Password-Authenticated Public-Key Encryption*

Tatiana Bradley[1], Jan Camenisch[2], Stanislaw Jarecki[1], Anja Lehmann[3], Gregory Neven[2], and Jiayu Xu[1]

[1] University of California, Irvine. Email: {tebradle@,sjarecki@,jiayux@}uci.edu.
[2] Dfinity. Email: {gregory@,jan@}dfinity.org.
[3] IBM Research – Zurich. Email: anj@zurich.ibm.com.

Abstract. We introduce password-authenticated public-key encryption (PAPKE), a new cryptographic primitive. PAPKE enables secure end-to-end encryption between two entities without relying on a trusted third party or other out-of-band mechanisms for authentication. Instead, resistance to man-in-the-middle attacks is ensured in a human-friendly way by authenticating the public key with a shared password, while preventing offline dictionary attacks given the authenticated public key and/or the ciphertexts produced using this key.

Our contributions are three-fold. First, we provide property-based and universally composable (UC) definitions for PAPKE, with the resulting primitive combining CCA security of public-key encryption (PKE) with password authentication. Second, we show that PAPKE implies Password-Authenticated Key Exchange (PAKE), but the reverse implication does not hold, indicating that PAPKE is a strictly stronger primitive than PAKE. Indeed, PAPKE implies a two-flow PAKE which remains secure if either party re-uses its state in multiple sessions, e.g. due to communication errors, thus strengthening existing notions of PAKE security. Third, we show two highly practical UC PAPKE schemes: a generic construction built from CCA-secure and anonymous PKE and an ideal cipher, and a direct construction based on the Decisional Diffie-Hellman assumption in the random oracle model.

Finally, applying our PAPKE-to-PAKE compiler to the above PAPKE schemes we exhibit the first 2-round UC PAKE's with efficiency comparable to (unauthenticated) Diffie-Hellman Key Exchange.

1 Introduction

A well-known Achilles' heel of end-to-end encryption is the distribution and trustworthiness of long-term cryptographic keys [27]. In particular, it is extremely hard for end users to judge the authenticity of public keys. They can therefore easily be tricked into encrypting the data under a wrong key and thereby lose all security. If the exchange of keys is facilitated by a third party such as a certificate authority or a service provider, as is the case for most public-key infrastructures (PKIs) as well as for end-to-end encrypted messengers such

* Full version of this paper appears in [12].

as Signal, WhatsApp, or iMessage, users need to trust that third party to provide the correct keys. Indeed, if a service provider is able to substitute its own keys for those of the intended recipients, it can mount a man-in-the-middle (MITM) attack and decrypt all subsequent communication.

An article in The Guardian [18] describes this trust required in the service provider and its capability of striking MITM attacks as a "backdoor" and a "security loophole" in the encryption scheme used by WhatsApp. This characterization was repudiated in an open letter signed by over seventy cryptographers and security experts [26], stating that this is not a "backdoor", but simply how cryptography works. While technically correct, this explanation is not very satisfactory from an end-user's perspective and prompts the question: *should cryptography work like that?* Is there really no way to protect encrypted communication between end users from such MITM and key substitution attacks?

Ad-hoc solutions against MITM attacks. Many approaches to preventing man-in-the-middle attacks in the context of secure end-to-end communication exist, but they either rely on trusted third parties, or on mostly ad-hoc solutions built on top of conventional encryption schemes, aiming to allow end-users to verify the correctness of public keys. None of these approaches provides the degree of usability and security that one can hope for, and which our solution provides.

Trust-on-first-use, as commonly used by Secure Shell (SSH), reduces the likelihood of MITM attacks, but cannot completely prevent them. The *web of trust* [24] as used e.g. in Pretty Good Privacy (PGP), establishes a distributed trust model via individual vetting: it requires users to endorse associations of public keys to specific people, and to endorse other people as trusted endorsers. Even though this approach was popular in the early days of cryptography, it was never widely adopted, possibly because of the strong level of involvement it requires from users to inspect each others' keys and to issue endorsements.

Today, the most common method to establish trust in end users' public keys is to let users manually verify a hash value of keys, known as a key *fingerprint*, using an out-of-band channel. Fingerprints are often represented in human-friendly formats to ease verification, e.g., as digits [28], pronounceable strings [20], ASCII art [23], or QR codes [28], but they require either physical proximity of communication partners (QR codes) or they are tedious to verify.[4] A crucial problem with key fingerprints is the far-from-optimal trade-off between security and usability: Strong fingerprints with 60 decimal or 32 hexadecimal digits are simply too long to verify by hand. Shorter fingerprints are more human-friendly but are vulnerable to preimage attacks, allowing an adversary to generate a key with the same fingerprint. A recent study comparing different manual key verification mechanisms found that all were subject to attacks whose success rates ranged between 6% and 72% [25].

Introducing a New Tool. We propose a new cryptographic primitive, *Password-Authenticated Public-Key Encryption (PAPKE)*, which authenticates an encryp-

[4] Users also struggle with the notion of key fingerprints, e.g. *all* Telegram users in one study [5] believed the fingerprint to be either the encryption key or a ciphertext.

tion public key using human-memorable passwords, but does so in a way which is not subject to offline dictionary attacks given the authenticated public key or the ciphertexts encrypted using that key.

More precisely, PAPKE modifies the notion of public-key encryption so that both the *key generation* and the *encryption* algorithms take as an additional input *a password*, i.e. an arbitrary human-memorable string. The semantics of this password input is that Alice, when generating her public key, implicitly authenticates and "locks" this key with a password, and to encrypt a message, Bob must use a matching password to "unlock" the authenticated public key correctly. The correctness guarantee is that Alice decrypts the message encrypted by Bob *if* Bob encrypted it using the same password that Alice used in key generation. The notion of password-authentication of the public key which PAPKE enforces is the following: If a man-in-the-middle attacker substitutes Alice's public key with its own, the confidentiality of messages which Bob encrypts under this key is guaranteed as long as the adversary fails to guess the password shared by Alice and Bob. Crucially, the attacker must guess this password at the time it creates the substituted public key, and the eventual leakage of the password after generation of the adversarial key has no impact on encryption security.

PAPKE thus enables end-to-end secure communication without relying on a trusted party or exchanging long fingerprints on an out-of-band channel, and instead it bootstraps security from a short human-memorizable password.

PAPKE and Offline Dictionary Attacks. The challenge of password-based schemes is obtaining strong security based on weak secrets. In particular, such a scheme must be resilient against *offline* password attacks. For PAPKE this means that an adversary who receives an authenticated public key and the ciphertexts created using this key cannot use offline computation to find the passwords used to create either object. In other words, the adversary cannot use the public key or the intercepted ciphertexts to locally test password guesses. Otherwise, the low entropy of passwords would hardly provide any extra security: according to NIST [13] even a 16-character human-memorizable password has only 30 bits of entropy on average, and hence can easily be brute-forced.

To illustrate this challenge, consider a few simple but failed attempts at constructing a secure PAPKE scheme. A natural way to password-authenticate any information, including a public key, would be to MAC it using the (hashed) password as a MAC key. This, however, would be subject to an offline dictionary attack, as the attacker can locally test password guesses until it finds the one for which the MAC verifies. More generally, any procedure which allows for *explicit verification* of the authenticated public key under a password would be subject to an offline attack. What if the key was not authenticated itself but the encrypting party included the password in the plaintext? This would be insecure against a man-in-the-middle attack which sends its public key to the encryptor and decrypts it to read the encryptor's password.

Indeed, in a secure PAPKE the authenticated public key must commit the receiver Alice to the password used in the key generation, and the sender Bob cannot verify this commitment explicitly, but it can create a ciphertext such

that (1) it is correct if Bob's and Alice's passwords match, (2) the plaintext is undecryptable if the two passwords differ, (3) the encryptor cannot tell which is the case, and (4) if the two passwords do not match then no one, including Alice who created the public key, can learn anything about Bob's password beyond the fact that it does not match the unique password she used to generate her public key.

We stress that PAPKE uses passwords to strictly enhance encryption security, i.e., for non-substituted keys, PAPKE provides standard CCA security that does not depend on the strength of the user's password. Thus the purpose of the password is solely to hedge security in case the encryptor uses a substituted key, but we stress that this hedging is applicable only if the encryptor shares a password with the party who generates the public key.

Our Contribution. We provide a thorough study of the proposed PAPKE primitive, and our contributions fall into the following three categories:

(Ia) Strong security notions for PAPKE. First, we formally introduce the concept of password-authenticated public key encryption, and define the desired security properties both via a universally composable (UC) functionality [14] and a set of property-based definitions. While property-based definitions are often more intuitive, a formalization in the UC framework provides stronger and more realistic security guarantees because it does not require any assumptions on the password distribution and correctly models real-world phenomena such as password reuse and making typing mistakes when entering the password. We prove that our UC security definition implies our property-based ones, hence proving a scheme secure in the UC setting implies its security under the more intuitive property-based notions.

(Ib) Relation to PAKE. To better understand the strength of the PAPKE primitive we compare it to the well-studied primitive of Password-Authenticated Key Exchange (PAKE) [7, 11, 15] The relation between PAPKE and PAKE is twofold. First, PAPKE immediately implies a two-round PAKE: Alice and Bob can perform password-authenticated key exchange if Alice sends to Bob a PAPKE public key authenticated by her password, and Bob encrypts a session key using the received key and his password. Indeed, we show that if this simple protocol is instantiated with any scheme satisfying our UC PAPKE notion then the resulting protocol satisfies the strong UC notion of PAKE [15].

Regarding the other direction it might seem at first glance that any 2-round PAKE protocol, e.g. [8, 7, 4], can generically imply a PAPKE scheme as follows: The PAKE requester's flow can define a PAPKE public key, and the PAKE responder's flow, together with an encryption of the plaintext under the established session key, can define a PAPKE ciphertext. However, we show that this intuition is in fact *incorrect*, as the non-interactive usage of encryption that is required by PAPKE is not compatible with standard PAKE security notions. Indeed, as we discuss below, the two-round PAKE implied by PAPKE has stronger security than what is implied by standard PAKE notions because it remains secure (and robust) even if either party re-uses its state. Summing up, the relation

of PAPKE and PAKE is that PAPKE implies a 2-round PAKE with a (novel) property of security under session state re-use.

(II) Efficient PAPKE constructions. We show two very practical constructions that securely realize the UC PAPKE functionality. Our first construction generically builds a PAPKE scheme from a public-key encryption (PKE) scheme and an ideal cipher: The authenticated public key is an encryption of the PKE public key under the password, with the encryption implemented using an ideal cipher over the space of PKE public keys. To obtain the desired UC-security, the PKE scheme must satisfy a number of properties beyond standard CCA security, such as key-anonymity [6] and strong robustness [1]. We show a concrete instantiation of this scheme using a variant of DHIES [2] which satisfies these properties under the so-called Oracle Diffie-Hellman (ODH) assumption. This results in a highly-efficient construction secure under ODH in the Ideal Cipher model, which uses 1 exponentiation for key generation, 2 for encryption, and 1 for decryption.

However, ideal ciphers over arbitrary cyclic groups, e.g. an elliptic curve, are not so easy to implement. While generic constructions for ideal ciphers from random oracles exist [19, 16], implementing ideal ciphers over a specific algebraic group is not straightforward, and if not done carefully can result in timing and/or offline password guessing attacks. Thus we also provide an alternative concrete construction that does not rely on ideal ciphers and therefore might be easier to implement. It uses the Fujisaki-Okamoto transform [17] of a twisted "twin-key" ElGamal construction of independent interest. This construction uses 2 exponentiations for key generation, 2 multi-exponentiations for encryption, and 1 exponentiation and 1 multi-exponentiation in decryption, and relies on the Decisional Diffie-Hellman (DDH) assumption in the Random Oracle Model.

(IIIa) Efficient 2-Round UC PAKE schemes. Our generic PAPKE-to-PAKE compiler discussed above implies two highly efficient UC PAKE protocols when instantiated with the above two PAPKE schemes. To the best of our knowledge these are the first two-round UC-secure PAKE's which rely on standard cyclic groups, i.e., do not use groups with bilinear maps or other trapdoor structure, and which resort instead to either the Ideal Cipher (IC) or the Random Oracle Model (ROM) to achieve practical efficiency. Specifically, our results imply a UC PAKE which uses 2 expentiations per party but relies on an ideal cipher over a group, and a UC PAKE which uses 4 (multi)-exponentiations for the requester and 2 exponentiations for the responder and relies on a random oracle model for hash functions. Note that the first scheme matches and the second scheme comes very close to the 2 exponentiations/party cost of *unauthenicated* Diffie-Hellman Key Exchange, with is the minimum cost for PAKE one can reasonably expect. The closest efficiency-wise UC PAKE we know of is by Abdalla et al. [3], which was shown secure under comparable assumptions, but which requires 3 message flows while our UC PAKE's use only 2 flows.

(IIIb) PAKE's with session re-use security. As we argued in (Ib) above, the PAPKE-to-PAKE compiler results in a 2-round PAKE which has novel security and reliability properties which follow from the fact that PAPKE enforces

ciphertext security when the same public key is used to encrypt multiple messages. Recall that the PAKE requester message is a PAPKE public key, and the PAKE responder message is a PAPKE ciphertext encrypting a random session key under this public key, and both the public key and the ciphertext are created using the passwords of resp. the requester and the responder. (See Section 3 for the full description of this PAKE.) The novel security property of this PAKE is that each of these keys is secure even though all sessions re-use the same session state and the first message flow of the requester. The standard model of PAKE security does not guarantee security in this case, but a PAKE which is secure in this way can be beneficial to higher-level applications. For example it can help handle communication faults: A responder session which believes that its response has not been delivered correctly can safely respond to the same requester message again, and a requester who gets multiple responses can securely spin off a subprocess for each of them without re-starting a new session from scratch.

Roadmap. In Section 2 we define PAPKE as a strengthened version of public-key encryption. Section 3 discusses the relation between PAKE and PAPKE, and shows a generic compiler from any UC PAPKE to UC PAKE. Section 4 presents our two highly efficient UC PAPKE schemes. In Appendix A we exemplify one highly-efficient concrete 2-round UC PAKE protocol obtained via the generic compiler of Section 3 applied to one of the PAPKE schemes of Section 4.

2 Security Model for PAPKE

In this section we introduce our security models for password-authenticated encryption. A peculiarity of formal security definitions for *password-based* primitives is that they must model the inherent probability of an adversary correctly guessing the low-entropy password. Property-based definitions [7] (sometimes also called game-based definitions) typically do so by requiring that the adversary's probability of winning the security game is negligibly more than a (non-negligible) threshold determined by its number of online queries and the entropy of the distribution from which the password is chosen. Composable security definitions [15] such as those in Canetti's Universal Composability (UC) framework [14], on the other hand, model the possibility of guessing the password directly into the ideal behavior of the primitive.

As argued by Canetti et al. [15], composable definitions provide stronger and more realistic security guarantees than property-based ones, because they do not make any implicit assumptions about the password distribution and correctly model real-world phenomena such as password reuse and typos while entering the password. Nevertheless, property-based definitions are often more intuitive and easier to understand than UC definitions. Below we present the property-based PAPKE security notions, and in the full version [12] we define UC notion of PAPKE and show that it implies the property-based notion.

Definition 1 (PAPKE). *Let $\mathcal{D}$ be a dictionary of possible passwords, and $\mathcal{M}$ be a message space. A password-authenticated public-key encryption scheme is a tuple of algorithms* $\mathsf{PAPKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following behavior:*

- $\mathsf{KGen}(\kappa, pwd) \to_R (apk, sk)$*: on input a security parameter κ and password $pwd \in \mathcal{D}$, output an authenticated public key apk and a secret key sk.*
- $\mathsf{Enc}(apk, pwd, m) \to_R c$*: on input an authenticated public key apk, password pwd and a message $m \in \mathcal{M}$, output a ciphertext c.*
- $\mathsf{Dec}(sk, c) \to m$*: on input a secret key sk and ciphertext c, output a message $m \in \mathcal{M} \cup \{\perp\}$ where $\perp$ indicates that the ciphertext is invalid.*

For correctness we require that for any password $pwd \in \mathcal{D}$, key pair $(apk, sk) \leftarrow_{\mathrm{R}} \mathsf{KGen}(\kappa, pwd)$, and ciphertexts $c \leftarrow_{\mathrm{R}} \mathsf{Enc}(apk, pwd, m)$, we have that $m = \mathsf{Dec}(sk, c)$.

Informally, the desired security properties of PAPKE schemes are:

Resistance against Offline-Attacks: None of the values that are (partially) derived from a password allows offline dictionary attacks on the passwords that were used to generate them: The authenticated public key apk does not leak anything about the setup password pwd, and ciphertexts c formed under apk do not leak any information about the password attempt pwd' that was used in the encryption. The only and inevitable information leaked is that the party who holds the secret key sk corresponding to apk learns whether $pwd' = pwd$, because that holds if and only if $\mathsf{Dec}(sk, c) \neq \perp$.

CCA Security: Ciphertexts encrypted under an *honestly* generated authenticated public key apk hide the encrypted message from any adversary who doesn't know the secret key. This property is modeled in the standard CCA setting, and it holds even if the adversary knows all passwords used.

Security against Man-in-the-Middle (MITM) Attacks: The choice of an authenticated public key apk^* commits the adversary to some single password guess pwd^*, and all ciphertexts encrypted under apk^* using any password $pwd \neq pwd^*$ hide the encrypted message. The only available attack is an *online* attack, where the adversary guesses password pwd used by the honest encryptor and generates apk^* so that it commits to $pwd^* = pwd$. Thus the MITM attack gains effectively one password guess per each adversarial public key apk^* which the honest party uses in encryption.

Long-Term Security: The security of encryptions under an adversarially chosen key apk^* is preserved in a forward-secure manner because it holds even if the adversary (eventually) learns the encryptor's password $pwd \neq pwd^*$.

Ciphertext Authenticity: The password also guarantees authenticity of ciphertexts. That is, an adversary who knows an honestly generated key apk, but not the password pwd (or the secret key sk), cannot create valid ciphertexts, i.e., ciphertexts that decrypt under sk into some message $m \neq \perp$.

2.1 Property-Based Security Definition

We formalize the above intuitive security requirements using two game-based definitions, namely indistinguishability against chosen-ciphertext and chosen-key attack (IND-CCKA), and ciphertext authenticity (AUTH-CTXT). For the

sake of brevity, we will refer to property IND-CCKA as the *privacy* property. The privacy experiment formalizes the first four properties listed above:

Experiment $\mathsf{Exp}^{\mathsf{IND\text{-}CCKA}}_{\mathcal{A},\mathsf{PAPKE}}(\kappa)$:
$pwd \leftarrow_{\mathrm{R}} \mathcal{D}$, $\mathbf{L} \leftarrow \emptyset$, $(apk, sk) \leftarrow_{\mathrm{R}} \mathsf{KGen}(\kappa, pwd)$
$b \leftarrow_{\mathrm{R}} \{0,1\}$, $\texttt{revealed} \leftarrow 0$
$b' \leftarrow_{\mathrm{R}} \mathcal{A}^{\mathsf{LoR}(b,pwd,\cdot,\cdot,\cdot),\mathsf{Dec}(sk,\cdot),\mathsf{Reveal}(pwd)}(apk)$
oracle LoR on input a public key apk^* and
two messages m_0 and m_1 where $|m_0| = |m_1|$
if $apk^* \neq apk$ and $\texttt{revealed} = 1$, return $\perp$
else, compute $C \leftarrow_{\mathrm{R}} \mathsf{Enc}(apk^*, pwd, m_b)$,
if $apk^* = apk$ add C to $\mathbf{L}$
return C
oracle Dec on input a ciphertext $C \notin \mathbf{L}$:
return $m \leftarrow \mathsf{Dec}(sk, C)$ where $m \in \mathcal{M} \cup \{\perp\}$
oracle Reveal: return pwd and set $\texttt{revealed} \leftarrow 1$
return 1 if $b' = b$

Definition 2 (IND-CCKA). *A* PAPKE *scheme is called* indistinguishable under chosen-ciphertext and key attacks *if for all efficient adversaries* $\mathcal{A}$*, and any password space* $\mathcal{D}$ *it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CCKA}}_{\mathcal{A},\mathsf{PAPKE}}(\kappa) = 1] \leq \frac{1}{2} \;+\; \frac{1}{2} \cdot \frac{q_{apk^*} + q_{\mathsf{Dec}}}{|\mathcal{D}|} + \mathsf{negl}(\kappa)$$

for a negligible function negl*, where* q_{apk^*} *denotes the number of public keys* $apk^* \neq apk$ *that* $\mathcal{A}$ *used in its queries to the* LoR *oracle, and where:*

- *if* $q_{apk^*} > 0$*, then* q_{Dec} *is the number of* $\mathcal{A}$*'s queries to the* Dec *oracle while* $\texttt{revealed} = 0$ (active/MITM security)
- *if* $q_{apk^*} = 0$*, then* $q_{\mathsf{Dec}} \leftarrow 0$ (passive/CCA security)

In the IND-CCKA definition above we set $q_{\mathsf{Dec}} = 0$ for passive attacks, i.e. if $q_{apk^*} = 0$, then the security bound is $1/2 + \mathsf{negl}(\kappa)$. In other words, if $\mathcal{A}$ does not stage any MITM attack, i.e. it never substitutes the challenge public key apk with $apk^* \neq apk$, then IND-CCKA is like standard CCA-security of PKE, i.e. $\mathcal{A}$ can make any number of encryption and decryption queries and they will not impact its success probability.

Authenticity (AUTH-CTXT). The ciphertext authenticity property (Def. 3) formalizes that the adversary $\mathcal{A}$, given apk generated for password pwd chosen at random in dictionary $\mathcal{D}$, cannot create a valid ciphertext except for probability $(1 + q_{apk^*} + q_{\mathsf{Dec}})/|\mathcal{D}|$, where q_{apk^*} is the number of encryption queries $\mathcal{A}$ makes under bad and distinct keys $apk^* \neq apk$, and q_{Dec} is the number of decryption queries. (See [12] for full discussion of these definitional choices.) Note that here we do not let $\mathcal{A}$ learn pwd because knowing pwd suffices to form a valid ciphertext. The password-guessing count is $q_{apk^*} + q_{\mathsf{Dec}}$ plus 1 because the final ciphertext $\mathcal{A}$ creates can itself be used to guess a password.

The authenticity experiment is defined as follows:

Experiment $\mathsf{Exp}^{\mathsf{AUTH\text{-}CTXT}}_{\mathcal{A},\mathsf{PAPKE}}(\kappa)$:
$pwd \leftarrow_{\mathrm{R}} \mathcal{D}$, $\mathbf{L} \leftarrow \emptyset$, $(apk, sk) \leftarrow_{\mathrm{R}} \mathsf{KGen}(\kappa, pwd)$
$C^* \leftarrow_{\mathrm{R}} \mathcal{A}^{\mathsf{Enc}(pwd,\cdot,\cdot),\mathsf{Dec}(sk,\cdot)}(apk)$
oracle Enc on input a key apk^* and message m :
compute $C \leftarrow_{\mathrm{R}} \mathsf{Enc}(apk^*, pwd, m)$
if $apk^* = apk$ add C to $\mathbf{L}$
return C
oracle Dec on input a ciphertext C :
return $m \leftarrow \mathsf{Dec}(sk, C)$, where $m \in \mathcal{M} \cup \{\perp\}$
return 1 if $\mathsf{Dec}(sk, C^*) \neq \perp$ and $C^* \notin \mathbf{L}$

Definition 3 (AUTH-CTXT). *A* PAPKE *scheme provides* authenticity of ciphertexts *if for all efficient adversaries $\mathcal{A}$, and any password space $\mathcal{D}$ it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{AUTH\text{-}CTXT}}_{\mathcal{A},\mathsf{PAPKE}}(\kappa) = 1] \leq \frac{q_{apk^*} + q_{\mathsf{Dec}} + 1}{|\mathcal{D}|} + \mathsf{negl}(\kappa)$$

for a negligible function negl*, where q_{apk^*} is the number of bad keys $apk^* \neq apk$ in $\mathcal{A}$'s* Enc *oracle queries and q_{Dec} is the number of $\mathcal{A}$'s* Dec *oracle queries.*

3 Relation between PAPKE and PAKE

PAPKE, the new cryptographic primitive we propose, is closely related to Password Authenticated Key Agreement (PAKE) [7, 11, 15]. Specifically, we show that it is easy to build a (UC-secure) two-round PAKE scheme from a (UC-secure) PAPKE scheme, but that while the converse looks like it should be true at first sight, it is not true in general, because PAPKE has stricter properties than a standard PAKE. In particular, we give a counterexample of a secure two-round PAKE scheme that, when converted into a PAPKE scheme in the straightforward fashion, yields an insecure PAPKE scheme. Indeed, PAPKE can be thought of as a two-round PAKE with a novel property of security under session state re-use, which to the best of our knowledge has not been observed and provably realized before.

Constructing PAKE from PAPKE. We show that any UC-secure PAPKE can be converted into a two-round UC-secure PAKE. This construction is shown in Figure 1, and it is fairly simple: The initiator $\mathcal{P}_i$ generates an authenticated public key apk from the input password pwd and sends it to $\mathcal{P}_j$. The responder $\mathcal{P}_j$, given its password pwd' and the received public key apk, picks a random session key $k \leftarrow_{\mathrm{R}} \{0,1\}^\kappa$, and responds to $\mathcal{P}_i$ with an encryption of k under apk and pwd'. $\mathcal{P}_i$ receives key k by decrypting the received ciphertext, or outputs $\perp$ if the decryption fails. Note that all communication is done over an insecure channel, fully controlled by the adversary. In particular, an adversary can replace $\mathcal{P}_i$'s public key and/or $\mathcal{P}_j$'s ciphertext. However, PAPKE security implies that neither $\mathcal{P}_i$'s public key nor $\mathcal{P}_j$'s ciphertext reveal anything about passwords, resp. pwd and pwd', and the only attack the adversary can stage is an on-line guessing

attack, because each substituted public key apk^* or ciphertext c^* commits the adversary to a *single* password guess pwd^*, and is guaranteed to fail (e.g. $\mathcal{P}_j$ fails to encrypt anything useful under apk^* or $\mathcal{P}_i$ fails to decrypt c^*) unless the guessed password pwd^* matches the password of resp. $\mathcal{P}_j$ or $\mathcal{P}_i$.

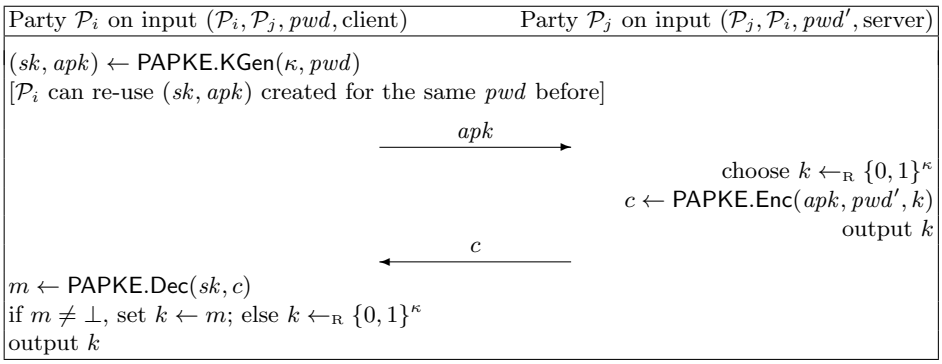
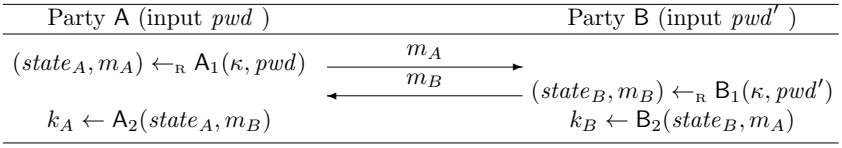


Fig. 1. Two-round PAKE protocol $\mathsf{PAPKE\text{-}2\text{-}PAKE}$ given $\mathsf{PAPKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$.

The proof of Theorem 1 is included in the full version [12]. Note that if the PAKE initiator $\mathcal{P}_i$ chooses to re-use its state (sk, apk) across protocol instances which share the same input pwd then $\mathcal{P}_i$ reveals that all these instances share the same input, hence such protocol can only realize functionality $\mathcal{F}_{\mathsf{PAKE}}$ modified so that a party can choose to reveal that two of its sessions run on the same password.

Theorem 1. *If* PAPKE *realizes the UC PAPKE functionality* $\mathcal{F}_{\mathsf{PAPKE}}$*, defined in [12], then the* $\mathsf{PAPKE\text{-}2\text{-}PAKE}$ *scheme shown in Figure 1 realizes the UC PAKE functionality* $\mathcal{F}_{\mathsf{PAKE}}$ *[15].*

An intuitive $\mathsf{PAKE\text{-}2\text{-}PAPKE}$ **compiler, and why it doesn't work.** It turns out that the intuitive approach of building PAPKE from two-round PAKE does not work due to subtle differences in the security notions of both primitives. Indeed, PAPKE has some security properties which are stronger than PAKE, and this in particular implies that the PAPKE-to-PAKE compiler shown above adds a new security property to the resulting PAKE. (We discuss that PAKE security property below.) For the ease of exposition, we state our results for the game-based representations of PAKE and PAPKE instead of using their UC variants, and refer to parties $\mathcal{P}_i$ and $\mathcal{P}_j$ as A and B respectively. On a first glance, it seems reasonable to generically build a PAPKE scheme from any two-round PAKE protocol, e.g. [8, 7, 4]. Specifically, any two-round PAKE protocol $\langle(\mathsf{A}_1, \mathsf{A}_2) \rightleftharpoons (\mathsf{B}_1, \mathsf{B}_2)\rangle$ can be abstracted as follows:



The natural approach to constructing PAPKE would combine a two-round PAKE with an authenticated encryption scheme AE: The PAKE message m_A from A would be A's static authenticated public key apk, and to encrypt message m under A's key $apk = m_A$ any party could complete the two-round PAKE protocol in the role of B and append the AE encryption of m under the derived session key k_B to B's PAKE message m_B. For decryption, A uses m_B to complete her side of the PAKE protocol to derive the same session key $k_A = k_B$ (if $pwd = pwd'$) and uses k_A to decrypt the attached ciphertext. More formally, given a 2-round $\mathsf{PAKE} = \langle(\mathsf{A}_1, \mathsf{A}_2) \rightleftharpoons (\mathsf{B}_1, \mathsf{B}_2)\rangle$ and authenticated encryption $\mathsf{AE} = (\mathsf{AE.Enc}, \mathsf{AE.Dec})$ sharing the same key space $\mathcal{K}$, one could consider the following PAPKE construction:

$\mathsf{PAPKE.KGen}(\kappa, pwd)$:
  run $(state_A, m_A) \leftarrow_R \mathsf{A}_1(\kappa, pwd)$, return $(sk \leftarrow state_A, apk \leftarrow m_A)$
$\mathsf{PAPKE.Enc}(apk, pwd', m)$:
  run $(state_B, m_B) \leftarrow_R \mathsf{B}_1(\kappa, pwd')$ and $k_B \leftarrow \mathsf{B}_2(state_B, apk)$
  encrypt $c \leftarrow \mathsf{AE.Enc}(k_B, m)$ and return $c' \leftarrow (m_B, c)$
$\mathsf{PAPKE.Dec}(sk, c')$:
  parse $c' = (m_B, c)$ and $sk = state_A$
  get $k_A \leftarrow \mathsf{A}_2(state_A, m_B)$ and return $m \leftarrow \mathsf{AE.Dec}(k_A, c)$

Intuitively, this should yield a secure PAPKE if PAKE is secure. However, this generic construction uses PAKE in a way that is not covered by its security definition: Whenever party A decrypts a PAPKE ciphertext it effectively *re-uses* the same local PAKE session state $state_A$ (and the same first-round message m_A) across multiple PAKE sessions. Indeed, this gap can be exploited to craft special PAKE and AE schemes that are secure by themselves but result in an insecure PAPKE when used in this natural compiler. (The full formal description of this counterexample is included in [12].)

Implications for UC PAKE protocols. We discuss the main conclusions we draw from the two technical facts above.

First 2-round UC PAKE's competitive with game-based PAKE's. In Appendix A we include two highly efficient UC PAKE protocols by instantiating the PAPKE-2-PAKE compiler with the PAPKE constructions of Section 4. To the best of our knowledge these are the first 2-round UC PAKE's which rely on standard cyclic groups with efficiency comparable to the Diffie-Hellman key exchange in the IC or RO model. While UC PAKE can be achieved using even 1 (simultaneous) round of communication, all 1-round UC PAKEs we know, e.g. [22, 21], use groups with bilinear maps and are significantly costlier. Thus practitioners are

likely to resort to constructions which require IC or ROM models but give much better concrete efficiency.

Concretely, we show two 2-round UC PAKE protocols: PAKE-IC-DHIES, secure under Oracle Diffie-Hellman (ODH) [2] in the IC model which uses 2 exponentiations per party, and PAKE-FO, secure under DDH in ROM which uses 4 (multi-)exps for the requester and 2 for the responder. The Universally Composable notion of PAKE security [15] has long been recognized as stronger than the game-based notions [7, 11], not only because it implies concurrent security and can be used in protocol composition, but also because, unlike the game-based notions, the UC PAKE implies security for non-uniform password distributions, password re-use, correlated passwords, misstyped passwords, and any other forms of information leakage. However, there has been an efficiency and round-complexity gap between UC PAKE's and PAKE's shown secure under game-based notions with the 3-round 2-exp/party UC PAKE of Abdalla et al. [3], which assumes DDH in IC model, coming closest to the 2-round 2-exp/party game-based PAKE of Abdalla-Pointcheval [4], which assumes DDH in ROM. Our UC PAKE constructions match [4] in round complexity, and our IC model construction also matches [4] in the number of exponentiation operations.[5]

PAKE with security on session re-use. As we argued above, the reason the compiler from 2-round PAKE to PAPKE does not work is that a standard PAKE security model does not extend to the case of the requester party, A, re-using the local state $state_A$ of a single PAKE session across many sessions, each of which would derive a session key k_A from same state k_A but potentially different responder messages m_B. By contrast, PAKE created from the secure PAPKE in Figure 1 does have this property: The requester party $\mathcal{P}_i$ can use the same local state, which is the PAPKE secret key sk, across many sessions, deriving $k_A \leftarrow \mathsf{PAPKE.Dec}(sk, c)$ on any number of responder messages c. By the same token, the responder $\mathcal{P}_j$ in this PAKE protocol is free to re-use $\mathcal{P}_i$'s first-round message apk in multiple sessions, because PAPKE ciphertexts created in each such session are all secure, and their plaintexts can all be used as session keys.

Indeed, this shows that protocol PAPKE-2-PAKE is a secure 2-round PAKE with security under re-use of requester's session state across multiple sessions. This can improve efficiency in PAKE applications where the initiator re-uses same password across multiple sessions (and does not mind revealing that fact), and it can also make it easier to handle communication faults, because both parties can keep their session information, the session state $state_A = sk$ for $\mathcal{P}_i$ and the requester's first message $m_A = apk$ for $\mathcal{P}_j$, and re-use them in case of communication faults instead of re-starting from scratch.

4 Efficient & UC-Secure PAPKE Constructions

First attempts to construct PAPKE schemes that authenticate public keys and plaintexts with a password would probably involve message authentication codes

[5] However, our local computation cost also includes Ideal Cipher operations.

(MACs) of the public key and/or the enrypted plaintext under a key derived from the password. Such solutions, however, fall prey to offline dictionary attacks, either given just the authenticated public key, or by substituting the real public key with an adversarial one and testing the decrypted MAC. Thus the challenge is to devise schemes that withstand offline attacks and achieve the strong security guarantees formalized in our UC and property-based definitions. We present two very practical PAPKE constructions that achieve this goal.

The first construction, PAPKE-IC in Section 4.1, combines any CCA secure public-key encryption and an ideal cipher, using the ideal cipher to encrypt the public key with the password as a key. We prove this PAPKE scheme secure in the ideal-cipher model if the PKE scheme satisfies a number of properties that go beyond the standard CCA security, namely key anonymity, robustness, and the requirement that public keys are uniform in the (ideal) cipher domain.

While the PAPKE-IC construction is conceptually simple, instantiating the combination of ideal ciphers and public-key encryption requires some care, and subtle implementation mistakes could render the PAPKE-IC construction insecure (see the discussion in Section 4.1 below). Hence we propose a second PAPKE construction, PAPKE-FO in Section 4.2, which is not generic, but it does not need an ideal cipher and therefore might be easier to implement. It is based on a twin-key version of the Fujisaki-Okamoto transform of ElGamal encryption, and it is secure under the DDH assumption in ROM.

4.1 PAPKE-IC: Generic Construction from PKE and Ideal Cipher

Our first construction, protocol PAPKE-IC in Figure 2, builds PAPKE generically from a public-key encryption PKE and an ideal cipher $\mathsf{IC} = (\mathsf{IC.Enc}, \mathsf{IC.Dec})$. The basic idea of the construction is simple and similar to the Encrypted Key Exchange (EKE) PAKE of Bellovin and Merritt [8]: The receiver generates a key pair for the PKE scheme and encrypts the public key under the ideal cipher using the password as a key. The resulting encrypted public key is used as PAPKE authenticated public key *apk*. To encrypt a message, the sender decrypts *apk* under the ideal cipher using the password as a key, and encrypts the message under the resulting public key. Our PAPKE-IC shares this basic design with EKE, except that we use a CCA-secure encryption while EKE implicitely uses a version of (CPA-secure) ElGamal whose security *as encryption* is less clear.

Protocol PAPKE-IC requires a number of properties of the PKE scheme that go beyond the standard notion of CCA security. First, its public keys must be *uniformly distributed* over the domain of the ideal cipher, because otherwise an attacker can test passwords offline by trying to decrypt *apk*. Second, ciphertexts of the PKE cannot reveal under which public key they were encrypted, as that would allow offline attacks as well. The second property is known as key privacy or *anonymity* [6]. Third, and perhaps a bit harder to see, is that an adversary should be unable to construct ciphertexts that decrypt correctly under multiple secret keys, but such ciphertext would allow the adversary to test multiple password guesses in one query to the decryption oracle. This property is known as *strong robustness* [1]. The latter two properties are formalized as, respectively,

Setup: Let $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with uniform public-key space $\mathcal{PK}$ and let $\mathsf{IC} = (\mathsf{IC.Enc}, \mathsf{IC.Dec})$ be an ideal cipher over $\mathcal{PK}$.

$\underline{\mathsf{PAPKE.KGen}(\kappa, pwd)}$:
Generate $(pk, sk) \leftarrow_{\mathrm{R}} \mathsf{PKE.KGen}(\kappa)$ and $apk \leftarrow \mathsf{IC.Enc}(pwd, pk)$ and output (sk, apk).

$\underline{\mathsf{PAPKE.Enc}(apk, pwd', m)}$:
Decrypt the public key, $pk \leftarrow \mathsf{IC.Dec}(pwd, apk)$ and output $c \leftarrow_{\mathrm{R}} \mathsf{PKE.Enc}(pk, m)$.

$\underline{\mathsf{PAPKE.Dec}(sk, c')}$:
Decrypt $m \leftarrow \mathsf{PKE.Dec}(sk, c')$ and output m.

Fig. 2. The generic PAPKE scheme PAPKE-IC.

AI-CCA and SROB-CCA (see [12]). Finally, PKE and IC have to be "compatible" in the sense that IC is an ideal cipher over the key space $\mathcal{PK}$ of PKE.

The proof of the following theorem appears in the full version [12]:

Theorem 2. *Protocol* PAPKE-IC *in Figure 2 securely realizes functionality* $\mathcal{F}_{\mathsf{PAPKE}}$ *in the* $\mathcal{F}_{\mathsf{IC}}$*-hybrid model, if the public key encryption* PKE *has uniform public-key space* $\mathcal{PK}$ *and is* AI-CCA *and* SROB-CCA*-secure.*

Implementing Ideal Ciphers over Groups. Our PAPKE-IC construction assumes an ideal cipher over a key space $\mathcal{PK}$ that for many PKE schemes will be a cyclic group $\mathbb{G}$. We stress that such an assumption is also used in several PAKE schemes, beginning from the Bellare et al. analysis [7] of the Encrypted Key Exchange (EKE) PAKE scheme of Bellovin and Merritt [8]. Ideal ciphers over variable domains can be implemented for a variety of domains, e.g. [10]. However, for many groups implementing an ideal cipher is somewhat cumbersome and can introduce possibilities for offline and/or timing attacks. Simply applying a block cipher to the public key doesn't work as not all strings of the same length are valid group elements, and an adversary could offline tests by decrypting the authenticated public key under a guessed password and testing if the decryption yields a valid group element. If $\mathcal{PK} = \mathbb{G}$ is any elliptic curve group, there are deterministic methods that map any string onto a group element [9] and hence offline and timing attacks are not a concern. The opposite direction can be implemented as in [9], but that encoding works only for subspace S of roughly $1/2$ of $\mathbb{G}$ elements. This slows down key generation, i.e. pair $(pk, sk) \leftarrow_{\mathrm{R}} \mathsf{PKE.KGen}$ has to be chosen s.t. $pk \in S$, but it does not lead to timing attacks on passwords. Still, these mappings complicate key generation and are non-trivial to implement, which motivates searching for alternative solutions that do not rely on ideal ciphers over arbitrary groups.

DHIES-based Instantiation. In Appendix A, we specify an efficient concrete instantiation of PAPKE-IC, called PAPKE-IC-DHIES, which uses a variant of DHIES as the robust and anonymous PKE. Scheme PAPKE-IC-DHIES is as efficient as one could hope for in a DH-based cryptosystem, i.e. it uses 1 exponentiation in key generation, 2 exponentiations in encryption, and 1 in decryption. The

DHIES variant we use (DHIES^*) was shown to satisfy the required properties under the Oracle-Diffie-Hellman assumption (ODH), using a collision-resistant hash function and a secure authenticated encryption scheme [1]. The authenticated encryption (or rather the combination of symmetric encryption and a MAC) needs to satisfy some additional, non-standard properties, and the ODH assumption also has an impact on the choice of the hash function. We refer to Appendix A for a more detailed discussion. Thus, similar to the challenges that arise when securely instantiating the ideal cipher, implementing DHIES^* also requires some care in the implementation and choice of its underlying primitives.

4.2 PAPKE-FO: Concrete Construction from DDH and ROM

Our second PAPKE construction, protocol PAPKE-FO in Figure 3, does not require an ideal cipher over a group of PKE public keys,and may thus be easier to implement. It is however slightly more costly, with 2 exponentiations for key generation, 2 multi-exponentiations (with two bases) for encryption, and 1 exponentiation and 1 (two base) multi-exponentiation for decryption. This construction is built using the Fujisaki-Okamoto (FO) transform [17] for ElGamal encryption but with a "twin" Diffie-Hellman key instead of a single key.

The high-level idea is to derive the authenticated public key apk by "blinding" the public key g^x of the ElGamal encryption scheme with the hash of the password as $apk \leftarrow g^x \cdot \mathsf{H}_0(pwd)$, where H_0 is a hash function onto $\mathbb{G}$, which can be implemented in deterministic way (to avoid timing attacks) using e.g. [9]. To encrypt message m under password pwd' and key apk, the encryptor "unblinds" the public key as $y \leftarrow apk \cdot \mathsf{H}_0(pwd')^{-1}$ and then encrypts m under y using FO-ElGamal, i.e. the Fujisaki-Okamoto transform applied to ElGamal which lifts its security from CPA to CCA, required to achieve the CCA-security and ciphertext authenticity properties of PAPKE.

None of the password-derived values apk or c allows an offline attack: Any "unblinding" of apk would yield a valid public key g^x for some x, and ElGamal ciphertexts are known to guarantee key anonymity [6], meaning that ciphertexts do not leak information about the public key used in encryption. (Note that the leakage of the unblinded public key $y = g^x$ used in encryption would allow an adversary who sees $apk = y \cdot \mathsf{H}_0(pwd)$ to mount an offline attack on pwd.) The scheme is correct because if $pwd' = pwd$ then the hash values cancel and encryption is done under the "original" public key $y = g^x$. However, if the passwords do not match then encryption is done under an effectively random public key $y \leftarrow_{\mathrm{R}} \mathbb{G}$. The latter gives us the desired security against active attacks: If an honest party is tricked into encryption under a malicious apk^* but uses a different password than the one which was used to blind apk^*, then the ciphertext will be indistinguishable from random, even if $\mathcal{A}$ knows the secret key to apk^*. Note, however, that unlike the ideal cipher encryption of apk under pwd used in PAPKE-IC, the method used here to blind key g^x and form the authenticated key apk is essentially a one-time pad over $\mathbb{G}$, and thus is not by itself a commitment to password pwd. Below we discuss how we modify the above sketch and in particular make this blinding password-committing.

Note that in Figure 3 the message space is $\mathcal{M} = \{0,1\}^n$ for fixed n but it can be extended to arbitrary messages e.g. using $\mathsf{H}_2(R)$ as a key in symmetric-key encryption instead of as a one-time pad.

Setup: Let $\mathbb{G}$ be a group of prime order p such that $2^{\kappa-1} < p < 2^\kappa$, and let g_1, g_2 be two random generators in $\mathbb{G}$. We use three hash functions modeled as random oracles: $\mathsf{H}_0 : \{0,1\}^* \to \mathbb{G}$, $\mathsf{H}_1 : \mathbb{G}^3 \times \{0,1\}^n \to \mathbb{Z}_p^2$ and $\mathsf{H}_2 : \mathbb{G} \to \{0,1\}^n$.

$\underline{\mathsf{PAPKE.KGen}(\kappa, pwd)}$:
- Choose $x \leftarrow_{\mathrm{R}} \mathbb{Z}_p$ and compute $y_1 \leftarrow g_1^x,\ y_2 \leftarrow g_2^x,\ Y_2 \leftarrow y_2 \cdot \mathsf{H}_0(pwd)$.
- Output (sk, apk) for $sk \leftarrow (x, y_1, y_2)$ and $apk \leftarrow (y_1, Y_2)$.

$\underline{\mathsf{PAPKE.Enc}(apk, pwd', m)}$:
- Abort if $|m| > n$.
- Parse $(y_1, Y_2) \leftarrow apk$, and "unblind" the second public key, $y_2 \leftarrow Y_2 \cdot \mathsf{H}_0(pwd)^{-1}$.
- Generate randomness via the RO: compute $(r_1, r_2) \leftarrow \mathsf{H}_1(R, y_1, y_2, m)$ for $R \leftarrow_{\mathrm{R}} \mathbb{G}$.
- Encrypt R under y_1 and y_2: $c_1 \leftarrow g_1^{r_1} g_2^{r_2},\ c_2 \leftarrow y_1^{r_1} y_2^{r_2} \cdot R,\ c_3 \leftarrow \mathsf{H}_2(R) \oplus m$.
- Output $c \leftarrow (c_1, c_2, c_3)$.

$\underline{\mathsf{PAPKE.Dec}(sk, c')}$:
- Parse $(x, y_1, y_2) \leftarrow sk$.
- Decrypt $c = (c_1, c_2, c_3)$: $R \leftarrow c_2 / c_1^x$, $m \leftarrow c_3 \oplus \mathsf{H}_2(R)$, and $(r_1, r_2) \leftarrow \mathsf{H}_1(R, y_1, y_2, m)$.
- Verify the correctness of decryption: if $c_1 = g_1^{r_1} g_2^{r_2}$ set $m' \leftarrow m$, else set $m' \leftarrow \perp$.
- Output m'

Fig. 3. Our DDH-based PAPKE scheme PAPKE-FO.

Achieving UC-Security via "Twin" Keys. To achieve UC security we have to ensure that both the key apk and the ciphertext c commit each party to a well-defined password choice. Technically, the simulator SIM must be able to extract (i) pwd from an adversarial apk^* and (ii) pwd' and m from an adversarial ciphertext c. While (ii) can be realized via the Fujisaki-Okamoto transform, case (i) requires more care. We need (i) for the reasons outlined above, i.e. a ciphertext encrypted by an honest party under an adversarial key apk^* must be decryptable only if apk^* commits to the encryptor's password. In the UC functionality $\mathcal{F}_{\mathsf{PAPKE}}$ this is enforced by SIM having to pass a single password guess pwd^* corresponding to the real-life adversary's choice of apk^*, and if $pwd^* \neq pwd'$, i.e., the guess does not match the encryptor's password pwd', then the encryption must reveal no information on the encrypted plaintext.

We achieve this by generating a "twin" public key using two generators g_1, g_2 in the CRS. The apk then consists of $y_1 \leftarrow g_1^x$ and $Y_2 \leftarrow g_2^x \cdot \mathsf{H}_0(pwd)$, i.e., we keep one public key in the clear and the other one is blinded with the password hash. In the security proof we set $g_2 \leftarrow g_1^s$, which allows the simulator to decrypt $\mathsf{H}_0(pwd)$ from apk, and look up pwd from the random oracle queries. Further, encryption is done under *both* public keys: y_1 and the "unblinded" $y_2 = Y_2 \cdot \mathsf{H}_0(pwd)^{-1}$. This double encryption under the plain and derived key is crucial, as it prevents an adversary $\mathcal{A}$ from providing a malformed apk^* which

would allow $\mathcal{A}$ to still decrypt, but from which SIM cannot extract a password. Thus, our "twin" key construction enforces that only a well-formed *apk* can lead to decryptable ciphertexts (if the passwords match), without requiring heavy tools such as zero-knowledge proofs.

For space-saving reasons the proof of the following theorem is relegated to the full version of the paper. Functionalities $\mathcal{F}_{\mathsf{CRS}}$ and $\mathcal{F}_{\mathsf{RO}}$ are UC models for resp. the CRS string and the RO hash functions we assume in this construction.

Theorem 3. *Protocol* $\mathsf{PAPKE\text{-}FO}$ *in Figure 3 securely realizes functionality* $\mathcal{F}_{\mathsf{PAPKE}}$ *under the DDH assumption in group* $\mathbb{G}$ *in the* $\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{RO}}$*-hybrid model.*

Acknowledgments. Anja Lehmann was supported by the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 786725 (OLYMPUS). Tatiana Bradley, Stanislaw Jarecki, and Jiayu Xu were supported by the NSF Cybersecurity Innovation for Cyberinfrastructure (CICI) Grant Award No. ACI-1547435.

References

1. Abdalla, M., Bellare, M., Neven, G.: Robust encryption. Cryptology ePrint Archive, Report 2008/440 (2008), `http://eprint.iacr.org/2008/440`
2. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (Apr 2001)
3. Abdalla, M., Catalano, D., Chevalier, C., Pointcheval, D.: Efficient two-party passwordbased key exchange protocols in the uc framework. In: The Cryptographers Track at RSA Conference (CT-RSA) (2008)
4. 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 (Feb 2005)
5. Abu-Salma, R., Sasse, M.A., Bonneau, J., Danilova, A., Naiakshina, A., Smith, M.: Obstacles to the adoption of secure communication tools. In: 2017 IEEE Symposium on Security and Privacy. pp. 137–153. IEEE Computer Society Press (May 2017)
6. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (Dec 2001)
7. 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 (May 2000)
8. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy. pp. 72–84. IEEE Computer Society Press (May 1992)
9. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13. pp. 967–980. ACM Press (Nov 2013)

10. Black, J., Rogaway, P.: Ciphers with Arbitrary Finite Domains, pp. 114–130. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
11. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (May 2000)
12. Bradley, T., Camenisch, J., Jarecki, S., Lehmann, A., Neven, G., Xu, J.: Password-authenticated public key encryption. Cryptology ePrint Archive, Report 2019/199 (2019), http://eprint.iacr.org/2019/199
13. Burr, W.E., Dodson, D.F., Newton, E.M., Perlner, R.A., Polk, W.T., Gupta, S., Nabbus, E.A.: Electronic authentication guideline. NIST Special Publication 800-63-1 (2011)
14. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001)
15. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (May 2005)
16. Dai, Y., Steinberger, J.P.: Indifferentiability of 8-round Feistel networks. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 95–120. Springer, Heidelberg (Aug 2016)
17. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999)
18. Guardian: Whatsapp design feature means some encrypted messages could be read by third party. https://www.theguardian.com/technology/2017/jan/13/whatsapp-design-feature-encrypted-messages/ (2017)
19. Holenstein, T., Künzler, R., Tessaro, S.: The equivalence of the random oracle model and the ideal cipher model, revisited. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 89–98. ACM Press (Jun 2011)
20. Huima, A.: The Bubble Babble binary data encoding. http://web.mit.edu/kenta/www/one/bubblebabble/spec/jrtrjwzi/draft-huima-01.txt/ (2000)
21. Jutla, C.S., Roy, A.: Dual-system simulation-soundness with applications to UC-PAKE and more. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 630–655. Springer, Heidelberg (Nov / Dec 2015)
22. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. Journal of Cryptology 26(4), 714–743 (Oct 2013)
23. OpenSSH 5.1 release announcement. https://www.openssh.com/txt/release-5.1/ (2008)
24. Rivest, R.L., Lampson, B.: SDSI – a simple distributed security infrastructure. http://people.csail.mit.edu/rivest/sdsi10.html/ (1996)
25. Tan, J., Bauer, L., Bonneau, J., Cranor, L.F., Thomas, J., Ur, B.: Can unicorns help users compare crypto key fingerprints? In: Mark, G., Fussell, S.R., Lampe, C., m. c. schraefel, Hourcade, J.P., Appert, C., Wigdor, D. (eds.) CHI Conference on Human Factors in Computing Systems. pp. 3787–3798. ACM (2017)
26. Tufekci, Z.: In response to guardians irresponsible reporting on whatsapp: A plea for responsible and contextualized reporting on user security. http://technosociology.org/?page_id=1687/ (2017)
27. Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., Smith, M.: SoK: Secure messaging. In: 2015 IEEE Symposium on Security and Privacy. pp. 232–249. IEEE Computer Society Press (May 2015)
28. WhatsApp encryption overview: Technical white paper. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf/ (2016)

A Concrete PAPKE and PAKE Instantiation Example

Here we show particular instantiations of some of our results, a PAPKE scheme PAPKE-IC-DHIES and a PAKE protocol PAKE-IC-DHIES. PAPKE-IC-DHIES is a particular instantiation of the generic PAPKE-IC scheme of Section 4.1 based on the DHIES* PKE by Abdalla et el. [1], and protocol PAKE-IC-DHIES is derived via the PAPKE-2-PAKE compiler of Section 3 applied to PAPKE-IC-DHIES.

Setup: $\mathbb{G}$ is a cyclic group of prime order p with generator g; $\mathsf{IC} = (\mathsf{IC.Enc}, \mathsf{IC.Dec})$ is an Ideal Cipher over $\mathbb{G}$ with key space $\{0,1\}^*$; $\mathsf{AE} = (\mathsf{AE.Enc}, \mathsf{AE.Dec})$ is an authenticated encryption with key space $\{0,1\}^\kappa$; $\mathsf{H} : \mathbb{G} \to \{0,1\}^\kappa$ is a collision-resistant hash function.

$\underline{\mathsf{PAPKE.KGen}(\kappa, pwd)}$:

- Pick $x \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, compute $y \leftarrow g^x$ and $apk \leftarrow \mathsf{IC.Enc}(pwd, y)$.
- Assign $sk \leftarrow x$ and output (sk, apk).

$\underline{\mathsf{PAPKE.Enc}(apk, pwd', m)}$:

- Compute $y \leftarrow \mathsf{IC.Dec}(pwd, apk)$, $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p^*$, $k \leftarrow \mathsf{H}(y^r)$, $c_1 \leftarrow g^r, c_2 \leftarrow \mathsf{AE.Enc}(k, m)$.
- Output $c = (c_1, c_2)$.

$\underline{\mathsf{PAPKE.Dec}(sk, c')}$:

- Parse $(c_1, c_2) \leftarrow c$, compute $k \leftarrow \mathsf{H}({c_1}^x)$.
- If $c_1 = 1$ set $m \leftarrow \perp$ otherwise set $m \leftarrow \mathsf{AE.Dec}(k, c_2)$, and output m.

Fig. 4. Concrete PAPKE instantiation PAPKE-IC-DHIES.

Concrete Instantiation of PAPKE-IC Using DHIES. In Section 4.1 we show a generic UC-secure PAPKE scheme that relies on an ideal cipher and a public-key encryption scheme that is both AI-CCA and SROB-CCA-secure. Abdalla et al. [1] show that these properties can be realized by DHIES*, a simple modification of DHIES [2] which excludes zero randomness at encryption, i.e., samples r from $\mathbb{Z}_p^*$ instead of $\mathbb{Z}_p$, and rejects ciphertexts that have 1 as first component. We specify DHIES* below relying on authenticated encryption AE, a hash function H and a cyclic group $(\mathbb{G}, p, g)$ of prime order p. Scheme PAPKE-IC-DHIES in Figure 4 is a (semi) concrete instantiation of PAPKE-IC using DHIES*, which uses 2 exponentiations for encryption and 1 for decryption, as well as an ideal cipher over group $\mathbb{G}$ and hashing onto $\mathbb{G}$.

$\mathsf{DHIES^*.KGen}(\kappa)$**:** $x \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $y \leftarrow g^x$, set $pk \leftarrow y, sk \leftarrow x$ and return (pk, sk)
$\mathsf{DHIES^*.Enc}(pk, m)$**:** parse $pk = y$, get $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p^*$, $k \leftarrow \mathsf{H}(y^r)$, $c_1 \leftarrow g^r$, $c_2 \leftarrow \mathsf{AE.Enc}(k, m)$ and return $c = (c_1, c_2)$.
$\mathsf{DHIES^*.Dec}(sk, c)$**:** parse $c = (c_1, c_2)$ and $sk = x$, get $k \leftarrow \mathsf{H}(c_1^x)$. If $c_1 = 1$ output $m \leftarrow \perp$ and $m \leftarrow \mathsf{AE.Dec}(k, c_2)$ else.

Concrete PAKE Protocols. We specify an example of a concrete UC PAKE instantiation obtained by applying the generic PAPKE-2-PAKE compiler shown

in Figure 1 to the PAPKE scheme PAPKE-IC-DHIES shown in Figure 4. In [12] we also specify PAKE protocol PAKE-FO implied by our second PAPKE construction, PAPKE-FO of Figure 3. To the best of our knowledge, these are the first two-round UC-secure PAKE's which rely on standard groups, i.e. no bilinear maps, but resort to the IC and/or ROM model to achieve practical efficiency. Concretely, PAKE-IC-DHIES uses from 2 exponentiations per party and PAKE-FO uses 4 (multi-)exponentiations for one party and 2 for the other. This *almost* matches the efficiency and assumptions used by two-round PAKE's which were shown secure under only game-based security notions, e.g. [7, 11, 4], and it reduces from 3 to 2 the rounds of previously known UC PAKE secure under comparable assumptions of Abdalla et al. [3].

Protocol PAKE-IC-DHIES shown in Figure 5 requires the same setup as the PAPKE scheme PAPKE-IC-DHIES in Figure 4, i.e. $\mathbb{G}$ is a cyclic group of prime order p with generator g , $\mathsf{IC} = (\mathsf{IC.Enc}, \mathsf{IC.Dec})$ is an ideal cipher over group $\mathbb{G}$ with key space $\{0,1\}^*$, $\mathsf{AE} = (\mathsf{AE.Enc}, \mathsf{AE.Dec})$ is an authenticated encryption with key space $\{0,1\}^\kappa$, and $\mathsf{H} : \mathbb{G} \to \{0,1\}^\kappa$ is a collision-resistant hash. The following security statement for PAKE-IC-DHIES follows from Theorem 1, Theorem 2, and the security properties of DHIES^* [1]:

Corollary 1. *The* PAKE-IC-DHIES *scheme described in Figure 5 securely realizes* $\mathcal{F}_{\mathsf{PAKE}}$ *in the* $\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{IC}}$*-hybrid model if the Oracle-Diffie-Hellman assumption is hard for* $\mathbb{G}$*,* H *is a collision-resistant hash, and* AE *is a secure, strongly unforgeable and collision-resistant authenticated encryption scheme.*

Party $\mathcal{P}_i$, upon input (NEWSESSION, $sid', \mathcal{P}_i, \mathcal{P}_j, pwd$, client):	Party $\mathcal{P}_j$, upon input (NEWSESSION, $sid', \mathcal{P}_j, \mathcal{P}_i, pwd'$, server):
get $x \leftarrow_{\mathrm{R}} \mathbb{Z}_p$, $y \leftarrow g^x$	wait for message with prefix sid'
set $apk \leftarrow \mathsf{IC.Enc}(pwd, y)$, store $sk \leftarrow x$	
$\xrightarrow{sid',\ apk}$	
	choose $k \leftarrow_{\mathrm{R}} \{0,1\}^\kappa$
	get $y \leftarrow \mathsf{IC.Dec}(pwd', apk)$, $r \leftarrow_{\mathrm{R}} \mathbb{Z}_p^*$, $k' \leftarrow \mathsf{H}(y^r)$,
	$c_1 \leftarrow g^r, c_2 \leftarrow \mathsf{AE.Enc}(k', k)$, set $c \leftarrow (c_1, c_2)$
	output (NEWKEY, sid', k)
$\xleftarrow{sid',\ c}$	
parse $c = (c_1, c_2)$, get $k' \leftarrow \mathsf{H}(c_1^x)$	
if $c_1 = 1$ set $m \leftarrow \perp$; else $m \leftarrow \mathsf{AE.Dec}(k', c_2)$	
if $m \neq \perp$, set $k \leftarrow m$; else $k \leftarrow_{\mathrm{R}} \{0,1\}^\kappa$	
output (NEWKEY, sid', k)	

Fig. 5. Two-round PAKE protocol PAKE-IC-DHIES.