Public-Key Encryption Indistinguishable Under Plaintext-Checkable Attacks

Michel Abdalla, Fabrice Benhamouda, and David Pointcheval

ENS, Paris, France [⋆]

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) [18], 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

[⋆] CNRS – UMR 8548 and INRIA

existing ciphertexts or inject new ones into the communication and obtain information about the decrypted messages. In fact, as shown by Bleichenbacher [9] 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) [28], indistinguishability under adaptive chosen-ciphertext attack (IND-CCA2 or IND-CCA) [30], non-malleability under chosen-plaintext attack (NM-CPA) or adaptive chosen-ciphertext attack (NM-CCA) [13,14]. Among these, as shown by Bellare *et al.* [5], 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.

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 [29], 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 [29] 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 [11], 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 [27].

In addition to being quite efficient, the new scheme can also be used with Groth-Sahai Non-Interactive Zero-Knowledge Proofs [20] and smooth projective hash functions (SPHF) [12], 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 [2].

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 [6], 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 [17], by Groce and Katz [19], and by Katz and Vaikuntanathan [26], 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 group element.

More precisely, in the case of the Gennaro-Lindell framework [17], which is a generalization of the PAKE scheme by Katz, Ostrovsky, and Yung [23], 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 [22] or 10 group elements in 3 flows in [16,24,25]. 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 [4,10,22], `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 [19], which is a generalization of the PAKE scheme by Jiang and Gong [21] 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 [26], we were able to obtain a PAKE scheme with a total communication complexity of 10 group elements instead of the current 12 in [8]. As in [8,26], 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.

```
$\mathsf{Exp}^{\mathtt{ind}-b}_{\mathsf{ES},\mathcal{A}}(\mathfrak{K})$
    $\mathsf{CTXT} \leftarrow$ 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`)

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 [11] 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 [17], by Groce and Katz [19], and by Katz and Vaikuntanathan [26], whose security proofs appear in the full version [1].

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

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

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

the bit b (i.e., it outputs $b' = b$) and (ℓ^*, C^*) is not in CTXT. The advantages are:

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

where $\mathcal{A} \leq t, q$ 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*;
– 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 [29].

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 [15]. 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 $a, b, c \xleftarrow{\$} \mathbb{Z}_q$:

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

Cramer-Shoup Encryption Scheme [11]. 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:

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

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

[1] Second-preimage resistance is actually sufficient.

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 $H \stackrel{\$}{\leftarrow} \mathcal{H}$, its probability to output $x \neq x'$ such that $H(x) = H(x')$ is bounded by ε. We denote $\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t)$ the best success probability any adversary can get within time t.

2.4 Smooth Projective Hash Functions

Projective hash function families were first introduced by Cramer and Shoup [12]. Here we use the formalization from [8]: 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.

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 $\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(L)$, $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, \perp)$, the distributions $\{(\mathsf{hp}, H) \mid H \leftarrow \mathsf{Hash}(\mathsf{hk}, L, C)\}$ and $\{(\mathsf{hp}, H) \mid H \stackrel{\$}{\leftarrow} \Pi\}$ 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 $\mathsf{hk} \stackrel{\$}{\leftarrow} \mathsf{HashKG}(L)$, $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, C)$, the distributions $\{(\mathsf{hp}, H) \mid H \leftarrow \mathsf{Hash}(\mathsf{hk}, L, C)\}$ and $\{(\mathsf{hp}, H) \mid H \stackrel{\$}{\leftarrow} \Pi\}$ are ε-close, then the SPHF is an ε-smooth GLSPHF. See [8] for more details on GLSPHF and KVSPHF.

Let us now recall SPHFs for the ElGamal and Cramer-Shoup encryption schemes, proposed in [8, 12, 17].

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:

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

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:

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

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

CS also admits an efficient KVSPHF for the language L_M^ℓ [8]:

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

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 $\mathsf{sk} = (s, a, b, a', b') \xleftarrow{\$} \mathbb{Z}_p$ and the public key $\mathsf{pk} = (h = g^s, c = g^a h^b, d = g^{a'} h^{b'})$;
- $\mathsf{SCS.Enc}^\ell(\mathsf{pk} = (h, c, d), M; r)$, for a label ℓ, a group element $M \in \mathbb{G}$ and a scalar $r \in \mathbb{Z}_p$, generates the ciphertext $C = (u = g^r, e = h^r M, v = (c d^\xi)^r)$, 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:

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

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 [8] 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:

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

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 [7, 31].

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 $\mathsf{sk} = (s, a, b, a', b') \overset{\$}{\leftarrow} \mathbb{Z}_p^5$, as well as an unlimited access to an 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 OPCA oracle, the adversary outputs a bit δ', its guess on the bit δ. Note that the adversary is not allowed to query the OPCA oracle on any tuple (ℓ^*, M, C^*).
More precisely, C^* is generated with a random scalar $r^* \overset{\$}{\leftarrow} \mathbb{Z}_p$, as $C^* = (u^* = g^{r^*}, e^* = h^{r^*} M_\delta, v^* = (cd^{\xi^*})^{r^*})$, where $\xi^* = H_{\mathsf{SCS}}(\ell^*, u^*, e^*)$. The 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, $\mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) = \mathsf{Adv}^{\mathtt{ind\text{-}pca}}_{\mathsf{SCS}}(\mathcal{A})$.

Game $\mathbf{G}_1$: In this game, we reject all queries $(\ell, M, C = (u, e, v))$ to the 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}: $|\mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) - \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A})| \leq \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t)$, where t is approximately the running time of $\mathcal{A}$.

Game $\mathbf{G}_2$: We first simplify the simulation of the 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 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: $a, b, a', b' \overset{\$}{\leftarrow} \mathbb{Z}_p$ and $\delta \overset{\$}{\leftarrow} \{0, 1\}$. 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 $\mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A})$:

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

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 `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 $\Delta_z = z - xy$,

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

$$\left\{\begin{array}{rcl} \log_g c & = & a + s \cdot b \\ \log_g d & = & a' + s \cdot b' \\ \log_g v^* & = & y \cdot (a + \xi^* a') + z(b + \xi^* b') \\ & = & y \cdot (\log_g c + \xi^* \log_g d) + \Delta_z \cdot (b + \xi^* b') \\ \log_g v & = & r \cdot (a + \xi a') + rs'(b + \xi b') \\ & = & r \cdot (\log_g c + \xi \log_g d + \Delta_s \cdot (b + \xi b')) \end{array}\right.$$

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 $e \overset{?}{=} u^x M$ and then, if that equality holds, that $v \overset{?}{=} u^{a+\xi a'} \cdot (e/M)^{b+\xi b'}$. 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 $v \overset{?}{=} u^{\alpha+\xi\beta}$, 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 $\delta \overset{\$}{\leftarrow} \{0, 1\}$;
- for the `OPCA` oracle on $(\ell, M, C = (u, e, v))$, one checks $e \overset{?}{=} u^x M$ and $v \overset{?}{=} u^{\alpha+\xi\beta}$, for $\xi = H_{\mathsf{SCS}}(\ell, u, e)$;
- for the challenge ciphertext, one sets $u^* \leftarrow Y$, $e^* \leftarrow ZM_\delta$, and $v^* \overset{\$}{\leftarrow} \mathbb{G}$.

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 [6] and the extension proposed by Abdalla, Fouque, and Pointcheval (AFP) [3]. 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:

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

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

Partnering. Before actually defining the secrecy of the session key, and thus implicit authentication, we need to introduce the notion of partnering: Two instances are partnered if they have matching transcripts, which means that, for one user, its view is a part of the view of the other user. One should note that the last flow can be dropped by the adversary, without letting the sender know. The sender of this last flow thus thinks that the receiver got the message and still computes the session key.

Security. To actually define the semantic security of a PAKE scheme, the adversary $\mathcal{A}$ has access to a challenge oracle $\texttt{Test}(U^i)$, available 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 `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 `Corrupt`(C) or via a query `Corrupt`$(\cdot, C, \cdot)$), where $C = U$ is U is a client or U^i's partner is an instance C^j of C
- forward-secure freshness: similar to basic freshness except for the last part, where only corruptions before U^i defined his key can make this instance unfresh.

In case of `Test`-query to an unfresh instance, the answer is $\perp$, which means that the adversary cannot have any advantage in these cases. A PAKE is considered BPR-secure if the advantage of any adversary $\mathcal{A}$, running within time t, in the previous experiment is bounded by $q_s \times 2^{-m} + \mathrm{negl}(\mathfrak{K})$, where q_s is the number of active sessions (handled with `Send` queries), 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 `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 [17]. The second one follows the Groce-Katz (GK) framework [19], and the third one follows the one-round Katz-Vaikuntanathan (KV) framework [26]. 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 `IND-PCA` encryption scheme $\mathsf{ES} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and an `IND-CPA` encryption scheme $\mathsf{ES}' = (\mathsf{KG}', \mathsf{Enc}', \mathsf{Dec}')$

so that SPHFs (either GLSPHFs or KVSPHFs according to the protocol) exist for the following families of languages:

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

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

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

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.

Client C (π_C)	CRS: (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

It requires an IND-CPA encryption scheme ES' with a KVSPHF, and an IND-PCA encryption scheme ES with a GLSPHF. In the full version [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:

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

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.

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.

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

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

Fig. 5. Generic GK–PAKE Construction

In the full version [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:

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

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.

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

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 $(\pi_{S,C})$
$r' \overset{\$}{\leftarrow} \mathbb{Z}_p$		
$(u' \leftarrow g^{r'}, e' \leftarrow y^{r'} g^{\pi_C}) \in \mathbb{G}^2$	$\xrightarrow{(u', e')}$	$(\alpha', \beta') \overset{\$}{\leftarrow} \mathbb{Z}_p^2$; $t' \leftarrow g^{\alpha'} y^{\beta'} \in \mathbb{G}$
		$H_C \leftarrow u'^{\alpha'} (e'/g^{\pi_{S,C}})^{\beta'}$
		$(K_S, r) \leftarrow \mathsf{PRG}(H_C)$
		$(u \leftarrow g^r, e \leftarrow h^r g^{\pi_{S,C}}, v \leftarrow (cd^{\xi})^r) \in \mathbb{G}^3$,
$H'_C \leftarrow t'^{r'}$	$\xleftarrow{t', (u, e, v)}$	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

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

Fig. 7. Generic KV–PAKE Construction

It requires an IND-PCA encryption scheme ES with a KVSPHF. In the full version [1], 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:

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

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

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption scheme indistinguishable under plaintext-checkable attacks. Cryptology ePrint Archive, Report 2014/609 (2014), http://eprint.iacr.org/2014/609
2. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth projective hashing for conditionally extractable commitments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 671–689. Springer (Aug 2009)
3. Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer (Jan 2005)
4. Abdalla, M., Pointcheval, D.: A scalable password-based group key exchange protocol in the standard model. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 332–347. Springer (Dec 2006)

5. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 26–45. Springer (Aug 1998)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer (May 2000)
7. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331 (2004), http://eprint.iacr.org/2004/331
8. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer (Aug 2013)
9. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 1–12. Springer (Aug 1998)
10. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer (May 2005)
11. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer (Aug 1998)
12. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer (Apr / May 2002)
13. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: 23rd ACM STOC. pp. 542–552. ACM Press (May 1991)
14. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)
15. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer (Aug 1984)
16. Gennaro, R.: Faster and shorter password-authenticated key exchange. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 589–606. Springer (Mar 2008)
17. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer (May 2003), http://eprint.iacr.org/2003/032.ps.gz
18. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
19. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 10. pp. 516–525. ACM Press (Oct 2010)
20. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer (Apr 2008)
21. Jiang, S., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 267–279. Springer (Aug 2004)
22. Katz, J., MacKenzie, P.D., Taban, G., Gligor, V.D.: Two-server password-only authenticated key exchange. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 05. LNCS, vol. 3531, pp. 1–16. Springer (Jun 2005)

23. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer (May 2001)
24. Katz, J., Ostrovsky, R., Yung, M.: Forward secrecy in password-only key exchange protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 02. LNCS, vol. 2576, pp. 29–44. Springer (Sep 2002)
25. Katz, J., Ostrovsky, R., Yung, M.: Efficient and secure authenticated key exchange using weak passwords. Journal of the ACM 57(1), 78–116 (2009)
26. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer (Mar 2011)
27. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS. pp. 458–467. IEEE Computer Society Press (Oct 1997)
28. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. pp. 427–437. ACM Press (May 1990)
29. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer (Apr 2001)
30. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 433–444. Springer (Aug 1991)
31. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), http://eprint.iacr.org/2004/332