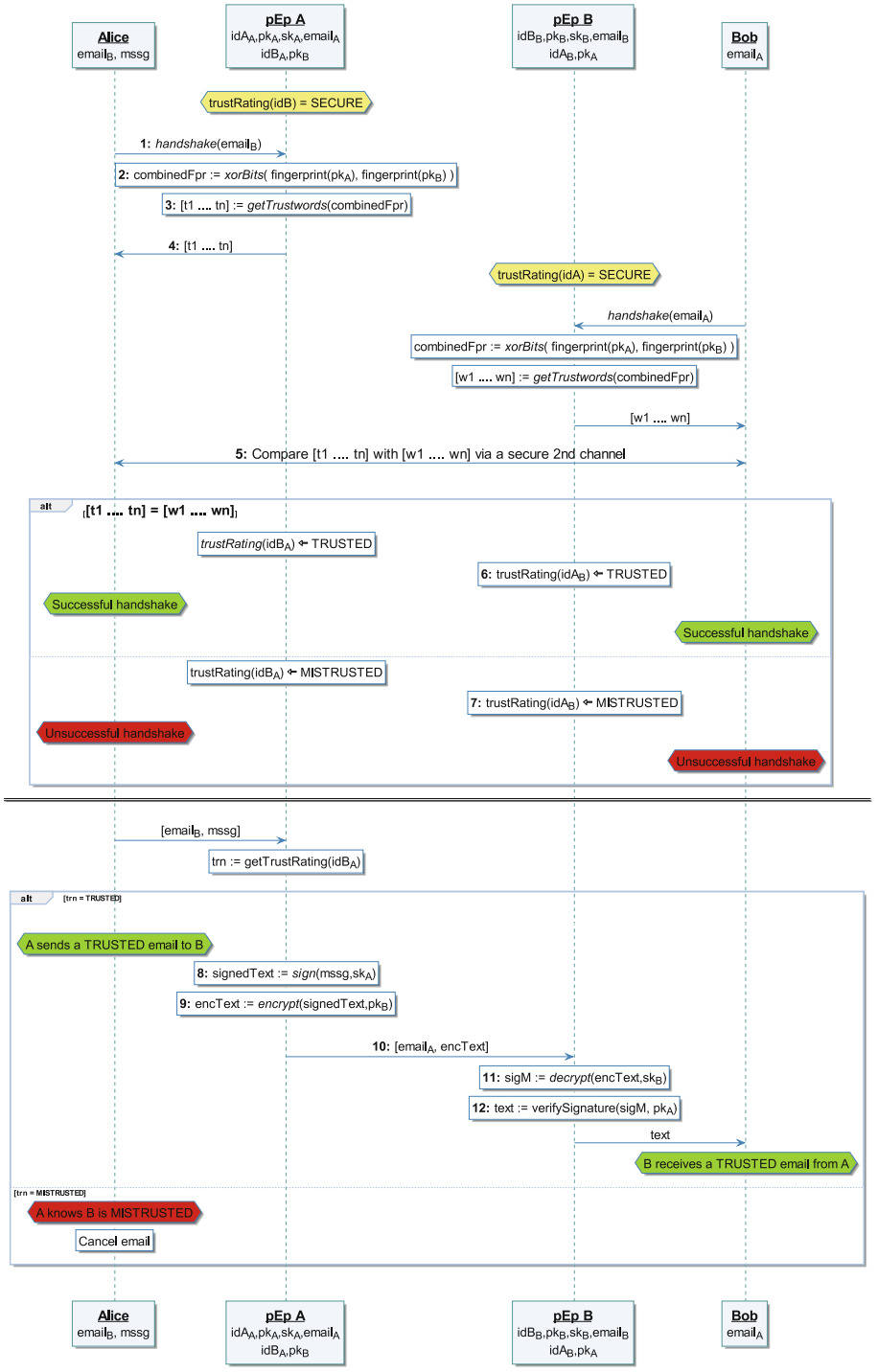

Fig. 2. $p \equiv p$ handshake protocol for authentication

Recall that a successful handshake is only reached if $\mathcal{B}$ confirms that the trustwords given by $\mathcal{A}$ match exactly those shown in his device, and vice versa; therefore, the agreement on the trustwords is implicit in the definition.

Property 2 (Trust-by-Handshake). Trust-by-Handshake holds for $\mathcal{B}$ if whenever $\mathcal{B}$ receives a message from $\mathcal{A}$ with privacy rating TRUSTED, then previously $\mathcal{B}$ executed a successful handhsake with $\mathcal{A}$.

Property 3 (Privacy-from-trusted). Privacy-from-trusted holds for $\mathcal{B}$ if, whenever $\mathcal{B}$ receives a message m from $\mathcal{A}$ who has a privacy rating TRUSTED, then $\mathcal{A}$ sent m to $\mathcal{B}$ and m is encrypted with $\mathcal{B}$'s public key.

Property 4 (Integrity-from-trusted). Integrity-from-trusted holds for $\mathcal{B}$ if, whenever $\mathcal{B}$ receives a message m from $\mathcal{A}$ who has a privacy rating TRUSTED, then $\mathcal{A}$ sent m to $\mathcal{B}$ and m is signed with a valid signature of $\mathcal{A}$.

Property 5 (MITM-detection). MITM-detection holds if whenever an unsuccessful handshake between $\mathcal{A}$ and $\mathcal{B}$ occurs, then $\mathcal{A}$ has registered a key for $\mathcal{B}$ that does not belong to him and/or vice versa.

Property 6 (Confidentiality). Confidentiality holds if $\mathcal{M}$ cannot learn the content of any message sent encrypted between $\mathcal{A}$ and $\mathcal{B}$.

6 Formal Security Analysis

A security analysis requires three elements: a protocol model, a set of security properties, and a threat model defining the capabilities of the adversary by which the scope of the verification is framed.

We model the $p \equiv p$ protocols in the *applied pi calculus* [4], a process calculus suitable for describing and reasoning about security protocols in the symbolic approach. Participants are represented as processes and their message exchanges are represented by terms sent over public or private channels. A so called equational theory defines how the cryptographic operations occurring in the protocol relate with each other, and how they can be applied to obtain equivalent terms.

6.1 Threat Model and Trust Assumptions

To determine a relevant attacker model we need to consider the decentralized architecture of $p \equiv p$. To an attacker with access to the user's device, not only the code but also the application databases and the keys repository are available. $\mathcal{M}$ can thus have $\mathcal{B}$ trusting her by simply modifying the corresponding record in the privacy ratings database, even if a handshake was never performed. Modifications to the trustwords database would also result in an attack, which although not threatening privacy, could prevent $\mathcal{A}$ and $\mathcal{B}$ from establishing a valid trusted communication as TRUSTED. Therefore, we restrict the threat model with the following assumptions:

1. $p \equiv p$ users are honest participants and their devices are secure;
2. The adversary cannot modify exchanges over the trustwords channel;
3. The adversary has complete control over the network used to exchange emails (Dolev-Yao attacker [15]);
4. The users execute the comparison of trustwords correctly, i.e., they confirm the trustwords in the system only when they match in the real world and they mistrust them only in the contrary case.

These assumptions allow $\mathcal{M}$ to eavesdrop, remove, and modify emails exchanged between $\mathcal{A}$ and $\mathcal{B}$, as well as to send them messages of her choice; this includes learning their public keys exchanged by email. $\mathcal{M}$ cannot however interfere with the channel used to corroborate trustwords. Remark that this is a secondary channel such as the phone or in-person, thus, not intended to replace the email communication channel. We elaborate on assumption 4 in Sect. 7.

6.2 Modeling the $p \equiv p$ Protocol

The $p \equiv p$ protocol consists of the sequential execution of the key distribution and the trust establishment protocols presented in Sect. 4.

$\mathcal{A}$ and $\mathcal{B}$ are represented by two processes, `senderA` and `receiverB`, whose parameters symbolize the knowledge that they have. To communicate with $\mathcal{B}$, $\mathcal{A}$ needs to know his contact details, which here we abstract with the type *userId*; in turn, $\mathcal{B}$ only needs to know his own id and his secret key. The actions for each participant come from the diagrams in Figs. 1 and 2. We run multiple instances of $\mathcal{A}$ as well as of $\mathcal{B}$, to simulate communication with multiple peers.

For the exchange of emails we use a public channel; on the contrary, a private channel models the trustwords' validation channel. In order to prove confidentiality of encrypted and authenticated communication, we introduce a private message *mssg* representing a message whose content is initially unknown to $\mathcal{M}$; then, we model $\mathcal{A}$ sending *mssg* to $\mathcal{B}$ via the public channel after a successful handshake between them. Since $\mathcal{B}$ is trusted, *mssg* is sent signed and encrypted (steps 8–9, Fig. 2), and thus, expected to remain unreadable by $\mathcal{M}$ at the end of the protocol.

According to the symbolic model assumption, our equational theory models a perfect behavior of asymmetric encryption and digital signatures. These equations capture the relationships allowed among the cryptographic primitives involved, determining the ways in which any participant, the attacker included, can reduce terms. Then, for M a message and SK a secret key:

$$adec(aenc(M, pubKey(SK)),\ SK) = M \tag{1}$$

$$verifSign(sign(M, SK),\ pubKey(SK)) = M \tag{2}$$

$$getMssg(sign(M, SK)) = M \tag{3}$$

Equation (1) expresses that a message M encrypted with a certain public key can be decrypted with the corresponding secret key; moreover, this is the only

way to obtain M from a ciphertext since there is no other equation involving the *aenc* primitive. Analogously, Eq. (2) returns M only if it was signed with the secret key associated to the public key used for the verification. Equation (3) allows the recovery of a message without verification of a digital signature and we introduce it here to model the capability of $\mathcal{M}$ for learning messages without the need of verifying the signature.

Also, we model a correct trustwords comparison as per assumption 4 in Sect. 6.1. We abstract fingerprints as public keys since a PGP fingerprint is uniquely derived from a public key. Then, for two public keys PK_1, PK_2, two trustwords lists W_1, W_2 and the trustwords generation function *trustwords*:

$$trustwordsMatch(trustwords(PK_1, PK_2), trustwords(PK_1, PK_2)) = true$$
$$trustwordsMatch(trustwords(PK_1, PK_2), trustwords(PK_2, PK_1)) = true$$

During its computations, $\mathcal{M}$ is allowed to apply all and only these primitives. Additionally, she has access to all the messages exchanged via the public channels and to any information declared as public. This models for instance $\mathcal{M}$'s real-life capability of generating the trustwords, which is possible because all the elements are public knowledge: the source code of the function, the trustwords database, $\mathcal{B}$'s public key and $\mathcal{A}$'s public key.

6.3 Privacy and Authentication Properties of $p \equiv p$

We formalize the properties introduced in Sect. 5 as correspondence and reachability queries based on events. Correspondences have the form $E \Longrightarrow e_1 \wedge ... \wedge e_n$; they model properties expressing: *if an event E is executed, then events $e_1, ..., e_n$ have been previously executed*. Events mark important states reached by the protocol and do not affect the protocol's behavior. Our properties are defined in terms of the next events, where s and r represent two $p \equiv p$ users:

- *endHandshakeOk(s,r,pk_s,pk_r,e_s,e_r)*: s and r completed a successful handshake with the public keys and emails (pk_s, e_s) and (pk_r, e_r) respectively.
- startHandshake(s,r): s starts a handshake via a second-channel with r
- *userKey(s,pk_s)*: the agent s is the owner of the key pk_s
- *userEmail(s,e_s)*: the agent s owns the email address e_s
- *receiveGreen(r,s,m)*: r received the message m from s as TRUSTED
- *receiverTrustsS(r,s)*: the contacted peer r sets the privacy rating of s as TRUSTED after confirming that the trustwords match
- *sendGreen(s,r,m)*: s sent the message m to r as TRUSTED
- *decryptionFails(r,s,m)*: r cannot decrypt a message m from a trusted peer s
- *signVerifFails(r,s,m)*: r cannot verify the signature attached to m as a valid signature of s
- *endHandshakeUnsucc(s,r,pk_s,pk_r)*: s and r completed an unsuccessful handshake with the public keys pk_s and pk_r respectively.
- *attacker(m)*: the adversary knows the content of the message m

Then, for a private message *mssg* and for all $p \equiv p$ users a and b, messages m and public keys ka, kb, pk_A, pk_B:

Full Agreement. For email addresses e_A and e_B,

$$\begin{aligned} endHandshakeOk(a, b, pk_A, pk_B, e_A, e_B) \implies & startHandshake(a, b) \wedge startHandshake(b, a) \\ & \wedge userKey(a, pk_A) \wedge userKey(b, pk_B) \\ & \wedge userEmail(a, e_A) \wedge userEmail(b, e_B) \end{aligned}$$

In our model the email address is abstracted as the identity itself, since we consider the case of one account per user. Therefore, in the verification the *userEmail* predicates are disregarded. We include them here for completeness.

Trust-by-Handshake

$$receiveGreen(b, a, m) \implies receiverTrustsS(b, a)$$

This formula matches exactly the definition of Property 2.

Privacy-from-Trusted. For a message z,

$$\begin{aligned} \big(receiveGreen(b, a, z) \implies & sendGreen(a, b, z) \wedge z = aenc(m, pk_B) \\ & \wedge userKey(b, pk_B)\big) \wedge \\ \big(decryptionFails(b, a, m) \implies & \neg\, sendGreen(a, b, m)\big) \end{aligned}$$

This formula is the conjunction of two correspondence assertions. The first one expresses Property 3; the second correspondence enforces the first by saying that it cannot be otherwise, i.e., when b receives a message m from a which for any reason cannot be decrypted—e.g. m is not encrypted—, then a did not send m to b.

Integrity-from-Trusted. For a message z and a secret key sk_A

$$\begin{aligned} \big(receiveGreen(b, a, z) \implies & sendGreen(a, b, z) \wedge z = aenc(sign(m, sk_A), kb) \\ & \wedge userKey(a, sk_A)\big) \wedge \\ \big(signVerifFails(b, a, m) \implies & \neg\, sendGreen(a, b, m)\big) \end{aligned}$$

Analogous to the previous formula, in this one we express Property 4 and reinforce it by proving that whenever the verification of the signature fails in message m, then a did not send m.

MITM-detection

$$\begin{aligned} endHandshakeUnsucc(a, b, ka, kb) \implies & (userKey(a, pk_A) \wedge pk_A \neq ka) \vee \\ & (userKey(b, pk_B) \wedge pk_B \neq kb) \end{aligned}$$

This formula matches exactly the definition of Property 5.

Confidentiality. *attacker* is a built in predicate in ProVerif, which evaluates to TRUE if by applying the derivation rules to the knowledge of the adversary, there exists a derivation that results in *mssg*. Therefore, the protocol achieves confidentiality if

$$\neg\, attacker(mssg)$$

6.4 Verification Results and Analysis

In order to determine whether or not the protocol satisfies the specified security properties we use ProVerif [9], an automatic symbolic cryptographic protocol verifier. We executed the verification[3] with ProVerif 2.0 on a standard PC (Intel i7 2.7GHz, 8GB RAM). The response time was immediate.

We analyzed three different models: of the key distribution protocol, of the trust establishment protocol and of the key distribution followed by the trust establishment (the $p \equiv p$ protocol).

For the key distribution protocol, the results confirmed its vulnerability to MITM attacks. The weakness resides in the exchange of public keys via a channel where $\mathcal{M}$ has complete access. An attack proceeds as follows: $\mathcal{M}$ can intercept the initial message from $\mathcal{A}$ to $\mathcal{B}$ and send him a new message attaching her own public key, pk_E, instead of $\mathcal{A}$'s one. $\texttt{pEp}_\texttt{B}$ will then link $\mathcal{M}$'s key with $\mathcal{A}$'s email in step (7) of Fig. 1, i.e., $storeId(idA_B, email_A, pk_E)$. When $\mathcal{B}$ replies, the message in step (10) is encrypted with pk_E, and thus $\mathcal{M}$ can intercept it again and decrypt it with her secret key, therefore obtaining pk_B attached. From this point, $\mathcal{M}$ can send encrypted emails to $\mathcal{B}$ using $\mathcal{A}$'s email address and she will be able to intercept and decrypt the responses sent by $\mathcal{B}$. In an analogous way, $\mathcal{M}$ can have $\mathcal{A}$ linking $\mathcal{M}$'s public key to $\mathcal{B}$'s identity, by sending her pk_E encrypted with pk_A obtained by intercepting the first message.

Regarding the trust establishment protocol, encryption and authentication hold since the trustwords comparison never mismatches due to the assumptions of the peer devices being secure and of a previous key distribution successfully executed.

The subsequent analysis of the $p \equiv p$ protocol determined that the six properties, full agreement, trust-by-handshake, privacy-from-trusted, integrity-from-trusted, MITM-detection and confidentiality are satisfied.

Regarding unsuccessful handshakes, even if $\mathcal{A}$ has the correct public key of $\mathcal{B}$, the handshake will fail if $\mathcal{B}$ has a key of $\mathcal{A}$ that does not correspond to her. Both partners will mistrust each other because the communication with those keys is threatened, however, once a peer is mistrusted, by $p \equiv p$ design such a privacy rating can not be reverted. This might be an issue, for instance if in the future $\mathcal{A}$ and $\mathcal{B}$ meet in person and exchange their public keys; they can then perform the handshake and $\mathcal{B}$ would be able to trust $\mathcal{A}$, but $\mathcal{A}$ would not be able to trust $\mathcal{B}$ in her device. In this case though, $\mathcal{M}$ misleading $\mathcal{A}$ to mistrust the intended partner is closer to a Denial of Service (DoS) attack but does not represent a threat to privacy.

We conclude that the execution of the $p \equiv p$ protocol fulfills the claimed security goals, i.e., after a successful handshake there is no undetectable way for $\mathcal{M}$ to modify the exchanges between $\mathcal{A}$ and $\mathcal{B}$, given that every message between them is always sent encrypted and signed with the corresponding keys. As a consequence, the privacy, authentication and integrity of the messages is preserved. Also, entity authentication is achieved by the $p \equiv p$ trust establishment protocol. These results depend on the assumptions of $p \equiv p$ residing in a

[3] https://www.dropbox.com/s/ste22xe2zfj9bnt/fullPepProtocol.pv?dl=0.

secure environment, of a secure second channel for the trustwords comparison and of $p \equiv p$ users owning a single instance of $p \equiv p$ with a single email account.

6.5 Limitations

This analysis focuses solely on the technical specification of the key distribution and handshake protocols. Social attacks such as impersonation or phishing are however still possible; for instance $\mathcal{M}$ can create a fake email account related to $\mathcal{A}$'s name and then use it to send $\mathcal{B}$ an email attaching $\mathcal{M}$'s public key and contact details. If $\mathcal{B}$ has never met $\mathcal{A}$, a handshake via trustwords comparison with $\mathcal{M}$ would succeed given that both partners are indeed executing the protocol, but the human $\mathcal{B}$ thinks that he is interacting with the human $\mathcal{A}$.

The assumption of perfect cryptography implies that we consider the libraries implementing cryptographic operations to be correct. Implementation flaws in $p \equiv p$ and side-channel attacks are not considered either; however, we highlight the requirement for the software to ensure that the trustwords database provided contains exactly the same data in all the distributions, to prevent introducing false mismatches during the trustwords generation.

7 Further Directions and Concluding Remarks

We reported a symbolic security analysis of the specifications of $p \equiv p$ protocols for key distribution and authentication, validating the exchange of authenticated end-to-end encrypted email between two $p \equiv p$ trusted peers. Here, we conclude by discussing some approaches that we have considered to extend our analysis.

How humans behave when comparing trustwords is not considered in this work; yet, incorrect input from users, such as mistrusting a trusted peer or vice-versa, might introduce security flaws. These situations happen, for instance, when users verify only the first two words of the list or when they click the trustwords confirmation button without comparing the trustwords. A formal model of human errors in human-to-machine authentication protocols is proposed in [8]; adapting such an approach to studying further the mentioned scenarios could give insights into how flaws introduced by users can be prevented. Understanding the causes and frequency of incorrect behavior requires a different kind of analysis mainly in the scope of usable security.

Regarding the decentralization of keys, we observe that trusting the user device instead of a third party key server could represent an issue, for instance if the user misplaces his device and does not have a protected repository. A comprehensive systematization and evaluation of current architectures and protocols for securing email is presented in [11], where authors discuss approaches achieving the strongest guarantees and their adoption decisions.

Since protocols for IM in general provide stronger security guarantees than those for email [11,25], we speculate whether solutions for automating IM security can be applied in the context of email. The Signal protocol [3], for instance, performs key agreement by mixing multiple Diffie-Hellman shared keys (X3DH)

and refreshing keys for every message exchange (double-ratchet), so that earlier keys cannot be calculated from later ones. The protocol has been formally analyzed and proved secure regarding secrecy and authentication of message keys [12]. The underlying reason preventing $p \equiv p$ from adopting a similar approach, hence upgrading security guarantees while relying less on the user, is Signal' use of a central server as a deposit for all the public keys involved and which is assumed to be trusted. This contradicts the decentralized paradigm adopted in $p \equiv p$'s design.

Following $p \equiv p$'s line of automating the processes as reasonably as possible, an idea to consider is how to automatically derive the trust from shared contacts with peers already trusted; a sort of an automatic web of trust. While there are many important considerations, for instance, how to get knowledge of shared contacts without violating privacy, we believe that this could be a direction worth studying.

As in the case of $p \equiv p$, in many systems that involve human-to-human authentication such a task is not mandatory to provide a service, but rather used to upgrade the security; therefore, users tend to neglect this step. Studying causes and solutions for those problems could be interesting from a usability perspective.

Finally, given that the human-to-human authentication relies on the trustwords shown to the user, as a next step we plan to verify $p \equiv p$'s trustwords generation function. Our approach considers taking advantage of the protocol verifier Tamarin, which recently added support for XOR operations [16]. Additionally, we foresee a verification closer to the implementation in the computational model.

Acknowledgments. Authors are supported by the project pEp Security SA/SnT "Protocols for Privacy Security Analysis".

References

1. OpenPGP. https://www.openpgp.org/
2. PGP word list. https://en.wikipedia.org/wiki/PGP_word_list
3. Signal technical specifications. https://signal.org/docs/
4. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: Acm Sigplan Notices, vol. 36, pp. 104–115. ACM (2001)
5. Basin, D., Cremers, C., Dreier, J., Meier, S., Sasse, R., Schmidt, B.: Tamarin prover. https://tamarin-prover.github.io/
6. Basin, D., Cremers, C., Meier, S.: Provably repairing the ISO/IEC 9798 standard for entity authentication. J. Comput. Secur. **21**(6), 817–846 (2013)
7. Basin, D., Dreier, J., Hirschi, L., Radomirovic, S., Sasse, R., Stettler, V.: A formal analysis of 5G authentication. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1383–1396. ACM (2018)
8. Basin, D., Radomirovic, S., Schmid, L.: Modeling human errors in security protocols. In: 2016 IEEE 29th Computer Security Foundations Symposium (CSF), pp. 325–340. IEEE (2016)

9. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: 14th IEEE Computer Security Foundations Workshop, pp. 82–96. IEEE (2001)
10. Blanchet, B.: Security protocol verification: symbolic and computational models. In: Degano, P., Guttman, J.D. (eds.) POST 2012. LNCS, vol. 7215, pp. 3–29. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28641-4_2
11. Clark, J., van Oorschot, P.C., Ruoti, S., Seamons, K., Zappala, D.: Securing email. arXiv preprint arXiv:1804.07706 (2018)
12. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. In: 2017 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 451–466. IEEE (2017)
13. Cremers, C.: Key exchange in IPsec revisited: formal analysis of IKEv1 and IKEv2. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 315–334. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23822-2_18
14. Dechand, S., Schürmann, D., Busse, K., Acar, Y., Fahl, S., Smith, M.: An empirical study of textual key-fingerprint representations. In: 25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 193–208 (2016)
15. Dolev, D., Yao, A.C.: On the security of public key protocols. In: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science, SFCS 1981, pp. 350–357. IEEE Computer Society, Washington, DC (1981)
16. Dreier, J., Hirschi, L., Radomirovic, S., Sasse, R.: Automated unbounded verification of stateful cryptographic protocols with exclusive OR. In: 2018 IEEE 31st Computer Security Foundations Symposium (CSF), pp. 359–373. IEEE (2018)
17. (IETF), I.E.T.F.: IANA registration of trustword lists. https://tools.ietf.org/html/draft-birk-pep-trustwords-03
18. (IETF), I.E.T.F.: pretty Easy privacy (pEp): privacy by default. https://www.ietf.org/id/draft-birk-pep-03.txt
19. Lowe, G.: Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 147–166. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61042-1_43
20. Lowe, G.: A hierarchy of authentication specifications. In: Proceedings 10th Computer Security Foundations Workshop, pp. 31–43. IEEE (1997)
21. Mauw, S., Cremers, C.: Operational Semantics and Verification of Security Protocols. Springer, Heidelberg (2012)
22. Pretty Easy Privacy: pep source code. https://pep.foundation/pep-software/index.html
23. Pretty Easy Privacy: pep user documentation. https://www.pep.security/docs/index.html
24. The Radicati Group: Email Statistics Report, 2018–2022. Technical report (2018)
25. Unger, N., et al.: SoK: secure messaging. In: 2015 IEEE Symposium on Security and Privacy, pp. 232–249. IEEE (2015)
26. Vazquez-Sandoval, I., Lenzini, G.: Experience report: how to extract security protocols' specifications from C libraries. In: IEEE 42nd Annual COMPSAC 2018, Tokyo, Japan, Vol. 2, pp. 719–724 (2018)
27. Whitten, A., Tygar, J.D.: Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In: USENIX Security Symposium, vol. 348 (1999)
28. Zimmermann, P.R.: The Official PGP User's Guide. MIT Press, Cambridge (1995)