This is the Full Version of the Extended Abstract that appears in
Public-Key Cryptography — PKC 2014 (March 30 – April 1, Maryland, USA)
Springer-Verlag, LNCS ????, pages ???–???.

Public-Key Encryption Indistinguishable Under Plaintext-Checkable Attacks

Michel Abdalla, Fabrice Benhamouda, and David Pointcheval

ENS, Paris, France [⋆]

January 6, 2015

Abstract. Indistinguishability under adaptive chosen-ciphertext attack (IND-CCA) is now considered the *de facto* security notion for public-key encryption. However, the security guarantee that it offers is sometimes stronger than what is needed by certain applications. In this paper, we consider a weaker notion of security for public-key encryption, termed indistinguishability under plaintext-checking attacks (IND-PCA), in which the adversary is only given access to an oracle which says whether or not a given ciphertext encrypts a given message. After formalizing the IND-PCA notion, we then design a new public-key encryption scheme satisfying it. The new scheme is a more efficient variant of the Cramer-Shoup encryption scheme with shorter ciphertexts and its security is also based on the plain Decisional Diffie-Hellman (DDH) assumption. Additionally, the algebraic properties of the new scheme also allow for proving plaintext knowledge using Groth-Sahai non-interactive zero-knowledge proofs or smooth projective hash functions. Finally, in order to illustrate the usefulness of the new scheme, we further show that, for many password-based authenticated key exchange (PAKE) schemes in the Bellare-Pointcheval-Rogaway security model, one can safely replace the underlying IND-CCA encryption schemes with our new IND-PCA one. By doing so, we were able to reduce the overall communication complexity of these protocols and obtain the most efficient PAKE schemes to date based on the plain DDH assumption.

1 Introduction

Public-key encryption (PKE) is one of the most fundamental primitives in cryptography, allowing users to exchange messages privately without the need for pre-established secrets. The basic security notion for (probabilistic) public-key encryption is indistinguishability of encryptions under chosen-plaintext attacks (IND-CPA) [GM84], also known as semantic security. Informally speaking, this notion states that any passive adversary capable of eavesdropping on the communication between two parties should not be able to obtain any information about the encrypted messages.

While IND-CPA security may suffice for certain applications, it does not provide any guarantee against active attacks, in which the adversary may modify existing ciphertexts or inject new ones into the communication and obtain information about the decrypted messages. In fact, as shown by Bleichenbacher [Ble98] in his attack against RSA PKCS #1, one can sometimes break an existing PKE scheme simply by knowing whether an existing ciphertext is valid or not.

In order to address the problem of active attacks, several notions of security have been proposed, such as indistinguishability under non-adaptive chosen-ciphertext attack (IND-CCA1) [NY90], indistinguishability under adaptive chosen-ciphertext attack (IND-CCA2 or IND-CCA) [RS91], non-malleability under chosen-plaintext attack (NM-CPA) or adaptive chosen-ciphertext attack (NM-CCA) [DDN91, DDN00]. Among these, as shown by Bellare *et al.* [BDPR98], the IND-CCA notion is the strongest one and implies all of the other ones. Unlike the IND-CPA notion, the IND-CCA security notion states that the adversary should not be capable to learning any information about the underlying message of a given ciphertext even when given access to the decryption of other ciphertexts of its choice.

[⋆] CNRS – UMR 8548 and INRIA

Indistinguishabiliy under Plaintext-Checkable Attacks. Even though IND-CCA is now considered the *de facto* security notion for public-key encryption, the security guarantee that it offers is sometimes stronger than what is needed by certain applications. Since stronger security guarantees usually result in a loss of efficiency, different security goals, such as oneway-ness, and different attack capabilities, such as plaintext-checkable attacks [OP01], have been considered as alternatives to the IND-CCA security notion. While in oneway-ness, the goal of the adversary is to recover the underlying encrypted message, in plaintext-checkable attacks, the adversary is given access to a plaintext-checking oracle that answers, on a given pair (m, c), whether c encrypts m or not.

In this paper, we first revisit the notion of oneway-ness under plaintext-checkable attacks (OW-PCA) by Okamoto and Pointcheval [OP01] and describe an indistinguishability-based variant for it. In the new notion, termed indistinguishability under plaintext-checkable attacks (IND-PCA), the adversary should not be able to learn any information about an encrypted message even when given access to a plaintext-checking oracle. As we show in Section 2, the new notion is also equivalent to the IND-CCA notion when the message space is small (polynomial in the security parameter) since it is possible to enumerate all the possible messages in this case.

A new IND-PCA encryption scheme. After defining the IND-PCA notion, our first main contribution is to design a new public-key encryption scheme which formally meets the new notion. The new scheme is a more efficient variant of the Cramer-Shoup encryption scheme [CS98], whose ciphertext consists of only 3 group elements. Like the Cramer-Shoup encryption scheme, the security of new scheme is also based on the plain Decisional Diffie-Hellman (DDH) assumption [NR97].

In addition to being quite efficient, the new scheme can also be used with Groth-Sahai Non-Interactive Zero-Knowledge Proofs [GS08] and smooth projective hash functions (SPHF) [CS02], for proving plaintext knowledge. To illustrate this fact, we design two different constructions of SPHFs for the new scheme, each providing a different security-efficiency trade-off.

Since IND-PCA implies IND-CCA for short messages, the new scheme can also replace IND-CCA schemes in applications where the message space is small. This is the case, for instance, when bits have to be encrypted as in [ACP09].

Applications to PAKE. After proposing the new scheme, our second main contribution is to show that, for many password-based authenticated key exchange (PAKE) in the Bellare-Pointcheval-Rogaway (BPR) security model [BPR00], one can safely replace the underlying IND-CCA encryption schemes with an IND-PCA one. In particular, we revisit the frameworks by Gennaro and Lindell [GL03], by Groce and Katz [GK10], and by Katz and Vaikuntanathan [KV11], and show that one can replace the underlying IND-CCA encryption schemes in their constructions with an IND-PCA encryption scheme. In all of these cases, we were able to reduce the overall communication complexity of the original protocols by at least one 1 group element.

More precisely, in the case of the Gennaro-Lindell framework [GL03], which is a generalization of the PAKE scheme by Katz, Ostrovsky, and Yung [KOY01], we were able to obtain a quite clean 2-flow protocol with 7 group elements in total, instead of 8 group elements in 3 flows [KMTG05] or 10 group elements in 3 flows in [KOY02, Gen08, KOY09]. The security of the new scheme is based on the DDH in the underlying group and assumes a trusted common reference string (CRS). In addition to avoiding the use of IND-CCA encryption schemes, our instantiation also avoids the use of one-time signatures and message authentication codes. 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.

In the case of the Groce-Katz (GK) framework [GK10], which is a generalization of the PAKE scheme by Jiang and Gong [JG04] that additionally provides mutual authentication, we were able to obtain a scheme with a total communication complexity of 7 group elements instead of the original 8 by using an IND-PCA encryption scheme to generate the second flow. 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. The security of the new scheme is based on the plain DDH assumption and on the security of the underlying pseudorandom number generator and assumes a trusted CRS.

Finally, in the case of Katz-Vaikuntanathan (KV) framework [KV11], we were able to obtain a PAKE scheme with a total communication complexity of 10 group elements instead of the current 12 in [BBC+13b]. As in [KV11, BBC+13b], our new scheme only has a single round of communication and assumes a trusted CRS. Its security proof is based on the plain DDH assumption.

Organization. Section 2 recalls standard definitions for public-key encryption and smooth projective hash proof functions (SPHFs) and describes some of the most classic instantiations of these primitives. Section 3 introduces our new IND-PCA encryption scheme and the associated SPHFs along with its security proof. The new scheme is a variant of the Cramer-Shoup encryption scheme [CS98] with shorter ciphertexts. Section 4 presents the security models for password-based authenticated key exchange (PAKE) used in our security proofs. Section 5 describes three PAKE constructions based on the frameworks by Gennaro and Lindell [GL03], by Groce and Katz [GK10], and by Katz and Vaikuntanathan [KV11], whose security proofs appear in Appendix A. Appendix B provides a brief overview of PAKE for completeness.

2 Public-Key Encryption

2.1 Definition

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;
- $\mathsf{Enc}^{\ell}(\mathsf{pk}, M; r)$ encrypts the message M 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 correctness requires that for all key pairs $(\mathsf{pk}, \mathsf{sk})$, all labels ℓ, all random coins r and all messages M,

$$\mathsf{Dec}^{\ell}(\mathsf{sk}, \mathsf{Enc}^{\ell}(\mathsf{pk}, M; r)) = M.$$

The main security notion is the so-called *indistinguishability of ciphertexts*, depicted in Fig. 1, in which the adversary chooses two messages M_0 and M_1 and a label ℓ^* (FIND phase), and then has to guess which of the two has been encrypted in the challenge ciphertext $C^* = \mathsf{Enc}^{\ell^*}(\mathsf{pk}, M_b; r)$ for a random bit b (GUESS phase). The adversary has access to an oracle ORACLE which may update some list of forbidden challenges CTXT, and it wins if and only if he guessed correctly the bit b (i.e., it outputs $b' = b$) and (ℓ^*, C^*) is not in CTXT. The advantages are:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{ind}}_{\mathsf{ES}}(\mathcal{A}) &= \Pr[\mathsf{Exp}^{\mathtt{ind}-1}_{\mathsf{ES},\mathcal{A}}(\mathfrak{K}) = 1] - \Pr[\mathsf{Exp}^{\mathtt{ind}-0}_{\mathsf{ES},\mathcal{A}}(\mathfrak{K}) = 1]\\ \mathsf{Adv}^{\mathtt{ind}}_{\mathsf{ES}}(t, q) &= \max_{\mathcal{A} \le t, q}\{\mathsf{Adv}^{\mathtt{ind}}_{\mathsf{ES}}(\mathcal{A})\},\end{aligned}$$

where $\mathcal{A} \le t, q$ are adversaries running within time t and asking at most q queries to ORACLE.

Depending on the definition of ORACLE, one gets three different security notions:

- if ORACLE $=\perp$, the adversary just has access to the public key, and one gets the IND-CPA notion, CPA meaning *Chosen-Plaintext Attack*;

```
$\mathsf{Exp}^{\mathtt{ind}-b}_{\mathsf{ES},\mathcal{A}}(\mathfrak{K})$
    $\mathsf{CTXT} \leftarrow$ empty list
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(\mathfrak{K})$
    $(\ell^*, M_0, M_1) \leftarrow \mathcal{A}(\mathsf{FIND} : \mathsf{pk}, \mathtt{ORACLE}(\cdot))$
    $C^* \leftarrow \mathsf{Enc}^{\ell^*}(\mathsf{pk}, M_b)$
    $b' \leftarrow \mathcal{A}(\mathsf{GUESS} : C^*, \mathtt{ORACLE}(\cdot))$
    **if** $(\ell^*, C^*) \in \mathsf{CTXT}$ **then return** $0$
    **else return** $b'$
```

Fig. 1. Indistinguishability Security Notions for Labeled Public-Key Encryption (IND-CPA when ORACLE $=\perp$, IND-PCA when ORACLE $=$ OPCA, and IND-CCA when ORACLE $=$ OCCA)

$$\begin{tikzcd}
\text{IND-CPA} \arrow[r, bend right, "/" marking, "\text{counter-example: ElGamal}"'] & \text{IND-PCA} \arrow[l, bend right] \arrow[r, bend right, dashed, "\text{for small messages}"'] & \text{IND-CCA} \arrow[l, bend right]
\end{tikzcd}$$

Fig. 2. Relations between IND-CPA, IND-PCA, and IND-CCA (normal arrows are implications, strike out arrows are separations)

- if ORACLE(ℓ, C) outputs the decryption of C under the label ℓ $(\mathsf{Dec}^{\ell}(\mathsf{sk}, C))$ and adds (ℓ, C) to CTXT, one gets the IND-CCA notion, CCA meaning *Chosen-Ciphertext Attack*;
- if ORACLE(ℓ, C, M) just answers whether the decryption of C under the label ℓ is M and adds (ℓ, C) to CTXT, one gets the IND-PCA notion, PCA meaning *Plaintext-Checking Attack*, as proposed in [OP01].

2.2 Relations with the IND-CPA and IND-CCA Security Notions

It is well known that IND-CCA implies IND-CPA (i.e., an encryption scheme IND-CCA-secure is IND-CPA-secure), and it is clear that IND-PCA implies IND-CPA. Let us now show that relations between IND-CPA, IND-PCA, IND-CCA are as depicted in Fig. 2. In all this paper, when we speak of small messages, we mean that it is possible to enumerate all the possible messages (i.e., the message space has a cardinal polynomial in the security parameter).

IND-CCA $\implies$ IND-PCA. One just has to remark that the OPCA oracle can be simulated by the OCCA oracle, and the restrictions are compatible (the same list CTXT will be generated): given a query (ℓ, M, C) to the OPCA oracle, the simulator can simply ask for (ℓ, C) to the OCCA oracle. This perfectly simulates the OPCA oracle.

IND-PCA $\implies$ IND-CCA, for Small Messages. In case of small messages for the encryption scheme, we remark that the OCCA oracle can be simulated by the OPCA oracle, and the restrictions are compatible too: given a query (ℓ, C) to the OCCA oracle, the simulator can simply ask for (ℓ, M, C) to the OPCA oracle, for all the messages M (we insist that by small messages, we mean we can enumerate them in polynomial time).If no message M matches, the simulator outputs $\perp$, otherwise it outputs the unique matching message (since the encryption is perfectly binding, at most one message can match). This perfectly simulates the OCCA oracle.

2.3 Classical Schemes

ElGamal Encryption Scheme [ElG84]. The ElGamal (EG) encryption scheme is defined as follows, in a cyclic group $\mathbb{G}$ of prime order p, with a generator g:

- $\mathsf{EG.KG}(1^{\mathfrak{K}})$ 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, which states that it is hard to distinguish a Diffie-Hellman tuple (g^a, g^b, g^{ab}) from a random tuple (g^a, g^b, g^c), for random scalars $a, b, c \xleftarrow{\$} \mathbb{Z}_q$:

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

Cramer-Shoup Encryption Scheme [CS98]. The labeled Cramer-Shoup (CS) encryption scheme is defined as follows, in a cyclic group $\mathbb{G}$ of prime order p, with two generators g_1, g_2, 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 $\mathbb{G}\backslash\{1\}$:

- $\mathsf{CS.KG}(1^{\mathfrak{K}})$ generates the secret key $\mathsf{sk} = (s, a, b, a', b') \xleftarrow{\$} \mathbb{Z}_p$ and the public key $\mathsf{pk} = (h = g_1^s, c = g_1^a g_2^b, d = g_1^{a'} g_2^{b'})$;
- $\mathsf{CS.Enc}^\ell(\mathsf{pk} = (h, c, d), M; r)$, for a label ℓ, a group element $M \in \mathbb{G}$ and a scalar $r \in \mathbb{Z}_p$, generates the ciphertext $C = (u_1 = g_1^r, u_2 = g_2^r, e = h^r M, v = (cd^\xi)^r)$, where $\xi = H_{\mathsf{CS}}(\ell, u_1, u_2, e)$;
- $\mathsf{CS.Dec}^\ell(\mathsf{sk} = (s, a, b, a', b'), C = (u_1, u_2, e, v))$ first checks whether $v = u_1^{a+\xi a'} \cdot u_2^{b+\xi b'}$, for $\xi = H_{\mathsf{CS}}(\ell, u_1, u_2, e)$. If the equality holds, it outputs $M = e/u_1^s$, otherwise it outputs $\perp$.

This encryption scheme is well-known to be IND-CCA under the DDH assumption and the collision-resistance of the hash function family:

$$\mathsf{Adv}_{\mathsf{CS}}^{\mathtt{ind\text{-}cca}}(t, q_d) \leq 2\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t) + \mathsf{Succ}_{\mathcal{H}}^{\mathsf{coll}}(t) + 3q_d/p,$$

where q_d is the number of queries to the OCCA oracle.

Remark 1. 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 $\mathsf{Succ}_{\mathcal{H}}^{\mathsf{coll}}(t)$ the best success probability any adversary can get within time t.

2.4 Smooth Projective Hash Functions

Projective hash function families were first introduced by Cramer and Shoup [CS02]. Here we use the formalization from [BBC+13b]: 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. More precisely, a smooth projective hash function (SPHF) over $L \subseteq X$ is defined by four algorithms.

- $\mathsf{HashKG}(L)$ 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.

[1] Second-preimage resistance is actually sufficient.

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 ProjKG does 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. A KVSPHF is stronger than a GLSPHF (in particular, a KVSPHF is a GLSPHF), and some applications require KVSPHF.

More precisely, if ProjKG does not use C and, if for any function from the set of projection keys to $X \setminus L$, on the probability space $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(L)$, $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, \perp)$, the distributions $\{(\mathsf{hp}, H) \mid H \leftarrow \mathsf{Hash}(\mathsf{hk}, L, C)\}$ and $\{(\mathsf{hp}, H) \mid H \overset{\$}{\leftarrow} \Pi\}$ are ε-close, where Π is the output set of the hash function, then the SPHF is an ε-smooth KVSPHF. If ProjKG uses C (or not) and if, for any $C \notin L$, on the probability space $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(L)$, $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C)$, the distributions $\{(\mathsf{hp}, H) \mid H \leftarrow \mathsf{Hash}(\mathsf{hk}, L, C)\}$ and $\{(\mathsf{hp}, H) \mid H \overset{\$}{\leftarrow} \Pi\}$ are ε-close, then the SPHF is an ε-smooth GLSPHF. See [BBC+13b] for more details on GLSPHF and KVSPHF.

Let us now recall SPHFs for the ElGamal and Cramer-Shoup encryption schemes, proposed in [CS02, GL03, BBC+13b].

ElGamal Encryption Scheme. EG admits 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:

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

Cramer-Shoup Encryption Scheme. CS admits an efficient GLSPHF for the language $L_M^\ell = \{C \mid \exists r,\ C = \mathsf{CS.Enc}^\ell(\mathsf{pk}, M; r)\}$, with L the superset of the ciphertexts:

$$\mathsf{hk} = \mathsf{HashKG}(L) = (\alpha, \beta, \gamma, \delta) \overset{\$}{\leftarrow} \mathbb{Z}_p^4 \qquad \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, L, C) = g_1^\alpha g_2^\beta h^\gamma (c d^\xi)^\delta$$
$$H = \mathsf{Hash}(\mathsf{hk}, L_M^\ell, C) = u_1^\alpha u_2^\beta (e/M)^\gamma v^\delta \qquad H' = \mathsf{ProjHash}(\mathsf{hp}, L_M^\ell, C, r) = \mathsf{hp}^r,$$

where $\xi = H_{\mathsf{CS}}(\ell, u_1, u_2, e)$.

CS also admits an efficient KVSPHF for the language L_M^ℓ [BBC+13b]:

$$\begin{aligned}
\mathsf{hk} &= \mathsf{HashKG}(L) = (\alpha_1, \alpha_2, \beta, \gamma, \delta) \overset{\$}{\leftarrow} \mathbb{Z}_p^5\\
\mathsf{hp} &= \mathsf{ProjKG}(\mathsf{hk}, L, C) = (\mathsf{hp}_1 = g_1^{\alpha_1} g_2^\beta h^\gamma c^\delta, \mathsf{hp}_2 = g_1^{\alpha_2} d^\delta)\\
H &= \mathsf{Hash}(\mathsf{hk}, L_M^\ell, C) = u_1^{\alpha_1 + \xi\alpha_2} u_2^\beta (e/M)^\gamma v^\delta\\
H' &= \mathsf{ProjHash}(\mathsf{hp}, L_M^\ell, C, r) = (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r,
\end{aligned}$$

where $\xi = H_{\mathsf{CS}}(\ell, u_1, u_2, e)$.

3 The Short Cramer-Shoup Encryption Scheme

The labeled Short Cramer-Shoup (SCS) encryption scheme is a variant of the above Cramer-Shoup 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 g, together with a hash function H_{SCS} randomly drawn from a collision-resistant[2] 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 $\mathsf{sk} = (s, a, b, a', b') \overset{\$}{\leftarrow} \mathbb{Z}_p$ 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 $C = (u = g^r, e = h^r M, v = (c d^\xi)^r)$, where $\xi = H_{\mathsf{SCS}}(\ell, u, e)$;

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

– $\mathsf{SCS.Dec}^\ell(\mathsf{sk} = (s, a, b, a', b'), C = (u, e, v))$ first computes $M = e/u^s$ and checks whether $v = u^{a+\xi a'}(e/M)^{b+\xi b'}$, for $\xi = H_{\mathsf{SCS}}(\ell, u, e)$. 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:

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

where q_p is the number of queries to the `OPCA` oracle. But before that, we build a GLSPHF and a KVSPHF for the SCS scheme.

3.1 Smooth Projective Hash Functions

Let us now define smooth projective hash functions. We use the formalization from [BBC+13b] to explain where these SPHFs come from. The reader not acquainted with it may skip the definitions via matrix/vectors and just look at the resulting GLSPHF and KVSPHF.

GLSPHF. The following matrix and vectors lead to an SPHF for the language $L^\ell_M = \{C \mid \exists r, C = \mathsf{SCS.Enc}^\ell(\mathsf{pk}, M; r)\}$, with L the superset of the ciphertexts:

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

where $\xi = H_{\mathsf{SCS}}(\ell, u, e)$. The matrix Γ depends on ξ, and thus on the word C. Hence, this is a GLSPHF:

$$\mathsf{hk} = \mathsf{HashKG}(L) = (\alpha, \beta, \gamma) \xleftarrow{\$} \mathbb{Z}_p^3 \qquad \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, L, C) = g^\alpha h^\beta (cd^\xi)^\gamma$$
$$H = \mathsf{Hash}(\mathsf{hk}, L^\ell_M, C) = u^\alpha (e/M)^\beta v^\gamma \qquad H' = \mathsf{ProjHash}(\mathsf{hp}, L^\ell_M, C, r) = \mathsf{hp}^r$$

KVSPHF. We could also use the following matrix and vectors:

$$\Gamma(C) = \begin{pmatrix} g & 1 & h & c \\ 1 & g & 1 & d \end{pmatrix} \qquad \begin{array}{rcl} \boldsymbol{\lambda} & = & (r) \\ \boldsymbol{\lambda} \cdot \Gamma & = & (g^r, g^{\xi r}, h^r, (cd^\xi)^r) \\ \Theta(C) & = & (u, u^\xi, e/M, v) \end{array}$$

where $\xi = H_{\mathsf{SCS}}(\ell, u, e)$. The matrix Γ does not depend anymore on ξ, nor on the word C in general. Hence, this is a KVSPHF:

$$\begin{aligned} \mathsf{hk} &= \mathsf{HashKG}(L) = (\alpha_1, \alpha_2, \beta, \gamma) \xleftarrow{\$} \mathbb{Z}_p^4 \\ \mathsf{hp} &= \mathsf{ProjKG}(\mathsf{hk}, L, C) = (\mathsf{hp}_1 = g^{\alpha_1} h^\beta c^\gamma, \mathsf{hp}_1 = g^{\alpha_2} d^\gamma) \\ H &= \mathsf{Hash}(\mathsf{hk}, L^\ell_M, C) = u^{\alpha_1 + \alpha_2 \xi} (e/M)^\beta v^\gamma \\ H' &= \mathsf{ProjHash}(\mathsf{hp}, L^\ell_M, C, r) = (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \end{aligned}$$

3.2 `IND-PCA` Security Proof

Let us now prove the `IND-CPA` security as advertised at the beginning of this section. We first recall the security game in Game $\mathbf{G}_0$, and present a series of indistinguishable games to show the advantage of the adversary is negligible [BR04, Sho04].

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 $\mathsf{sk} = (s, a, b, a', b') \xleftarrow{\$} \mathbb{Z}_p^5$, as well as an unlimited access to an $\mathtt{OPCA}$ 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 $\mathtt{OPCA}$ oracle, the adversary outputs a bit δ', its guess on the bit δ. Note that the adversary is not allowed to query the $\mathtt{OPCA}$ oracle on any tuple (ℓ^*, M, C^*).

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 $\xi^* = H_{\mathsf{SCS}}(\ell^*, u^*, e^*)$. The $\mathtt{OPCA}$ oracle, on input $(\ell, M, C = (u, e, v))$, unless $(\ell, C) = (\ell^*, C^*)$, checks both equations: $e \overset{?}{=} u^s M$ and $v \overset{?}{=} u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$, for $\xi = H_{\mathsf{SCS}}(\ell, u, e)$. Then, $\mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) = \mathsf{Adv}_{\mathsf{SCS}}^{\mathtt{ind\text{-}pca}}(\mathcal{A})$.

Game $\mathbf{G}_1$: In this game, we reject all queries $(\ell, M, C = (u, e, v))$ to the $\mathtt{OPCA}$ oracle, where $(\ell, u, e) \neq (\ell^*, u^*, e^*)$ but $\xi^* = \xi$. This game is computationally indistinguishable from the previous one under the collision-resistance of H_{SCS}: $|\mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A})| \leq \mathsf{Succ}_{\mathcal{H}}^{\mathsf{coll}}(t)$, where t is approximately the running time of $\mathcal{A}$.

Game $\mathbf{G}_2$: We first simplify the simulation of the $\mathtt{OPCA}$ oracle: it just checks the second equation: $v \overset{?}{=} u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$, for $\xi = H_{\mathsf{SCS}}(\ell, u, e)$, 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: this means that $e = u^{s'} M$ and $v = u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$, for $\xi = H_{\mathsf{SCS}}(\ell, u, e)$, 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 (actually upper-bounded by $1/p$) since a, a', b, b' are unknown. See a more complex case in Game $\mathbf{G}_6$, where even more information is available to the adversary. One thus gets $|\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})| \leq q_p/p$, where q_p is the number of queries to the $\mathtt{OPCA}$ 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 $\xi^* = H_{\mathsf{SCS}}(\ell^*, u^*, e^*)$. Since $z = xy$, we have a perfect simulation of v^* as in the previous game, hence $\mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A})$:

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

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. Hence we have: $|\mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A})| \leq 1/p$.

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 $\mathtt{OPCA}$ 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: this means that $e = u^{s'} M$ and $v = u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$, for $\xi = H_{\mathsf{SCS}}(\ell, u, e)$, with $h = g^s$ and $\Delta_s = s' - s \neq 0$.

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 have to show 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 than in Game $\mathbf{G}_2$, since more relations are available to the adversary. This proof would thus also apply for the Game $\mathbf{G}_2$. 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

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

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_{SCS}, the determinants are all non-zero, in which cases the expected values for v are unpredictable, hence $|\mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A})| \leq q_p/p$, where q_p is the number of queries to the `OPCA` oracle.

Game $\mathbf{G}_7$: We now choose v^* at random, independently of Y and Z.
To show this does not change anything, we first show that what $\mathcal{A}$ sees does never depend on the four variables a, b, a', b', but only depends on $\alpha = a + xb$ and $\beta = a' + xb'$, except for v^*: The only information $\mathcal{A}$ has from a, b, a', b' comes from the answers of the `OPCA` oracle, where we first check that $e \stackrel{?}{=} u^x M$ and then, if that equality holds, that $v \stackrel{?}{=} u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$. But when $e = u^x M$, $u^{a+\xi a'} \cdot (e/M)^{b+\xi b'} = u^{(a+\xi a')+x(b+\xi b')} = u^{(a+xb)+\xi(a'+xb')} = u^{\alpha+\xi\beta}$, therefore the second verification can be replaced by $v \stackrel{?}{=} u^{\alpha+\xi\beta}$, which only depends on α and β.
If we denote $v^* = g^\gamma$, we have $\gamma = y(a + \xi^* a) + z(b + \xi^* b')$, which is linearly independent of α and β (when a, a', b, b' are unknowns) since $z \neq xy$, and so γ looks completely random to the adversary, and so does v^* too: $\mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A})$.

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. Hence we have: $|\mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A})| \leq 1/p$.

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 $\delta \stackrel{\$}{\leftarrow} \{0, 1\}$;
– for the `OPCA` oracle on $(\ell, M, C = (u, e, v))$, one checks $e \stackrel{?}{=} u^x M$ and $v \stackrel{?}{=} u^{\alpha+\xi\beta}$, for $\xi = H_{\mathsf{SCS}}(\ell, u, e)$;
– for the challenge ciphertext, one sets $u^* \leftarrow Y$, $e^* \leftarrow ZM_\delta$, and $v^* \stackrel{\$}{\leftarrow} \mathbb{G}$.

Since Z was used in e^* only (and nowhere 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, hence $\mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A}) = 0$.

4 PAKE Security Models

In this section, we recall the BPR security model [BPR00] and the extension proposed by Abdalla, Fouque, and Pointcheval (AFP) [AFP05]. Then, in the next section, we will present several protocols secure in the basic BPR model, but also in the AFP model and with forward-secrecy.

4.1 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:

- `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);
- `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;
- `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.
- `Corrupt`(C): this query models the client corruption. The output of this query is the password π_C.
- `Corrupt`(S, C, π): 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 `Test`(U^i), available only once, 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 $\texttt{Reveal}$-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 $\texttt{Test}$-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 $\texttt{Send}$ queries), and m is the min-entropy of the password distribution. 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.

4.2 The Abdalla-Fouque-Pointcheval Security Model

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

5 PAKE Constructions

In this section, we present three PAKE constructions: the first one follows the Gennaro-Lindell (GL) framework [GL03]. The second one follows the Groce-Katz (GK) framework [GK10], and the third one follows the one-round Katz-Vaikuntanathan (KV) framework [KV11]. They all make use of public-key encryption schemes that admit SPHFs on the languages of the ciphertexts of a given message.

5.1 Public-Key Encryption Schemes

In all our constructions, we will consider a labeled $\texttt{IND-PCA}$ encryption scheme $\mathsf{ES} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and an $\texttt{IND-CPA}$ encryption scheme $\mathsf{ES}' = (\mathsf{KG}', \mathsf{Enc}', \mathsf{Dec}')$ so that SPHFs (either GLSPHFs or KVSPHFs according to the protocol) exist for the following families of languages:

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

with the global parameters and the public keys pk and pk' in the common reference string CRS. We also suppose that HashKG and ProjKG, for both L_π^ℓ and L'_π, do not depend on π nor ℓ, and thus, just (respectively) on the supersets

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

5.2 GL–PAKE Construction and GL–SPOKE

GL–PAKE. Our first two-flow construction is depicted in Fig. 3, 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). The session key generated by the client is denoted K_C, while the one generated by the server is denoted K_S.

It requires an $\texttt{IND-CPA}$ encryption scheme ES' with a KVSPHF, and an $\texttt{IND-PCA}$ encryption scheme ES with a GLSPHF. In Appendix A.1, we prove the following result, with perfectly-smooth SPHFs, which applies for the basic freshness in the BPR setting, or for the forward-secure freshness in the AFP setting with static corruptions only:

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

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

Client C (π_C)	CRS: $(\text{parameters}, \mathsf{pk}, \mathsf{pk}')$	Server S $(\pi_{S,C})$
$\mathsf{hk}_C \overset{\$}{\leftarrow} \mathsf{HashKG}(L')$		
$\mathsf{hp}_C \leftarrow \mathsf{ProjKG}(\mathsf{hk}_C, L', \perp)$		
$\ell = (C, S, \mathsf{hp}_C)$		
$r_C \overset{\$}{\leftarrow}$; $c_C \leftarrow \mathsf{Enc}^\ell(\mathsf{pk}, \pi_C; r_C)$	$\xrightarrow{\mathsf{hp}_C, c_C}$	$\ell = (C, S, \mathsf{hp}_C)$
		$r_S \overset{\$}{\leftarrow}$; $c_S \leftarrow \mathsf{Enc}'(\mathsf{pk}', \pi_{S,C}; r_S)$
		$\mathsf{hk}_S \overset{\$}{\leftarrow} \mathsf{HashKG}(L^\ell_{\pi_{S,C}})$
	$\xleftarrow{\mathsf{hp}_S, c_S}$	$\mathsf{hp}_S \leftarrow \mathsf{ProjKG}(\mathsf{hk}_S, L^\ell_{\pi_{S,C}}, c_C)$
$H'_C \leftarrow \mathsf{ProjHash}(\mathsf{hp}_S, L^\ell_{\pi_C}, c_C, r_C)$		$H'_S \leftarrow \mathsf{ProjHash}(\mathsf{hp}_C, L'_{\pi_{S,C}}, c_S, r_S)$
$H_S \leftarrow \mathsf{Hash}(\mathsf{hk}_C, L'_{\pi_C}, c_S)$		$H_C \leftarrow \mathsf{Hash}(\mathsf{hk}_S, L^\ell_{\pi_{S,C}}, c_C)$
$K_C \leftarrow H'_C \times H_S$		$K_S \leftarrow H'_S \times H_C$

Fig. 3. Generic GL–PAKE Construction

GL–SPOKE: GL – Simple Password-Only Key Exchange (Fig. 4). Combining our new Short Cramer-Shoup encryption scheme, with the basic ElGamal encryption scheme, we obtain the most efficient PAKE with implicit authentication.

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:

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

where q_s is the number of `Send`-queries, Q is the global number of oracle queries, and m is the min-entropy of the passwords.

Client C (π_C)	CRS: $(\mathsf{param} = (\mathbb{G}, p, q), \mathsf{pk} = (h, c, d) \in \mathbb{G}^3, \mathsf{pk}' = y \in \mathbb{G})$	Server S $(\pi_{S,C})$
$(\alpha', \beta') \overset{\$}{\leftarrow} \mathbb{Z}_p^2$; $t' \leftarrow g^{\alpha'} y^{\beta'} \in \mathbb{G}$		
$r \overset{\$}{\leftarrow} \mathbb{Z}_p$		
$(u \leftarrow g^r, e \leftarrow h^r g^{\pi_C}, v \leftarrow (cd^\xi)^r) \in \mathbb{G}^3$,		
with $\xi \leftarrow H_{\mathsf{SCS}}(C, S, t', u, e)$	$\xrightarrow{t', (u, e, v)}$	$(\alpha, \beta, \gamma) \overset{\$}{\leftarrow} \mathbb{Z}_p^3$
		$t \leftarrow g^\alpha h^\beta (cd^\xi)^\gamma \in \mathbb{G}$
		with $\xi \leftarrow H_{\mathsf{SCS}}(C, S, t', u, e)$
		$r' \overset{\$}{\leftarrow} \mathbb{Z}_p$
	$\xleftarrow{t, (u', e')}$	$(u' \leftarrow g^{r'}, e' \leftarrow y^{r'} g^{\pi_{S,C}}) \in \mathbb{G}^2$
$H'_C \leftarrow t^r$; $H_S \leftarrow u'^{\alpha'} (e'/g^{\pi_C})^{\beta'}$		$H'_S \leftarrow t'^{r'}$; $H_C \leftarrow u^\alpha (e/g^{\pi_{S,C}})^\beta v^\gamma$
$K_C \leftarrow H'_C \times H_S$		$K_S \leftarrow H'_S \times H_C$

Fig. 4. GL–SPOKE

We remark that one encrypted g^π where π is the password, instead of π. This makes it hard to recover π from the decryption of a ciphertext, but that is not a problem in the proofs, where one only needs to check whether a ciphertext contains a given password or not.

5.3 GK–PAKE Construction and GK–SPOKE

GK–PAKE. Our second two-flow construction is depicted in Fig. 5. It additionally provides explicit server authentication to the client. It requires an `IND-CPA` encryption scheme ES' with a GLSPHF, and an `IND-PCA` encryption scheme ES (no need of SPHF for it). It also makes use of a Pseudo-Random Generator PRG, which on a random input returns a longer output that looks indistinguishable to random.

Client C (π_C)	CRS: ($\mathsf{parameters}, \mathsf{pk}, \mathsf{pk}'$)	Server S ($\pi_{S,C}$)
$r_C \xleftarrow{\$}$; $c_C \leftarrow \mathsf{Enc}'(\mathsf{pk}', \pi_C; r_C)$	$\xrightarrow{\quad c_C \quad}$	$\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(L'_{\pi_{S,C}})$
		$\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L'_{\pi_{S,C}}, c_C)$
		$H \leftarrow \mathsf{Hash}(\mathsf{hk}, L'_{\pi_{S,C}}, c_C)$
		$(K_S, r_S) \leftarrow \mathsf{PRG}(H)$
		$\ell = (C, S, c_C, \mathsf{hp})$
$H' \leftarrow \mathsf{ProjHash}(\mathsf{hp}, L'_{\pi_C}, c_C, r_C)$	$\xleftarrow{\quad \mathsf{hp}, c_S \quad}$	$c_S \leftarrow \mathsf{Enc}^{\ell}(\mathsf{pk}, \pi_{S,C}; r_S)$
$(K_C, r'_S) \leftarrow \mathsf{PRG}(H')$		
$\ell = (C, S, c_C, \mathsf{hp})$		
$c'_S \leftarrow \mathsf{Enc}^{\ell}(\mathsf{pk}, \pi_C; r'_S)$		
If $c'_S \neq c_S$, abort		

Fig. 5. Generic GK–PAKE Construction

In Appendix A.2, we prove the following result, with perfectly-smooth SPHFs, which applies for the basic freshness in the BPR setting, or for the forward-secure freshness in the AFP setting with static corruptions only:

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

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.

GK–SPOKE: GK – Simple Password-Only Key Exchange (Fig. 6). Combining our new Short Cramer-Shoup encryption scheme, with the basic ElGamal encryption scheme, we obtain the most efficient PAKE known so far: It is based on the plain DDH assumption, and consists of 2 group elements to be sent by the client and 4 group elements by the server. They both have to compute less than 9 exponentiations.

It also uses a PRG from $\mathbb{G}$ to $\{0,1\}^k \times \mathbb{Z}_p$, where k is the bit-length of the eventual common session key. In practice, one would just need a randomness extractor to extract a seed, and then one extends the seed to get the session key K and the random coins r for the encryption scheme.

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:

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

where q_s is the number of `Send`-queries, Q is the global number of oracle queries, and m is the min-entropy of the passwords.

Client C (π_C)	CRS: ($\mathsf{param} = (\mathbb{G}, p, g), \mathsf{pk} = (h, c, d) \in \mathbb{G}^3, \mathsf{pk}' = y \in \mathbb{G}$)	Server S ($\pi_{S,C}$)
$r' \xleftarrow{\$} \mathbb{Z}_p$		
$(u' \leftarrow g^{r'}, e' \leftarrow y^{r'} g^{\pi_C}) \in \mathbb{G}^2$	$\xrightarrow{\quad (u', e') \quad}$	$(\alpha', \beta') \xleftarrow{\$} \mathbb{Z}_p^2$; $t' \leftarrow g^{\alpha'} y^{\beta'} \in \mathbb{G}$
		$H_C \leftarrow u'^{\alpha'} (e'/g^{\pi_{S,C}})^{\beta'}$
		$(K_S, r) \leftarrow \mathsf{PRG}(H_C)$
		$(u \leftarrow g^r, e \leftarrow h^r g^{\pi_{S,C}}, v \leftarrow (cd^{\xi})^r) \in \mathbb{G}^3$,
$H'_C \leftarrow t'^{r'}$	$\xleftarrow{\quad t', (u, e, v) \quad}$	with $\xi \leftarrow H_{\mathsf{SCS}}(C, S, u', e', t', u, e)$
$(K_C, r'') \leftarrow \mathsf{PRG}(H'_C)$		
$(u'' \leftarrow g^{r''}, e'' \leftarrow h^{r''} g^{\pi_C}, v'' \leftarrow (cd^{\xi''})^{r''}) \in \mathbb{G}^3$, with $\xi'' \leftarrow H_{\mathsf{SCS}}(C, S, u', e', t', u'', e'')$		
If $(u'', e'', v'') \neq (u, e, v)$, abort		

Fig. 6. GK–SPOKE

5.4 KV–PAKE Construction and KV–SPOKE

KV–PAKE. Our third construction is a one-round PAKE, depicted in Fig. 7, from the client point of view, but the server does exactly the same thing, since this is a one-round protocol, where the two flows can be sent independently to each other.

Client C (π_C)	CRS: (parameters, $\mathsf{pk}, \mathsf{pk}'$)	Server S $(\pi_{S,C})$
$\mathsf{hk}_C \xleftarrow{\$} \mathsf{HashKG}(L')$; $\mathsf{hp}_C \leftarrow \mathsf{ProjKG}(\mathsf{hk}_C, L', \perp)$		
$\ell_C = (C, S, \mathsf{hp}_C)$; $r_C \xleftarrow{\$}$; $c_C \leftarrow \mathsf{Enc}^{\ell_C}(\mathsf{pk}, \pi_C; r_C)$	$\xrightarrow{\mathsf{hp}_C, c_C}$	
$\ell_S = (S, C, \mathsf{hp}_S)$	$\xleftarrow{\mathsf{hp}_S, c_S}$	
$H'_C \leftarrow \mathsf{ProjHash}(\mathsf{hp}_S, L^{\ell_C}_{\pi_C}, c_C, r_C)$		
$H_S \leftarrow \mathsf{Hash}(\mathsf{hk}_C, L^{\ell_S}_{\pi_C}, c_S)$		
$K_C \leftarrow H'_C \times H_S$		

Fig. 7. Generic KV–PAKE Construction

It requires an IND-PCA encryption scheme ES with a KVSPHF. In Appendix A.3, we prove the following 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:

$$\mathsf{Adv}(\mathcal{A}) \leq q_s \times 2^{-m} + (2q_e + q_s) \times \mathsf{Adv}^{\texttt{ind-pca}}_{\mathsf{ES}}(t) + \frac{q_e q_s}{2^{2n}},$$

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.

KV–SPOKE: KV – Simple Password-Only Key Exchange (Fig. 8). Using our new Short Cramer-Shoup encryption scheme and its associated KVSPHF, we obtain the most efficient one-round PAKE known so far: It is based on the plain DDH assumption, and consists of 5 group elements to be sent by the each user. They both have to compute 14 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:

$$\mathsf{Adv}(t) \leq q_s \times 2^{-m} + 4Q \times (\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t)) + \frac{2Q^2}{p},$$

where q_s is the number of `Send`-queries, Q is the global number of oracle queries, and m is the min-entropy of the passwords.

Client C (π_C)	CRS: ($\mathsf{param} = (\mathbb{G}, p, q), \mathsf{pk} = (h, c, d) \in \mathbb{G}^3$)	Server S $(\pi_{S,C})$
$(\alpha'_1, \alpha'_2, \beta', \gamma') \xleftarrow{\$} \mathbb{Z}_p^4$		
$(t'_1 \leftarrow g^{\alpha'_1} h^{\beta'} c^{\gamma'}, t'_2 \leftarrow g^{\alpha'_2} d^{\gamma'}) \in \mathbb{G}^2$		
$r \xleftarrow{\$} \mathbb{Z}_p$; $(u \leftarrow g^r, e \leftarrow h^r g^{\pi_C}, v \leftarrow (cd^\xi)^r) \in \mathbb{G}^3$,		
with $\xi \leftarrow H_{\mathsf{SCS}}(C, S, t'_1, t'_2, u, e)$	$\xrightarrow{t'_1, t'_2, (u, e, v)}$	
$H'_C \leftarrow (t_1 t_2^\xi)^r$	$\xleftarrow{t_1, t_2, (u', e', v')}$	
$H_S \leftarrow u'^{\alpha'_1 + \xi' \alpha'_2} (e'/g^{\pi_C})^{\beta'} v'^{\gamma'}$		
with $\xi' \leftarrow H_{\mathsf{SCS}}(S, C, t_1, t_2, u', e')$		
$K_C \leftarrow H'_C \times H_S$		

Fig. 8. KV–SPOKE

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. In *ASIACRYPT 2013, Part I*, *LNCS* 8269, pages 214–234. Springer, December 2013. (Page 26.)

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. (Pages 2 and 26.)

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 9.)

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 25.)

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. (Pages 2 and 26.)

BBC$^+$13a. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New smooth projective hash functions and one-round authenticated key exchange. Cryptology ePrint Archive, Report 2013/034, 2013. http://eprint.iacr.org/2013/034. (Page 23.)

BBC$^+$13b. 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 3, 5, 6, 7, and 26.)

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 25.)

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 25.)

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 25.)

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 1.)

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/2006/214. (Page 26.)

Ble98. D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *CRYPTO'98*, *LNCS* 1462, pages 1–12. Springer, August 1998. (Page 1.)

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 25.)

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 25.)

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 25.)

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 2, 9, and 25.)

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 25.)

BR04. M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. http://eprint.iacr.org/2004/331. (Page 7.)

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 26.)

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 25.)

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. (Pages 2 and 26.)

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. (Pages 2, 3, and 5.)

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, 5, 6, and 26.)

DDN91. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991. (Page 1.)

DDN00. D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. (Page 1.)

DH76. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Page 25.)

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 4.)

Gen08. R. Gennaro. Faster and shorter password-authenticated key exchange. In *TCC 2008*, *LNCS* 4948, pages 589–606. Springer, March 2008. (Pages 2 and 26.)

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 25.)

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, 3, 11, 21, and 26.)

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 25.)

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, 3, 6, 11, 17, and 26.)

GM84. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. (Page 1.)

GS08. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, *LNCS* 4965, pages 415–432. Springer, April 2008. (Page 2.)

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 25.)

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 26.)

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. (Pages 2 and 26.)

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 25.)

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, 17, 20, and 26.)

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, 17, 20, and 26.)

KV11. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *TCC 2011*, *LNCS* 6597, pages 293–310. Springer, March 2011. (Pages 2, 3, 11, 23, and 26.)

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 25.)

Mac02. P. D. MacKenzie. The PAK suite: Protocols for password-authenticated key exchange. Contributions to IEEE P1363.2, 2002. (Page 25.)

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 25.)

NR97. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997. (Page 2.)

NY90. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990. (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 4.)

RS91. C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO'91*, *LNCS* 576, pages 433–444. Springer, August 1991. (Page 1.)

Sho99. V. Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 1999. (Page 25.)

Sho04. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/2004/332. (Page 7.)

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 25.)

A Security Proofs

A.1 Security Proof of GL–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 $\mathbf{G}_6$).

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 $\underline{\texttt{Execute}\text{-queries between two compatible users}}$ are answered. Since the hashing keys are known, we compute the common session key as

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

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

Game $\mathbf{G}_2$: Since the random coins are not needed anymore, 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 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

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

(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 $\mathsf{Hash}(\mathsf{hk}_S, L^{\ell}_{\pi_C}, c_C)$ 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 OPCA 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 $|\mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A})| \leq q_{s1} \times \mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{ES}'}(t)$, where q_{s1} is the number of $\mathtt{Send}_1$-queries.

Game $\mathbf{G}_9$: We now modify the way the $\underline{\mathtt{Send}_2\text{-queries}}$ 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 `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 `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 $\mathbf{G}_{10}$: We now modify the way the $\underline{\mathtt{Send}_0\text{-queries}}$ are answered. 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 `IND-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 `OPCA` 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 `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 `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 `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 $\underline{\text{final game}}$, Win is initially set to `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 `True` at the very end only, or when a corruption happens, using the decryption keys sk and sk':

- `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;

- $\texttt{Send}_1$-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.
- $\texttt{Send}_2$-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 remark 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).

A.2 Security Proof of GK–PAKE

This proof is now close to the proof from [GK10], but a bit more intricate since we allow executions between players with incompatible passwords. 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 $\mathtt{True}$ or not. This flag is initially set to $\mathtt{False}$, 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 $\mathtt{True}$, see Game $\mathbf{G}_{12}$).

Game $\mathbf{G}_1$: We first modify the way $\underline{\mathtt{Execute}\text{-queries between two compatible}}$ users are answered. Since the hashing key is known, we compute H', from the client side as $H' = H = \mathsf{Hash}(\mathsf{hk}, L'_{\pi_C}, c_C)$. 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 r_C are not needed anymore, we replace c_C by the encryption of a dummy password 0_C. This is indistinguishable from $\mathbf{G}_1$ under the $\mathtt{IND\text{-}CPA}$ property of the encryption scheme ES', for each $\mathtt{Execute}$-query between two compatible users. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})| \leq q_{e,1} \times \mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{ES}'}(t)$, where $q_{e,1}$ is the number of $\mathtt{Execute}$-queries between two compatible users.

Game $\mathbf{G}_3$: We replace the hash value $H' = H$ by a truly random value. Since the language L'_π is 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 we now replace the outputs of the PRG by truly random outputs $K_C = K_S$ and $r_S = r'_S$. The security of the PRG leads to: $|\mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A})| \leq q_{e,1} \times \mathsf{Adv}^{\mathtt{prg}}_{\mathsf{PRG}}(t)$.

Game $\mathbf{G}_5$: Since the client and the server use the same random coins and the random password, we never make the client abort. This does not change anything, and one thus gets: $\mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A})$.

Game $\mathbf{G}_6$: We replace c_S by the encryption of a dummy password 0_S. This is indistinguishable from $\mathbf{G}_5$ under the $\mathtt{IND\text{-}CPA}$ property of the encryption scheme ES, for each $\mathtt{Execute}$-query between two compatible users. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A})| \leq q_{e,1} \times \mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{ES}}(t)$.

Game $\mathbf{G}_7$: We now modify the way $\underline{\mathtt{Execute}\text{-queries between two incompatible users}}$ are answered: since the client and the server use different passwords, we always make the client abort. This does not change anything: $\mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_6}(\mathcal{A})$.

Game $\mathbf{G}_8$: Since the random coins r_C are not needed anymore, we replace c_C by the encryption of a dummy password 0_C. This is indistinguishable from $\mathbf{G}_7$ under the $\mathtt{IND\text{-}CPA}$ property of the encryption scheme ES', for each $\mathtt{Execute}$-query between two incompatible users. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A})| \leq q_{e,2} \times \mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{ES}'}(t)$, where $q_{e,2}$ is the number of $\mathtt{Execute}$-queries between two incompatible users.

Game $\mathbf{G}_9$: We replace H by a random value. Thanks to the perfect smoothness of the SPHF, $\mathsf{Hash}(\mathsf{hk}, L'_{\pi_{S,C}}, c_C)$ is completely random values: $\mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A})$.

Game $\mathbf{G}_{10}$: We we now replace the outputs of the PRG by truly random outputs K_S and r_S. The security of the PRG leads to: $|\mathsf{Adv}_{\mathbf{G}_{10}}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A})| \leq q_{e,2} \times \mathsf{Adv}^{\mathtt{prg}}_{\mathsf{PRG}}(t)$.

Game $\mathbf{G}_{11}$: We replace c_S by the encryption of a dummy password 0_S. This is indistinguishable from $\mathbf{G}_{10}$ under the $\mathtt{IND\text{-}CPA}$ property of the encryption scheme ES, for each $\mathtt{Execute}$-query between two incompatible users. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_{11}}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_{10}}(\mathcal{A})| \leq q_{e,2} \times \mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{ES}}(t)$.

Game $\mathbf{G}_{12}$: $\underline{\text{To simplify the following games}}$, we now consider the games in which a collision appears between flows c_C generated via an $\mathtt{Execute}$-query and a $\mathtt{Send}_0$-query: we set Coll to $\mathtt{True}$, and stop the simulation. Since this makes the adversary win, this change can only increase the advantage of the adversary. Therefore, we have: $\mathsf{Adv}_{\mathbf{G}_{11}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_{12}}(\mathcal{A})$.

Game $\mathbf{G}_{13}$: We now modify the way the $\underline{\mathtt{Send}_2\text{-queries}}$ are answered, by using a $\mathtt{OPCA}$ oracle on ES, or alternatively knowing the decryption key sk. More precisely, when a message (hp, 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 c_C, four cases can appear (note we have already excluded collisions on c_C via Execute and $\mathtt{Send}_0$-queries in $\mathbf{G}_6$):

– it is not a message generated by the simulator in the name of this server S in response to the first flow c_C sent by C^j, then we first check whether c_S contains the expected password π_C or not, with the label $\ell = (C, S, c_C, \mathsf{hp})$:
  1. if the expected password is encrypted, then we set Win to True, and stop the simulation;
  2. otherwise, we abort;

– it is a message generated in the name of some $S^{i'}$ (for the same server S):
  3. if S and C are compatible, we know the associated hk, from the simulation of hp, so we can compute H' using hk (as H, and so without the random coins r_C of c_C);
  4. otherwise, we abort.

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 since c_S does not encrypt the correct password. The correctness of the SPHF implies the indistinguishability of the third case. Therefore, we have: $\mathsf{Adv}_{\mathbf{G}_{12}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_{13}}(\mathcal{A})$.

Game $\mathbf{G}_{14}$: We now modify the way the $\underline{\mathtt{Send}_0\text{-queries}}$ are answered: since the random coins r_C are not needed anymore by the simulated clients to compute H', we replace c_C by an encryption of the dummy password 0_C (up to the corruption of π_C). This is indistinguishable from $\mathbf{G}_{13}$ under the IND-CPA property of the encryption scheme ES'. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_{14}}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_{13}}(\mathcal{A})| \leq q_{s0} \times \mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{ES}'}(t)$, 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 hash value H' and thus the session key K_C, whereas the adversary might be able: using a 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 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 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 $\mathbf{G}_{15}$: We now modify the way the $\underline{\mathtt{Send}_1\text{-queries}}$ are answered, knowing the decryption key sk'. More precisely, when a message c_C is sent, in the name of some client instance C^i and to some server instance S^j, three cases can happen:

– it is not a message generated by the simulator in the name of this client C (via a $\mathtt{Send}_0$-query or an Execute-query), then we first decrypt the ciphertext to get the password π used by the adversary:
  1. if this is the expected password $\pi_{S,C}$, then we set Win to True, and stop the simulation;
  2. otherwise, we choose the hash value H at random;

– it is a message generated in the name of some $C^{i'}$ (for the same client C):
  3. we choose the hash value H at random;

The changes in the first case can only increase the advantage of the adversary, while the changes in the second and third cases are indistinguishable under the smoothness of the SPHF, since the encrypted passwords are wrong. Thus, we have: $\mathsf{Adv}_{\mathbf{G}_{14}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_{15}}(\mathcal{A})$.

Game $\mathbf{G}_{16}$: We we now replace the outputs of the PRG by truly random outputs $K_C = K_S$ and $r'_S = r_S$. The security of the PRG leads to: $|\mathsf{Adv}_{\mathbf{G}_{16}}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_{15}}(\mathcal{A})| \leq q_{s1} \times \mathsf{Adv}^{\mathtt{prg}}_{\mathsf{PRG}}(t)$, where q_{s1} is the number of $\mathtt{Send}_1$-queries.

Game $\mathbf{G}_{17}$: In $\mathbf{G}_{13}$, we remark that we do not need to check the correct construction of c_S from a simulated server, so we can simply encrypt the dummy password 0_S instead of the correct password $\pi_{S,C}$ in all ciphertexts c_S, generated as responses to $\mathtt{Send}_1$ queries (up to the corruption of π_C or $\pi_{S,C}$). This is indistinguishable under the IND-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 $\mathtt{OPCA}$ oracle for ES. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_{17}}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_{16}}(\mathcal{A})| \leq q_{s1} \times \mathsf{Adv}^{\mathtt{ind\text{-}pca}}_{\mathsf{ES}}(t)$.

Game $\mathbf{G}_{18}$: 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'.
By a simple summary (in the previous proof), we can see that $\mathsf{Adv}_{\mathbf{G}_{18}}(\mathcal{A}) \leq q_s \times 2^{-m} + q_e q_{s0}/2^{2n}$.

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

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

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

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

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

Forward-Secrecy and Multiple $\mathtt{Test}$*-Queries.* The same remarks as in the previous proof still hold.

A.3 Security Proof of KV–PAKE

This proof is now close to the proof from [KV11], but also to [BBC+13a]. 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 $\mathtt{True}$ or not. This flag is initially set to $\mathtt{False}$, 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 $\mathtt{True}$, see Game $\mathbf{G}_6$).

Game $\mathbf{G}_1$: We first modify the way $\underline{\mathtt{Execute}\text{-queries between two compatible users}}$ are answered. Since the hashing keys are known, we compute the common session key as

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

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

Game $\mathbf{G}_2$: Since the random coins are not needed anymore, 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 $\mathtt{IND\text{-}CPA}$ property of the encryption scheme, for each $\mathtt{Execute}$-query between two compatible users. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})| \leq 2q_{e,1} \times \mathsf{Adv}^{\mathtt{ind\text{-}cpa}}_{\mathsf{ES}}(t)$, where $q_{e,1}$ is the number of $\mathtt{Execute}$-queries between two compatible users.

Game $\mathbf{G}_3$: 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

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

(for the client and the server) by two independent truly random values. Thanks to the perfect smoothness of the KVSPHF, $\mathsf{Hash}(\mathsf{hk}_C, L^{\ell_S}_{\pi_{S,C}}, c_S)$ and $\mathsf{Hash}(\mathsf{hk}_S, L^{\ell_C}_{\pi_C}, c_C)$ 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 scheme, for each Execute-query between two incompatible users. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_5}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_4}(\mathcal{A})| \leq 2q_{e,2} \times \mathsf{Adv}^{\texttt{ind-cpa}}_{\mathsf{ES}}(t)$, 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) generated via an Execute-query and a Send-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 Send$_1$-queries are answered, by using a OPCA oracle on ES, or alternatively knowing the decryption key sk. More precisely, when a message (hp, c) is sent, in the name of some instance U^i and to some simulated instance V^j, four cases can happen:

- it is not a message generated by the simulator in the name of this instance U (via a Send$_0$-query or an Execute-query), then we first check whether c contains the expected password or not, with the label $\ell = (U, V, \mathsf{hp})$:
  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 $U^{i'}$ (for the same player U):
  3. if U and V are compatible, we know the associated hk_U, so we can compute H'_V using hk_U (as H_V, and so without the random coins r_V of c_V). 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 KVSPHF. 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 $\mathbf{G}_8$: We now modify the way the Send$_0$-queries are answered: since the random coins r are not needed anymore by the simulated players to compute hash values, we replace c by an encryption of the dummy password 0_U (up to the corruption of the associated password). This is indistinguishable under the IND-PCA property of the encryption scheme ES, since we still need to be able to test the correct password encrypted by the adversary (see $\mathbf{G}_7$): the simulator has still access to the OPCA oracle for ES. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_{\mathbf{G}_8}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_7}(\mathcal{A})| \leq q_{s0} \times \mathsf{Adv}^{\texttt{ind-pca}}_{\mathsf{ES}}(t)$, where q_{s0} is the number of Send$_0$-queries.

Game $\mathbf{G}_9$: In this final game, Win is initially set to 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 `True` at the very end only, or when a corruption happens, using the decryption key sk. Again, as in the previous proofs, in this game, $\mathsf{Adv}_{\mathbf{G}_9}(\mathcal{A}) \leq q_s \times 2^{-m} + q_e q_{s0}/2^{2n}$.

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

$$q_s \times 2^{-m} + (2q_e + q_s) \times (\mathsf{Adv}^{\mathtt{ind\text{-}pca}}_{\mathsf{ES}}(t)) + \frac{q_e q_{s0}}{2^{2n}},$$

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

$$\mathsf{Adv}_{\mathbf{G}_{\mathtt{real}}}(\mathcal{A}) \leq q_s \times 2^{-m} + (2q_e + q_s) \times \mathsf{Adv}^{\mathtt{ind\text{-}pca}}_{\mathsf{ES}}(t)) + \frac{q_e q_{s0}}{2^{2n}},$$

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

Forward-Secrecy and Multiple `Test`*-Queries.* The same remarks as in the first proof still hold.

B An Overview of PAKE

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].

The first PAKE protocols to be proven secure in the standard model were proposed by Katz, Ostrovsky, and Yung (KOY) [KOY01] based on the DDH assumption and by Goldreich and Lindell [GL01] based on general assumptions. While the KOY protocol assumed the existence of a trusted 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+13b, 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] are among the most efficient protocols. Moreover, as shown by [KOY02, KOY09], these protocols were shown to even achieve a weak flavor of forward secrecy. The framework by Groce and Katz [GK10] based on the Jiang-Gong PAKE [JG04] also yields very efficient protocols in the BPR model, but they additionally make use of pseudorandom number generators.