Software-Only Two-Factor Authentication Secure Against Active Servers

Julien Bringer[1], Hervé Chabanne[1,2], and Roch Lescuyer[1(✉)]

[1] Morpho, Issy-les-moulineaux, France
roch.lescuyer@morpho.com
[2] Télécom ParisTech, Paris, France

Abstract. In most password-based authentication protocols, the server owns a value, the so-called verifier, that depends on the registered password. This verifier is often a one-way function of the password. Despite this protection, an unauthorized person who gets access to the verifier can mount a brute-force attack to recover the password. If the entropy of the password is low, which is often the case in practice, such an attack might be successful. Motivated by the growing need to face databases compromises, we propose a two-factor password-based authentication protocol where no information about the password leak from the server's side nor from the client's side, and where the password is not sent to the server when the user authenticates. During the registration, a user gets a value, called the token, while the server records the verifier. Our security model ensures that brute-force attacks are impossible if the server is compromised. Moreover, only on-line attempts are possible if a token is stolen. The solutions that we describe fit well into scenarios where the token is stored on a mobile phone. We provide constructions, proven secure in the random-oracle model, under standard assumptions.

1 Introduction

Password-based authentication (PA) is the most wide-spread way deployed to authenticate users. A lot of advanced forms of authentications have been developed by the research community. However, the simplest form of PA, consisting of a (login, password) pair, widely remains the method in use. Moreover, this form of authentication takes more and more place in citizen's life. Each entity and service, own their websites and information systems, and each of them asks the user for a password to get accessed.

The server managing the access rights stores the users' information. Fortunately, most of the time, the password is not directly stored. Instead, a one-way, hard to invert, function is applied to the password, and the output, aka the verifier, is stored on the server. This is a first step towards password protection. Given a verifier, a brute-force recovering of the password consists of computing the output of the function for all possible values of the password and comparing the results with the verifier, sometimes with the help of dictionaries. Such an attack is implicitly assumed to be impossible. However, the password is often

D. Pointcheval et al. (Eds.): AFRICACRYPT 2016, LNCS 9646, pp. 285–303, 2016.
DOI: 10.1007/978-3-319-31517-1_15

chosen within a limited set of passwords, which makes the brute-force recovering possible. A lot of protocols aims at being secure up to dictionary attacks. In other words, the protocol ensures that the best possible attack is the dictionary attack. In this paper, we ask whether it would be possible to go beyond this bound. Ideally, we would like the verifier to leak no information at all about the password. Thus, the consequences of server compromises would be mitigated.

Our approach. First of all, we would like to enhance the security of the most common password based authentication, so we do not want to rely on specific hardware. We rather use a two-factor software-only approach. During the registration process, a user gets a value, called the token, while the server records the verifier. The two factors needed for the authentication are the password and the token. We want the brute-force recovering of the password to be impossible given only the verifier or the token. If a token is stolen, we want that only on-line attempts are possible. As usual, a bound on the number of attempts will protect the password. Such an authentication is well-suited to a mobile scenario, where the token is stored on the phone. Moreover, we do not want to rely on advanced cryptographic mechanisms, such as bilinear pairings in group of prime orders. All operations in our constructions are based on operations in a group of prime order. We propose two solutions, called pw-com and pw-hom, we now briefly introduce in this introduction. In the main body of this paper, we prove them secure in the random oracle model.

High-level view of our solution based on commitments. Our first solution, pw-com, uses the standard notion of commitments and zero-knowledge proofs. A commitment scheme enables a user to commit to a value without revealing it. The commitment binds the user to the committed value, but the user is ensured that the value is not disclosed. Then, with a zero-knowledge (ZK) proofs of knowledge (PK) of a committed value, the user proves that he knows a pair (m, r) such that $c = \mathtt{Commit}(m; r)$ without revealing any information about (m, r). Let Com be a commitment scheme, $\mathcal{P}$ be a set of password and $\mathtt{H}_h : \mathcal{P} \to \mathcal{M}_{\mathsf{C}}$ be an injective encoding from the set $\mathcal{P}$ to the message space $\mathcal{M}_{\mathsf{C}}$ of the commitment scheme. The main idea is to store a statistically hiding commitment as verifier (on the server), and the random value used to commit to the password as token (on the client). The global parameters of the scheme are a commitment key ck. In the registration phase, the user draws a uniform value $t \leftarrow \mathcal{R}_{\mathsf{C}}$ in the random space of the commitment scheme, stores t as token and sends $c := \mathtt{Commit}(\mathtt{H}_h(pw); t)$. The server stores c as verifier. In the authentication phase, the user supplies a ZKPK of (h, t) such that $c = \mathtt{Commit}(h; t)$. From a token t and a verifier c, a brute-force attack can be used to recover a password pw such that $c = \mathtt{Commit}(\mathtt{H}_h(pw); t)$. However, the knowledge of t or c alone does not help to recover the password. On the one hand, the token t leaks no information about the password (in the information-theoretic sense), so if the token is disclosed, only guesses on-line may help to recover the password. On the other hand, the verifier c statistically hides the password, so brute-force attacks are impossible, even for an unlimited adversary. Last, but not least, an authentication session does not leak any information about the password, nor the token, thanks to the zero-knowledge property of the proof of knowledge.

High-level view of our solution based on a homomorphic encryption scheme. Updates of the password in the user's side are not possible in the pw-com solution. We now ask the question whether it is possible for a user to update its password without interaction with the server. We introduce a solution, denoted pw-hom, based on a homomorphic encryption scheme over a prime order group that achieves this property. The basic idea is the following. Let $\mathbb{G}$ be a group of prime order – in additive notation here by pure convention – and hom-pke be a public key homomorphic encryption scheme over $\mathbb{G}$ as message space. After an interactive registration, the client got a pair of ciphertexts encrypting two elements $(K, [h] \cdot K)$ where $K \in \mathbb{G}$ is a user-specific element and $h \in \mathbb{Z}_p^*$ an encoding of the password. The verifier contains the decryption keys (there is one key pair per user) and a hash value of K. No information about the password leak from the verifier. In the authentication step, the client computes a proof of knowledge of the password over the ciphertexts thanks to the homomorphic properties of the encryption scheme. In other words, he computes a Schnorr signature [35] over the ciphertexts. Thanks to the verifier, the server is able to decrypt the ciphertexts and to check the proof. However the server could retrieve the password from an authentication, since it could retrieve the pair $(K, [h] \cdot K)$, then the password by brute-force recovering. Therefore, the password is first masked with a fresh random value, then the proof of knowledge is performed. As a result, no information about the password leaks from the server point of view. The user-specific element K allows for authentication on behalf of the user. A crucial point is to use two independent keys to produce the pair of ciphertexts, to prevent the computation of a ciphertext of $[h] \cdot K$ from a ciphertext of K.

On the salting. In both solutions, $\mathtt{H}_h$ is just an encoding function without any security property. Common solutions include a salt in the password hashing, but there is no need here to include such a salt. A point to be noticed is that, without salt, the $\mathtt{H}_h(pw)$ values are not uniform, because of the distribution of pw. If $\mathtt{H}_h$ were a programmable random oracle, the defect would disappear. However, the hiding property of the commitment scheme and the semantic security of the encryption scheme are sufficient to hide the password and to avoid the random oracle assumption on $\mathtt{H}_h$, a probably too strong and not realistic assumption. The only property we require is that $\mathtt{H}_h$ must be injective.

Related work. The literature about password based authentication is vast. A lot of protocols are designed to derive a session key, an issue we do not address here. The seminal work of [3] addresses authentication base on passwords only (without additional assumptions) and proposes the Encrypted Key-Exchange protocol (EKE). EKE was followed by several works [4,21,31,38,40]. The IEEE P1363.2 Password-Based Public-Key Cryptography Working Group [20] contains several of the proposals designed during the nineties. Formal models for Password-based Authenticated Key Exchange (PAKE) appeared in [2,7]. The GL's framework [16] is an abstraction of the construction of [26] (KOY), and was the first to propose a solution in the standard model. This framework underlies a lot of subsequent constructions [15,24,25,27,28]. The GK's framework [18] is an abstraction of the construction in [23], also achieves PAKE in the standard

model, but without trusted setup. A third framework (KV) [29] achieves one-round PAKE in the standard model. Recent work explicitly includes the verifier in Authenticated Key Exchange [5,30]. However, they only consider security up to the brute-force attack, according to the min-entropy of the passwords.

Turning our attention to two-factor password authentication, the constructions of [36] achieve some of the properties we look for. However, they are based on pairings, a tool we want to avoid in this paper, and they lack a formal analysis. Several commercial solutions exist, such as Google Authenticator [13], Duo Security [10], HotPin [19], and PhoneFactor [33]. The work of [37] introduces a framework to analyse these two-factor authentication protocols, then proposes several efficient constructions, and apply them in different scenarios. The participants in the protocols above are, apart from the user, a client (say a web browser), a server, and a device (say a smartphone). When authenticating, the user submits a password and some additional information supplied by the device. Our model is not the one they follow. We only assume a device (say a smartphone) authenticating to a server; *i.e.*, we do not split between a client and a device. Nevertheless, our solutions can be adapted in some of the scenarios described in [37]. We elaborate on this point in the full version of our paper.

In most existing solutions, including [37], a hashed password is stored on the server and the password is sent during the authentication protocol. To the contrary, in our solutions, the password is never sent when the user authenticates.

In anonymous password authentication [39,41], several password-based sessions from the same user cannot be linked. Although, several constructions of anonymous password authentication use homomorphic encryption, our solutions do not address the same problem – we protect the passwords against the servers, without privacy of the identities –, and are more efficient.

Finally, let us mention recent concurrent and independent work which also aims at mitigating server breaches for diverse authentication tasks. [8] introduces Virtual Smart Card, a software-only solution for signature generation, in which a signature is jointly generated by a device and a server while the user owning the device authenticates to the server with a password. The signing key is distributed between the device and the server, however the server's data alone is not sufficient to produce a signature or to mount a brute-force password-recovery attack, and the same holds for the device's data. Another work [22] introduces Device-Enhanced PAKE, in which the presence of a device in the client's side is integrated into the notion of Password-based AKE. Their model is not exactly the same as ours: the value stored in the device in their model could be the token in our model, but they also assume computation abilities in the user's side, whereas we only consider the user as the owner of a device. Last, [6] also aims at mitigating server breaches in PAKE, but with a different approach: the password database is split among several servers.

Organisation of the paper. Section 2 introduces some notations. Section 3 formally defines the two factor authentication we consider in this paper. Section 4 describes our solution based on commitments, and Sect. 5 our solution based on homomorphic encryption.

2 Notations

NOTATIONS. If x and y are strings over some alphabet, $x\|y$ denotes their concatenation. ε denotes the empty string. $\mathbb{R}$ denotes the set of real numbers, $\mathbb{R}^+$ the set of non-negative real numbers, $\mathbb{N}$ the set of non-negative integers, and $\mathbb{Z}_n$ the ring $\mathbb{Z}/n\mathbb{Z}$ of modular integers modulo the integer n. 1^n denotes the unary representation of the integer n. For an integer $n \geq 1$, $[1, n]$ denotes the set $\{1, \ldots, n\}$. If S is a set and D a distribution, $x \leftarrow_D S$ means that x is drawn from S according to D. $x \leftarrow S$ means that x is drawn according to the uniform distribution. $D \approx E$ denotes that two distributions D and E are indistinguishable (in a computational, statistical, or perfect sense depending on the context). If $\mathtt{A}$ is a (probabilistic) algorithm, $x \leftarrow \mathtt{A}(y)$ means that x is the result of the execution of $\mathtt{A}$ on input y, for some internal random coins. If these random coins used by $\mathtt{A}$ are made explicit, we note $x := \mathtt{A}(y; r)$, and $\mathtt{A}$ is deterministic. We note $x \in \mathtt{A}(y)$ to denote that x belongs to the support of $\mathtt{A}$ on input y, *i.e.*, that x might be an output of $\mathtt{A}$ on input y. The assignment phrase $a := E$ means that the value a receives the result of the evaluation of the (deterministic) expression E. A function $f : \mathbb{N} \to \mathbb{R}^+$ is said negligible if it decreases faster than any polynomial. $\mathsf{negl}(\cdot)$ denotes some unspecified negligible function.

3 Security Model

In this section, we formally defined the primitive we consider in this paper and the security properties it should satisfy. A *two-factor password-based authentication* TFPA scheme is given by a finite set $\mathcal{U} \subseteq \mathbb{N}$ of users and a set of functionalities {`Setup`, `Join`, `Issue`, `Prove`, `Verify`} described as follows.

Setup. This algorithm derives global parameters param together with a master key mk, according to a security parameter λ. We note: $\mathtt{Setup}(1^\lambda) \rightarrow (\mathsf{param}, \mathsf{mk})$.

Registration: Join↔Issue. During the registration step, the user supplies a password $pw \in \{0,1\}^*$ (possibly with low entropy) and gets a token T. The token is recorded on the user's side and the password pw is discarded. The issuer owns the master key mk, and might additionally use some user-specific auxiliary information $info \in \{0,1\}^*$ as input. The issuer outputs a user-specific value V, called the verifier, stored on a server. We note: $T \leftarrow \mathtt{Join}(\mathsf{param}, i, pw) \leftrightarrow \mathtt{Issue}(\mathsf{param}, \mathsf{mk}, i, info) \rightarrow V$.

Authentication: Prove↔Verify. On input a token T and a fresh password $\tilde{pw}$, a user authenticates to a server. The latter knows the verifier V recorded during the registration, and outputs a decision $dec \in \{\mathsf{accept}, \mathsf{reject}\}$. We note: $\mathtt{Prove}(\mathsf{param}, i, \tilde{pw}, T) \leftrightarrow \mathtt{Verify}(\mathsf{param}, i, V) \rightarrow dec$.

We assume that the registration protocol is carried out over a secure channel. We stress that the password is discarded after the registration. If the token is stolen (for instance on a mobile phone), the UF-pw security property below ensures that only guesses on-line are possible with the token. The password chosen by

the user might depend on the information used to identify him. For instance, a 4 digits PIN on a mobile phone might be chosen by the user according to its mobile number (say the last 4 digits), the phone number being precisely the information used to enrol the user and to index the verifier. Our model takes into account such dependencies through the auxiliary information $info$.

Parameters and password entropy. A TFPA scheme is parametrized by three integers $\lambda, \beta, \tau \in \mathbb{N}$. λ manages the length of the keys, as in standard cryptographic primitives. β manages the min-entropy of the password. τ is a bound on the number of attempts to authenticate on behalf of a given user. An adversary could always try to guess the password on-line and brute-forcing the password takes 2^β attempts on average. In practice, τ is set according to β. One usually sets $\tau < \beta$ (such that $\tau \ll 2^\beta$). For instance, let us assume that a PIN number is composed of four uniform digits. The server usually aborts after $\tau = 3 < \beta \approx 13$ attempts that failed.

Security properties for TFPA. Intuitively, we address two kinds of problems. From the *authentication* point of view, we want that only a registered user can authenticate to the server, knowing a valid (token, password) pair. We handle this with two unforgeability games UF-token and UF-pw. In each game, the adversary tries to authenticate knowing a factor among (password, token) and ignoring the other. From the *password protection* point of view, we do not want the password to be guessable, neither from an adaptive external adversary, nor from corrupted authorities. In the security game, Password-Leakage, the adversary tries to guess the password, knowing the server's data (including the master key) but without knowing the client's data.

Game-based definitions. Properties are expressed by games played between an adversary A against a scheme Π and a challenger C. The adversary has access to a set of oracles, described below. The set $\mathcal{L}$ records what leaks to A. The tables pw, client and server record respectively the password, data on the client's side (aka the token) and data on the server's database (aka the verifier).

Password sampling. First of all, each security game is carried out according to a set $\mathcal{P}$ of passwords and a password distribution P with min-entropy β. The challenger uses them to sample the passwords (if needed).

AddUser. A supplies $(i, info, pw)$ where i is new. If $pw \neq \perp$, the challenger adds (i, pw) to $\mathcal{L}$. If $pw = \perp$, the challenger picks a random password in $pw \leftarrow_P \mathcal{P}$ according to the distribution P. In both cases the challenger computes the token T and the verifier V from pw, $info$, mk, records $\mathsf{client}[i] := T$, $\mathsf{server}[i] := V$ and $\mathsf{pw}[i] := pw$.

SendToIssuer. A may interact with the issuer, and potentially deviate from the protocol during the registration. A supplies $(i, info)$. Values $\mathsf{pw}[i]$ and $\mathsf{client}[i]$ stay undefined. The challenger adds (i, pw), (i, client) to $\mathcal{L}$.

UserData. The adversary might ask for the i-th user's data: $\mathsf{pw}[i]$, $\mathsf{client}[i]$, or $\mathsf{server}[i]$. The corresponding pair $(i, table)$ is added to $\mathcal{L}$, where $table \in \{\mathsf{pw}, \mathsf{client}, \mathsf{server}\}$.

IssuerImpersonation: $\mathsf{SendToUser}_{issuer}$. The adversary might impersonate the issuer in front of a new user i (and deviate from the protocol). The challenger plays the role of the user i, records the corresponding $\mathsf{pw}[i]$ and $\mathsf{client}[i]$, and sets $\mathsf{server}[i] := \perp$. The pair (i, server) is added to $\mathcal{L}$.

SendToServer. An adversary tries to authenticate on behalf of a user i of her choice. According to the functionality, the challenger accepts up to τ attempts per registered user.

The verification procedure enables authentication of *registered* users only. As a consequence, the challenger responds only if i has already been enrolled.

ServerImpersonation: $\mathsf{SendToUser}_{server}$. The adversary might impersonate the server in front of a registered user i. There is no restriction on the number of attempts for this oracle.

UF: UnForgeability. In the unforgeability games, the adversary tries to authenticate to the challenger on behalf of an existing user. The adversary may attempt an authentication without a token (she might know the password of the target user, and data from other users) or without knowing a password (she might know the token of the target user, and data from other users). The first property prevents an adversary to authenticate itself without being registered. The second property prevents an adversary to authenticate itself if it stole the token. We stress that the adversary knows whether an authentication attempt is successful or not (Fig. 1).

Property UF-token. Given a scheme Π, a probabilistic polynomial adversary $\mathtt{A}$ and security parameters $\lambda, \beta, \tau \in \mathbb{N}$, the probability of success in the $\mathsf{Experiment}^{\mathsf{UF-token}}_{\Pi,\mathtt{A}}$ game is negligible as a function of λ:

$$\Pr\left[\mathsf{Experiment}^{\mathsf{UF-token}}_{\Pi,\mathtt{A}}(\lambda,\beta,\tau) \Rightarrow 1\right] < \mathsf{negl}(\lambda).$$

Property UF-pw. Given a scheme Π, a probabilistic polynomial adversary $\mathtt{A}$ and security parameters $\lambda, \beta, \tau \in \mathbb{N}$, the probability of success in the $\mathsf{Experiment}^{\mathsf{UF-pw}}_{\Pi,\mathtt{A}}$ game is negligible as a function of λ, up to on-line guesses:

$$\Pr\left[\mathsf{Experiment}^{\mathsf{UF-pw}}_{\Pi,\mathtt{A}}(\lambda,\beta,\tau) \Rightarrow 1\right] < \tau/2^{\beta} + \mathsf{negl}(\lambda).$$

Remarks. In the UF-pw game, the possibility to query $\mathsf{server}[i^*]$ is disallowed, and it is inherent to the notion: from $\mathsf{server}[i^*]$ and $\mathsf{client}[i^*]$, an adversary could brute-force recover $\mathsf{pin}[i^*]$. Moreover, we assume that the registration protocol is carried out over a secure channel; so the adversary has no access to a $\mathtt{SendToUser}_{issuer}$ oracle, neither to transcripts of registration sessions.

PL: Password Leakage. We do not want the issuer or the verifier to be able to recover the password. This point is not addressed by the UF games above. We formalize a game where the adversary knows the master key and the verifiers, but is not allowed to know the password nor the token for a target user i (it might know these data for other users). If it knew the token, it could brute-force recover the password (Fig. 2).

$\mathsf{Experiment}^{\mathsf{UF\text{-}\{token|pw\}}}_{\Pi,\mathtt{A}}(\lambda,\beta,\tau)$:
– $\mathcal{L} := \emptyset$; $table[i] := \perp$, $\forall table \in \{\mathsf{pw}, \mathsf{client}, \mathsf{server}\}$
– $(\mathsf{param}, \mathsf{mk}) \leftarrow \mathtt{Setup}(1^\lambda)$
– $(i^*, state) \leftarrow \mathtt{A}^{\mathcal{O}}(\mathsf{param})$ where $\mathcal{O} := \{\mathtt{AddUser}, \mathtt{UserData}, \mathtt{SendToIssuer}, \mathtt{SendToServer}, \mathtt{SendToUser}_{server}\}$
– if ($\mathsf{server}[i^*] = \perp$): return 0
– C engages an authentication with A (acting as user i^*):
  $\mathtt{A}(state) \leftrightarrow \mathtt{Verify}(\mathsf{param}, i^*, \mathsf{server}[i^*]) \rightarrow dec$
– return 1 if ($dec = \mathsf{accept}$) and
  · *in the weak* UF-token *case:* $((i^*, \mathsf{server}) \notin \mathcal{L})$ and $((i^*, \mathsf{client}) \notin \mathcal{L})$
  · *in the strong* UF-token *case:* $((i^*, \mathsf{client}) \notin \mathcal{L})$
  · *in the* UF-pw *case:* $((i^*, \mathsf{server}) \notin \mathcal{L})$ and $((i^*, \mathsf{pw}) \notin \mathcal{L})$
  otherwise return 0

Fig. 1. The unforgeability experiment

$\mathsf{Experiment}^{\mathsf{PL}}_{\Pi,\mathtt{A}}(\lambda,\beta,\tau)$:
– $\mathcal{L} \leftarrow \emptyset$; $table[i] := \perp$, $\forall table \in \{\mathsf{pw}, \mathsf{client}, \mathsf{server}\}$
– $(\mathsf{param}, \mathsf{mk}) \leftarrow \mathtt{Setup}(1^\lambda)$
– $\mathcal{O} := \{\mathsf{UserData}, \mathsf{SendToUser}_{issuer}, \mathsf{SendToUser}_{server}\}$
– $(i^*, pw^*) \leftarrow \mathtt{A}^{\mathcal{O}}(\mathsf{param}, \mathsf{mk}, \mathsf{sk})$
– return 1 if $(\mathsf{pw}[i^*] = pw^*)$ and $(\{(i^*, \mathsf{pw}), (i^*, \mathsf{client})\} \cap \mathcal{L} = \emptyset)$,
  otherwise return 0

Fig. 2. The password leakage experiment

Property Password-Leakage. Given a scheme Π, a probabilistic polynomial adversary A and security parameters $\lambda, \beta, \tau \in \mathbb{N}$, the probability of success in the $\mathsf{Experiment}^{\mathsf{PL}}_{\Pi,\mathtt{A}}$ game is negligible as a function of λ, up to a simple guess of the password:

$$\Pr\left[\mathsf{Experiment}^{\mathsf{PL}}_{\Pi,\mathtt{A}}(\lambda,\beta,\tau) \Rightarrow 1\right] < 1/2^\beta + \mathsf{negl}(\lambda).$$

The non-interactive authentication setting. By non-interactive setting, we mean that the authentication procedure is the non-interactive signing of a random message. The message to be signed is fresh for each authentication. It might be a random challenge chosen by the verifier, or a hash value of a context-dependent message (time, verifier identity, *etc.*), determined by the protocol specification. In this setting, the $\mathtt{SendToUser}_{server}$ oracle becomes a signature oracle and the SendToServer oracle becomes a verification oracle. According to the standard existential unforgeability notion for signatures [17], we allow the adversary to choose the message to be signed in the security experiment.

$\mathsf{SendToUser}_{server}$. The adversary supplies (i, m), where i is a user, and m a message; and receives a signature σ on m on behalf of i.

SendToServer. The adversary supplies (i, m, σ); and receives the decision of the verifier about the validity of the signature.

At the end of the game, there is no interaction with the challenger. The adversary eventually outputs a tuple (i^*, m^*, σ^*) and wins if the non-triviality conditions hold ($\mathsf{server}[i^*] \neq \perp$, *etc.*), if m^* has not been queried to the signature oracle, and if $\mathtt{Verify}(\mathsf{param}, i^*, \mathsf{server}[i^*], m^*, \sigma^*) = \mathsf{accept}$.

Local updates. We now discuss the possibility of password updates on the client's side into our definition. May the user change its password and update its token accordingly without interaction with the server? In fact, this property is not contradictory with the notion. However, if such a procedure exists, the client cannot check if the token is correctly updated. By contradiction, he could carry out a verification protocol on its own, and therefore the adversary could try, given a token, to guess the password and check its guess. In practice, one can imagine that the old token is backed up and an authentication protocol is done with the new token. If successful, then the old token is removed. If not, the new token is removed and the old one is kept.

4 A Solution Based on Commitments

In this section, we give a simple solution that uses the standard notions of commitments and proofs of knowledge of committed values.

Commitments and ZK proofs. A *commitment* scheme Com is composed of a message space $\mathcal{M}_{\mathsf{C}}$, a random value space $\mathcal{R}_{\mathsf{C}}$, a commitment space $\mathcal{C}_{\mathsf{C}}$, and a set of functionalities $\{\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open}\}$ as follows. On input a security parameter $\lambda \in \mathbb{N}$, the setup procedure Setup outputs a commitment key ck. The (deterministic) commitment procedure Commit takes as input a commitment key ck, a message $m \in \mathcal{M}_{\mathsf{C}}$ and a random value $r \in \mathcal{R}_{\mathsf{C}}$, and outputs a committed value $c \in \mathcal{C}_{\mathsf{C}}$. Given a commitment key ck and a commitment c, the Open procedure simply reveals a pair (m, r): everyone can check whether $c = \mathtt{Commit}(\mathsf{ck}, m; r)$. A commitment scheme is *binding* if it is impossible to reveal a distinct pair $(m', r') \neq (m, r)$ such that $c = \mathtt{Commit}(m'; r')$. It is *hiding* if c does not leak any information about m. Both security notions can be defined in a computational, statistical or perfect (information-theoretic) sense. In our constructions, we use standard ZK proofs of knowledge, known as Σ-protocols, and their standard transformation into signatures of knowledge [9] through the Fiat-Shamir heuristic [12]. A Σ-protocol is proof of knowledge that consists of three messages: a commitment message R, a random uniform challenge $c \leftarrow \{0,1\}^{\lambda_c}$ for some security parameter λ_c, and a response s. The Fiat-Shamir heuristic makes this ZKPK non-interactive by generating the random challenge with a hash function. The resulting signature – denoted SoK – is as secure as the underlying Σ-protocol in the programmable random oracle model.

user(ck, pw)		issuer($info$)
[user]: hash $h := \mathtt{H}_h(pw)$; pick $t \leftarrow \mathcal{R}_{\mathsf{C}}$		
[user]: set $C :=$ Com.$\mathtt{Commit}(\mathsf{ck}, h; t)$; store $T = t$		
	send C $\longrightarrow$	
[issuer]:		store $V = C$

The pw-com registration protocol

user($\mathsf{ck}, T = t, \tilde{pw}, \mathsf{m}$)		server($\mathsf{ck}, V = C, \mathsf{m}$)
[user]: $\tilde{h} := \mathtt{H}_h(\tilde{pw})$; compute $\sigma \leftarrow$ SoK.$\mathtt{Sign}(\mathsf{ck}, (\tilde{h}, t), \mathsf{m})$		
	send σ $\longrightarrow$	
[server]:		check SoK.$\mathtt{Verify}(\mathsf{ck}, C, \sigma, \mathsf{m}) = \mathsf{accept}$

The two factor pw-com authentication protocol

Fig. 3. The two factor pw-com solution

Description of the solution. Let Com be a computationally binding, statistically hiding commitment scheme, and $\mathcal{P}$ be a set of passwords. Let $\mathtt{H}_h : \mathcal{P} \rightarrow \mathcal{M}_{\mathsf{C}}$ be an injective encoding function from the passwords to the message space of the Com scheme. The $\mathtt{Setup}$ procedure picks a commitment key $\mathsf{ck} \leftarrow$ Com.$\mathtt{Setup}(1^{\lambda})$ and returns ck as global parameter. The registration and authentication protocols are described Fig. 3.

Security analysis. The pw-com solution is token-unforgeable in the programmable random oracle for $\mathtt{H}_c$ if the SoK scheme is sound, zero-knowledge, and if the Com scheme is computationally binding and statistically hiding. It is password-unforgeable in the programmable random oracle for $\mathtt{H}_c$ if the SoK scheme is sound, zero-knowledge, and if the Com scheme is computationally binding and statistically hiding. Finally, no information about the password is available from the issuer's and server's point of view, in the random oracle for $\mathtt{H}_c$ under the zero-knowledge property of the SoK scheme and the statistical hiding property of the Com scheme. For the sake of reading, we use the following notations in the proofs: $\mathtt{S}$ is the signature oracle (instead of $\mathsf{SendToUser}_{server}$), $\mathtt{V}$ the verification oracle (instead of SendToServer). Moreover we analyse the security for a single user, the extension to several users being straightforward.

Theorem 1. *Let* $\mathtt{A}$ *be an adversary against the* UF-token *property of the* pw-com *scheme with a single user, running in time at most* t*, making at most* q_s *signature queries, and at most* q_v *verification queries. Then:*

$$\mathbf{Adv}^{\mathsf{UF\text{-}token}}_{\mathsf{pw\text{-}com},\mathsf{ck}}(\mathtt{A}) \leq \frac{q_s \cdot (q_s + q_h)}{|\mathcal{C}_{\mathsf{C}}|} + \frac{q_h}{2^{\lambda_c}} + \sqrt{q_h \cdot \mathbf{Adv}^{\mathsf{bind}}_{\mathsf{ck}}(2 \cdot t)}.$$

Proof. Let $T = t$ be the token of the user, pw its password, and $V = C$ its verifier. Game G_0 is the token-unforgeability security experiment. The adversary

gets pw and V but does not have access to T. In game G_1, the $\mathtt{S}$ oracle simulates the signature with the simulator of signature of knowledge, in the random oracle, without knowing the password, nor the token t, but knowing the verifier C. Games G_0 and G_1 are identical up to the simulation failure. Game $\mathsf{G}_2(\mathsf{ck}, pw, C)$ takes as input a commitment key ck, a random password $pw \leftarrow \mathcal{P}$, and a random commitment $C \leftarrow \mathcal{C}_{\mathsf{C}}$. During the registration step, G_2 sets $T = \perp$ and $V = C$. The token is not available to the simulation, but is not needed to simulate the signatures. From the statistical property of the commitment scheme, we know that for all pw and commitment C, there exists $t \in \mathcal{R}_{\mathsf{C}}$ such that $C = \mathtt{Commit}(\mathtt{H}_h(pw); t)$ with overwhelming probability. It remains to show that the probability of success of $\mathtt{A}$ in game G_2 is negligible. Let $\mathtt{B}$ be the following reduction. $\mathtt{B}$ receives a challenge ck for the computational binding property of the commitment scheme, picks $pw \leftarrow_D \mathcal{P}$, $C \leftarrow \mathcal{C}_{\mathsf{C}}$, and runs $\mathtt{A}$ simulating game $\mathsf{G}_2(\mathsf{ck}, pw, C)$. $\mathtt{A}$ eventually outputs a valid signature σ for some message m. If $\mathtt{A}$ is successful, then the soundness of the SoK scheme is broken. □

Theorem 2. *Let* $\mathtt{A}$ *be an adversary against the* UF-pw *property of the* pw-com *scheme with a single user, running in time at most* t*, making at most* q_h *random oracle queries,* q_s *signature queries, and* $q_v < \tau$ *verification queries. Then:*

$$\mathbf{Adv}^{\mathsf{UF\text{-}pw}}_{\mathsf{pw\text{-}com},\mathsf{ck}}(\mathtt{A}) \leq \frac{\tau}{2^{\beta}} + \frac{(q_s \cdot (q_s + q_h) - q_v)}{|\mathcal{C}_{\mathsf{C}}|} + \frac{q_h}{2^{\lambda_c}} + \sqrt{q_h \cdot \mathbf{Adv}^{\mathsf{bind}}_{\mathsf{ck}}(2 \cdot t)}.$$

The proof is very similar to the proof of Theorem 1. However, we must take care of the password distribution, which is not assumed to induce a uniform distribution over the message space of the commitment scheme.

Proof. Let $T = t$ be the token of the user, pw its password, and $V = C$ its verifier. Game G_0 is the password-unforgeability security experiment. The adversary gets t but does not have access to pw nor to C. The only difference between G_0 and G_1 lies in the responses from the verification oracle for 'fresh' queries, *i.e.*, message/signature pairs that do not correspond to a previous query/response pair from the signature oracle. When a valid fresh verification query is supplied, G_1 stops and returns 1. G_0 and G_1 returns 1 with the same probability, but by doing this we ensure that all fresh verification queries returns reject before the forgery. In game G_2, the signatures are simulated in the random oracle for $\mathtt{H}_c$ with the simulator of the signature of knowledge. Game $\mathsf{G}_3(\mathsf{ck}, t, C)$ is a transition step. It takes as input a commitment key ck, a random value $t \leftarrow \mathcal{R}_{\mathsf{C}}$, a commitment $C := \mathtt{Commit}(\mathtt{H}_h(pw); t)$ for some password $pw \leftarrow_D \mathcal{P}$, and sets $T = t$ and $V = C$. The password pw is not given, but is not needed for the simulation. Game $\mathsf{G}_4(\mathsf{ck}, t, C)$ is as game G_3, except that $C \leftarrow \mathcal{C}_{\mathsf{C}}$. If $2^{\beta} \ll |\mathcal{C}_{\mathsf{C}}|$, it is unlikely that (event E): there exists pw such that $C = \mathtt{Commit}(\mathtt{H}_h(pw); t)$. The probability of E is at most $2^{\beta}/|\mathcal{C}_{\mathsf{C}}|$, given that the encoding $\mathtt{H}_h$ is injective. However, the commitment key, the token and the simulated signatures are identical in both games. The $\mathtt{S}$ oracle can still simulate signatures since the SoK simulator can simulate signatures even for false statements. $\mathtt{A}$ gets information about C only through the verification oracle. If E does not happen, a potential bias of $1/2^{\beta}$ is introduced per verification query from $\mathtt{A}$'s point of

view. This is because a negative response from the verification oracle reveals at most one bit of information about the password (recall that all responses from $\mathtt{V}$ are negative). So we bound the difference between G_3 and G_4 as follows: $\Pr\left[\mathtt{A}^{\mathsf{G}_3} \Rightarrow 1\right] - \Pr\left[\mathtt{A}^{\mathsf{G}_4} \Rightarrow 1\right] \leq q_v/2^{\beta} \cdot \left(1 - 2^{\beta}/|\mathcal{C}_{\mathsf{C}}|\right)$. It remains to show that the probability of success of $\mathtt{A}$ in game G_4 is negligible. This is shown by reduction to the soundness of the SoK signature, as in Theorem 1. □

Theorem 3. *Let* $\mathtt{A}$ *be an adversary against the* PL *property of the* pw-com *scheme making at most* q_h *queries to the random oracle and* q_s *queries to the signature oracle. Then:*

$$\mathbf{Adv}^{\mathsf{PL}}_{\mathsf{pw\text{-}com},N}(\mathtt{A}) \leq \frac{1}{2^{\beta}} + \frac{q_s \cdot (q_s + q_h)}{|\mathcal{C}_{\mathsf{C}}|}.$$

Proof. Game G_0 is the PL security game. $\mathtt{A}$ eventually outputs (i^*, pw^*) and wins if it correctly guesses the password. In game G_1, signatures are simulated as in Theorem 1. In game G_2, in the registration protocol, a random commitment $C \leftarrow \mathcal{C}_{\mathsf{C}}$ is drawn, instead of computing C according to the user's password. Thanks to the mask value t, the simulated C is statistically close from a real one, under the hiding property of the Com scheme. Finally, in game G_2, no password is used. The probability to win is then bound by the probability to guess a password. □

A DL instantiation. Let $(p, \mathbb{G}, G)$ be a group of prime order. During the registration step, the client takes a password hash $h := \mathtt{H}_h(pw)$, picks $t \leftarrow \mathbb{Z}_p$, and sets $C := [h] \cdot G + [t] \cdot H$. It stores t as token and sends C to the server which stores it as verifier. During the authentication step, the client takes a password hash $\tilde{h} := \mathtt{H}_h(\tilde{pw})$, picks $r_h, r_t \leftarrow \mathbb{Z}_p$, sets $R := [r_h] \cdot G + [r_t] \cdot H;$, $c := \mathtt{H}_c(R, \mathsf{m})$, $s_h := r_h + c \cdot \tilde{h} \mod p$, and $s_t := r_t + c \cdot t \mod p$. It sends (c, s_h, s_t) to the server, which computes $\tilde{R} := [s_h] \cdot G + [s_t] \cdot H - [c] \cdot C$, $\tilde{c} := \mathtt{H}_c(\tilde{R}, \mathsf{m});$, and checks $c = \tilde{c}$. For 128 bits of security, the user's response (c, s_h, s_t) takes 640 bits.

5 A Solution Based on Homomorphic Encryption

In this section, we give a construction based on a homomorphic encryption scheme over group of prime order, which includes local updates of the password.

Hash functions. A *hash function* H is given by two procedures $\{\mathtt{KeyGen}, \mathtt{Eval}\}$ as follows. On input a security parameter λ, the $\mathtt{KeyGen}$ procedure outputs some public parameters. In case of *keyed* hash function, the procedure also outputs a random uniform evaluation key $\mathsf{ek} \leftarrow \{0,1\}^{\lambda}$. On input a message $m \in \{0,1\}^{\ell(\lambda)}$ for some polynomial ℓ, the evaluation function $\mathtt{Eval}$ outputs a hash value in some space $\mathcal{H}$. We need *collision-resistant* (unkeyed) hash functions and *pseudo-random* keyed hash functions. The collision-resistance requires that it should impossible to exhibit two distinct messages that share the same hash value. A keyed hash function is pseudo-random if a polynomial adversary cannot distinguish whether it interacts with a pseudo-random function or a truly random one.

Homomorphic Encryption. Our constructions make use of an encryption scheme that supports some homomorphic operation and its efficient iteration. By pure convention we use here the additive symbol. A *homomorphic public key encryption* scheme hom-pke is composed of a message space $\mathcal{M}_{\mathsf{E}}$ supporting some operation $+$, a ciphertext space $\mathcal{C}_{\mathsf{E}}$, and a set of functionalities $\{\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Add}, \mathtt{SMul}\}$ as follows.

On input a security parameter $\lambda \in \mathbb{N}$ and the particular message space $\mathcal{M}_{\mathsf{E}}$, the key generation procedure $\mathtt{KeyGen}$ outputs a pair of public/private keys $(\mathsf{pk}, \mathsf{sk})$. The (probabilistic) encryption procedure $\mathtt{Enc}$ takes as input a public key pk and a message $m \in \mathcal{M}_{\mathsf{E}}$, and outputs a ciphertext c. The (deterministic) decryption procedure takes as input a secret key sk and a ciphertext c, and outputs a message m. The homomorphic procedure takes as input a public key pk and two ciphertexts c_1, c_2, and outputs a ciphertext c'. The homomorphic operation $\mathtt{Add}$ is extended to an efficient scalar multiplication $\mathtt{SMul}$ which takes as input a public key pk, a ciphertext c and a scalar $n \in \mathbb{N}$, $n > 1$, and outputs a ciphertext c' such that for all $m \in \mathcal{M}_{\mathsf{E}}$ we have: $n \times m \leftarrow \mathtt{Dec}(\mathsf{sk}, \mathtt{SMul}(\mathsf{pk}, \mathtt{Enc}(\mathsf{pk}, m), n))$. Sometimes we note $c_1 \oplus c_2$ as a shortcut for $\mathtt{Add}(\mathsf{pk}, c_1, c_2)$, and $[n] \cdot c$ for $\mathtt{SMul}(\mathsf{pk}, c, n)$, the public key being clear from the context.

The one-wayness OW property states that it is impossible given a ciphertext to recover the underlying plaintext. The semantic security, or indistinguishability against chosen plaintext attacks $\mathsf{IND\text{-}CPA}$, state that no information about the plaintext leak from the ciphertext. We also assume that ciphertexts produced by the homomorphic procedures are indistinguishable from those directly produced by the encryption procedure.

Description of the solution. The Setup procedure picks a prime order group $(p, \mathbb{G}, G)$ in additive notation, with null element $\mathbf{0}$, and a master key $\mathsf{mk} \leftarrow \{0,1\}^{\lambda}$. Let hom-pke $:= \{\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Add}, \mathtt{SMul}\}$ be an additively homomorphic encryption scheme over $\mathcal{M}_{\mathsf{E}} := (p, \mathbb{G}, G)$. Let $\mathtt{H}_u : \{0,1\}^{\lambda} \times \{0,1\}^* \to \mathbb{Z}_p$ be a pseudo-random hash function, $\mathtt{H}_c : \{0,1\}^{\lambda} \times {\mathcal{C}_{\mathsf{E}}}^2 \times \{0,1\}^{\ell} \to \{0,1\}^{\lambda}$ a hash function (for a message length ℓ), $\mathtt{H}_v : \mathcal{C}_{\mathsf{E}} \to \{0,1\}^{\lambda}$ another hash function, and $\mathtt{H}_h : \mathcal{P} \to \mathbb{Z}_p^*$ an injective encoding function of the passwords.

The Setup procedure returns the global parameters $\mathsf{param} := (p, \mathbb{G}, G)$. The registration and authentication protocols are described Fig. 4. In the registration step, the user has got a password pw, the server knows the master key mk and some user information $info$, and both know the parameters $(p, \mathbb{G}, G)$. In the authentication step, the user supplies a fresh $\tilde{pw}$ value and owns a token $(\mathsf{pk}_1, \mathsf{pk}_2, B, C)$, the server knows the verifier $(H, \mathsf{sk}_1, \mathsf{sk}_2)$, and both formerly agreed on a message m to be signed.

Local updates. Local updates on the client's side are possible in the $\mathsf{pw\text{-}hom}$ construction: (i) ask the user for pw_{old}, pw_{new}; set $h_{\mathsf{old}} := \mathtt{H}_h(pw_{\mathsf{old}})$, $h_{\mathsf{new}} := \mathtt{H}_h(pw_{\mathsf{new}})$; (ii) given $T := (\mathsf{pk}_1, \mathsf{pk}_2, B, C)$, update $C \leftarrow \mathtt{SMul}(\mathsf{pk}_2, C, h_{\mathsf{new}} \cdot (h_{\mathsf{old}})^{-1} \mod p)$.

user(pw)	parameters $\lambda, (p, \mathbb{G}, G)$	issuer($\mathsf{mk}, info$)
[user]: compute $h := \mathtt{H}_h(pw)$; pick $a \leftarrow \mathbb{Z}_p^*$; set $Y := [a \cdot h] \cdot G$		
	send Y $\longrightarrow$	
[issuer]:		compute $k := \mathtt{H}_u(\mathsf{mk}, info)$
[issuer]:		pick $(\mathsf{pk}_1, \mathsf{sk}_1)$, $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathtt{KeyGen}(1^\lambda, (p, \mathbb{G}, G))$
[issuer]:		set $\tilde{C} \leftarrow \mathtt{Enc}(\mathsf{pk}_2, [k] \cdot Y)$ $B \leftarrow \mathtt{Enc}(\mathsf{pk}_1, [k] \cdot G)$; $H := \mathtt{H}_v([k] \cdot G)$
[issuer]:		store $(H, \mathsf{sk}_1, \mathsf{sk}_2)$
	$\longleftarrow$ send $\mathsf{pk}_1, \mathsf{pk}_2, B, \tilde{C}$	
[user]: set $C \leftarrow \mathtt{SMul}(\mathsf{pk}_2, \tilde{C}, a^{-1} \bmod p))$		
[user]: store $(\mathsf{pk}_1, \mathsf{pk}_2, B, C)$		

The pw-hom registration protocol

user($T = (\mathsf{pk}_1, \mathsf{pk}_2, B, C), p\tilde{w}, \mathsf{m}$)	server($V = (H, \mathsf{sk}_1, \mathsf{sk}_2), \mathsf{m}$)
[user]: set $\tilde{h} := \mathtt{H}_h(p\tilde{w})$; pick $a, r \leftarrow \mathbb{Z}_p^*$	
[user]: $E \leftarrow \mathtt{SMul}(\mathsf{pk}_1, B, r)$; $F \leftarrow \mathtt{SMul}(\mathsf{pk}_2, C, a)$	
[user]: $c := \mathtt{H}_c(E, F, \mathsf{m})$; $s := r + c \cdot a \cdot \tilde{h} \bmod p$	
	send E, F, s $\longrightarrow$
[server]:	$\tilde{c} := \mathtt{H}_c(E, F, \mathsf{m})$; $R := \mathtt{Dec}(\mathsf{sk}_1, E)$; $A := \mathtt{Dec}(\mathsf{sk}_2, F)$
[server]:	check $A \neq \mathbf{0}$; $K := [s^{-1} \bmod p] \cdot (R + [\tilde{c}] \cdot A)$; check $\mathtt{H}_v(K) = H$

The pw-hom two factor authentication protocol

Fig. 4. The pw-hom two factor solution

Security analysis. The pw-hom scheme is UF-pw secure under the semantic security of hom-pke, the pseudo-randomness of $\mathtt{H}_u$ and the collision-resistance of $\mathtt{H}_v$ in the random oracle for $\mathtt{H}_c$. It is (weakly) UF-token secure under the one-wayness of hom-pke, the pseudo-randomness of $\mathtt{H}_u$ and the collision-resistance of $\mathtt{H}_v$ in the random oracle for $\mathtt{H}_c$. It is PL resistant in the random oracle for $\mathtt{H}_c$.

Theorem 4. *Let* $\mathtt{A}$ *be an adversary against the* UF-pw *property of the* pw-hom *scheme with a single user, running in time at most* t*, making at most* q_h *random oracle queries,* q_s *signature queries and* $q_v < \tau$ *verification queries. Then:*

$$\mathbf{Adv}^{\mathsf{UF\text{-}pw}}_{\mathsf{pw\text{-}hom},\mathbb{G}}(\mathtt{A}) \leq \frac{\tau^2}{2^\beta} + \frac{(q_s(q_s+q_h)-\tau^2)}{p} + \mathbf{Adv}^{\mathsf{PRF}}_{\mathtt{H}_u}(t) + \tau \cdot \Big(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\text{HOM-PKE},\mathbb{G}}(t) + \frac{q_h}{2^{\lambda_c}} + \sqrt{q_h \cdot \big(\mathbf{Adv}^{\mathsf{OW}}_{\text{HOM-PKE},\mathbb{G}}(2\cdot t) + \mathbf{Adv}^{\mathsf{CR}}_{\mathtt{H}_v}(2\cdot t)\big)}\Big).$$

Note on the bound. We are not able to prove that the bound is negligible beyond $\tau/2^\beta$, but only beyond $\tau^2/2^\beta$. This means that one should set $\tau^2 < \beta$ rather than $\tau < \beta$ in practice. If the encryption scheme is one-way under a computational problem, this bound could be taken back to $\tau/2^\beta$ at the cost of an interactive

Gap assumption [32], according to which the adversary has access to an oracle for the corresponding decision problem.

Proof. Let $T = (\mathsf{pk}_1, \mathsf{pk}_2, B, C)$ be the token of the user, pw its password, and $V = (H, \mathsf{sk}_1, \mathsf{sk}_2)$ its verifier. Game G_0 is the UF-pw security experiment. The adversary gets T but does not have access to pw nor to V. In game G_1, the $\mathtt{S}$ oracle simulates the signatures, in the random oracle, without knowing the password, nor the ciphertext C, but knowing the ciphertext B and the public encryption keys. During the n-th query, on input a message m_n, game G_1 generates the signature as follows: pick $s_n \leftarrow \mathbb{Z}_p$; $c_n \leftarrow \{0,1\}^\lambda$; $L_n \leftarrow \mathbb{G}$, set $F_n \leftarrow \mathtt{Enc}(\mathsf{pk}_2, L_n)$; $E_n \leftarrow \mathtt{Enc}(\mathsf{pk}_1, [-c_n] \cdot L_n) \oplus \mathtt{SMul}(\mathsf{pk}_1, B, s_n)$, if $\mathtt{H}_c(E_n, F_n, m_n) \neq \perp$, abort, otherwise program $\mathtt{H}_c(E_n, F_n, m_n) := c_n$, return $\sigma := (E_n, F_n, s_n)$. Game G_2 is game G_1, except that the master key mk is dropped and G_2 has an oracle access to $\mathtt{H}_u(\mathsf{mk}, \cdot)$. In game G_3, the oracle access to $\mathtt{H}_u$ is replaced by an oracle which draws random values in $\mathbb{Z}_p^*$. The success probability of a distinguisher between G_2 and G_3 is bound by the advantage to break the PRF property of $\mathtt{H}_u$ within the same time. In game G_3, $\mathtt{A}$ supplies $q_v + 1$ message/signature pairs, either to the verification oracle, or at the end of the game. Game G_4 guesses the first query $\mathsf{j} \in [1, q_v + 1]$ where $\mathtt{A}$ gives a valid 'fresh' message/signature pair. For all $j < \mathsf{j}$, the verification oracle returns 0. At the j-th query (or at the end of the game if $\mathsf{j} = q_v + 1$), game G_4 returns 1 and stops. The oracle $\mathtt{V}$ no longer needs the secret keys, at the price of a security loss linear in the number of verification queries. Game $\mathsf{G}_5(\mathsf{pk}_1, B, \mathsf{pk}_2, C)$ is a transition step. It takes as input two public keys pk_1, pk_2, a ciphertext B of a group element $[k] \cdot G$ under pk_1, and a ciphertext C of a group element $[\mathtt{H}_h(pw) \cdot k] \cdot G$ under pk_2 for some $pw \leftarrow_P \mathcal{P}$ and $k \leftarrow \mathbb{Z}_p$. The password pw is not given, but the simulation of signatures is not affected. Likewise, the secret keys are not given, but they are not needed in the simulation of the verification oracle. In game G_6, a random exponent $h \leftarrow \mathbb{Z}_p^*$ is taken instead of computing h from the password. With probability at most $2^\beta/p$, there exists $pw \in \mathcal{P}$ such that $h = \mathtt{H}_h(pw)$. Otherwise, the distribution of C is not as in the real protocol. Under the semantic security of HOM-PKE, C does not leak information about the password. However the verification queries might leak information about the password. We bound the difference between G_5 and G_6 as follows: $\Pr\left[\mathtt{A}^{\mathsf{G}_5} \Rightarrow 1\right] - \Pr\left[\mathtt{A}^{\mathsf{G}_6} \Rightarrow 1\right] \leq \left(q_v/2^\beta + \mathbf{Adv}^{\text{IND-CPA}}_{\text{HOM-PKE},\mathbb{G}}(t)\right) \cdot \left(1 - 2^\beta/p\right)$. It remains to bound the probability of success of $\mathtt{A}$ is game G_6. We adapt the standard forking lemma techniques [1,34] to our case, which do not involve difficulties. For the sake of place, we postpone it to the full version of our paper. □

Theorem 5. *Let* $\mathtt{A}$ *be an adversary against the* UF-token *property of the* pw-hom *scheme with a single user, running in time at most* t*, making at most* q_h *random oracle queries,* q_s *signature queries and* q_v *verification queries. Then:*

$$\mathbf{Adv}^{\mathsf{UF\text{-}token}}_{\mathsf{pw\text{-}hom},\mathbb{G}}(\mathtt{A}) \leq \frac{(q_s \cdot (q_s + q_h) + q_v \cdot q_h)}{p} + \mathbf{Adv}^{\mathsf{PRF}}_{\mathtt{H}_u}(t) + (q_v + 1) \cdot \sqrt{q_h \cdot \left(\mathbf{Adv}^{\mathsf{OW}}_{\text{HOM-PKE},\mathbb{G}}(2 \cdot t) + \mathbf{Adv}^{\mathsf{CR}}_{\mathtt{H}_v}(2 \cdot t)\right)}.$$

Proof. Let $T = (\mathsf{pk}_1, \mathsf{pk}_2, B, C)$ be the token of the user, pw its password, and $V = (H, \mathsf{sk}_1, \mathsf{sk}_2)$ its verifier. Game G_0 is the token-unforgeability security experiment. The adversary gets pw but does not have access to T nor V. Games G_1 to G_4 are as in Theorem 4. Game $\mathsf{G}_5(\mathsf{pk}, \mathsf{B})$ takes as input a public key pk, and the ciphertext B of $[k] \cdot G$ for some uniform $k \leftarrow \mathbb{Z}_p$. G_4 sets $\mathsf{pk}_1 := \mathsf{pk}$ and $B := \mathsf{B}$. The distributions of B in G_4 and G_5 are identical. The ciphertext C is not computed and is never used. It remains to show that the probability of success of $\mathtt{A}$ in game G_5 is negligible. This is done as in Theorem 4, under the one-wayness of HOM-PKE and the collision-resistance of $\mathtt{H}_v$. □

Theorem 6. *Given an adversary* $\mathtt{A}$ *making at most* q_h *requests to the random oracle and* q_s *requests to the signature oracle, we have:*

$$\mathbf{Adv}^{\mathsf{PL}}_{\mathsf{pw\text{-}hom},\mathbb{G}}(\mathtt{A}) \leq \frac{1}{2^\beta} + \frac{q_s \cdot (q_s + q_h)}{p}.$$

Proof. The proof is fairly straightforward, as in the pw-com scheme. □

Security in presence of local updates. We add an oracle to the security game, to catch the possibility of local updates. The adversary supplies (i, pw'), for i such that $\mathsf{pw}[i]$ and $\mathsf{client}[i]$ are well-defined. If $pw' = \perp$, $\mathtt{C}$ picks a new password $pw' \leftarrow_P \mathcal{P}$ at random, according to the password distribution P. In both cases, $\mathtt{C}$ updates the tables. If UserData has already been called on $\mathsf{pw}[i]$ (or $\mathsf{client}[i]$), $\mathtt{C}$ sends the corresponding updated value to $\mathtt{A}$. This oracle has an effect during the game G_6 in the UF-pw proof. The ciphertext C is replaced by another ciphertext $C = \mathtt{SMul}(\mathsf{pk}_2, C, t)$ for some uniform $t \leftarrow \mathbb{Z}_p^*$. The value t is not distributed as in the real protocol, but the simulation remain correct under the semantic security of the encryption scheme.

A concrete instantiation with ElGamal. The black-box construction above might be instantiated with the ElGamal encryption scheme [14]. Let ElG be the following scheme. The key generation procedure takes as input a prime order group $(p, \mathbb{G}, G)$, picks a secret key $\mathsf{sk} \leftarrow \mathbb{Z}_p^*$, computes the public key $\mathsf{pk} := [\mathsf{sk}] \cdot G$, and returns the key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathbb{G} \times \mathbb{Z}_p^*$. Then encryption procedure takes as input an element $M \in \mathbb{G}$ and a public key pk, picks $r \leftarrow \mathbb{Z}_p$ and outputs a ciphertext $C([r] \cdot G, M + [r] \cdot \mathsf{pk}) \in \mathbb{G}^2$. The decryption step takes as input a ciphertext (C_1, C_2) and a secret key sk, and outputs $M = C_2 - [\mathsf{sk}] \cdot C_1$. The one-wayness of the ElG scheme is equivalent to the CDH problem and its semantic security is equivalent to the DDH problem. When instantiated with the ElG scheme, a token is given by 6 group elements, a verifier by two scalars and a hash, and a signature by 4 elements plus a scalar. According to the state of the art (see for instance [11]), for 100 bits of security, the ElG scheme might be instantiated with an elliptic curve over a prime field $\mathbb{F}_q$ with a 200-bits prime q.

As a result, for 128 bits of security, a token needs ≈3k bits to be stored without compression (and half this value with compression), a verifier takes ≈640 bits, and a signature ≈2300 bits without compression (≈1300 with compression).

Acknowledgements. This work has been partially funded by the European FP7 EKSISTENZ (SEC-2013-607049) project. The opinions expressed in this document only represent the authors' view. They reflect neither the view of the European Commission nor the view of their employer. The authors would like to thanks Rodolphe Hugel, Olivier Cipière and Victor Servant for useful discussions, and the anonymous reviewers for their valuable comments and suggestions.

References

1. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: CCS 2006, pp. 390–399. ACM (2006)
2. 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)
3. Bellovin, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: SP 1992, pp. 72–84. IEEE (1992)
4. Bellovin, S.M., Merritt, M.: Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In: Computer and Communications Security (CCS 1993), pp. 244–250. ACM (1993)
5. Benhamouda, F., Pointcheval, D.: Verifier-based password-authenticated key exchange: new models and constructions. IACR ePrint Archive, 2013/833 (2013)
6. Blazy, O., Chevalier, C., Vergnaud, D.: Mitigating server breaches in password-based authentication: secure and efficient solutions. In: CT-RSA 2016 (2016). to appear
7. 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 (2000)
8. Camenisch, J., Lehmann, A., Neven, G., Samelin, K.: Virtual smart cards: how to sign with a password and a server. IACR ePrint Archive, 2015/1101 (2015)
9. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006)
10. Duo Security two-factor authentication. https://www.duosecurity.com/
11. ECRYPT II NoE. Yearly report on algorithms and keysizes. D.SPA.20 Rev. 1.0, ICT-2007-216676 ECRYPT II, 09/2012
12. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
13. Google Authenticator. http://code.google.com/p/google-authenticator/
14. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
15. Gennaro, R.: Faster and shorter password-authenticated key exchange. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 589–606. Springer, Heidelberg (2008)
16. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. ACM Trans. Inf. Syst. Secur. **9**(2), 181–234 (2006)

17. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
18. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: CCS 2010, pp. 516–525. ACM Press (2010)
19. Celestix HotPin. http://www.celestixworks.com/HOTPin.asp
20. IEEE P1363.2. Password-based public-key cryptography working group
21. Jablon, D.P.: Extended password key exchange protocols immune to dictionary attacks. In: WET-ICE 1997, pp. 248–255. IEEE Computer Society (1997)
22. Jarecki, S., Krawczyk, H., Shirvanian, M.: Saxena device-enhanced password protocols with optimal online-offline protection. IACR Archive, 2015/1099 (2015)
23. Jiang, S., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 267–279. Springer, Heidelberg (2004)
24. Katz, J., MacKenzie, P.D., Taban, G., Gligor, V.D.: Two-server password-only authenticated key exchange. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 1–16. Springer, Heidelberg (2005)
25. Katz, J., MacKenzie, P.D., Taban, G., Gligor, V.D.: Two-server password-only authenticated key exchange. J. Comput. Syst. Sci. **78**(2), 651–669 (2012)
26. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
27. Katz, J., Ostrovsky, R., Yung, M.: Forward secrecy in password-only key exchange protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 29–44. Springer, Heidelberg (2003)
28. Katz, J., Ostrovsky, R., Yung, M.: Efficient and secure authenticated key exchange using weak passwords. J. ACM **57**(1), 78–116 (2009)
29. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. J. Cryptol. **26**(4), 714–743 (2013)
30. Kiefer, F., Manulis, M.: Zero-knowledge password policy checks and verifier-based PAKE. In: Kutyłowski, M., Vaidya, J. (eds.) ICAIS 2014, Part II. LNCS, vol. 8713, pp. 295–312. Springer, Heidelberg (2014)
31. Lucks, S.: Open key exchange: how to defeat dictionary attacks without encrypting public keys. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361. Springer, Heidelberg (1998)
32. Okamoto, T., Pointcheval, D.: The Gap-problems: a new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992. Springer, Heidelberg (2001)
33. Microsoft PhoneFactor. https://www.phonefactor.com/
34. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000)
35. Schnorr, C.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991)
36. Scott, M.: Replacing username/password with software-only two-factor authentication. IACR IACR ePrint Archive, 2012/148 (2012)
37. Shirvanian, M., Jarecki, S., Saxena, N., Nathan, N.: Two-factor authentication resilient to server compromise using mix-bandwidth devices. In: Network and Distributed System Security - NDSS 2014. The Internet Society (2014)
38. Steiner, M., Tsudik, G., Waidner, M.: Refinement and extension of encrypted key exchange. Oper. Syst. Rev. **29**(3), 22–30 (1995)

39. Viet, D.Q., Yamamura, A., Tanaka, H.: Anonymous password-based authenticated key exchange. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 244–257. Springer, Heidelberg (2005)
40. Wu, T.D.: The secure remote password protocol. In: Network and Distributed System Security - NDSS 1998. The Internet Society (1998)
41. Yang, Y., Zhou, J., Weng, J., Bao, F.: A new approach for anonymous password authentication. In: ACSAC 2009, pp. 199–208. IEEE Computer Society (2009)