PakeMail: Authentication and Key Management in Decentralized Secure Email and Messaging via PAKE

Itzel Vazquez Sandoval[1(✉)], Arash Atashpendar[1,2], Gabriele Lenzini[1], and Peter Y. A. Ryan[1]

[1] SnT, University of Luxembourg, Esch-sur-Alzette, Luxembourg
{itzel.vazquezsandoval,gabriele.lenzini,peter.ryan}@uni.lu
[2] itrust consulting s.á r.l., Niederanven, Luxembourg
atashpendar@itrust.lu

Abstract. We propose the use of password-authenticated key exchange (PAKE) for achieving and enhancing entity authentication (EA) and key management (KM) in the context of decentralized end-to-end encrypted email and secure messaging, i.e., without a public key infrastructure or a trusted third party. This not only simplifies the EA process by requiring users to share only a low-entropy secret such as a memorable word, but it also allows us to establish a high-entropy secret key. This approach enables a series of cryptographic enhancements and security properties, which are hard to achieve using out-of-band (OOB) authentication. We first study a few vulnerabilities in voice-based OOB authentication, in particular a combinatorial attack against lazy users, which we analyze in the context of a secure email solution. We then propose tackling public key authentication by solving the problem of *secure equality test* using PAKE and discuss various protocols and their properties. This method enables the automation of important KM tasks such as key renewal and future key pair authentications, reduces the impact of human errors and lends itself to the asynchronous nature of email and modern messaging. It also provides cryptographic enhancements including multi-device synchronization, and secure secret storage/retrieval, and paves the path for forward secrecy, deniability and post-quantum security. We also discuss the use of auditable PAKEs for mitigating a class of online guess and abort attacks in authentication protocols. We present an implementation of our proposal, called PakeMail, to demonstrate the feasibility of the core idea and discuss some of its cryptographic details, implemented features and efficiency aspects. We conclude with some design and security considerations, followed by future lines of work.

Keywords: Password-authenticated key exchange · Public key authentication · Key management · Secure email · Secure messaging · Implementation · Decentralized trust model

1 Introduction

Largely owing to cryptography, modern messaging tools (e.g., Signal) have reached a considerable degree of sophistication, balancing advanced security features, ranging from end-to-end encryption to forward secrecy and deniability, with high usability.

M. S. Obaidat and J. Ben-Othman (Eds.): ICETE 2020, CCIS 1484, pp. 102–128, 2021.
https://doi.org/10.1007/978-3-030-90428-9_5

However, this has not been the case for email, even though it has a long history and remains the most pervasive and interoperable form of digital communication, with billions of emails exchanged on a daily basis [15]. Yet, secure messaging and email share two long-standing challenges, namely entity authentication and key management.

The primary concern is entity authentication, which invariably involves a mechanism that associates some cryptographic material with an identity, e.g., public key authentication. Key management, affecting email more acutely, is intertwined with authentication and the need for automating it has been known for a long time, e.g., see [39].

Over the years, several methods have been established for accomplishing key authentication, and indirectly key management: manual validation of key fingerprints, web of trust, public key infrastructure (PKI) and hierarchical validation, public key directories as well as server-derived public keys such as identity-based encryption (IBE).

The set of viable techniques becomes much smaller once we consider a decentralized setting, i.e., without a PKI or a trusted third party (TTP). In this context, approaches based on the use of out-of-band (OOB) channels and short authentication string (SAS) comparisons (see Sect. 1.3) have received a great deal of attention from the research community. Due to the required user interaction in these approaches—e.g., manually verifying public key fingerprints—usability plays a key role in achieving authentication. Therefore, reducing the gap between security and usability by finding optimal trade-offs has been a central theme of research for decades, with a plethora of long-standing open problems [15,46].

In an attempt to improve usability in the entity authentication process, Alexander and Goldberg [3] proposed a modified solution to the socialist millionaires' problem (SMP) by Boudot et al. [14], also known as secure equality test, for authentication in the off-the-record messaging (OTR) protocol [13]. To the best of our knowledge, this is the only work that proposes an approach for key validation, mainly suitable for online (synchronous) settings, that relies on users pre-sharing a low-entropy secret.

Here we revisit the problem of public key authentication in a decentralized setting to propose a user-friendly and robust approach based on password-authenticated key exchange (PAKE) for solving SMP via low-entropy secrets. These secrets are not expected to be sampled from a large, uniformly distributed space, but rather from a small set of values, e.g., typical human-memorable passwords or pin numbers. The task of SMP boils down to two parties verifying equality of their inputs $\pi_{\mathcal{A}}$ and $\pi_{\mathcal{B}}$ in a zero-knowledge manner such that by the end they learn nothing but the boolean result of the test.

Apart from offering improved usability properties and eliminating a host of vulnerabilities present in OOB-based protocols, as discussed in Sect. 3, we show how the PAKE-generated secret key can be used to pave the path towards providing a series of cryptographic enhancements in secure email and messaging. These include automation in key management and key renewal, forward secrecy in a symmetric-key setting, deniability, post-quantum security, secure secret retrieval and storage, and auditability for mitigating a certain class of online guess and abort attacks in authentication protocols.

To demonstrate the feasibility of the proposed approach, we provide a complete implementation of the core ideas. This also shows how the suggested approach would not only work naturally in the context of secure messaging, but also in the inherently asynchronous setting of email.

By applying PAKE to this problem, we advance the state-of-the-art in the use of shared low-entropy secrets for entity authentication, an idea considered only in [3]. Moreover, while SMP is a subproblem solved naturally by PAKE, the latter has not been applied to tackle the problem of authenticating public keys in decentralized settings.

1.1 Motivation

Despite its crucial importance, secure email and messaging solutions tend to brush aside the entity authentication step and as such, this feature often tends to go unnoticed,[1] which contributes to users neglecting the process. Solutions that do consider the entity authentication problem in decentralized non-PKI environments, typically rely on users correctly executing a manual comparison, which has been repeatedly shown to be error-prone and inconvenient for users (e.g. [26]). Our incentive for replacing OOB authentication with a cryptographic protocol lies in the significant impact of failures occurring in methods highly-dependent on user behavior, which could completely jeopardize security.

Our motivation for using PAKE—a method that does not seem to have enjoyed enough recognition due to a lack of mature implementations, reluctance towards client side cryptography, patent-encumbered designs and perhaps even unawareness of its usefulness—is grounded not only in its independence from a PKI or a TTP, but also in its provision of a zero-knowledge (ZK) solution for the secure equality test problem using a low number of rounds, which makes it compatible with asynchronous settings, and in the fact that it enables additional cryptographic enhancements.

Additionally, the need for addressing common challenges such as key management automation and device synchronization spurred us on. By implementing our PAKE-based solution for entity authentication, we address two open problems in secure email and messaging [15,46]: bridging the gap between known theoretical results and real-world solutions, and the need for more robust authentication methods that also improve the trade-off between security and usability in secure solutions.

1.2 Contributions and Structure

A brief review of the state-of-the-art is covered in Sect. 1.3, followed by an overview of background concepts in Sect. 2. In Sect. 3, we discuss a few vulnerabilities in the use of OOB channels for authentication, including a partial preimage attack targeting lazy users, which we analyze in the context of the p ≡ p [34] secure email solution.

Next, in Sect. 4 we describe how solving the *secure equality test* using PAKE leads not only to entity authentication but also to the establishment of a shared high-entropy secret key that can be used to achieve additional cryptographic tasks and properties. We provide a concrete illustrative scheme along with an analysis of various PAKE constructions and properties relevant for our work, and briefly analyze network transport mechanisms and security.

[1] For example, in order to access the authentication menu in Signal or WhatsApp, users need to (1) select a chat (2) click on the contact's name (3) select "View safety number/Encryption".

In Sect. 5, we elaborate on the said cryptographic enhancements, such as inattentive user resistance, automated key renewal, automated future key pair authentication and multi-device synchronization, along with security properties such as deniability, forward secrecy, post-quantum security, auditability for detecting guess and abort attacks, and secure secret storage and retrieval with applications in email and secure messaging.

Extended Version. The main contributions of this work, which is an extended version of our previous paper [48], are as follows:

- PAKE-based Public Key Authentication and Key Exchange over Email (PakeMail): in Sect. 6, we present a complete implementation of the main idea for a PAKE-based authentication and key management approach in the context of decentralized secure email, serving as a proof of feasibility. The source code, along with the corresponding documentation, can be found at [4].
- In Sects. 2 and 4 we provide more theoretical details on PAKE protocols and cryptographic elements relevant for a concrete implementation.
- We provide an analysis in Sect. 5.4, comparing our proposal with state-of-the-art trust establishment approaches.
- We extend and improve our descriptions of the cryptographic enhancements in Sect. 5, including the notion of secure secret storage and retrieval, a variant of which has been recently implemented for the Signal messaging system, but using a combination of different cryptographic constructions.

In Sect. 7 we review the main security properties of our solution and elaborate on a few methods for low-entropy secret agreement and improving usability. We conclude in Sect. 8 with a more detailed outline of future directions and open questions.

1.3 Related Work

Unger et al. [46] and Clark et al. [15] provide extensive systematic surveys covering numerous aspects of secure messaging and email. We limit ourselves to the decentralized setting without elaborating on the drawbacks of web of trust approaches covered in the above mentioned works.

The literature contains a sizeable body of work on OOB-based approaches, considered first by Rivest [36], many of which are inspired by the original work of Vaudenay [47] based on SAS comparisons, e.g., [26,27,32,45], to name a few. This area has also been investigated by the formal methods community, see e.g. [18] for a recent formal analysis of SAS-based schemes in the symbolic model.

As for low-entropy secret-based authentication, to the best of our knowledge, the only work in the literature is by Alexander and Goldberg [3] using a modified version of the SMP protocol by Boudot et al. [14] for improving the OOB-based authentication process in off-the-record messaging (OTR) [13]. OTR is a cryptographic protocol originally designed by Borisov and Goldberg, aimed at enabling encrypted, authenticated and deniable instant messaging conversations with forward secrecy; this protocol was proposed as an alternative to PGP for "casual" conversations.

The variant proposed by Alexander and Goldberg sacrifices the fairness property of [14] for efficiency and the authentication process requires $\mathcal{A}$ and $\mathcal{B}$ to be both online when entering their secret and for the subsequent exchange of messages.

2 Framework and Preliminaries

We use $\mathcal{A}$ and $\mathcal{B}$ to refer to honest parties Alice and Bob, and $\mathcal{M}$ for the adversary, Mallory. We use $\leftarrow\!\!\$$ to denote an element sampled uniformly at random, and $\|$ to denote concatenation. We denote low-entropy secrets provided by users with π.

Security Model. We consider the standard Dolev-Yao model [21]. We do not assume any additional trusted infrastructure. In one of our proposed methods for transport protocol, we assume the existence of untrusted buffer/relay servers, somewhat akin to the ones used in the design of Signal or OTR4 (see Sect. 4.4). Regarding PAKEs, we will consider several constructions in Sect. 4, largely proven secure in the so-called BPR model [8] under various hardness assumptions.

System Requirements. Our proposal does not require any format modifications and preserves compatibility between existing email clients and servers; therefore, we assume standard requirements for email transfer. As for secure messaging, we do not introduce any extra trust assumptions and no additional infrastructure requirements. Any exchanges relayed or buffered by intermediate servers can be done by untrusted ones.

Cryptographic Notions. Due to space limitations, we assume familiarity with common cryptographic concepts, in particular with Diffie-Hellman (DH)-based computational hardness assumptions.

We discuss schemes based on the Ring Learning With Errors (RLWE) problem, a special case of the Learning With Errors (LWE) problem whose security may be reducible to the hardness of solving the Shortest Vector Problem (SVP) in lattices, for which no efficient quantum algorithms are known, thus conjectured to be quantum-secure. Post-quantum (PQ) cryptography encompasses schemes that are considered to be safe against adversaries equipped with scalable, cryptographically relevant quantum computers.

We use $\mathsf{KDF}(s)$ to denote a key derivation function that takes a source s of keying material, typically with a fair amount of entropy but not uniformly distributed, and produces one or more cryptographically strong secret keys, see [29] for details. We denote with $\mathsf{MAC}(k,m)$ a keyed message authentication code scheme that computes a tag on m under key k. We use "Curve25519" to refer to the underlying elliptic curve used in the elliptic-curve-Diffie-Hellman function by Bernstein [10].

Socialist Millionaires' Problem. In the realm of secure multi-party computation (MPC), Yao's millionaires' problem [50] is a famous example in which two parties want to find out whose input is greater without revealing any more information on the actual value. SMP is a variant of this and a ZK proof of knowledge protocol, with the difference that the parties only wish to know if their inputs are equal.

A series of works has been dedicated to solving SMP, including a well-known solution by Boudot et al. [14] that provides a fair and efficient protocol, where fairness roughly means that no party can evaluate the function and walk away with the result without the other party learning the output.

Garay et al. [23] showed that the fairness and the security definition of [14] are not compatible with the simulation paradigm and that their solution would not be secure

when composed concurrently; they present a construction that can be composed arbitrarily, with similar complexity results.

PAKE. Password-authenticated key exchange (PAKE) protocols enable the establishment of secure channels without the need for a PKI, TTP or empirical OOB channels. In essence, they address a secure two-party computation problem and allow two parties $\mathcal{A}$ and $\mathcal{B}$ who share only a low-entropy secret/password $\pi \in \mathcal{D}$, with $\mathcal{D}$ being some relatively small dictionary, to agree on a high-entropy cryptographic secret key k, using π for authentication. Since the seminal work of Bellovin and Merritt [9], numerous PAKE protocols have been proposed, which largely fall into the two categories of balanced (symmetric) and augmented (or asymmetric), referred to as aPAKE. The latter stores one-way mappings of passwords on the server side in client-server settings.

Intuitively, a core property of PAKE is that a run of the protocol should not leak any information about the password. Moreover, apart from protection against man-in-the-middle (MITM) attacks and variants thereof such as replay/reuse and mixing attacks, they should also provide security against offline dictionary attacks by passive and active adversaries. While due to the use of low-entropy passwords, any PAKE protocol is vulnerable to an online guessing attack, the goal is to ensure that at most one test per run constitutes the optimal attack strategy for an active $\mathcal{M}$ interacting with a party. Similar to SMP, $\mathcal{M}$ can mask failed guessing attempts as network failures, thus allowing numerous attempts without raising suspicion. This is in general unavoidable, however, we will see in Sect. 4 how a recent work by Roscoe and Ryan [38] can mitigate this.

Most well-known PAKE protocols rely on different variants of the Diffie-Hellman Problem (DHP), which means that their security is ultimately reduced to that of the Discrete Logarithm Problem (DLP). These typically make use of a cyclic group $\mathbb{G}$ of prime order p, generated by $g \in \mathbb{G}$, along with a hash function H, modelled as a random oracle, plus a few other public parameters, e.g., $M, N \in \mathbb{G}$ in the case of SPAKE2, as shown in Fig. 1. Moreover, the passwords are viewed as elements of $\mathbb{Z}_p$ (obtained by hashing user passwords to some $\pi \in \mathbb{Z}_p$), which are used to blind the DH terms by multiplying these terms by randomly chosen elements of $\mathbb{G}$ raised to π, e.g., $g^x \cdot M^\pi$, where $x \leftarrow\$ \mathbb{Z}_p$. The final session key is then derived by computing a hash of the entire transcript T_P of a run of protocol P, which includes all of the parties' public and private values, the user identities, the DH terms and the password, i.e., $k = H(\pi, id_\mathcal{A}, id_\mathcal{B}, T_P)$.

Often, how passwords are agreed upon and the actual details pertaining to the exchange of user identities are left out, i.e., deferred to higher-level applications implementing the protocol. It is typically required for a higher-level application to be able to refer to a session using a globally unique identifier, a channel binding often also called a session ID, which for technical reasons rooted in composability should be computed as a function of user instance roles and information exchanged over the network during the execution of the protocol, e.g., user IDs and public randomness. The session identification (ID) is usually defined as the transcript T_P of the communication/conversation between $\mathcal{A}$ and $\mathcal{B}$, which can also be viewed as a random variable, being a function of the random values generated by $\mathcal{A}$ and $\mathcal{B}$. These, among other things, protect against MITM, unknown key share attacks and replay attacks.

3 Pitfalls in Out-of-Band Authentication

In OOB authentication, users typically compare some representation of a cryptographic hash (fingerprint) of their partners' public keys via a separate authenticated channel. This representation is usually in the form of a list of words, numbers or images.

Strong security properties can be achieved if users execute the manual verification correctly. Yet, the difficulty of having users do the corresponding tasks correctly while finding the right balance between usability and security is the root cause of security drawbacks, which have been amply discussed by research on fingerprint and SAS comparison via OOB channels (e.g., [26,27,45]). Naturally, usability studies encourage the replacement of manual comparisons by automated software whenever possible [45]. Some of the problems rooted in OOB authentication are as follows.

Selection of an Adequate OOB Channel. In practice, the theoretical and strong authentication requirements of OOB methods are not easy to satisfy. While face-to-face conversations provide a strong authenticated channel [32], they are often not viable. It is usually assumed that an OOB channel cannot be forged, but it can be blocked, overheard, delayed or replayed. Typical instantiations are done via voice-based channels, e.g., a phone call. However, some already consider voice-based SAS comparison to be obsolete from a security perspective [46] as nowadays messages can be forged by voice synthesizers with a small sample of the victim's voice. Indeed, a voice impersonation attack on users comparing PGP words [41] reported the fake voice to be indistinguishable in about 50% of the cases.

Social Engineering Attacks. Although there are multiple options for users to interact via OOB, little effort has been gone into designing precise protocols for humans to carry out the authentication process in a privacy-preserving and fair manner. This leads to various attack vectors based on misleading users as opposed to finding technical vulnerabilities. For instance, without knowing the authentication value, $\mathcal{M}$ can fool $\mathcal{A}$ into trusting her key by pretending to be $\mathcal{B}$, asking $\mathcal{A}$ to read her fingerprint representation first, and then simply confirming that the fingerprints match.

3.1 Inattentive Users and Partial Preimage Attacks

Inattentive and Lazy Users. Here we consider users misreading words (inattentive) or comparing only subsets of them (lazy). A recent paper by Naor et al. [31] analyzes approaches based on SAS authentication that are vulnerable to MITM attacks w.r.t. lazy users. For instance, the approach in WhatsApp and Signal would be flawed if users compared only either the first or the second half of the value, since it would amount to verifying only one peer's fingerprint. To fix this, the authors propose an influence spreading technique in which every bit of the value to be authenticated influences the generation of each element of the OOB representation.

Partial Preimage Attack. Dechand et al. [17] study an attack aimed at finding a partial preimage for a fingerprint verified by lazy users; specifically, they assume that users check subsets of bits at the boundaries and in the middle.

We now give a more detailed description of their analysis. Let p denote the probability of finding a partial preimage for a given fingerprint f and q its complementary

event. To calculate $p = 1 - q$, we work out q (i.e., the absence of partial preimages for a specific bit permutation). Let b be the length of the fingerprint f and assuming that r consecutive boundary bits are fixed (checked by the user), in this case, the leftmost and rightmost bits of f, we let ℓ denote the number of remaining bits in the middle from which a possible variation of u bits could be fixed, i.e., checked by the user.

Thus, we have $2 \cdot r + u$ fixed bits that the adversary cannot invert without the user noticing. Valid preimages can thus be obtained by flipping up to $t = \ell - u$ bits within the middle bits; by removing these from the total space of size 2^b, we obtain the number of invalid ones. With k denoting a given number of positions to modify, the valid strings are then given by $\binom{\ell}{k}$ choices of positions to flip. Thus, q is given by

$$q = \frac{2^b - \sum_{k=1}^{t} \binom{\ell}{k}}{2^b}. \tag{1}$$

Expressing p as a function of the computational effort in terms of e brute-force attempts, we have $p = 1 - q^e$. To estimate the number of steps needed for finding partial preimages with a success probability $\geq p$, we simply compute $e = \log_q(1 - p)$. Expressing e in base 2 gives results comparable to [17].

3.2 Case Study

Pretty Easy Privacy (p ≡ p) is a software aimed at providing usable privacy-by-default in email via end-to-end opportunistic encryption. The tool largely automates key management tasks. The public key of a user is attached to outgoing emails when a key of the recipient has not been stored. Received keys are automatically stored for future use (*trust-on-first-use*) and outgoing emails are automatically encrypted when a public key of the intended receiver is available. This approach requires neither a PKI nor a TTP.

Similar to the PGP word list, $p \equiv p$ *trustwords* [12] are natural language words that two users compare via a low-bandwidth OOB authenticated channel to prevent MITM attacks. The trustwords generation algorithm $\mathsf{tws}(\cdot)$ is a deterministic algorithm that runs locally taking as input the public key of the peer obtained by email and the user's own public key. Informally, $\mathsf{tws}(\cdot)$ performs an XOR over the fingerprints of each of the input arguments, and then maps each block of 16 bits from the resulting 160-bit long string to a word in a predefined dictionary of size 2^{16}, thus yielding a list of ten words.

To encourage users to perform the OOB authentication, by default p ≡ p shows only five words; this means that the peers compare the first 80 out of the 160 bits of a PGP fingerprint, assuming that they check all the words. Since an "influence spreading" property, similar to Naor et al.'s, is already present, the best adversarial strategy is a brute-force attack over the public key space requiring $O(2^{80})$ steps to find a key k such that the first 80 bits of $\mathsf{fpr}(k)$ are equal to those of $\mathsf{fpr}(\mathsf{pk}_B)$, with pk_B being the public key of $\mathcal{B}$.

We consider lazy users and compute estimates for partial preimage attacks similar to the one presented above. We consider the two cases where, out of five words, the user verifies (i) the first and last words as well as two from the middle (ii) the first and last words, along with one of the three in the middle. Let $b = 80$, $\ell = 48$ and for (i) we have $u = 32$ and we get $e \approx 2^{38}$; for (ii), with $u = 16$, we get $e \approx 2^{32}$. These results show

that $\mathcal{M}$ would succeed with costs equal to and lower than the computational power estimated for an average adversary [17].

Clearly the decision to show five words instead of ten by default needs to be reconsidered. Users might feel less annoyed by having to compare fewer words, however, its adverse effect on security is considerable as it practically renders brute-force attacks viable.

4 Authentication in Secure Email and Messaging via PAKE

We now show how PAKE can be used to perform a secure equality test and thereby authentication, yielding a more efficient solution, compared to OOB methods and the OTR approach, with better security guarantees and further cryptographic features.

Trust Establishment using Low-entropy Secrets. For now, we assume that $\mathcal{A}$ and $\mathcal{B}$ share a low-entropy secret—e.g., a short password—either agreed upon beforehand or decided by posing and answering a question at the beginning of the mutual authentication.

Intuitively, the goal is for $\mathcal{A}$ and $\mathcal{B}$ to authenticate their public keys via a secure equality test of their respective secrets π_A and π_B, without revealing any information about the latter; hence the need for a zero-knowledge protocol guaranteeing that upon termination of the protocol, the resulting transcript of the exchanges does not leak any information on π_A and π_B, allowing $\mathcal{A}$ and $\mathcal{B}$ to learn only whether or not their respective secrets were equal. In addition, it should not be possible for $\mathcal{M}$ to brute-force the password via offline dictionary attacks. Thus, $\mathcal{M}$'s only strategy would amount to making online attempts.

4.1 Public Key Authentication via PAKE

To determine at the end of a PAKE run whether the user secrets π_A and π_B are equal, without revealing anything else, we enforce explicit authentication using a key confirmation (KC) step after the key establishment phase. While this step may be optional in the general case for PAKE protocols, here it would be necessary in order to bind the cryptographic material with an identity. The information that $\mathcal{A}$ and $\mathcal{B}$ wish to authenticate—e.g., public key fingerprints for email addresses or phone numbers in Signal—can be incorporated either into the KC phase or into the initial user secrets.

Next, we delve into the details of how this can be achieved using a concrete PAKE protocol. The literature contains several well-studied instances of PAKE, therefore, we first pick a candidate to demonstrate how it can be used to validate public keys, and then compare a few prominent schemes according to specific properties of interest. For the moment, we do not focus on engineering aspects related to (a)synchronicity and message transport mechanisms, but we will come back to these in Sects. 4.4 and 6.

4.2 An Instantiation Based on SPAKE2

For illustration, in Fig. 1 we propose an extension of SPAKE2, a one-round protocol, with a KC step to achieve explicit authentication, thus binding a public key to

an entity. This yields a 2-round scheme, the minimum when KC is enforced; see [28] for optimal-round PAKEs. For KC we can use the generic refresh-then-MAC transformation. Despite its long history, this transform was only recently proved secure [22].

With $\mathbb{G}$ being a finite cyclic group of prime order p, generated by an element $g \in \mathbb{G}$, let $\mathbb{G}, g, p, M \leftarrow\$ \mathbb{G}, N \leftarrow\$ \mathbb{G}$ and hash function $H(\cdot)$ denote public parameters and $\pi \in \mathbb{Z}_p$ the private low-entropy secret, with the user password assumed to be appropriately mapped to an element in $\mathbb{Z}_p$. The parties perform the key exchange phase, as shown in Fig. 1, which concludes with the generation of a symmetric key. Upon termination of the key establishment, $\mathcal{A}$ and $\mathcal{B}$ each use the symmetric key to carry out a key-refreshing step via a key derivation function in order to generate fresh MAC keys (for both parties), along with a new session key, K, which will be the final shared secret key. Next, under the freshly generated keys, they each compute a MAC on the fingerprints of both parties' public keys. The authentication now amounts to exchanging and verifying the obtained tags τ^a and τ^b, i.e., to see if the received tag and its locally computed counterpart match.

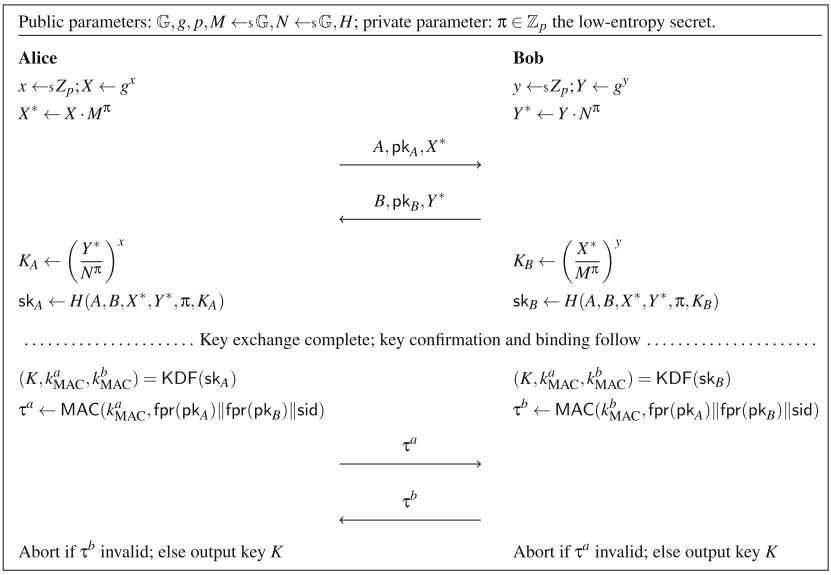

Fig. 1. pk authentication using SPAKE2 with refresh-then-MAC key confirmation for entity binding. (Originally presented as such in our previous work [48]).

The addition of the KC step increases the number of rounds and flows to 2 and 4, respectively. Note that this is merely an illustrative example and as already mentioned, other possibilities for KC do exist, some of which offer additional properties. For instance, in [7] the authors showed that a modified version of SPAKE2, called PFS-SPAKE2, coupled with a KC step can achieve perfect forward secrecy (PFS) at the cost of increasing the number of rounds from 1 to 3. More recently, Abdalla et al. [1]

showed that SPAKE2 does indeed satisfy PFS even without KC under a different hardness assumption. They also prove a version with a KC step (yielding a better bound) almost identical to the one given in Fig. 1, except that the protocol has one less flow.

Alternatively, the public key fingerprints can be embedded in the secret π, but note that even in that case, the KC step cannot be skipped as an explicit authentication of the public keys would be still needed. More precisely, we would let $\pi = \pi' \| \mathsf{fpr}(\mathsf{pk}_{\mathcal{A}}) \| \mathsf{fpr}(\mathsf{pk}_{\mathcal{B}})$, where π' denotes the original user provided secrets, and we would compute the tags as $\tau^a \leftarrow \mathsf{MAC}(k^a_{\mathrm{MAC}}, \mathsf{sid})$, where the session identifier sid is defined as the transcript of conversation between $\mathcal{A}$ and $\mathcal{B}$, with τ^b computed similarly. The IETF documents for SPAKE2[2] and J-PAKE[3] provide similar one round KC methods.

Table 1. Comparison of PAKE protocols. (Originally presented in [48].)

Protocol	Rounds/ Flows	KC	Forward secrecy	Security model	Hardness assumption
SPAKE2	1/2	✗	✓	ROM	CDH
PFS-SPAKE2	3/3	✓	✓	ROM	CDH
OPAQUE	2/3	✓	✓	ROM	OMDH
J-PAKE	2/4	✗	✓	ROM-AAM	DSDH
KV-SPOKE	1/2	✗	-	CRS	DDH
RLWE-PAK	3/3	✓	✓	ROM	RLWE
RLWE-PPK	2/2	✗	✓	ROM	RLWE

ROM: Random Oracle Model; AAM: Algebraic Adversary Model; CRS: Common Reference String
DH: Diffie-Hellman; CDH: Computational DH; DDH: Decisional DH; DSDH: Decision Square DH;
OMDH: One-More DH; RLWE: Ring Learning With Errors

Note that the inclusion of pk_A and pk_B in the key exchange phase in Fig. 1 merely illustrates that they could be exchanged in one round. However, this exchange can be decoupled from the original SPAKE2 specification; indeed, the exchange of public keys may occur long before their authentication. This allows us to preserve the original description of the protocol and the computation of the transcript; otherwise, the key fingerprints would have to be included in the SPAKE2 transcript and in turn, in the input of the hash function computing the session key. In our case, the security guarantee is independent from this particular choice due to our strict enforcement of explicit authentication: fingerprints are included in the computation of the transcript (or session ID) in the KC step. We will elaborate further on this in Sect. 6.

[2] https://tools.ietf.org/id/draft-irtf-cfrg-spake2-08.html.
[3] https://tools.ietf.org/html/rfc8236.

4.3 Selecting a PAKE Protocol

We consider a number of representative PAKE protocols and analyze their properties w.r.t. our use case: SPAKE2 [2], OPAQUE [25], PFS-SPAKE2 [7], J-PAKE [24], KV-SPOKE [28], RLWE-PAK and PPK [20]. PAKEs are typically evaluated according to the security model in which they are proven secure, support for forward secrecy, the number of rounds, along with their communication and computational complexity. The complexity related aspects become more relevant in a client-server setting wherein a server has to process a high number of requests and sessions in a short time span. In a decentralized peer-to-peer setting, such properties no longer play a major role.

In Table 1, we present some relevant properties of the said constructions. Except for RLWE-PAK and RLWE-PPK that make use of lattice-based cryptography, all other schemes are Diffie-Hellman-based. In terms of PQ security, this implies that the latter cases would not be quantum-safe, whereas the first two would provide conjectured quantum-security due to the underlying RLWE problem.

Minimizing the number of rounds is more important for secure email than it is for messaging, especially if the transport mechanism is based on attachments or hidden emails (see Sect. 4.4). As for secure messaging, this may be equally relevant for solutions that do not operate in a purely decentralized and peer-to-peer setting in which one may wish to reduce the load on relay or buffer servers, e.g., Signal or OTR4, but the number of rounds would in general be arguably less of a concern. Note that KC can be added to schemes that do not have it by default at the cost of an extra round.

Intuitively, the notion of forward secrecy (FS) captures the requirement that a long-term secret compromise should not result in prior session keys getting compromised and consequently the corresponding exchanges. Weak FS (wFS) refers to those schemes satisfying FS against passive adversaries who did not interfere in the previous sessions and perfect FS to those achieving the same against active adversaries. We will come back to this in Sect. 5.2.

We limit the discussion on security models to practical considerations. In the random oracle model (ROM), an ideal truly random function being accessible to the parties through oracle calls is typically instantiated using cryptographic hash functions, and the common reference string (CRS) model implies the accessibility of a random string to all parties, generated in a trusted way. The latter may be less obvious to implement in the case of email due to the constraints of decentralization given that the generation of the CRS would be typically done by a trusted party or via a secure MPC protocol, e.g., the decentralized CRS generation shown in [40]. Finally, regarding the RLWE-based schemes, their proofs are unfortunately in the ROM, as opposed to the quantum ROM (QROM), which would allow adversaries to query the random oracle in superposition.

4.4 Transport Mechanism

Email-Based Approach. Given the small number of rounds required by PAKE protocols, in the case of email we can afford to use standard email attachments or specially formatted hidden emails as message carriers, processed in the background by the email

client. Since we primarily deal with authentication, these exchanges would have minimal impact in terms of communication and computational complexity as the protocol would have to take place only once per peer.

For the case of attachments, a PAKE-based implementation could give $\mathcal{A}$ the option to enter her secret $\pi_{\mathcal{A}}$ upon sending her first email to $\mathcal{B}$, thus allowing the first flow of the protocol to occur via an attachment; the initial PAKE round would be completed when $\mathcal{B}$ replies after entering his secret $\pi_{\mathcal{B}}$. The subsequent exchange for the KC step can be done automatically.

Alternatively, the implementations could encapsulate cryptographic messages in specially crafted emails, kept hidden from the user (e.g., archived separately) and processed automatically—as $\mathrm{p} \equiv \mathrm{p}$ does for multi-device key synchronization.

Untrusted Server Approach. Although early instant messaging (IM) tools were entirely online services that maintained an active session for each conversation, modern IM tools follow an asynchronous model similar to that of email. For instance, both Signal and the latest version of OTR [33] achieve offline messaging by using "buffer servers" for hosting pre-key bundles that can be fetched without the other party being online.

We can use a similar mechanism to overcome transport engineering obstacles in email more elegantly, since all aspects related to the exchange of emails remain unchanged and thus interoperable. In fact, the use of an intermediate server would not introduce additional trust assumptions as the transcript of a PAKE protocol does not leak useful information to the adversary; such a server would be untrusted and any entity would be able to set up their own instance.

5 Enhancements to Secure Email and Messaging by PAKE

Our PAKE-based approach for authentication satisfies and improves a number of key properties related to security and usability that have been identified in the literature [46]. We first discuss how these properties are satisfied and then introduce novel uses of PAKE in secure email and messaging. Note that once a PAKE-generated symmetric key is established, subsequent PAKE instances can be run automatically via a chaining self-sustaining mechanism; moreover, while we primarily focus on enhancements for existing paradigms that depend on public keys—e.g. PGP-based or OTR-inspired systems such as Signal—one could also consider the benefits of transitioning to symmetric-key constructions, e.g., MAC-based authentication and symmetric-key encryption schemes.

5.1 Key Management and Authentication Improvements

The improvements presented here mainly deal with key management automation and error resilience.

Automation of Future Key Pair Authentications. Once authentication between $\mathcal{A}$ and $\mathcal{B}$ is bootstrapped from an initial PAKE, the authentication of new key pairs belonging to either $\mathcal{A}$ or $\mathcal{B}$ can be automated by using the PAKE-generated key as input, without prompting the users to yet again enter new secrets. Authentication of new keys is needed for instance when keys expire, when a new key pair needs to be associated with an

existing identity, or when new email addresses need to be associated with key pairs. These can be automatically authenticated by running a PAKE with the stored shared symmetric key as input. Note that each execution of a PAKE refreshes the stored PAKE-generated symmetric keys. Automating the authentication of future keys enables the achievement of the other properties in this category.

Immediate Enrolment. This property holds if when a user reinitializes their keys, other parties can verify and use them immediately. The PAKE-generated key allows to automate the new key exchange and the corresponding authentication as explained above.

Alert-Less Key Renewal. Complementing the previous property, this one refers to users not receiving alerts or warnings prompting them to take action when other parties renew their public keys. This would be automated similarly to immediate enrolment.

Low Key Maintenance. This property refers to minimizing users efforts related to key management tasks, such as signing keys or renewing expired keys. By achieving immediate enrolment and alert-less key renewal as explained above, the PAKE-based approach improves key maintenance too.

Inattentive User Resistance. As discussed earlier, manual OOB key/fingerprint verification methods are susceptible to human error and inattentiveness. In the PAKE-based approach, even if users enter the wrong password, the result would not be as catastrophic as trusting a key prepared by the adversary. At worst, it would be inconvenient as the authentication would fail, prompting the user to eventually repeat the process.

5.2 Cryptographic Properties Enabled by PAKE

Symmetric Key Cryptography. An immediate and rather evident advantage of using a PAKE protocol in this context is that the resulting cryptographic secret key can be already used for performing cryptographic tasks, whereas in a general PKI setting, upon authenticating the public keys, one would then typically make use of a Key Encapsulation Mechanism (KEM) in order to establish a symmetric key.

Perfect Forward Secrecy (PFS). Once, more popular in the context of secure messaging (e.g., Signal and OTR), PFS is now a requirement for cipher suites supported in TLS 1.3. PFS means that in the event of a password disclosure, previously derived session keys remain secure. To minimize the impact of a long-term key disclosure, one could implement a PAKE-chaining mechanism that automatically performs key rotations and periodically refreshes the symmetric key; this would provide limited windows of opportunity for $\mathcal{M}$ to compromise the channel, past which point, the fresh key would be secure again. If there is evidence that $\mathcal{M}$ has corrupted the channel, the cryptographic key would have to be discarded and replaced by a new PAKE execution. This refreshing paradigm might be expensive, however it would be relevant when PAKE-based approaches are used for synchronization purposes, either device-to-device or device-to-server, where PAKE can be used to both authenticate and establish a secure channel, thus providing PFS for the session keys used for syncing.

Several PAKE constructions provide PFS by default, some of which are listed in Table 1; moreover, PFS can be obtained by adding explicit authentication via a KC step

to constructions that do not have this property [8]. Alternatively, to improve efficiency we could resort to symmetric-key schemes that provide PFS, e.g., SAKE [5]. In this case, a PAKE can be used once to bootstrap authentication via a low-entropy secret and to generate the initial symmetric master key required by SAKE.

The use of PAKEs could for instance improve the approach based on regular sub-key rotations, adopted by the Sequoia-PGP project for adding FS to OpenPGP-based solutions; a PAKE-based solution could automate authentication in case the master key, certifying the short-term sub-keys, needs to be refreshed. For additional security, with slightly hampered usability, a separation of storage can be enforced by for example storing such PAKE long-term keys in dedicated hardware, e.g., hardware security modules or smart key storage devices such as YubiKey or Nitrokey, to protect against a device compromise; see Sect. 5.3 for more details on this.

Deniability. This is another subtle and fundamental property that has been of particular interest in recent secure messaging systems such as Signal and OTR. Deniable exchange, applied to tasks ranging from authentication to encryption, has a long and somewhat controversial history due to the subtleties in various existing security definitions. We limit ourselves to the case of key exchange and the seminal framework of Di Raimondo et al. [19], which provides security definitions in the simulation paradigm for deniable key exchange and authentication, where both message and participation repudiation are considered as requirements.

Assuming that the secret keys cannot be traced back to identities, we conjecture that sender/receiver-only deniability for symmetric PAKE would satisfy the said definition of deniability in the symmetric-key setting: in a two-party setup, a malicious accusing party $\mathcal{M}$ would not be able to produce binding cryptographic proofs from communication transcripts, associating another party with a particular exchange, as all messages could have been simulated by the accusing party $\mathcal{M}$. More specifically, in terms of distribution indistinguishability, a simulator in the said framework [19] can be constructed given that π is the only private input, symmetrically shared by both parties, and all other parameters are public and drawn at random. Indeed, this may not be surprising as Di Raimondo et al. [19] consider deniability in the symmetric key setting to be trivially satisfied.

Finally, assuming composability, using the PAKE-generated key with symmetric ciphers and MAC-based authentication would preserve deniability. Clearly, this and other forms of deniability for PAKE need to be studied rigorously in future work.

As a side note, deniability of messages and FS were among OTR's original goals, however, such features are independent from their SMP solution for authentication; they are implemented separately, e.g., by using MAC-based authentication and revealing keys. In the case of PAKE, these properties are rather built into the scheme.

Post-Quantum Security. As pointed out in Sect. 4.3, in the event that secure messaging and email tools transition to post-quantum cryptography, there are already candidate PAKE constructions that provide conjectured PQ security (e.g. see Table 1). Moreover, the recent symmetric-key authenticated key exchange (SAKE) by Avoine et al. [5] is conjectured to be PQ-secure due to its use of symmetric-key primitives. Thus, a quantum-resistant PAKE can be combined with SAKE, to obtain a low cost and effi-

cient PQ-AKE with PFS suitable for settings with limited computational power, e.g., the IoT.

5.3 Cryptographic Enhancements to Email and Messaging

The uses of PAKEs for securing email and messaging go beyond entity authentication and KM. Here, we discuss some areas that could benefit from the use of these schemes.

Multi-device Synchronization. A quite natural application of PAKE is in the realm of device pairing and secure multi-device synchronization, where the goal is to create an authenticated and private channel between devices, usually by the same user. Most solutions typically rely on a human interactive security protocol (HISP) and OOB channels, thus requiring manual intervention, which can give rise to new and subtle attacks. The application of PAKEs for device pairing in other contexts has been considered before [30]; it is thus natural to consider its use in multi-device syncing of secure email and messaging systems, for instance, to synchronize a user's keys for encryption and keys of trusted contacts.

A secure email solution can display a screen in each of $\mathcal{A}$'s devices that are to be paired, D_1 and D_2, so that after $\mathcal{A}$ enters a password in both, a PAKE protocol is triggered. Alternatively, this process can even be done asynchronously, i.e., without the two devices being online: D_1 pushes its state (e.g., key store, chat or email archive) to a server in encrypted form and later D_2 retrieves the secrets stored on the server in an oblivious manner w.r.t. the server. We discuss this further in the following part regarding secure secret retrieval.

For instance, the current implementation of p $\equiv$ p resorts to an ad-hoc pairing technique for key synchronization based on OOB comparison of SAS. Instead, it could benefit from such a PAKE-based solution. The established channel could be used not only for sharing key material but also contact lists, calendars, etc.

Secure Secret Sharing and Retrieval. This feature is inspired by the notion of password-protected secret sharing (PPSS) schemes formalized by Bagherzandi et al. [6], which are (t,n)-threshold constructions wherein security is preserved against an adversary controlling up to t servers out of n. A problem that PPSS addresses is protecting $\mathcal{A}$'s secret data d (e.g., a secret key used for decryption, authentication credentials, crypto-currency wallet key, etc.) in the event of a device compromise or failure.

An implementation of PPSS would secret-share d among a set of n entities so that only a collusion of more than t corrupt ones would compromise the data. A password-based mechanism would allow the authentication of the owner of d to the secret-share holders in order to trigger a reconstruction protocol and then retrieve the secret. The private storage of d can be shared among n external network entities; alternatively, if $\mathcal{A}$ does not trust external entities, her device can instead partake in the secret-sharing by storing multiple shares, thus preventing online dictionary attacks by a network attacker and not allowing $\mathcal{M}$ to learn anything about the secret without corrupting $\mathcal{A}$'s device.

Secret retrieval would have several use cases in secure messaging. For instance, a general anonymity/privacy related criticism directed at messaging services has to do with the identification of users via their phone numbers. This can be dealt with by securely storing long-term identities in encrypted form on the server, accessible only

to the users. Servers could also store per user lists of contacts in encrypted form; this would enable asynchronous syncing of contacts across multiple devices without the service provider learning the content.

Another use case would be to secret-share user data among several of their own devices, e.g., smartphone, laptop and tablet, so that a device compromise would not provide any useful information to an attacker; this can also be used for performing key synchronization among multiple devices. All these mechanisms would work in a similar manner from the user's point of view, i.e., simply by providing a password.

Recently, the Signal messaging system was enhanced with a functionality referred to as "Secure Value Recovery" [42], which aims at storing encrypted backups of user's data that can be recovered using a PIN. Among other things, the design involves a key stretching of the user's PIN along with a master key derivation from the stretched key and a piece of server-side stored randomness. The same core functionality can be achieved with the use of either PPSS or PAKE constructions such as OPAQUE, a recent aPAKE construction that, among other things, offers a secure secret retrieval mechanism based on oblivious pseudo-random functions, to fetch a secret stored in encrypted form on a server, using only a low-entropy password. It also offers protection against breaches and server password file compromises.

Signal's developers also mention secret sharing and oblivious pseudo-random functions as future possibilities [43], both of which could be achieved using existing cryptographic primitives, as explained above.

Auditable PAKEs for Thwarting Online Guessing Attacks. As is the case for SMP in OTR, online guessing attacks are unavoidable in PAKEs. This is usually dealt with by fixing a limit on the number of failed attempts that can be tolerated before invalidating a password.

However, in certain cases, another subtle adversarial strategy aimed at sidestepping the (at most) one online test per run would be to resort to a class of guess and abort attacks in which $\mathcal{M}$ intercepts a message in a given session (or initiates a session of her own) at a crucial step of a protocol run, verifies her guess at the password and in case of an incorrect guess, drops the said message to disguise her attempt as a network communication failure.

This can be done in both directions to double the chance of discovering the password, or in parallel against many network nodes depending on the setting. Such an attack can be carried out repeatedly without raising an alarm as the honest parties may simply view this as a network failure.

We identify a similar vulnerability in the use of a modified version of SMP in OTR: just before the last phase where the parties perform their secure equality test, when $\mathcal{A}$ and $\mathcal{M}$ exchange their blinded DH terms incorporating the low-entropy password in the exponent, i.e., $(g_3^a, g_1^a g_2^{\pi_{\mathcal{A}}})$, $\mathcal{M}$ could make a guessing attempt at $\pi_{\mathcal{A}}$ and in case of obtaining 0 (not equal), drop the message and force an abort, see Sects. 4.2 and 4.3 in [3]. Note that the non-interactive zero-knowledge (NIZK) proofs that are attached to the messages at every exchange are not meant to protect against this type of attack.

In a relatively recent work, Roscoe and Ryan [38] apply a mechanism based on commitment schemes and delay functions (e.g., timed-release encryption), originally developed by Roscoe [37] for protecting against online attacks in HISPs that use SAS,

to the setting of PAKEs in order to make them auditable by achieving *stochastic fair exchange*.

Roughly speaking, this is achieved by a transformation for PAKEs at the level of KC using a combination of blinding, randomization, commitments and delay functions such that a series of messages consisting of fake ones and the real intended message are exchanged and the parties will only get to know which is the *right* one until their exchange is complete. In a follow-up work, Couteau et al. [16] generalize this result to achieve ε-fair exchange using oblivious transfer and timed-release encryption.

This transformation can be used to enhance any PAKE with auditability, thus lending itself quite naturally to the authentication method suggested in this work. An important limitation here is that, due to the highly interactive design of the solution, it would be more suitable to the setting of secure messaging than email, unless a given email solution were to opt for untrusted buffer servers for transport, see Sect. 4.4.

Table 2. Comparison of trust establishment approaches. (Partial modifications to the original presented in [48]).

Paradigm	Example	Security									Usability									Adoption		
		Network MitM Prevented	Operator MitM Prevented	Operator MitM Detected	Operator Accountability	Key Revocation Possible	Privacy Preserving	Deniability Facilitated	Forward Secrecy Facilitated	Post-quantum Security	Automatic Key Initialization	Low Key Maintenance	Easy Key Discovery	Easy Key Recovery	In-Band	No Shared Secrets	Alert-less Key Renewal	Immediate Enrollment	Inattentive User Resistant	No Service Provider	Asynchronous	Multiple Key Support
Web of Trust	PGP	●	●	●	◐	◐	✗	✗	-	✗	✗	✗	◐	◐	✗	✗	✗	✗	✗	●	●	●
KD + SaL	CONIKS	●	✗	◐	●	●	●	✗	-	✗	●	●	●	●	●	●	●	●	●	●	●	●
OE + TOFU	TextSecure	◐	◐	◐	◐	✗	●	-	-	-	●	●	●	●	●	●	✗	●	✗	●	●	✗
OE + TOFU + OOB	p ≡ p	◐	◐	◐	◐	◐	●	-	-	-	●	◐	●	◐	✗	●	✗	✗	✗	●	✗	✗
OE + SMP	OTR	◐	◐	◐	◐	●	●	-	-	✗	✗	●	✗	✗	◐	◐	✗	●	✗	●	✗	●
OE + PAKE	PakeMail	◐	◐	⊛	⊛	◐	●	⊛	⊛	⊛	●	●	●	●	●	✗	●	●	●	●	●	●
KFV: OOB	SilentText	●	●	●	●	◐	●	-	-	-	✗	✗	✗	✗	✗	●	✗	✗	✗	●	✗	✗
KFV: SMP	OTR	●	●	●	●	◐	●	-	-	✗	✗	✗	✗	✗	●	✗	✗	✗	✗	●	✗	✗
KFV: PAKE	PakeMail	●	●	●	●	◐	●	●	●	⊛	●	●	●	●	●	✗	●	●	●	●	●	●

The property is: ● = satisfied; ◐ = partially satisfied; ✗ = not satisfied; ⊛ = implementation dependent; - = N/A
KD = Key directory; KFV = Key fingerprint verification; OE = Opportunistic encryption; SaL = Self-auditable logs; TOFU = Trust-on-first-use

Finally, note that some of the ideas in this transformation, specifically those related to enforcing fairness, have common elements with the original SMP [14] solution aimed at providing fairness, a property that was removed from the modified version of SMP used in OTR [3] on account of achieving efficiency.

5.4 Comparison

Table 2 shows a comparison of our proposal with a select set of approaches for trust establishment extracted from a relatively recent survey by Unger et al. on secure messaging [46]. We limit our analysis to the most relevant aspects with respect to our proposal and refer the reader to the cited source for a more detailed explanation of the approaches and their properties. If the reason behind a given evaluation is not specified in [46], we provide our own interpretation and evaluate our approach accordingly.

Most of the properties have self-explanatory names, except perhaps operator accountability, which is considered to be satisfied if the paradigm provides support for verifying the correct behavior of service providers during the trust establishment process, when a centralized infrastructure is required. The network and operator attackers considered for MITM refer respectively to adversaries controlling large segments of the internet and infrastructure operators (service providers).

PAKE-based approaches satisfy privacy preservation as the transcript of a PAKE execution does not leak information. *Deniability facilitated, FS facilitated* and post-quantum security are subject to the selection and exact usage of the PAKE scheme.

Approaches built upon opportunistic encryption (OE) partially provide MITM prevention because an attack can be successful during the initial communication round, before a key is authenticated. When combined with SMP, operator accountability and MITM detection are also partially satisfied given that if the execution of the SMP protocol fails, the users do not learn whether this was due to mismatching passwords or an adversarial attempt at compromising the channel. However, when it comes to our PAKE-based approach, these last two properties could be potentially satisfied with the use of auditable PAKEs (see Sect. 5.3), mainly in the context of messaging.

It is somewhat ambiguous as to why the authors of [46] consider key revocation—users being able to revoke and renew keys—to be fully satisfied for SMP applied to OE. While revocation is possible, the process would still suffer from the known limitations of a truly decentralized setting, e.g., informing all users of an expired key. The latter is indeed stated to be the reason for considering that KFV approaches only partially satisfy this property. Therefore, PAKE applied to OE would also partially satisfy key revocation. Thanks to the derived cryptographic key, the main advantages of OE with PAKE can be observed at the level of usability related properties, e.g., automation of tasks.

In key fingerprint verification (KFV) approaches, the verification is considered to occur before using the public keys, which leads to achieving most of the security properties. The evaluations for the OOB approach assume that the manual comparison is executed correctly; this assumption is not needed for SMP or PAKE. As we can observe, PAKE-based KFV significantly improves usability compared to OOB and SMP fingerprint verification.

Key directory combined with self-auditable logs (KD+SaL) is arguably the most promising approach identified by Unger et al. due to the wide range of properties that it provides. It allows users to efficiently verify the consistency of their own entry in a central key directory and therefore to detect and expose misbehavior by a third party.

The set of properties that KD+SaL and KFV:PAKE can achieve is similar, yet, the latter has the advantage of enhancing security with the properties discussed in Sect. 5.2.

Overall, PAKE-based key fingerprint verification offers the most complete set of properties with reasonable trade-offs between security and usability in a purely decentralized setting.

Clark et al. [15] present a similar table evaluating primitives used to enhance email security. Considering end-to-end encryption as a baseline, PAKE-based key verification/management would perform as shared secret key verification (R14 in [15]), except that, additionally, our PAKE-based approach partially satisfies the property that refers to providing support for server-side content processing (P12) as this can be enabled without exposing the encrypted content, e.g., via secure secret retrieval (see Sect. 5.3).

6 Implementation: PakeMail

Here we present PakeMail, an implementation of the core set of features of our proposal, mainly aimed at demonstrating the feasibility of the key ideas presented in this work. The source code and related documentation are available at [4].

PakeMail is a complete implementation of the main functionalities, namely, carrying out a PAKE protocol in a decentralized setting to authenticate public keys and establish a shared symmetric cryptographic key, using standard email and attachments as transport mechanism for networking, while preserving interoperability and without introducing any extra trust assumptions. However, this implementation should be rather viewed as a proof of concept given that a full-fledged version would not only require additional design and security considerations, but it would also provide support for the other remaining features that we have discussed in Sect. 5.

Our solution is implemented in Python 3, specifically targeted at version 3.6, with minimal dependencies, largely using standard Python libraries for tasks such as email formatting (MIME), encoding and exchange (IMAP, SMTP, TLS) as well as networking and file system operations. In terms of design, we have mainly adopted an object-oriented programming paradigm, enabling well-established properties such as a modular implementation with better separation of concerns via encapsulation, extensibility and re-usability. The current implementation is geared towards Unix-like operating systems, but it can be easily ported to other platforms.

6.1 Cryptographic Details

PakeMail makes use of the SPAKE2 library developed by Warner [49], which by default uses "Curve25519"[4] for the underlying elliptic curve, offering 128 bits of security. It is however possible to switch to 1024/2048/3072-bit integer groups as well. For the key confirmation phase described in Fig. 1, we use HKDF (HMAC-based Extract-and-Expand Key Derivation Function)[5] by H. Krawczyk for implementing the key derivation function, and HMAC[6] keyed-hashing for message authentication to derive the authentication tags. Finally, we use the PyNaCl library, which is a wrapper for the

[4] https://mailarchive.ietf.org/arch/msg/cfrg/-9LEdnzVrE5RORux3Oo_oDDRksU/.
[5] https://tools.ietf.org/html/rfc5869.html.
[6] https://tools.ietf.org/html/rfc2104.html.

well-known NaCl library, for performing cryptographic tasks such as encryption using 256-bit PAKE-derived secret keys.

PAKE messages and passwords are stored and transferred as byte strings. While an encoding at the application layer can be applied, ultimately, the underlying SPAKE2 Application Programming Interface (API) requires byte strings, thus leaving such choices to the users of the library. Moreover, due to the inherently asymmetric design of the SPAKE2 implementation, we assign distinct roles to PAKE instances, which in our implementation are referred to as "initiator" and "responder". Also, among other things, to prevent message reuse in different contexts and in line with the original protocol description [2] and the SPAKE2 library, we also enforce identities—again as byte strings—at the level of PAKE instances, which can refer to a username, user ID or server names, to name a few. As detailed in Sect. 4.2, the public key fingerprints could be included in the transcript and thus in the input of the hash function computing the intermediate shared key before KC, however the SPAKE2 API accepts only the user IDs and the weak password. We deal with this using the KC step and the inclusion of the public key fingerprints as associated data into the HMAC-authenticated message.

For further information on the details of the underlying SPAKE2 implementation, we refer the reader to the corresponding documentation by Warner [49].

6.2 PAKE Protocol Carried Out over Email

We have implemented the email-based approach suggested in Sect. 4.4, mainly because it corresponds to the solution that preservers compatibility and interoperability without imposing any additional requirements on standard email exchange solutions. PakeMail essentially makes use of email messages and attachments as transport mechanism for exchanging cryptographic messages and key confirmation tags belonging to PAKE protocol sessions as well as other data such as public keys that are to be authenticated by PAKE messages, effectively implementing the communication channel via mailboxes. In the case of secure messaging, the networking would be rather trivial given that most current solutions make use of intermediary servers, which in our case can be untrusted.

6.3 Implemented Scenarios

The solution provides PAKE clients and email services designed to deal with the requirements of PAKE exchanges and state maintenance in a decentralized and distributed computing setting. The PAKE clients have been implemented such that they take on either the role of an "initiator" or that of a "responder", consistent with the original SPAKE2 protocol design and the requirements of the SPAKE2 Python API.

Moreover, we provide a module containing easy to use executable implementations of the following scenarios: (i) a local execution of two independent threads of PAKE clients running a PAKE session with key confirmation, followed by some cryptographic tasks using the established key; (ii) an online execution of two clients (an initiator and a responder instance) running on the same hardware but routing their messages via email exchanges and attachments, currently implemented to work with Gmail but adapting it to other services would simply amount to providing the appropriate access data, e.g., the corresponding mail server credentials and port numbers; (iii) and (iv) provide the

execution of initiator and responder instances, respectively, on two different machines, again using email as transport mechanism.

6.4 Performance

In terms of performance, the main scenario of interest, namely that of running two separate instances of PakeMail on two different machines, carrying out a PAKE protocol with explicit key confirmation over Gmail, requires $\approx 3 \cdot 10^{-3}$ s, averaged over 10 runs. The results were obtained from executions on two laptops running at 1.6 GHz (Dual-Core Intel Core i5) with 8 GB of RAM, 256 KB and 4 MB of L2 and L3 cache, respectively.

Given the setting for which this approach is designed, i.e., distributed peer-to-peer connections between entities running point-to-point PAKE sessions, we consider the current overall execution time to be fast enough for all practical purposes. Table 3 provides a concise comparison of execution times for pure SPAKE2 sessions with its PakeMail counterpart, providing some information on the overall overhead added by our email-based networking and other non-PAKE computations.

Note that once both parties have entered their passwords, the added networking overhead due to email exchanges triggered by PakeMail will arguably not be perceptible by users given the inherent delay in email exchanges.

Table 3. Execution time comparison averaged over 10 runs.

Group	Pure SPAKE2	Local PakeMail	PakeMail via Gmail
Curve25519	26 ms	50 ms	350 ms

Finally, in terms of the underlying SPAKE2 library's performance on the same hardware, the average execution times using Curve25519 and 1024/2048/3072-bit integer groups are 26 ms, 9 ms, 42.1 ms and 82.6 ms, respectively. The delta would simply contribute additively to the PakeMail executions as the additional overhead incurred by switching to different representations is independent from the details of PakeMail.

6.5 Further Design and Security Considerations

Due to the nature of the current proof of concept implementation, certain design decisions have been made simply to ensure the implementation of a functional tool capable of demonstrating the feasibility, usability and efficiency of the proposed approach. However, a mature and robust implementation would have to account for a number of nuances. For instance, for the purpose of our proof of concept, we simply use universally unique identifier (UUID) numbers along with other user identifiers, which are stored in the email subject, to synchronize and map initiator and responder messages belonging to the same session to one another, coupled with a persistent per client session history to track and resolve sessions. A robust networking component capable

of addressing distributed systems corner cases such as deadlocks and race conditions remains to be done.

Regarding the cryptographic details of the implementation, it should be pointed out that a secure and scalable industrial implementation would have to at the very least rely on a constant-time implementation of the PAKE library as the currently used SPAKE2 library is by no means constant-time and is thus vulnerable to timing attacks.

Finally, note that dedicated optimization efforts remain to be done as future work. Clearly, the alternative transport mechanism based on intermediary servers, enabling more natural communication channels and networking, would lead to far lower communication overhead, albeit at the cost of somewhat hampering interoperability and compatibility, unless projects such as Matrix[7] and MLS[8] gain widespread adoption.

7 Security and Low-Entropy Secrets

The schemes considered thus far come with proofs of security, see Table 1 for the corresponding models and assumptions. The security guarantees can be traced back to the core properties of PAKEs: they can in effect fulfill the role of ZK proof of knowledge schemes such that a run of the protocol does not leak any information on the password and upon termination only reveals whether the secrets were equal; they resist offline dictionary attacks against passive and active adversaries, and online guessing attacks by limiting adversarial tests to one password per run; compromised session keys do not compromise the security of other established session keys; depending on the choice of PAKE, FS would ensure that session keys remain secure in case of password disclosure.

The only way for $\mathcal{M}$ to gain knowledge about the secret would be via active online guessing attempts, typically dealt with by fixing a limit on the number of failed attempts, e.g., SMP in OTR. As we previously discussed, the possibility of making PAKEs auditable can be used to mitigate this class of attacks by distinguishing between failed adversarial attempts and network failures to minimize the adversary's tries to one, under the assumption of correct input entry by honest users.

Low-Entropy Secret Agreement. Our proposal does come with a caveat, namely the need for either presharing or agreeing on a low-entropy secret in-band. As already discussed in [3], users can either share a secret over a secure channel, e.g. OOB, or agree on one via an in-band solution without revealing sensitive information about the secret itself, e.g., $\mathcal{A}$ asking $\mathcal{B}$ to use the name of their favorite restaurant. The user interface of a tool implementing this could warn users not to include the secret itself, similar to standard email warnings reminding users to attach documents in case they have mentioned it in the body of the message.

Assuming already bootstrapped authentication to avoid circularity, another possibility would be to use another already authenticated and secure channel to agree on a secret. For instance, given the widespread use of tools such as Signal, parties could simply use it to agree on a secret for a one-time entity authentication of their secure email solution. While it may not be appealing from a theoretical point of view, due to

[7] https://matrix.org/docs/spec/.

[8] https://messaginglayersecurity.rocks/.

the assumption of there being an already authenticated and secure channel, practically speaking, this approach would in fact provide a realistic and usable solution.

Usability Aspects. Particular attention must be paid to the implementation of an adequate interface for entering the low-entropy secret, along with the corresponding documentation and manuals with simple explanations for users. A lesson learned from a usability study on the OTR/SMP tool [44] stresses the need for further research on how to guide users towards establishing a secure shared human-memorable secret.

For instance, adding a list pre-populated with questions might serve to reduce user effort by allowing them to choose one from the list, or as a guide for users to generate similar questions. The questions should not lead to evident answers or to answers belonging to very small known sets, such as "yes/no" or colors, as such cases increase the successful guessing probability of the adversary. Another measure for dealing with disparities due to letter cases would be to for example simply convert the secret to upper-case, at the cost of reducing entropy.

8 Further Directions

A clear and promising line of future work consists of improving the current implementation and adding the various enhancements discussed here.

Producing secure implementations of cryptographic primitives and protocols is a notoriously difficult task. Consequently, over the past decades, a considerable amount of research in formal verification has focused on developing techniques for ensuring that security software preserves the security guarantees of the underlying cryptographic constructions. Although our solution builds on provably secure cryptographic constructions, the actual implementation makes use of cryptographic software that has not been proven secure. Therefore, pursuing the development of a verified implementation of a PAKE protocol would be another promising research direction. This could be achieved using dedicated languages such as F* [35], which has been used, among other things, to produce a verified reference implementation of the TLS (1.2) protocol [11].

Alternatively, a robust PAKE implementation in a language designed for performance and safety such as RUST would be yet another viable path. An initial rough implementation of SPAKE2 in RUST is already available and subject to ongoing work.[9]

Follow-up theoretical work on all the suggested cryptographic enhancements and implementations thereof represents another line of research. In particular, given the fact that mature PAKE implementations are quite rare, we consider further theoretical work on the design and analysis of a quantum-secure PAKE, proven secure in the QROM, accompanied by an actual implementation to be worth pursuing. Similarly, to the best of our knowledge, an implementation, let alone practical and efficient, of the secure secret storage and retrieval tasks (e.g., using PPSS or OPAQUE) represents yet another promising line of work.

Moreover, research on effective and usable methods for assisting users in agreeing on low-entropy secrets while reducing the mental effort and the likelihood of mistakes, is also encouraged. Other interesting directions include the application of

[9] https://github.com/RustCrypto/PAKEs.

PAKE to authentication for encrypted mailing lists, and studying the possibility of sharing/synchronizing existing trust assignments for contacts across different services—e.g., from Signal to p ≡ p or vice versa. In this case, once an entity is trusted in one application, other applications that recognize this entity could inherit the trust stored in the user's device; clearly, it is vital to do this in a secure and privacy-preserving manner.

References

1. Abdalla, M., Barbosa, M.: Perfect forward security of SPAKE2. Cryptology ePrint Archive, Report 2019/1194 (2019). https://eprint.iacr.org/2019/1194
2. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_14
3. Alexander, C., Goldberg, I.: Improved user authentication in off-the-record messaging. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society. ACM (2007)
4. Atashpendar, A., Vazquez Sandoval, I.: PakeMail (2020). https://github.com/CryptographySandbox/PakeMail
5. Avoine, G., Canard, S., Ferreira, L.: Symmetric-Key Authenticated Key Exchange (SAKE) with perfect forward secrecy. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 199–224. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_10
6. Bagherzandi, A., Jarecki, S., Saxena, N., Lu, Y.: Password-protected secret sharing. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 433–444 (2011)
7. Becerra, J., Ostrev, D., Škrobot, M.: Forward secrecy of SPAKE2. In: Baek, J., Susilo, W., Kim, J. (eds.) ProvSec 2018. LNCS, vol. 11192, pp. 366–384. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01446-9_21
8. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_11
9. Bellovin, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: 1992 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 72–84. IEEE Computer Society (1992)
10. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006). https://doi.org/10.1007/11745853_14
11. Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P.Y.: Implementing TLS with verified cryptographic security. In: 2013 IEEE Symposium on Security and Privacy, pp. 445–459. IEEE (2013)
12. Birk, V., Marques, H., Hoeneisen, B.: pEp Foundation: IANA registration of trustword lists (2019). https://tools.ietf.org/html/draft-birk-pep-trustwords-03
13. Borisov, N., Goldberg, I., Brewer, E.: Off-the-record communication, or, why not to use PGP. In: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society (2004)
14. Boudot, F., Schoenmakers, B., Traore, J.: A fair and efficient solution to the socialist millionaires' problem. Discrete Appl. Math. **111**, 23–36 (2001)
15. Clark, J., van Oorschot, P.C., Ruoti, S., Seamons, K., Zappala, D.: Securing email. arXiv preprint arXiv:1804.07706 (2018)
16. Couteau, G., Roscoe, A.W., Ryan, P.Y.A.: Partially-fair computation from timed-release encryption and oblivious transfer. Cryptology ePrint Archive, Report 2019/1281 (2019). https://eprint.iacr.org/2019/1281

17. 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, pp. 193–208 (2016)
18. Delaune, S., Kremer, S., Robin, L.: Formal verification of protocols based on short authenticated strings. In: 2017 IEEE 30th Computer Security Foundations Symposium (CSF), pp. 130–143. IEEE (2017)
19. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Deniable authentication and key exchange. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 400–409 (2006)
20. Ding, J., Alsayigh, S., Lancrenon, J., RV, S., Snook, M.: Provably secure password authenticated key exchange based on RLWE for the post-quantum world. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 183–204. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_11
21. 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 (1981)
22. Fischlin, M., Günther, F., Schmidt, B., Warinschi, B.: Key confirmation in key exchange: a formal treatment and implications for TLS 1.3. In: 2016 IEEE Symposium on Security and Privacy (SP). IEEE (2016)
23. Garay, J.A., MacKenzie, P.D., Yang, K.: Efficient and secure multi-party computation with faulty majority and complete fairness. IACR Cryptol. ePrint Arch. **2004**, 9 (2004)
24. Hao, F., Ryan, P.: J-PAKE: authenticated key exchange without PKI. In: Gavrilova, M.L., Tan, C.J.K., Moreno, E.D. (eds.) Transactions on Computational Science XI. LNCS, vol. 6480, pp. 192–206. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17697-5_10
25. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: an asymmetric PAKE protocol secure against pre-computation attacks. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 456–486. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_15
26. Kainda, R., Flechais, I., Roscoe, A.: Usability and security of out-of-band channels in secure device pairing protocols. In: Proceedings of the 5th Symposium on Usable Privacy and Security, p. 11. ACM (2009)
27. Kainda, R., Flechais, I., Roscoe, A.: Secure mobile ad-hoc interactions: reasoning about out-of-band (OOB) channels. IWSSI/SPMU **2010**, 10–15 (2010)
28. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_18
29. Krawczyk, H.: Cryptographic extraction and key derivation: the HKDF scheme. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 631–648. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_34
30. Kumar, A., Saxena, N., Tsudik, G., Uzun, E.: A comparative study of secure device pairing methods. Pervasive Mob. Comput. **5**(6), 734–749 (2009)
31. Naor, M., Rotem, L., Segev, G.: The security of lazy users in out-of-band authentication. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11240, pp. 575–599. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_21
32. Nguyen, L.H., Roscoe, A.W.: Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey. J. Comput. Secur. **19**(1), 139–201 (2011)
33. OTRv4-development: Specification of OTR version 4, October 2019. https://github.com/otrv4/otrv4/blob/master/otrv4.md
34. pEp Security: Pretty Easy Privacy (pEp). https://www.pep.security

35. Microsoft Research, I.: F* (2020). https://fstar-lang.org/
36. Rivest, R.L., Shamir, A.: How to expose an eavesdropper. Commun. ACM **27**(4), 393–394 (1984)
37. Roscoe, A.W.: Detecting failed attacks on human-interactive security protocols (transcript of discussion). In: Anderson, J., Matyáš, V., Christianson, B., Stajano, F. (eds.) Security Protocols 2016. LNCS, vol. 10368, pp. 198–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62033-6_22
38. Roscoe, A.W., Ryan, P.Y.A.: Auditable PAKEs: approaching fair exchange without a TTP. In: Stajano, F., Anderson, J., Christianson, B., Matyáš, V. (eds.) Security Protocols 2017. LNCS, vol. 10476, pp. 278–297. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71075-4_31
39. Ruoti, S., Andersen, J., Monson, T., Zappala, D., Seamons, K.: A comparative usability study of key management in secure email. In: Fourteenth Symposium on Usable Privacy and Security, pp. 375–394 (2018)
40. Sasson, E.B., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474 (2014)
41. Shirvanian, M., Saxena, N.: Wiretapping via mimicry: short voice imitation man-in-the-middle attacks on crypto phones. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014, pp. 868–879 (2014)
42. Signal: Improving registration lock with secure value recovery, February 2020. https://signal.org/blog/improving-registration-lock
43. Signal: Technology preview for secure value recovery (2020). https://signal.org/blog/secure-value-recovery
44. Stedman, R., Yoshida, K., Goldberg, I.: A user study of off-the-record messaging. In: 4th Symposium on Usable Privacy and Security, pp. 95–104 (2008)
45. Tan, J., Bauer, L., Bonneau, J., Cranor, L.F., Thomas, J., Ur, B.: Can unicorns help users compare crypto key fingerprints? In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 3787–3798. ACM (2017)
46. Unger, N., et al.: SoK: secure messaging. In: 2015 IEEE Symposium on Security and Privacy, pp. 232–249. IEEE (2015)
47. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 309–326. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_19
48. Vazquez Sandoval, I., Atashpendar, A., Lenzini, G.: Authentication and key management automation in decentralized secure email and messaging via low-entropy secrets. In: Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2: SECRYPT, Lieusaint, Paris, France (2020)
49. Warner, B.: Pure-Python SPAKE2 (2010). https://github.com/warner/python-spake2
50. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982), pp. 160–164. IEEE (1982)