SPOKE: Simple Password-Only Key Exchange in the Standard Model

Michel Abdalla, Fabrice Benhamouda, and David Pointcheval

ENS, Paris, France *

Abstract. In this paper, we propose a simple and efficient password-only authenticated key exchange (PAKE) protocol with a proof of security in the standard model. In its most efficient instantiation, the new protocol has only two flows of communication and a total of 7 group elements and its proof of security is based on the plain DDH assumption. To achieve this goal, we first propose a variant of the Gennaro-Lindell/Katz-Ostrovsky-Yung (GL/KOY) PAKE protocol, in which the encryption schemes used to generate the first- and second-flow messages are only required to be semantically secure against plaintext-checking attacks (IND-PCA) and chosen-plaintext attacks (IND-CPA), respectively. Unlike semantic security against chosen-ciphertext attacks (IND-CCA), an IND-PCA adversary is only given access to an oracle which says whether or not a given ciphertext encrypts a given message. Next, we design a more efficient variant of the Cramer-Shoup encryption scheme with shorter ciphertexts together with an associated hash proof system and we prove its IND-PCA security under the plain DDH assumption. We believe that the new IND-PCA scheme is of independent interest, since it can also replace the Cramer-Shoup encryption scheme in many other PAKE schemes in the standard model, and it yields the most efficient "algebraic" IND-CCA encryption scheme, under plain DDH, for small messages.

1 Introduction

Password-only authenticated key exchange (PAKE) protocols allow users to establish a secure channel over a public, possibly adversarially controlled, network, with the help of a simple password. Unlike MAC-based or signature-based authenticated key exchange protocols, which usually require a special-purpose hardware capable of storing high-entropy secret keys or certified public-keys, PAKE protocols only require the knowledge of easily memorizable password.

The setting in which only passwords are used for authentication was first proposed by Bellovin and Merritt in 1992 [BM92], who also proposed a candidate protocol, known as the Encrypted Key Exchange (EKE) protocol. Though their protocol had no formal security proofs, it became the basis of several follow-up works (e.g., [BM93, Luc97, Jab97, STW95]) due to its simplicity. In a nutshell, their protocol can be seen as an encrypted version of the Diffie-Hellman key exchange protocol [DH76], where the password is used as the encryption key.

Due to the low entropy of passwords, the problem of modeling the security of PAKE schemes is not straightforward as these protocols are always subject to online dictionary attacks. In these attacks, the attacker can simply guess the value of the password among the set of possible values (i.e., the dictionary) and then verify whether its guess was correct by interacting with the system. While these attacks are unavoidable, their damage can be mitigated by using appropriate organizational restrictions such as limiting the number of failed login attempts.

The first ones to propose security models for PAKE schemes were Bellare, Pointcheval, and Rogaway (BPR) [BPR00] and Boyko, MacKenzie, and Patel (BMP) [BMP00]. While the BPR security model was based on the game-based security model by Bellare and Rogaway for secure key distribution [BR95], the BMP security model was based on the simulation-based model by Shoup for authenticated key exchange [Sho99]. In addition to introducing new security models for PAKE schemes, Bellare, Pointcheval, and Rogaway [BPR00] and Boyko, MacKenzie, and Patel [BMP00] also proved the security of certain variants of the EKE schemes in idealized models such as the ideal-cipher and the random-oracle models. These results were soon improved in a series of works [Mac02, BCP03, BCP04, AP05]. Unfortunately, the use of idealized models in the security proof of these schemes is a very strong assumption as these models are known not to be sound [CGH98, Nie02, GK03, BBP04].

* CNRS – UMR 8548 and INRIA

The first PAKE protocols to be proven secure in the standard model were proposed by Katz, Ostrovsky, and Yung (KOY) [KOY01] based on the decisional Diffie-Hellman assumption and by Goldreich and Lindell [GL01] based on general assumptions. While the KOY protocol assumed the existence of a trusted common reference string (CRS), the work of Goldreich and Lindell did not assume any trusted setup assumption. Due to its efficiency, the KOY protocol soon became the basis of several other protocols, starting with the work of Gennaro and Lindell (GL) [GL03] who abstracted and generalized it using the notion of smooth projective hash functions (SPHFs), introduced by Cramer and Shoup [CS02], and followed by many others [JG04, CHK+05, KMTG05, BGS06, AP06, Gen08, ACP09, GK10].

Among the different extensions that were proposed, the work of Canetti *et al.* [CHK+05] was the first to consider security in the universal composability (UC) framework [Can01]. While they showed that a variant of the KOY/GL protocol could realize the new security model, their protocol required several rounds of communication and was only known to be secure against *static* adversaries. To get around these limitations, several other constructions have been proposed [ACP09, KV11, BBC+13, ABB+13] achieving adaptive security and/or better round complexity.

In the BPR model, the PAKE protocol by Gennaro [Gen08], which relies on MACs instead of one-time signatures, and the KOY/GL variants proposed by [CHK+05, KMTG05, AP06], which shows that one of the flows of the original KOY/GL protocol can be computed using a semantically secure (IND-CPA) encryption scheme, remain the most efficient protocols. Moreover, as shown by [KOY02, KOY09], these protocols were shown to even achieve a weak flavor of forward secrecy. Note that the slightly different framework from [JG04, GK10] is hard to compare since it either makes use of an additional pseudo-random generator to expand the hash value, or requires a larger hash value and thus a larger encryption scheme in the first flow. Anyway, our improvement applies to this framework too.

In this work, our main goal is to further improve the efficiency of PAKE schemes in the BPR model. Towards this goal, we first revisit the KOY/GL framework and show that one can instantiate it using weaker forms of public-key encryption schemes. More precisely, all existing instantiations of the KOY/GL framework require either the use of encryption schemes which are semantically secure against chosen-ciphertext attacks (IND-CCA), which is one of the strongest security notions for encryption schemes [BDPR98], or the use non-interactive perfectly-binding commitments with non-malleability for multiple commitments. Even though the latter could in principle be based on semantically secure (IND-CPA) public-key encryption schemes using results from [PasV06, CDSMW09], the most efficient instantiations are based on IND-CCA encryption schemes. In contrast, in our new variant, we need at most an encryption scheme which is semantically secure against plaintext-checking attacks (IND-PCA) [OP01]. Unlike the notion of chosen-ciphertext security in which the adversary has access to a full decryption oracle, an IND-PCA adversary only has access to a plaintext-checking oracle that answers, on a given pair (m, c), whether c encrypts m or not. Finally, in order to instantiate our new variant of the KOY/GL framework, we also design a more efficient variant of the Cramer-Shoup encryption scheme together with an associated hash proof system. The new encryption scheme has shorter ciphertexts and achieves IND-PCA security under the plain DDH assumption.

Although it was already known that one of the two ciphertexts could be generated using an IND-CPA encryption scheme [CHK+05, KMTG05, AP06], IND-CCA security was always required for the generation of the other ciphertext in all concrete instantiations of the KOY/GL framework. Hence, by fully avoiding the use of IND-CCA encryption schemes, we are able to avoid the use of Cramer-Shoup encryption scheme, and to use more efficient SPHFs. Both improvements together lead to the most efficient PAKE in the BPR model: a quite clean 2-flow protocol with 7 group elements in total, with a security just relying on the DDH in that group, instead of at least 8 group elements in 3 flows [KMTG05], 10 group elements in 3 flows [KOY02, Gen08, KOY09], or 12 group elements in one round [BBC+13].

We also stress that most of the security proofs of PAKE protocols in the BPR security model would remain essentially unchanged if the underlying encryption scheme is assumed to be IND-PCA instead of IND-CCA. For instance, the PAKE schemes with mutual authentication in [JG04, GK10] would still remain secure if we use an IND-CPA encryption scheme to generate the first flow and an IND-PCA encryption scheme to generate the second. Moreover, in cases where mutual authentication is not needed, one could further improve the overall efficiency of these protocols by removing the third flow. The resulting scheme would only have 2 flows and require the exchange of 6 group elements in total. Likewise, the one-round PAKE protocol in [BBC+13] can also be instantiated with our new IND-PCA construction and its associated KVSPHF instead of the Cramer-Shoup encryption scheme. The resulting scheme would only require the exchange of 10 group elements in total, instead of the current 12.

In addition, our new IND-PCA encryption scheme, whose ciphertext consists of only 3 group elements, yields the most efficient SPHF-friendly IND-CCA encryption scheme for small messages under the plain DDH assumption. As a consequence, we believe this scheme is of independent interest.

2 Security Model

In this section, we recall the BPR security model [BPR00] and the extension proposed by Abdalla, Fouque, and Pointcheval (AFP) [AFP05].

2.1 The Bellare-Pointcheval-Rogaway Security Model

Users and Passwords. Each client $C \in \mathcal{C}$ holds a password π_C, while each server $S \in \mathcal{S}$ holds passwords $\pi_{S,C}$ for each client C.

Protocol Execution. The adversary $\mathcal{A}$ can create several concurrent instances U^i of each user $U \in \mathcal{C} \cup \mathcal{S}$, and can interact with them via the following oracle queries:

- $\texttt{Execute}(C^i, S^j)$: this query models a passive attack in which the adversary eavesdrops on honest executions between a client instance C^i and a server instance S^j. The output of this query consists of the messages that are exchanged during an honest execution of the protocol between C^i and S^j (i.e., the transcript of the protocol);
- $\texttt{Send}(U^i, U'^j, m)$: this query models an active attack, in which the adversary may intercept a message and modify it, create a new message, or simply replay or forward an existing message, to the user instance U'^j in the name of the user instance U^i. The output of this query is the message that U'^j would generate after receiving m. A specific message Start can be sent to a client, in the name of a server, to initiate a session between this client and this server;
- $\texttt{Reveal}(U)$: this query models the misuse of the session key that has been established. The output of this query is the session key, if it has been set.
- $\texttt{Corrupt}(C)$: this query models the client corruption. The output of this query is the password π_C.
- $\texttt{Corrupt}(S, C, \pi)$: this query models the server corruption. The output of this query is the stored password $\pi_{S,C}$. In addition, if $\pi \neq \perp$, $\pi_{S,C}$ is then changed to π.

This is a slight variant of the so-called *weak corruption* model in BPR, since the long term secrets (passwords) only are leaked, and not the internal states, in case of corruption. But contrarily to BPR, in case of server corruption, we also leak the password even in case of a password change request. However, this does not affect the security notion since, in both the original BPR model and in ours, any corruption query makes the password *corrupted*, and so the Test-query is not allowed anymore on instances of these players (see below), since they are no longer fresh.

Partnering. Before actually defining the secrecy of the session key, and thus implicit authentication, we need to introduce the notion of partnering: Two instances are partnered if they have matching

transcripts, which means that, for one user, its view is a part of the view of the other user. One should note that the last flow can be dropped by the adversary, without letting the sender know. The sender of this last flow thus thinks that the receiver got the message and still computes the session key.

Security. To actually define the semantic security of a PAKE scheme, the adversary $\mathcal{A}$ has access to a challenge oracle $\texttt{Test}(U^i)$, available once only, to evaluate the indistinguishability of a specific session key. A random bit b is chosen and the `Test`-query, for some user instance U^i is answered as follows: if $b = 1$, return the session key of U^i, and otherwise, return a random session key. At the end of the game, the adversary $\mathcal{A}$ has to output a bit b', as a guess for b. The success probability Succ of $\mathcal{A}$ is the probability that $b' = b$, while its advantage is defined by $\mathsf{Adv} = 2 \cdot \mathsf{Succ} - 1$.

Note that there are natural restrictions for the `Test`-query: the tested instance must be *fresh*, which means that this is not a trivial case, where trivial cases are no key or known key. More precisely, there are two definitions of freshness, whether we consider the forward-secrecy, or not:

- basic freshness: an instance U^i is fresh (fresh) if,
  - a session key has been defined;
  - no `Reveal`-query has been asked to U^i, or to his partner, if there is one;
  - the password of the client C has not been corrupted (either via a query $\texttt{Corrupt}(C)$ or via a query $\texttt{Corrupt}(\cdot, C, \cdot)$), where $C = U$ is U is a client or U^i's partner is an instance C^j of C
- forward-secure freshness: similar to basic freshness except for the last part, where only corruptions before U^i defined his key can make this instance unfresh.

In case of `Test`-query to an unfresh instance, the answer is $\perp$, which means that the adversary cannot have any advantage in these cases. A PAKE is considered BPR-secure if the advantage of any adversary $\mathcal{A}$, running within time t, in the previous experiment is bounded by $q_s \times 2^{-m} + \mathrm{negl}(\mathfrak{K})$, where q_s is the number of active sessions (handled with `Send` queries). Intuitively this means that to win, the adversary has to do an on-line dictionary attack, which only enables it to test one password per session.

2.2 The Abdalla-Fouque-Pointcheval Security Model

It extends the model with multiple `Test`-queries, which are all answered with the same bit b. Queries asked to unfresh instances are answered by $\perp$.

3 PAKE Construction

Our construction follows the GL framework [GL03] based on smooth projective hash functions (SPHFs), which is a generalization of the KOY protocol [KOY01].

3.1 Tools

Smooth Projective Hash Functions. Projective hash function families were first introduced by Cramer and Shoup [CS02]. Here we use the formalization from [BBC+13]: Let X be the domain of these functions and let L be a certain subset of this domain (a language). A key property of these functions is that, for words c in L, their values can be computed by using either a *secret* hashing key hk or a *public* projection key hp but with a witness w of the fact that c is indeed in L:

- $\mathsf{HashKG}(L)$ generates a hashing key hk for the language L;
- $\mathsf{ProjKG}(\mathsf{hk}, L, c)$ derives the projection key hp, possibly depending on the word c;
- $\mathsf{Hash}(\mathsf{hk}, L, c)$ outputs the hash value from the hashing key, for any word $c \in X$;
- $\mathsf{ProjHash}(\mathsf{hp}, L, c, w)$ outputs the hash value from the projection key hp, and the witness w, for a word $c \in L$.

On the one hand, the *correctness* of the SPHF assures that if $c \in L$ with w a witness of this fact, then $\mathsf{Hash}(\mathsf{hk}, L, c) = \mathsf{ProjHash}(\mathsf{hp}, L, c, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $c \notin L$, $\mathsf{Hash}(\mathsf{hk}, L, c)$ is *statistically* indistinguishable from a random element, even knowing hp.

Note that HashKG and ProjKG can just depend partially on L (i.e., can only depend on a superset $\hat{L}$): we then note $\mathsf{HashKG}(\hat{L})$ and $\mathsf{ProjKG}(\mathsf{hk}, \hat{L}, c)$. In addition, if HashKG and ProjKG do not depend on c, and verify a slightly stronger smoothness property (called adaptive smoothness, which holds even if c is chosen after hp), we say the SPHF is a KVSPHF. Otherwise, it is said to be a GLSPHF. See [BBC+13] for details on GLSPHF and KVSPHF and language definitions.

Encryption Scheme. A (labeled) public-key encryption scheme is defined by three algorithms:
– $\mathsf{KG}(1^\mathfrak{K})$ generates a key pair: a public key pk and a secret key sk;
– $\mathsf{Enc}^\ell(\mathsf{pk}, x; r)$ encrypts the message x under the key pk with label ℓ, using the random coins r;
– $\mathsf{Dec}^\ell(\mathsf{sk}, c)$ decrypts the ciphertext c, using the secret key sk, with label ℓ.
The main security notion for an encryption scheme is the so-called *indistinguishability of ciphertexts*, in which the adversary has to distinguish, for two messages m_0 and m_1, and possibly a label ℓ, all of its choice, which one has been encrypted when it receives $c = \mathsf{Enc}^\ell(\mathsf{pk}, m_b; r)$ for a random bit b. It does so by answering its guess b' on b. Its advantage is denoted $\mathsf{Adv}^{\mathtt{ind}}(\mathcal{A}) = 2 \times \Pr[b' = b] - 1$.

To achieve its goal, the adversary can just have access to the public key, which allows it to encrypt any message of its choice, hence the Chosen-Plaintext Attack (denoted `CPA`), or to the decryption oracle (except on the challenge ciphertext with the challenge label), hence the Chosen-Ciphertext Attack (denoted `CCA`). Another scenario has been proposed in [OP01], with access to a Plaintext-Checking oracle, which answers whether for a given tuple (ℓ, m, c), c is indeed an encryption of m with the label ℓ (except on the challenge ciphertext with the challenge label), hence the Chosen-Plaintext Attack (denoted `PCA`). In this paper, we suppose that encryption schemes are `IND-CPA` (without labels) or `IND-PCA` (with labels) only.

While previous constructions either required non-malleable or `IND-CCA` encryption schemes, the proof in the BPR security model essentially stops when the adversary encrypts the correct password: the simulator thus just has to be able to check it, hence the `PCA` notion.

Collision-Resistant Hash Function Family. A family $\mathcal{H}$ of hash functions from a set X to a set Y is said (t, ε)-collision-resistant if for any adversary $\mathcal{A}$ running within time t, on a random element $H \xleftarrow{\$} \mathcal{H}$, its probability to output $x \neq x'$ such that $H(x) = H(x')$ is bounded by ε.

We denote $\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t)$ the best success probability any adversary can get within time t.

3.2 Generic Two-Flow Construction

Let us consider a labeled `IND-PCA` encryption scheme $\mathsf{ES} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and an `IND-CPA` encryption scheme $\mathsf{ES}' = (\mathsf{KG}', \mathsf{Enc}', \mathsf{Dec}')$ so that a GLSPHF and a KVSPHF (respectively) exist for the following families of languages:

$$L_\pi = \{(\ell, c) \mid \exists r,\, c = \mathsf{Enc}^\ell(\mathsf{pk}, \pi; r)\} \qquad L'_\pi = \{c \mid \exists r,\, c = \mathsf{Enc}'(\mathsf{pk}', \pi; r)\},$$

with $\mathsf{param}, \mathsf{pk}$ and pk' global parameters in the common reference string CRS, generated as follows: $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^\mathfrak{K}, 1^n)$, $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KG}(1^\mathfrak{K})$ and $(\mathsf{sk}', \mathsf{pk}') \xleftarrow{\$} \mathsf{KG}'(1^\mathfrak{K})$. We also suppose that HashKG and ProjKG, for both L_π and L'_π, do not depend on π, and thus, just (respectively) on.

$$L = \{(\ell, c) \mid \exists \pi, \exists r,\, c = \mathsf{Enc}^\ell(\mathsf{pk}, \pi; r)\} \qquad L' = \{c \mid \exists \pi, \exists r,\, c = \mathsf{Enc}'(\mathsf{pk}', \pi; r)\},$$

Then our two-flow construction is depicted in Figure 1, where $\times$ is a commutative operation between hash values such that if A is a uniform hash value and B is any hash value, $A \times B$ is uniform (often hash values live in a group and $\times$ is just the group law). In Section 4.3, we prove the following

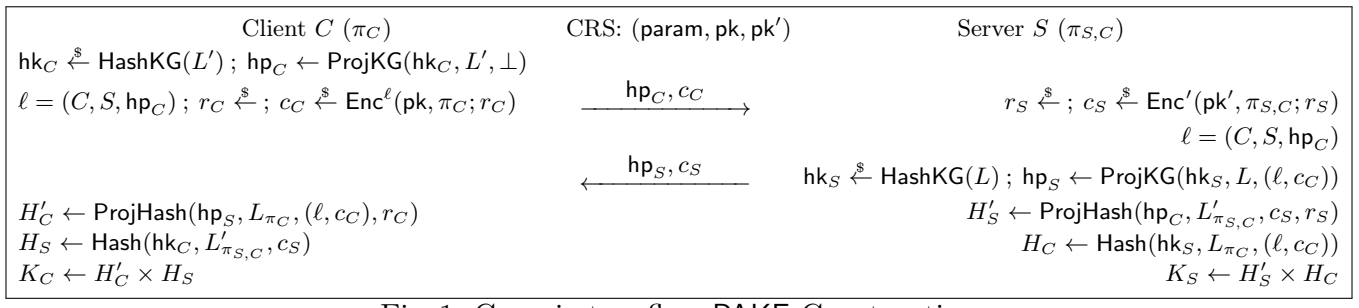

Fig. 1: Generic two-flow PAKE Construction

result, which applies for the basic freshness in the BPR setting, or for the forward-secure freshness in the AFP setting with static corruptions only:

$$\mathsf{Adv}(\mathcal{A}) \leq q_s \times 2^{-m} + (q_e + q_s) \times (\mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{ES}'}(t) + \mathsf{Adv}^{\mathtt{ind\text{-}pca}}_{\mathsf{ES}}(t)) + \frac{q_e q_s}{2^{2n}},$$

where q_e and q_s are the number of `Execute` and `Send`-queries, n is the entropy of both the projected keys and the ciphertexts, and m is the entropy of the passwords.

3.3 Instantiation

ElGamal Encryption Scheme. The ElGamal (EG) encryption scheme [ElG84] is defined as follows, in a cyclic group $\mathbb{G}$ of prime order p, with a generator p:
- $\mathsf{EG.KG}(1^{\mathfrak{K}})$ generates the secret key $\mathsf{sk} = x \xleftarrow{\$} \mathbb{Z}_p$ and the public key $\mathsf{pk} = y = g^x$;
- $\mathsf{EG.Enc}(\mathsf{pk} = y, M; r)$, for a group element $M \in \mathbb{G}$ and a scalar $r \in \mathbb{Z}_p$, generates the ciphertext $c = (u = g^r, e = y^r M)$;
- $\mathsf{EG.Dec}(\mathsf{sk} = x, c = (u, e))$ computes $M = e/u^x$.

This encryption scheme is well-known to be `IND-CPA` under the DDH assumption:

$$\mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{EG}}(t) \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t).$$

It also provides an efficient KVSPHF for the language $L_M = \{c \mid \exists r,\ c = \mathsf{EG.Enc}(\mathsf{pk}, M; r)\}$, with $L = \mathbb{G}^2$ the superset of the ciphertexts:

$$\mathsf{hk} = \mathsf{HashKG}(L) = (\alpha, \beta) \xleftarrow{\$} \mathbb{Z}_p^2 \qquad \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, L, \perp) = g^{\alpha} y^{\beta}$$
$$H = \mathsf{Hash}(\mathsf{hk}, L_M, c) = u^{\alpha}(e/M)^{\beta} \qquad H' = \mathsf{ProjHash}(\mathsf{hp}, L_M, c, r) = \mathsf{hp}^r$$

Shorter Cramer-Shoup Encryption Scheme. The labeled Shorter Cramer-Shoup (SCS) encryption scheme is a variant of the well-known Cramer-Shoup [CS98] encryption scheme, but with one less element. It is defined as follows, in a cyclic group $\mathbb{G}$ of prime order p, with a generator p, together with a hash function H_{CS} randomly drawn from a collision-resistant[1] hash function family $\mathcal{H}$ from the set $\{0,1\}^* \times \mathbb{G}^2$ to the set $\mathbb{G}\backslash\{1\}$:
- $\mathsf{SCS.KG}(1^{\mathfrak{K}})$ generates the secret key $\mathsf{sk} = (s, a, b, a', b') \xleftarrow{\$} \mathbb{Z}_p$ and the public key $\mathsf{pk} = (h = g^s, c = g^a h^b, d = g^{a'} h^{b'})$;
- $\mathsf{SCS.Enc}^{\ell}(\mathsf{pk} = (h, c, d), M; r)$, for a label ℓ, a group element $M \in \mathbb{G}$ and a scalar $r \in \mathbb{Z}_p$, generates the ciphertext $c = (u = g^r, e = h^r M, v = (cd^{\xi})^r)$, where $\xi = H_{\mathsf{CS}}(\ell, u, e)$;
- $\mathsf{SCS.Dec}^{\ell}(\mathsf{sk} = (x, a, b, a', b'), c = (u, e, v))$ first computes $M = e/u^s$ and checks whether $v = u^{a+\xi a'} \times (e/M)^{b+\xi b'}$, for $\xi = H_{\mathsf{CS}}(\ell, u, e)$. If the equality holds, it outputs M, otherwise it outputs $\perp$.

We show below it is `IND-PCA` under the DDH and the collision-resistance assumptions:

$$\mathsf{Adv}^{\mathtt{ind\text{-}pca}}_{\mathsf{SCS}}(t) \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) + 2(q_{\mathtt{PCA}} + 1)/p,$$

where $q_{\mathtt{PCA}}$ is the number of queries to the `PCA` oracle.

[1] Second-preimage resistance is actually enough, as for the original Cramer-Shoup encryption scheme.

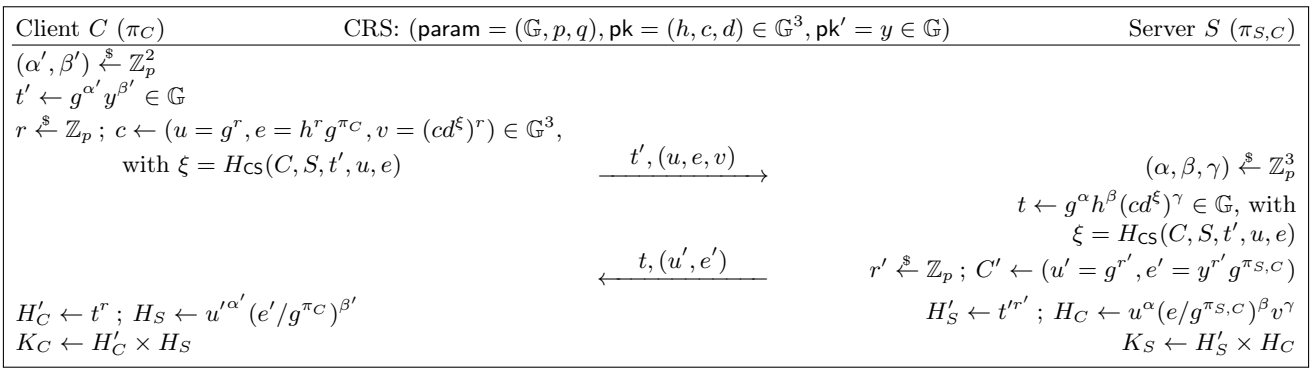

Fig. 2: SPOKE

It also provides an efficient GLSPHF for the language $L_M = \{c \mid \exists r,\ c = \mathsf{SCS.Enc}^\ell(\mathsf{pk}, M; r)\}$, with L the superset of the ciphertexts:

$$\mathsf{hk} = \mathsf{HashKG}(L) = (\alpha, \beta, \gamma) \xleftarrow{\$} \mathbb{Z}_p^2 \qquad \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, L, c) = g^\alpha h^\beta (cd^\xi)^\gamma$$
$$H = \mathsf{Hash}(\mathsf{hk}, L_M, c) = u^\alpha (e/M)^\beta v^\gamma \qquad H' = \mathsf{ProjHash}(\mathsf{hp}, L_M, c, r) = \mathsf{hp}^r$$

Simple Password-Only Key Exchange. Combining these concrete primitives leads to the most efficient PAKE known so far, with a proof of security in the standard model, as presented on Figure 2. It is based on the plain DDH assumption, and consists of 4 group elements to be sent by the client and 3 group elements by the server. They both have to compute 10 exponentiations.

Using the above security bounds for the encryption schemes, one gets, for the basic freshness in the BPR setting, or for the forward-secure freshness in the AFP setting with static corruptions:

$$\mathsf{Adv}(t) \leq q_s \times 2^{-m} + (q_e + q_s) \times (2\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t)) + \frac{2(q_e + q_s)(q_s + 1)}{p} + \frac{q_e q_s}{p^2},$$

where q_e and q_s are the number of `Execute` and `Send`-queries, and m is the min-entropy of the passwords. Note that with the above encryption schemes and SPHFs, ciphertexts, and projection keys have entropy $\log_2 p$.

4 Security Proof

4.1 ElGamal Encryption Scheme

The `IND-CPA` security of the ElGamal encryption scheme is well-known, and the SPHF comes from the following matrix (using the formalism from [BBC+13]):

$$\Gamma = \begin{pmatrix} g & y \end{pmatrix} \qquad \begin{array}{rcl} \boldsymbol{\lambda} & = & (r) \\ \boldsymbol{\lambda} \cdot \Gamma & = & (g^r, y^r) \\ \Theta(c) & = & (u, e/M) \end{array}$$

The matrix Γ does not depend on c and, thus, this is a KVSPHF.

4.2 Shorter Cramer Shoup Encryption Scheme

Before proving the `IND-PCA` security, we show the SPHF comes from the following matrix:

$$\Gamma(c) = \begin{pmatrix} g & h & cd^\xi \end{pmatrix} \qquad \begin{array}{rcl} \boldsymbol{\lambda} & = & (r) \\ \boldsymbol{\lambda} \cdot \Gamma & = & (g^r, h^r, (cd^\xi)^r) \\ \Theta(c) & = & (u, e/M, v) \end{array}$$

The matrix Γ depends on ξ, and thus on c. Hence, this is a GLSPHF. Even though we could have built a KVSPHF for the new scheme, as done in [BBC+13] for the full-fledged Cramer-Shoup encryption scheme, the latter would be less efficient, requiring two group elements instead of one (hence the 3+2

group elements for each user in the one-round PAKE variant using this encryption scheme). Moreover, since there is already one interaction in our protocol, there is no need to have a KVSPHF as the more efficient GLSPHF is already sufficient.

For the IND-PCA security, we first recall the security game in Game $\mathbf{G}_0$, and present a series of games to show the advantage of the adversary is negligible.

Game $\mathbf{G}_0$: The adversary $\mathcal{A}$ is given a public key $\mathsf{pk} = (h = g^s, c = g^a h^b, d = g^{a'} h^{b'})$, generated with the secret key $\mathsf{sk} = (s, a, b, a', b') \xleftarrow{\$} \mathbb{Z}_p$, as well as an unlimited access to a PCA oracle with input a tuple (ℓ, M, c) that consists of a ciphertext c and an alleged plaintext M with the label ℓ. This oracle answers whether c really encrypts M or not. At some point, the adversary outputs a label ℓ^* and two message M_0 and M_1, and receives the encryption $c^* = (u^*, e^*, v^*)$ of M_δ with the label ℓ^*. After more calls to the PCA' oracle, the adversary outputs a bit δ', its guess on the bit δ. Note that in this second phase, the PCA' oracle slightly differs from the PCA oracle: on the target ciphertext c^*, with the same label ℓ^*, it outputs $\perp$. In the sequel, when we talk about the PCA oracle, we mean the PCA and the PCA' oracles.
More precisely, c^* is generated with a random scalar $r^* \xleftarrow{\$} \mathbb{Z}_p$, as $c^* = (u^* = g^{r^*}, e^* = h^{r^*} M_\delta, v^* = (cd^{\xi^*})^{r^*})$, where $\xi^* = H(\ell^*, u^*, e^*)$. The PCA oracle, on input $(\ell, M, c = (u, e, v))$, unless $(\ell, c) = (\ell^*, c^*)$, checks both equations: $e \stackrel{?}{=} u^s M$ and $v \stackrel{?}{=} u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$, for $\xi = H(\ell, u, e)$. Then, $\mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) = \mathsf{Adv}^{\mathtt{ind\text{-}pca}}(\mathcal{A})$.

Game $\mathbf{G}_1$: In this game, we reject all queries $(\ell, M, c = (u, e, v))$ to the PCA (and PCA') oracle, where $(\ell, u, e) \neq (\ell', u', e')$ but $\xi' \neq \xi$. This game is computationally indistinguishable from the previous one under the collision-resistance of H_{CS}: $|\mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A})| \leq \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t)$, where t is approximately the running time of $\mathcal{A}$.

Game $\mathbf{G}_2$: We first simplify the simulation of the PCA oracle: it just checks the second equation: $v \stackrel{?}{=} u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$, for $\xi = H(\ell, u, e)$, so that we do not need to know s in this game anymore. It can only make a difference if this equation is satisfied while the first was not: $e = u^{s'} M$ and $v = u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$, for $\xi = H(\ell, u, e)$, with $h = g^s$ and $\Delta_s = s' - s \neq 0$.
However, we can see that the probability for v to satisfy the above equation while e does not is negligible since a, a', b, b' are unknown. Indeed, with the given relations, any v could be possible: a powerful adversary might know, where $u = g^r$,

$$\begin{cases} c &= g^a h^b \\ d &= g^{a'} h^{b'} \\ v &= u^{a+\xi a'} \cdot (e/M)^{b+\xi b'} \\ &= u^{a+\xi a'} \cdot u^{s'(b+\xi b')} \end{cases} \qquad \begin{cases} \log_g c &= a + s \cdot b \\ \log_g d &= a' + s \cdot b' \\ \log_g v &= \log_g u \cdot ((a + \xi a') + s'(a' + \xi b')) \\ &= r \cdot (\log_g c + \xi \log_g d + \Delta_s \cdot (b + \xi b')) \end{cases}$$

Since b and b' are unpredictable and $\Delta = s' - s \neq 0$, the expected value for v is unpredictable (see more details in Game $\mathbf{G}_6$), hence $|\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})| \leq q_{\mathtt{PCA}}/p$, where $q_{\mathtt{PCA}}$ is the number of queries to the PCA oracle.

Game $\mathbf{G}_3$: We are now given a Diffie-Hellman tuple $(g, X = g^x, Y = g^y, Z = g^z)$, with $z = xy$. We set $h \leftarrow X$ (which means that $s = x$), but let the rest of the setup as before: $a, b, a', b' \xleftarrow{\$} \mathbb{Z}_p$ and $\delta \xleftarrow{\$} \{0, 1\}$. This is possible since we do not know s anymore since Game $\mathbf{G}_2$. For the challenge ciphertext, we set $u^* \leftarrow Y$ (which means that $r^* = y$) and $e^* \leftarrow Z M_\delta$. For v^*, since we do not know r^*, we use the verification equation: $v^* \leftarrow Y^{a+\xi^* a'} \cdot Z^{b+\xi^* b'}$, for $\xi^* = H(\ell^*, u^*, e^*)$. Since $z = xy$,

$$v^* \leftarrow g^{y(a+\xi^* a') + xy(b+\xi^* b')} = (g^{(a+xb)} \cdot g^{\xi^*(a'+xb')})^y = ((g^a h^b) \cdot (g^{a'} h^{b'})^{\xi^*})^y = (cd^{\xi^*})^{r^*}.$$

This is thus perfectly indistinguishable from the previous game: $\mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A})$.

Game $\mathbf{G}_4$: We are now given a random tuple $(g, X = g^x, Y = g^y, Z = g^z)$, with z independently chosen. The simulation is the same as in the previous game: $|\mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{ddh}}(t)$, where t is essentially the running time of the adversary $\mathcal{A}$.

Game $\mathbf{G}_5$: We now choose z uniformly at random in $\mathbb{Z}_p \setminus \{xy\}$ instead of $\mathbb{Z}_p$. This game is statistically indistinguishable from the previous one: $|\mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A})| \leq 1/p$.

Game $\mathbf{G}_6$: We now randomly choose $g \xleftarrow{\$} \mathbb{G}$, and $x, y, z \xleftarrow{\$} \mathbb{Z}_p$ (with $z \neq xy$) to define the random tuple $(g, X = g^x, Y = g^y, Z = g^z)$ as in the previous game, but with the knowledge of the exponents. We thus know again $s = x$. We can go back with the full simulation of the PCA oracle: it additionally checks whether $e = u^s M$ or not. It can again make a difference if this equation is not satisfied while the other one was: $e = u^{s'} M$ and $v = u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$, for $\xi = H(\ell, u, e)$, with $h = g^s$ and $\Delta_s = s' - s \neq 0$.

Let us suppose that $(\ell, u, e, v) \neq (\ell^*, u^*, e^*, v^*)$, otherwise the oracle just answers $\perp$.

First, if $(\ell, u, e) = (\ell^*, u^*, e^*)$ but $v \neq v^*$, since that implies $\xi = \xi^*$, we can safely answer negatively. We thus now have to deal with the cases $(\ell, u, e) \neq (\ell^*, u^*, e^*)$, where $\xi^* \neq \xi$ (since we have already dealt with collisions in ξ and ξ' in Game $\mathbf{G}_1$).

As in Game $\mathbf{G}_2$, we can see that the probability for v to satisfy the above equation while e does not is negligible since a, b, a', b' are unknown. This is a bit more subtle since more relations are available to the adversary. Anyway, with the given relations, any v could be possible: a powerful adversary might know, where $u = g^r$ and $\Delta_z = z - xy$,

$$\left\{\begin{array}{rcl} c &=& g^a h^b \\ d &=& g^{a'} h^{b'} \\ v^* &=& u^{*a+\xi^* a'} \cdot (e^*/M_\delta)^{b+\xi^* b'} \\ &=& g^{y(a+\xi^* a')} \cdot g^{z(b+\xi^* b')} \\ v &=& u^{a+\xi a'} \cdot (e/M)^{b+\xi b'} \\ &=& g^{r(a+\xi a')} \cdot g^{rs'(b+\xi b')} \end{array}\right. \qquad \left\{\begin{array}{rcl} \log_g c &=& a + s \cdot b \\ \log_g d &=& a' + s \cdot b' \\ \log_g v^* &=& y \cdot (a + \xi^* a') + z(b + \xi^* b') \\ &=& y \cdot (\log_g c + \xi^* \log_g d) + \Delta_z \cdot (b + \xi^* b') \\ \log_g v &=& r \cdot (a + \xi a') + rs'(b + \xi b') \\ &=& r \cdot (\log_g c + \xi \log_g d + \Delta_s \cdot (b + \xi b')) \end{array}\right.$$

This system can be turned into

$$\begin{pmatrix} \log_g c \\ \log_g d \\ \log_g v^* - y \cdot (\log_g c + \xi^* \log_g d) \\ \log_g v - r \cdot (\log_g c + \xi \log_g d) \end{pmatrix} = \begin{pmatrix} 1 & 0 & s & 0 \\ 0 & 1 & 0 & s \\ 0 & 0 & \Delta_z & \Delta_z \xi^* \\ 0 & 0 & \Delta_s & \Delta_s \xi \end{pmatrix} \cdot \begin{pmatrix} a \\ a' \\ b \\ b' \end{pmatrix}$$

where the determinant is clearly $\Delta_z \Delta_s (\xi^* - \xi)$. Since we assumed $z \neq xy$, $\Delta_z \neq 0$, and no collision on the hash function H, the determinants are all non-zero, in which cases the expected values for v are unpredictable, hence $|\mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A})| \leq q_{\mathtt{PCA}}/p$, where $q_{\mathtt{PCA}}$ is the number of queries to the PCA oracle.

Game $\mathbf{G}_7$: We now choose v^* at random, independently of Y and Z.

To show this does not change anything, we basically show that what the adversary sees never depends on the four variables a, b, a', b', but only depends on $\alpha = a + xb$ and $\beta = a' + xb'$, except for v^*. This implies that $\gamma = y(a + \xi^* a) + z(b + \xi^* b')$ is linearly independent of α and β (when a, a', b, b' are unknowns), and so γ looks completely random to the adversary (if he does not sees v^*. Since $v^* = g^\gamma$, we conclude that, for the adversary v^* looks completely random. That would prove that $\mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A})$.

It remains to show that what the adversary sees never depends on the four variables a, b, a', b', but only depends on $\alpha = a + xb$ and $\beta = a' + xb'$, except for v^*. This comes from the fact that the only times where a, b, a', b' are used are in the PCA oracle, where we first check that $e \stackrel{?}{=} u^x$ and then, if that equality holds, checks that $v \stackrel{?}{=} u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$. Therefore, this last check can be replaced by $v \stackrel{?}{=} u^{\alpha+\xi\beta}$, which only depends on α and β, since when $e = u^x M$, $u^{a+\xi a'} \cdot (e/M)^{b+\xi b'} = u^{(a+\xi a')+x(b+\xi b')} = u^{(a+xb)+\xi(a'+xb')} = u^{\alpha+\xi\beta}$

Game $\mathbf{G}_8$: We now choose z uniformly at random in $\mathbb{Z}_p$ instead of $\mathbb{Z}_p \setminus \{xy\}$. This game is statistically indistinguishable from the previous one: $|\mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A})| \leq 1/p$.

Game $\mathbf{G}_9$: We now choose e^* at random, independently of Z and M_δ.

To show this does not change anything either, we review the previous game:

- the simulator chooses random scalars x, y, z to define the random tuple $(g, X = g^x, Y = g^y, Z = g^z)$, as well as random scalars α, β to define $c = g^\alpha, d = g^\beta$, and $\delta \xleftarrow{\$} \{0, 1\}$;
- for the PCA oracle on $(\ell, M, c = (u, e, v))$, one checks $e \stackrel{?}{=} u^x M$ and $v \stackrel{?}{=} u^{\alpha+\xi\beta}$, for $\xi = H(\ell, u, e)$;

– for the challenge ciphertext, on sets $u^* \leftarrow Y$, $e^* \leftarrow ZM_\delta$, and $v^* \xleftarrow{\$} \mathbb{G}$.

Since Z was used in e^* only, and anywhere else, a random Z or a random e^* are indistinguishable: $\mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A})$.

In addition, δ does not appear anywhere: $\mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A}) = 0$.

4.3 Security of the Generic Two-Flow PAKE

In this proof, we first deal with the BPR setting (one `Test`-query only) and with the basic freshness (once a password is corrupted, all the associated instances are considered unfresh, even before the corruption), but allowing adaptive corruptions. This notion of adaptive corruptions, which allows the adversary to ask `Corrupt`-queries at any time, makes the last game complex since the simulator does not know, at the simulation time of the ciphertext c_C, whether the password will get corrupted before the end of the execution of the protocol or not. In the case of static corruptions, where the adversary is only allowed to corrupt a password when no instance of the associated players is involved in an execution of the protocol, this is much simpler since the simulator knows from the beginning of a session which party is corrupted or not.

The AFP setting and forward-secrecy are discussed later. But adaptive corruptions seem hard to deal with in the AFP setting since the simulator cannot handle a `Reveal`-query on a client instance if the server gets corrupted after c_C has been sent (and thus simulated with a dummy password): the adversary can correctly generate c_S and compute the session key K_S, whereas the simulator is not able to compute K_C that should be equal to K_S. However, forward-security in the AFP setting, but with static corruptions only, easily follows from the proof below.

This proof is close to the proof from [GL03]. The proof consists in proving that the real attack game $\mathbf{G}_{\mathtt{real}} = \mathbf{G}_0$ is indistinguishable from a game in which no actual password is used, but just at the end or at the corruption time to control whether the flag Win has to be set to `True` or not. This flag is initially set to `False`, and at the end of the game, we say the adversary wins if $b' = b$, or if $\mathsf{Win} = \mathtt{True}$. We will also make the adversary win the game when a quite unlikely collision appears between two simulated flows (flag Coll set to `True`, see Game $\mathbf{G}_1$).

Let us consider a polynomial time adversary $\mathcal{A}$. The proof is done by a series of games $\mathbf{G}_i$, and $\mathsf{Adv}_i(\mathcal{A}, \mathfrak{K})$ is the advantage of $\mathcal{A}$ in the game $\mathbf{G}_i$. We separate `Send` queries in three types:

– $\mathtt{Send}_0(S^i, C^j, \mathtt{Start})$ queries which enable an adversary to ask C^j to initiate the protocol with S^i and which return the first flow for C^j and S^i.

– $\mathtt{Send}_1(C^i, S^j, m)$ queries which enable an adversary to send the first flow for C^i and S^j and which return the second flow answered back by S^j;

– $\mathtt{Send}_2(S^i, C^j, m)$ queries which enable an adversary to send the second flow for C^j and S^i and which return nothing but set the session key K of S^i.

`Reveal`-queries and the `Test`-query are answered using the defined session key, and according to the bit b and the freshness of instance. As already said, we first consider the basic freshness only, in which case a corrupted password makes all the instances of the associated client and server unfresh.

In the following proof, we assume the SPHFs to be perfectly smooth. This is the case of our candidates, but the proof could be extended to statistical smoothness only.

We say that two users (a client C and a server S) are *compatible* if $\pi_C = \pi_{S,C}$. Passwords are initially all set as the same for each client and the server, but a corruption of the server with a new password can replace $\pi_{S,C}$ by a different value than π_C: C and S are then said *incompatible*. Note that as in [KOY02, KOY09], the compatibility is defined at the beginning of the execution of the protocol (by uploading passwords in the local memory), which means that even in case of password change in the database during the protocol, this does not affect the passwords used during this execution.

Game $\mathbf{G}_1$: We first modify the way `Execute`-queries between two compatible users are answered.

Since the hashing keys are known, we compute the common session key as

$$K = K_S = K_C = \mathsf{Hash}(\mathsf{hk}_C, L'_{\pi_{S,C}}, c_S) \times \mathsf{Hash}(\mathsf{hk}_S, L_{\pi_C}, (\ell, c_C)).$$

This does not change anything thanks to the correctness of the SPHFs: $\mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A})$.

Game $\mathbf{G}_2$: Since the random coins are not needed anymore for these `Execute`-queries between two compatible users, we replace c_S and c_C by encryptions of the dummy passwords 0_S and 0_C. This is indistinguishable from $\mathbf{G}_1$ under the `IND-CPA` property of the encryption schemes, for each `Execute`-query between two compatible users. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})| \leq q_{e,1} \times (\mathsf{Adv}^{\texttt{ind-cpa}}_{\mathsf{ES}}(t) + \mathsf{Adv}^{\texttt{ind-cpa}}_{\mathsf{ES}'}(t))$, where $q_{e,1}$ is the number of `Execute`-queries between two compatible users.

Game $\mathbf{G}_3$: We modify again the way `Execute`-queries between two compatible users are answered: we replace the common session key by a truly random value. Since the languages are not satisfied, the perfect smoothness guarantees perfect indistinguishability: $\mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A})$.

Game $\mathbf{G}_4$: We now modify the way `Execute`-queries between two incompatible users are answered: we replace both session keys

$$K_C = \mathsf{ProjHash}(\mathsf{hp}_S, L_{\pi_C}, (\ell, c_C), r_C) \times \mathsf{Hash}(\mathsf{hk}_C, L'_{\pi_{S,C}}, c_S)$$
$$K_S = \mathsf{Hash}(\mathsf{hk}_S, L_{\pi_C}, (\ell, c_C)) \times \mathsf{ProjHash}(\mathsf{hp}_C, L'_{\pi_{S,C}}, c_S, r_S)$$

(for the client and the server) by two independent truly random values. Thanks to the perfect smoothness of the SPHFs, $\mathsf{Hash}(\mathsf{hk}_C, L'_{\pi_{S,C}}, c_S)$ and $\mathsf{Hash}(\mathsf{hk}_S, L_{\pi_C}, (\ell, c_C))$ are completely independent random values, since the passwords are different. And we have $\mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A})$.

Game $\mathbf{G}_5$: Since the random coins are not needed anymore for these `Execute`-queries between two incompatible users, we replace c_S and c_C by encryptions of the dummy passwords 0_S and 0_C. This is indistinguishable from $\mathbf{G}_4$ under the `IND-CPA` property of the encryption schemes, for each `Execute`-query between two incompatible users. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A})| \leq q_{e,2} \times (\mathsf{Adv}^{\texttt{ind-cpa}}_{\mathsf{ES}}(t) + \mathsf{Adv}^{\texttt{ind-cpa}}_{\mathsf{ES}'}(t))$, where $q_{e,2}$ is the number of `Execute`-queries between two incompatible users.

Game $\mathbf{G}_6$: To simplify the following games, we now consider the games in which a collision appears between flows (hp_C, c_C) generated via an `Execute`-query and a $\texttt{Send}_0$-query: we set Coll to `True`, and stop the simulation. Since this makes the adversary win, this change can only increase the advantage of the adversary. Therefore, we have: $\mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A})$.

Game $\mathbf{G}_7$: We now modify the way the $\texttt{Send}_1$-queries are answered, by using a `PCA` oracle on ES, or alternatively knowing the decryption key sk. More precisely, when a message (hp_C, c_C) is sent, in the name of some client instance C^i and to some server instance S^j, four cases can happen:

- it is not a message generated by the simulator in the name of this client C (via a $\texttt{Send}_0$-query or an `Execute`-query), then we first check whether c_C contains the expected password $\pi_{S,C}$ or not, with the label $\ell = (C, S, \mathsf{hp}_C)$:
  1. if the expected password is encrypted, then we set Win to `True`, and stop the simulation;
  2. otherwise, we choose the session key K at random;
- it is a message generated in the name of some $C^{i'}$ (for the same client C):
  3. if S and C are compatible, we know the associated hk_C, so we can compute H'_S using hk_C (as H_S, and so without the random coins r_S of c_S). They both come up with the same key K;
  4. otherwise, we choose a random session key K.

The change in the first case can only increase the advantage of the adversary, while the changes in the second and fourth cases are indistinguishable under the perfect smoothness of the GLSPHF. The correctness of the KVSPHF implies the indistinguishability of the third case. Therefore, we have: $\mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A})$.

Game $\mathbf{G}_8$: Since the random coins r_S are not needed anymore by the simulated servers to compute hash values, we replace c_S by an encryption of the dummy password 0_S (up to the corruption of π_C or $\pi_{S,C}$). This is indistinguishable from $\mathbf{G}_7$ under the `IND-CPA` property of the encryption scheme ES'. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A})| \leq q_{s1} \times \mathsf{Adv}^{\texttt{ind-cpa}}_{\mathsf{ES}'}(t)$, where q_{s1} is the number of $\texttt{Send}_1$-queries.

Game $\mathbf{G}_9$: We now modify the way the $\mathtt{Send}_2$-queries are answered, knowing the decryption key sk'. More precisely, when a message (hp_S, c_S) is sent, in the name of some server instance S^i and to some simulated client instance C^j that previously answered the $\mathtt{Send}_0$-query by (hp_C, c_C), four cases can appear (note we have already excluded collisions on (hp_C, c_C) via $\mathtt{Execute}$ and $\mathtt{Send}_0$-queries in $\mathbf{G}_6$):

- it is not a message generated by the simulator in the name of a server in response to the first flow (hp_C, c_C) sent by C^j, then we first decrypt the ciphertext to get the password π used by the adversary:
  1. if this is the expected password π_C, then we set Win to $\mathtt{True}$, and stop the simulation;
  2. otherwise, we choose the session key K at random;
- it is a message simulated in the name of some $S^{i'}$ (via a $\mathtt{Send}_1$-query) after receiving the first flow (hp_C, c_C), and thus C^j and $S^{i'}$ are partners:
  3. if S and C are compatible, we use the same key as $S^{i'}$;
  4. otherwise, we choose a random session key K.

The changes in the first case can only increase the advantage of the adversary, while the changes in the second and fourth cases are indistinguishable under the smoothness of the KVSPHF. The third case is identical. Therefore, we have: $\mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A})$.

We remark that H'_S can now be computed without using random coins of c_S. That was the goal of this game.

Game $\mathbf{G}_{10}$: In $\mathbf{G}_9$, we remark that we do not need to know the random coins r_C used by the ciphertext c_C generated in response to a $\mathtt{Send}_0$ query. So, we can simply encrypt the dummy password 0_C instead of the correct password π_C in all ciphertexts c_C, generated as responses to $\mathtt{Send}_0$ queries (up to the corruption of π_C). This is indistinguishable under the $\mathtt{IND\text{-}PCA}$ property of the encryption scheme ES, since we still need to be able to test the correct password encrypted by the adversary: the simulator thus knows the decryption key sk' of ES', but just has access to the $\mathtt{PCA}$ oracle for ES. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_{10}}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A})| \leq q_{s0} \times \mathsf{Adv}^{\mathtt{ind\text{-}pca}}_{\mathsf{ES}}(t)$, where q_{s0} is the number of $\mathtt{Send}_0$-queries.

Unfortunately, in the case of the corruption of π_C after the $\mathtt{Send}_0$-query, we are not able to generate the correct session key K_C, whereas the adversary might be able: using a $\mathtt{Reveal}$-query, he can detect this mistake by the simulator. To overcome this problem, we can simply guess which client-password will be involved in the $\mathtt{Test}$-query, and thus apply the above modifications only on flows from or to an instance of this client. We know such a password cannot get corrupted, otherwise the $\mathtt{Test}$-query answers $\perp$ whenever it is asked (since we are dealing with the basic freshness only), and so the advantage of the adversary is 0.

Game $\mathbf{G}_{11}$: In this final game, Win is initially set to $\mathtt{False}$, and we target a specific client C: for all the executions and flows not involving this client, we do the simulation with the correct password. However, we do not choose π_C from the beginning. We thus ignore the cases 1 during the game, simulating them as cases 2, and we will just check whether Win has to be set to $\mathtt{True}$ at the very end only, or when a corruption happens, using the decryption keys sk and sk':

- $\mathtt{Execute}$-queries: encryptions of the dummy passwords 0_C and 0_S are generated together with projection keys. If the users are compatible, they are given the same random key $K = K_C = K_S$. If they are incompatible, they are given two independent random keys K_C and K_S;
- $\mathtt{Send}_0$-queries: a projection key hp_C is generated, together with an encryption c_C of the dummy password 0_C;
- $\mathtt{Send}_1$-queries:
  - if this is not a message generated by the simulator in the name of a client, then we store the input (hp_C, c_C) in Λ_C;
  - in any case, a projection key hp_S is generated, together with an encryption c_S of the dummy password 0_S, and we choose a random session key K_S.
- $\mathtt{Send}_2$-queries:

- if this is not a message generated by the simulator in the name of a server in response to the first flow (hp_C, c_C), then we store the input (C, S, c_S) in Λ_S, and choose a random session key K_C;
- otherwise, if C and S are compatible, it gets the same keys as its partner, otherwise we choose a random session key K_C.

– $\texttt{Corrupt}(C)$ and $\texttt{Corrupt}(S, C, \perp)$-queries: one first chooses a random password π, and checks for all the tuples in Λ_C and Λ_S involving this client, whether one of them encrypts π. In such a case, we set Win to $\texttt{True}$.
– $\texttt{Corrupt}(S, C, \pi)$-query: set $\pi_{S,C} \leftarrow \pi$.
– $\texttt{Reveal}$-queries and the $\texttt{Test}$-query are answered using the defined session key, and according to the bit b and the freshness of instance.

In case of collision between the first flow of an $\texttt{Execute}$-answer and a $\texttt{Send}_0$-answer, we sent Coll to $\texttt{True}$. At the very end, all the undefined passwords are dealt as above: for each client, one chooses a random password π, and checks for all the tuples in Λ_C and Λ_S involving this client, whether one encrypts π. In such a case, one sets Win to $\texttt{True}$. If $b' = b$, $\mathsf{Win} = \texttt{True}$, or $\mathsf{Coll} = \texttt{True}$, we say the adversary has won.

– Since the passwords are chosen at random, but never used, the probability for Win to be set to $\texttt{True}$ is bounded by $(q_{s0} + q_{s1}) \times 2^{-m}$, where m is the entropy of the passwords;
– The probability to generate two identical first flows is upper-bounded by $q_e q_{s0}/2^{2n}$, where n is the entropy of both the projected keys and the ciphertexts, and q_{s0} is the number of $\texttt{Send}_0$-queries;
– Since all the session keys are chosen uniformly at random, in all the other cases, the advantage is 0.

Hence, $\mathsf{Adv}_{\mathbf{G}_{11}}(\mathcal{A}) \leq q_s \times 2^{-m} + q_e q_{s0}/2^{2n}$.

In conclusion, the global advantage of any adversary against a specific client C in the PAKE protocol is bounded by

$$q_s \times 2^{-m} + (q_e + q_s) \times (\mathsf{Adv}^{\texttt{ind-cpa}}_{\mathsf{ES}'}(t) + \mathsf{Adv}^{\texttt{ind-pca}}_{\mathsf{ES}}(t)) + \frac{q_e q_{s0}}{2^{2n}},$$

where q_e and q_s are queries involving C. By summing on all the clients, one gets

$$\mathsf{Adv}_{\mathbf{G}_{\mathrm{real}}}(\mathcal{A}) \leq q_s \times 2^{-m} + (q_e + q_s) \times (\mathsf{Adv}^{\texttt{ind-cpa}}_{\mathsf{ES}'}(t) + \mathsf{Adv}^{\texttt{ind-pca}}_{\mathsf{ES}}(t)) + \frac{q_e q_{s0}}{2^{2n}},$$

where q_e and q_s are the number of $\texttt{Execute}$ and $\texttt{Send}$-queries, and n is the entropy of both the projected keys and the ciphertexts.

Forward-Secrecy and Multiple $\texttt{Test}$-Queries. One can note that the forward-secrecy notion achieved by the KOY protocol [KOY02, KOY09] is essentially the above one, since in their proof they assume that, once a user is corrupted, all the sessions involving this user are unfresh (even before the corruption). But this is a quite weak notion.

The above proof extends in a straightforward way to the forward-secure freshness setting (with all the clients at once) if we assume static corruptions only: no corruption of U is allowed during the execution of a protocol involving an instance of U. In addition, in the case of static corruption, our proof also extends to multiple $\texttt{Test}$-queries (the AFP setting).

Acknowledgments

This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project, the CFM Foundation, and the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

References

ABB+13. M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, and D. Pointcheval. Sphf-friendly non-interactive commitments. Cryptology ePrint Archive, Report 2013/588, 2013. http://eprint.iacr.org/. (Page 2.)

ACP09. M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. In *CRYPTO 2009*, *LNCS* 5677, pages 671–689. Springer, August 2009. (Page 2.)

AFP05. M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In *PKC 2005*, *LNCS* 3386, pages 65–84. Springer, January 2005. (Page 3.)

AP05. M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In *CT-RSA 2005*, *LNCS* 3376, pages 191–208. Springer, February 2005. (Page 1.)

AP06. M. Abdalla and D. Pointcheval. A scalable password-based group key exchange protocol in the standard model. In *ASIACRYPT 2006*, *LNCS* 4284, pages 332–347. Springer, December 2006. (Page 2.)

BBC+13. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 449–475. Springer, August 2013. (Pages 2, 3, 4, 5, and 7.)

BBP04. M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *EUROCRYPT 2004*, *LNCS* 3027, pages 171–188. Springer, May 2004. (Page 1.)

BCP03. E. Bresson, O. Chevassut, and D. Pointcheval. Security proofs for an efficient password-based key exchange. In *ACM CCS 03*, pages 241–250. ACM Press, October 2003. (Page 1.)

BCP04. E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. In *PKC 2004*, *LNCS* 2947, pages 145–158. Springer, March 2004. (Page 1.)

BDPR98. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO'98*, *LNCS* 1462, pages 26–45. Springer, August 1998. (Page 2.)

BGS06. J.-M. Bohli, M. I. Gonzalez Vasco, and R. Steinwandt. Password-authenticated constant-round group key establishment with a common reference string. Cryptology ePrint Archive, Report 2006/214, 2006. http://eprint.iacr.org/. (Page 2.)

BM92. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. (Page 1.)

BM93. S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM CCS 93*, pages 244–250. ACM Press, November 1993. (Page 1.)

BMP00. V. Boyko, P. D. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *EUROCRYPT 2000*, *LNCS* 1807, pages 156–171. Springer, May 2000. (Page 1.)

BPR00. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, *LNCS* 1807, pages 139–155. Springer, May 2000. (Pages 1 and 3.)

BR95. M. Bellare and P. Rogaway. Provably secure session key distribution: The three party case. In *27th ACM STOC*, pages 57–66. ACM Press, May / June 1995. (Page 1.)

Can01. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. (Page 2.)

CDSMW09. S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Simple, black-box constructions of adaptively secure protocols. In *TCC 2009*, *LNCS* 5444, pages 387–402. Springer, March 2009. (Page 2.)

CGH98. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. (Page 1.)

CHK+05. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In *EUROCRYPT 2005*, *LNCS* 3494, pages 404–421. Springer, May 2005. (Page 2.)

CS98. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98*, *LNCS* 1462, pages 13–25. Springer, August 1998. (Page 6.)

CS02. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002*, *LNCS* 2332, pages 45–64. Springer, April / May 2002. (Pages 2 and 4.)

DH76. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Page 1.)

ElG84. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*, *LNCS* 196, pages 10–18. Springer, August 1984. (Page 6.)

Gen08. R. Gennaro. Faster and shorter password-authenticated key exchange. In *TCC 2008*, *LNCS* 4948, pages 589–606. Springer, March 2008. (Page 2.)

GK03. S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003. (Page 1.)

GK10. A. Groce and J. Katz. A new framework for efficient password-based authenticated key exchange. In *ACM CCS 10*, pages 516–525. ACM Press, October 2010. (Pages 2 and 3.)

GL01. O. Goldreich and Y. Lindell. Session-key generation using human passwords only. In *CRYPTO 2001*, *LNCS* 2139, pages 408–432. Springer, August 2001. http://eprint.iacr.org/2000/057. (Page 2.)

GL03. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT 2003*, *LNCS* 2656, pages 524–543. Springer, May 2003. http://eprint.iacr.org/2003/032.ps.gz. (Pages 2, 4, and 10.)

Jab97. D. P. Jablon. Extended password key exchange protocols immune to dictionary attacks. In *6th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 1997)*, pages 248–255, Cambridge, MA, USA, June 18–20, 1997. IEEE Computer Society. (Page 1.)

JG04. S. Jiang and G. Gong. Password based key exchange with mutual authentication. In *SAC 2004*, *LNCS* 3357, pages 267–279. Springer, August 2004. (Pages 2 and 3.)

KMTG05. J. Katz, P. D. MacKenzie, G. Taban, and V. D. Gligor. Two-server password-only authenticated key exchange. In *ACNS 05*, *LNCS* 3531, pages 1–16. Springer, June 2005. (Page 2.)

KOY01. J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT 2001*, *LNCS* 2045, pages 475–494. Springer, May 2001. (Pages 2 and 4.)

KOY02. J. Katz, R. Ostrovsky, and M. Yung. Forward secrecy in password-only key exchange protocols. In *SCN 02*, *LNCS* 2576, pages 29–44. Springer, September 2002. (Pages 2, 10, and 13.)

KOY09. J. Katz, R. Ostrovsky, and M. Yung. Efficient and secure authenticated key exchange using weak passwords. *Journal of the ACM*, 57(1):78–116, 2009. (Pages 2, 10, and 13.)

KV11. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *TCC 2011*, *LNCS* 6597, pages 293–310. Springer, March 2011. (Page 2.)

Luc97. S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Workshop on Security Protocols*, École Normale Supérieure, 1997. (Page 1.)

Mac02. P. D. MacKenzie. The PAK suite: Protocols for password-authenticated key exchange. Contributions to IEEE P1363.2, 2002. (Page 1.)

Nie02. J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO 2002*, *LNCS* 2442, pages 111–126. Springer, August 2002. (Page 1.)

OP01. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT-RSA 2001*, *LNCS* 2020, pages 159–175. Springer, April 2001. (Pages 2 and 5.)

PasV06. R. Pass, abhi shelat, and V. Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *CRYPTO 2006*, *LNCS* 4117, pages 271–289. Springer, August 2006. (Page 2.)

Sho99. V. Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 1999. (Page 1.)

STW95. M. Steiner, G. Tsudik, and M. Waidner. Refinement and extension of encrypted key exchange. *ACM SIGOPS Operating Systems Review*, 29(3):22–30, July 1995. (Page 1.)