Smooth Projective Hashing for Conditionally Extractable Commitments

Michel Abdalla, Céline Chevalier, and David Pointcheval

École Normale Supérieure, CNRS-INRIA, Paris, France

Abstract. The notion of smooth projective hash functions was proposed by Cramer and Shoup and can be seen as special type of zero-knowledge proof system for a language. Though originally used as a means to build efficient chosen-ciphertext secure public-key encryption schemes, some variations of the Cramer-Shoup smooth projective hash functions also found applications in several other contexts, such as password-based authenticated key exchange and oblivious transfer. In this paper, we first address the problem of building smooth projective hash functions for more complex languages. More precisely, we show how to build such functions for languages that can be described in terms of disjunctions and conjunctions of simpler languages for which smooth projective hash functions are known to exist. Next, we illustrate how the use of smooth projective hash functions with more complex languages can be efficiently associated to extractable commitment schemes and avoid the need for zero-knowledge proofs. Finally, we explain how to apply these results to provide more efficient solutions to two well-known cryptographic problems: a public-key certification which guarantees the knowledge of the private key by the user without random oracles or zero-knowledge proofs and adaptive security for password-based authenticated key exchange protocols in the universal composability framework with erasures.

1 Introduction

In [16], Cramer and Shoup introduced a new primitive called smooth projective hashing and showed how to use it to generalize their chosen-ciphertext secure public-key encryption scheme [15]. The new abstraction not only provided a more intuitive description of the original encryption scheme, but also resulted in several new instantiations based on different security assumptions such as quadratic residuosity and N-residuosity [31].

The notion of smooth projective hash functions (SPHF, [16], after slight modifications [22]) has been proven quite useful and has found applications in several other contexts, such as password-based authenticated key exchange (PAKE, [22]) and oblivious transfer [27]. In the context of PAKE protocols, the work of Gennaro and Lindell abstracted and generalized (under various indistinguishability assumptions) the earlier protocol by Katz, Ostrovsky, and Yung [28] and has become the basis of several other schemes [3, 1, 8]. In the context of oblivious transfer, the work of Kalai [27] also generalized earlier protocols by Naor and Pinkas [30] and by Aiello, Ishai, and Reingold [2].

To better understand the power of SPHF, let us briefly recall what they are. First, the definition of SPHF requires the existence of a domain X and an underlying NP language L such that it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$. For instance, in the particular case of the PAKE scheme in [13], the language L is defined as the set of triples $\{(c, \ell, m)\}$ such that c is an encryption of m with label ℓ under a public key given in the common reference string (CRS). The semantic security of the encryption scheme guarantees computational indistinguishability between elements from L and from X.

One of the key properties that make SPHF so useful is that, for a point $x \in L$, the hash value can be computed using either a *secret* hashing key hk, or a *public* projected key hp (depending on x [22] or not [16]) together with a witness w to the fact that $x \in L$. Another important property of these functions is that, given the projected key hp, their output is uniquely defined for points $x \in L$ and statistically indistinguishable from random for points $x \in X \setminus L$. Moreover, without the knowledge of the witness w to the fact that $x \in L$, the output of these functions on x is also pseudo-random.

The first main contribution of this paper is to extend the line of work on SPHF, the element-based version proposed by [22], to take into account more complex NP languages. We show how to build SPHF for languages that can be described in terms of disjunctions and conjunctions of simpler languages for which SPHF are known to exist. For instance, let H_m represent a family of SPHF for the language $\{(c)\}$, where c is the encryption of m under a given public key. Using our tools, one can build a family of SPHF for the language $\{(c)\}$, where c is the encryption of either 0 or 1, by combining H_0 and H_1.

One of the advantages of building SPHF for more complex languages is that it allows us to simplify the design of the primitives to which they are associated. To demonstrate this, we consider in this paper the specific case of extractable commitment schemes. In most protocols in which extractable commitments are used, the capability of extracting the committed message usually depends on the commitment being properly generated. To achieve this goal and enforce the correct generation of the commitment, it is often the case that additional mechanisms, such as zero-knowledge proofs, may have to be used. This is the case, for instance, of several protocols where a specific public-key registration phase is required, such as most of the cryptographic protocols with dynamic groups (multisignatures [9, 29], group signatures [18], etc). Such a framework is sometimes named *registered public-key model*, where a proof of knowledge of the secret key is required before any certification.

To be able to build more efficient extractable commitment schemes and avoid the use of possibly expensive concurrent zero-knowledge proofs, a second main contribution of this paper is to generalize the concept of extractable commitments so that extraction may fail if the commitment is not properly generated. More specifically, we introduce a new notion of L-extractable commitments in which extraction is only guaranteed if the committed value belongs to the language L and may fail otherwise. The main intuition behind this generalization is that, when used together with a SPHF for the language L, the cases in which

extraction may fail will not be very important as the output of the SPHF will be statistically indistinguishable from random in such cases.

Applications.
Registered Public-Key Setting. For many cryptographic protocols, for proving the security even when users can dynamically join the system, the simulator described in the security proof often needs to know the private keys of the authorized users, which is called the *registered public-key setting*, in order to avoid rogue-attacks [9]. This should anyway be the correct way to proceed for a certification authority: it certifies a public key to a user if and only if the latter provides a proof of knowledge of the associated private key. However, in order to allow concurrency, intricate zero-knowledge proofs are required, which makes the certification process either secure in the random oracle model [6] only, or inefficient in the standard model.

In this paper, we show how SPHF with conditionally extractable commitments can help to solve this problem efficiently, in the standard model, by establishing a secure channel between the players, with keys that are either the same for the two parties if the commitment has been correctly built, or perfectly independent in the other case.

Adaptively-secure PAKE schemes. We thereafter study more involved key exchange schemes. In 1992, Bellovin and Merritt [7] suggested a method to authenticate a key exchange based on simple passwords, possibly drawn from a space so small that an adversary might enumerate off-line all possible values. Because of the practical interest of such a primitive, many schemes have been proposed and studied. In 2005, Canetti *et al.* [13] proposed an ideal functionality for PAKE protocols, in the universal composability (UC) framework [11, 14], and showed how a simple variant of the Gennaro-Lindell methodology [22] could lead to a secure protocol. Though quite efficient, their protocol is not known to be secure against adaptive adversaries, where they can corrupt players at any time, and learn their internal states. The first ones to propose an adaptively-secure PAKE in the UC framework were Barak *et al.* [3] using general techniques from multi-party computation (MPC). Though conceptually simple, their solution yields quite inefficient schemes.

Here, we take a different approach. Instead of using general MPC techniques, we extend the Gennaro-Lindell methodology to deal with adaptive corruptions by using a non-malleable conditionally-extractable and equivocable commitment scheme with an associated SPHF family. The new scheme is adaptively secure in the common reference string model in the UC framework under standard complexity assumptions with erasures.

Related work.
Commitments. Commitment schemes are one of the most fundamental cryptographic primitives, being used in several cryptographic applications such as zero-knowledge proofs [25] and secure multi-party computation [24]. Even quite practical protocols need them, as already explained above in the public-key registration setting, but also in password-based authenticated key exchange [22].

They allow a user to commit a value x into a public value C, such that the latter does not reveal any information about x (the hiding property), but C can be opened later to x only: one cannot change its mind (the binding property). Various additional properties are often required, such as non-malleability, extractability and equivocability. Canetti and Fischlin [12] provided an ideal functionality for such a primitive and showed that achieving all these properties at the same time was impossible in the UC plain model. They also provided the first candidate in the CRS model. Damgård and Nielsen [17] later proposed another construction of universally composable commitments, that is more efficient for some applications. Since we want to avoid the use of possibly inefficient proofs of relations present in the Damgård-Nielsen construction and given that the Canetti-Fischlin construction is well suited for our purpose of designing an associated smooth hash function, we opted to use the latter as the starting point for our constructions.

PAKE. The password-based setting was first considered by Bellovin and Merritt [7] and followed by many proposals. In 2000, Bellare, Pointcheval, and Rogaway [5] as well as Boyko, MacKenzie, and Patel [10] proposed security models and proved variants of the protocol of [7], under ideal assumptions, such as the random oracle model [6]. Soon after, Katz, Ostrovsky, and Yung [28] and Goldreich and Lindell [23] proposed the first protocols with a proof of security in the standard model, with the former being based on the decisional Diffie-Hellman assumption and the latter on general assumptions. Later, Gennaro and Lindell [22] proposed an abstraction and generalization of the KOY protocol and became the basis of several other variants, including ours in the last section.

Organization of the Paper. In Section 2, we review the basic primitives needed in this paper. Then, in Section 3, we describe our first contribution: SPHF families on conjunctions and disjunctions of languages. In Section 4 we combine that with our second contribution, conditionally-extractable commitments. We focus on the ElGamal-based commitment, since this is enough to build more efficient public-key certification protocols. Finally, in Section 5, we add equivocability to the commitment, borrowing techniques from Canetti and Fischlin [12]. Then, we add the non-malleability property, granted the Cramer-Shoup encryption scheme, which can then be used to build an adaptively-secure PAKE in the UC framework, based on the Gennaro and Lindell [22] framework. Due to space restrictions, formal definitions, proofs, and application details were postponed to the appendix.

2 Commitments

In the following, we focus on Pedersen commitments, and certification of Schnorr-like public keys, hence, we work in the discrete logarithm setting. As a consequence, to get extractable commitments, we use encryption schemes from the same family: the ElGamal encryption [21] and the labeled version of the Cramer-Shoup encryption scheme [15] (for achieving non-malleability).

Labeled Public-Key Encryption. Labeled encryption [32] is a variation of the usual encryption notion that takes into account the presence of labels in the encryption and decryption algorithms. More precisely, both the encryption and decryption algorithms have an additional input parameter, referred to as a label, and the decryption algorithm should only correctly decrypt a ciphertext if its input label matches the label used to create that ciphertext.

The security notion for labeled encryption is similar to that of standard encryption schemes. The main difference is that, whenever the adversary wishes to ask a query to its Left-or-Right encryption oracle in the indistinguishability security game (IND-CPA) [26, 4], in addition to providing a pair of messages (m_0, m_1), it also has to provide a target label ℓ to obtain the challenge ciphertext c. When chosen-ciphertext security (IND-CCA) is concerned, the adversary is also allowed to query its decryption oracle on any pair (ℓ', c') as long as $\ell' \neq \ell$ or the ciphertext c' does not match the output c of a query to its Left-or-Right encryption oracle whose input includes the label ℓ. For formal security definitions for labeled encryption schemes, please refer to [13, 1].

One of the advantages of using labeled encryption, which we exploit in this paper, is that we can easily combine several IND-CCA labeled encryption schemes with the help of a strongly unforgeable one-time signature scheme so that the resulting scheme remains IND-CCA [20].

ElGamal and Cramer-Shoup Encryption. We denote by G a cyclic group of prime order q where q is large (n bits), and g a generator for this group. Let $\mathsf{pk} = (g_1, g_2, c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z, H)$ be the public key of the Cramer-Shoup scheme, where g_1 and g_2 are random group elements, x_1, x_2, y_1, y_2 and z are random scalars in $\mathbb{Z}_q$, and H is a collision-resistant hash function (actually, second-preimage resistance is enough), and $\mathsf{sk} = (x_1, x_2, y_1, y_2, z)$ the associated private key. Note that (g_1, h) will also be seen as the public key of the ElGamal encryption, with z the associated private key. For the sake of simplicity, we assume in the following that public keys will additionally contain all the global parameters, such as the group G.

If $M \in G$, the multiplicative ElGamal encryption is defined as $\mathsf{EG}^{\times}_{\mathsf{pk}}(M; r) = (u_1 = g_1^r, e = h^r M)$, which can be decrypted by $M = e/u_1^z$. If $M \in \mathbb{Z}_q$, the additive ElGamal encryption is defined as $\mathsf{EG}^{+}_{\mathsf{pk}}(M; r) = (u_1 = g_1^r, e = h^r g^M)$. Note that $\mathsf{EG}^{\times}_{\mathsf{pk}}(g^M; r) = \mathsf{EG}^{+}_{\mathsf{pk}}(M; r)$. It can be decrypted after an additional discrete logarithm computation: M must be small enough. Similarly, if $M \in G$, the multiplicative labeled Cramer-Shoup encryption is defined as $\mathsf{CS}^{\times\,\ell}_{\mathsf{pk}}(M; r) = (u_1, u_2, e, v)$, such that $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r M$, $\theta = H(\ell, u_1, u_2, e)$ and $v = (cd^\theta)^r$. Decryption works as above, with $M = e/u_1^z$, but only if the ciphertext is valid: $v = u_1^{x_1 + \theta y_1} u_2^{x_2 + \theta y_2}$. If $M \in \mathbb{Z}_q$, its additive encryption $\mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(M; r)$ is such that $e = h^r g^M$. The following relation holds $\mathsf{CS}^{\times\,\ell}_{\mathsf{pk}}(g^M; r) = \mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(M; r)$. The decryption applies as above if M is small enough.

As already noted, from any Cramer-Shoup ciphertext (u_1, u_2, e, v) of a message M with randomness r, whatever the label ℓ is, one can extract (u_1, e) as an ElGamal ciphertext of the same message M with the same randomness r.

This extraction applies independently of the additive or multiplicative version since the decryption works the same for the ElGamal and the Cramer-Shoup ciphertexts, except for the validity check that provides the CCA security level to the Cramer-Shoup encryption scheme, whereas the ElGamal encryption scheme achieves IND-CPA security level only.

Commitments. With a commitment scheme, a player can commit to a secret value x by publishing a commitment $C = \mathsf{com}(x; r)$ with randomness r, in such a way that C reveals nothing about the secret x, which is called the *hiding* property. The player can later open C to reveal x, by publishing x and a decommitment, also referred to as witness, in a publicly verifiable way: the player cannot open C to any other value than x, which is the *binding* property. In many cases, the decommitment consists of the random r itself or some part of it. In this paper, we only consider commitment schemes in the common reference string (CRS) model in which the common parameters, referred to as the CRS, are generated honestly and available to all parties.

Note that an IND-CPA public-key encryption scheme provides such a commitment scheme: the binding property is guaranteed by the uniqueness of the plaintext (perfectly binding), and the hiding property is guaranteed by the IND-CPA security (computationally hiding). In this case, the CRS simply consists of the public-key of the encryption scheme. The Pedersen commitment $C = \mathsf{comPed}(x; r) = g^x h^r$ provides a perfectly hiding, but computationally binding commitment under the intractability of the discrete logarithm of h in basis g.

We now present additional properties that can be satisfied by the commitment. First, we say that a commitment is *extractable* if there exists an efficient algorithm, called an extractor, capable of generating a new set of common parameters (*i.e.*, a new CRS) whose distribution is equivalent to that of an honestly generated CRS and such that it can extract the committed value x from any commitment C. This is of course only possible for computationally hiding commitments, such as encryption schemes: the decryption key is the extraction trapdoor. Second, we say that a commitment is *equivocable* if there exists an efficient algorithm, called an equivocator, capable of generating a new CRS and a commitment with similar distributions to those of the actual scheme and such that the commitment can be opened in different ways. Again, this is possible for computationally binding commitments only, such as the Pedersen commitment: the knowledge of the discrete logarithm of h in basis g is a trapdoor that allows the opening of a commitment in more than one way. Finally, a *non-malleable* commitment ensures that if an adversary that receives a commitment C of some unknown value x can generate a valid commitment for a related value y, then a simulator could perform as well without seeing C. A public-key encryption scheme that is IND-CCA provides such a non-malleable commitment [22]. For formal security definitions for commitment schemes, please refer to [22, 19, 12].

In the following, we use encryption schemes in order to construct commitments, which immediately implies the hiding, binding and extractable properties, as said above. However, when one uses the additive versions of ElGamal or Cramer-Shoup encryption schemes, extractability (or decryption) is only possible

if the committed values (or plaintexts) are small enough, hence our notion of L-extractable commitments (see Section 4) which will mean that the commitment is extractable if the committed value lies in the language L. More precisely, we will split the value to be committed in small pieces (that lie in the language L), but we will then need to be sure that they actually lie in this language to guarantee extractability. We thus introduce smooth hash functions in order to allow communications if the commitments are valid only.

3 Smooth Hash Functions on Conjunctions and Disjunctions of Languages

Smooth Projective Hash Functions. Projective hash function families were first introduced by Cramer and Shoup [16] as a means to design chosen-ciphertext secure encryption schemes. We here use the definitions of Gennaro and Lindell [22], who later showed how to use such families to build secure password-based authenticated key exchange protocols, together with non-malleable commitments. In addition to commitment schemes, we also consider here families of SPHF associated to labeled encryption as done by Canetti *et al.* [13] and by Abdalla and Pointcheval [1].

Let X be the domain of these functions and let L be a certain subset of points of this domain (a language). A key property of these functions is that, for points in L, their values can be computed by using either a *secret* hashing key or a *public* projected key. While the computation using the *secret* hashing key works for all points in the domain X of the hash function, the computation using a *public* projected key only works for points $x \in L$ and requires the knowledge of the witness w to the fact that $x \in L$. A projective hash function family is said to be *smooth* if the value of the function on inputs that are outside the particular subset L of the domain are independent of the projected key. Another important property of these functions is that, given the projected key hp, their output is uniquely defined for points $x \in L$. Moreover, if L is a *hard partitioned subset* of X (*i.e.*, it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$), this output is also *pseudo-random* if one does not know a witness w to the fact that $x \in L$ [22]. The interested reader is referred to the full version for more formal definitions.

In the particular case of the Gennaro-Lindell scheme [22], the subset $L_{\mathsf{pk},m}$ was defined as the set of $\{(c)\}$ such that c is a commitment of m using public parameters pk: there exists r for which $c = \mathsf{com}_{\mathsf{pk}}(m; r)$ where com is the committing algorithm of the commitment scheme. In the case of the CHKLM scheme [13], the subset $L_{\mathsf{pk},(\ell,m)}$ was defined as the set of $\{(c)\}$ such that c is an encryption of m with label ℓ, under the public key pk: there exists r for which $c = \mathcal{E}^{\ell}_{pk}(m; r)$ where $\mathcal{E}$ is the encryption algorithm of the labeled encryption scheme. In the case of a standard encryption scheme, the label is simply omitted. The interested reader is referred to [22, 13, 1] for more details.

Languages. Since we want to use more general languages, we need more detailed notations. Let **LPKE** be a labeled encryption scheme with public key pk.

Let X be the range of the encryption algorithm. Here are three useful examples of languages L in X:

- the valid ciphertexts c of m under pk, $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m)} = \{c | \exists r\ c = \mathcal{E}^{\ell}_{\mathsf{pk}}(m;r)\}$;
- the valid ciphertexts c of m_1 or m_2 under pk (that is, a disjunction of two versions of the former languages), $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_1 \vee m_2)} = L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_1)} \cup L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_2)}$;
- the valid ciphertexts c under pk, $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,*)} = \{c | \exists m\ \exists r\ c = \mathcal{E}^{\ell}_{\mathsf{pk}}(m;r)\}$.

If the encryption scheme is IND-CPA, the first two are hard partitioned subsets of X. The last one can also be a hard partitioned subset in some cases: for the Cramer-Shoup encryption, $L \subsetneq X = G^4$ and, in order to distinguish a valid ciphertext from an invalid one, one has to break the DDH problem. However, for the ElGamal encryption scheme, all the ciphertexts are valid, hence $L = X = G^2$.

More complex languages can be defined, with disjunctions as above, or conjunctions: the pairs of ciphertexts (a, b) such that $a \in L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,0\vee1)}$ and $b \in L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,2\vee3)}$. This set can be obtained by $(L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,0\vee1))} \times X) \cap (X \times L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,2\vee3)})$.

Likewise, we can define more general languages based on other primitives such as commitment schemes. The definition would be similar to the one above, with pk playing the role of the common parameters, $\mathcal{E}_{\mathsf{pk}}$ playing the role of the committing algorithm, (m, ℓ) playing the role of the input message, and c playing the role of the commitment.

More generally, in the following, we denote the language by the generic notation $L_{(\mathbf{Sch},\rho),aux}$ where aux denotes all the parameters useful to characterize the language (such as the label used, or a plaintext), ρ denotes the public parameters such as a public key pk, and **Sch** denotes the primitive used to define the language, such as an encryption scheme **LPKE** or a commitment scheme **Com**. When there is no ambiguity, the associated primitive **Sch** will be omitted.

We now present new constructions of SPHF to deal with more complex languages, such as disjunctions and conjunctions of any languages. The constructions are presented for two languages but can be easily extended to any polynomial number of languages. We then discuss about possible information leakage at the end of this section. The properties of correctness, smoothness and pseudo-randomness are easily verified by these new smooth hash systems. Due to the lack of space, the formal proofs can be found in the full version.

Conjunction of two Generic Smooth Hashes. Let us consider an encryption or commitment scheme defined by public parameters and a public key aggregated in ρ. X is the range of the elements we want to study (ciphertexts, tuples of ciphertexts, commitments, etc), and $L_1 = L_{1,\rho,aux}$ and $L_2 = L_{2,\rho,aux}$ are hard partitioned subsets of X, which specify the expected properties (valid ciphertexts, ciphertexts of a specific plaintext, etc). We consider situations where X possesses a group structure, which is the case if we consider ciphertexts or tuples of ciphertexts from an homomorphic encryption scheme. We thus denote by $\oplus$ the commutative law of the group (and by $\ominus$ the opposite operation, such that $c \oplus a \ominus a = c$).

We assume to be given two smooth hash systems SHS_1 and SHS_2, on the sets corresponding to the languages L_1 and L_2: $\mathsf{SHS}_i = \{\mathsf{HashKG}_i, \mathsf{ProjKG}_i, \mathsf{Hash}_i, \mathsf{ProjHash}_i\}$. Here, HashKG_i and ProjKG_i denote the hashing key and the projected key generators, and Hash_i and $\mathsf{ProjHash}_i$ the algorithms that compute the hash function using hk_i and hp_i respectively.

Let c be an element of X, and r_1 and r_2 two elements chosen at random. We denote by $\mathsf{hk}_1 = \mathsf{HashKG}_1(\rho, aux, r_1)$, $\mathsf{hk}_2 = \mathsf{HashKG}_2(\rho, aux, r_2)$, $\mathsf{hp}_1 = \mathsf{ProjKG}_1(\mathsf{hk}_1; \rho, aux, c)$, and $\mathsf{hp}_2 = \mathsf{ProjKG}_2(\mathsf{hk}_2; \rho, aux, c)$ the keys. A smooth hash system for the language $L = L_1 \cap L_2$ is then defined as follows, if $c \in L_1 \cap L_2$ and w_i is a witness that $c \in L_i$, for $i = 1, 2$:

$$\begin{aligned}
\mathsf{HashKG}_L(\rho, aux, r = r_1 \| r_2) &= \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2) \\
\mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) \\
\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, c) \\
\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, c; (w_1, w_2)) &= \mathsf{ProjHash}_1(\mathsf{hp}_1; \rho, aux, c; w_1) \\
&\quad \oplus \mathsf{ProjHash}_2(\mathsf{hp}_2; \rho, aux, c; w_2)
\end{aligned}$$

Disjunction of two Generic Smooth Hashes. Let L_1 and L_2 be two languages as described above. We assume to be given two smooth hash systems SHS_1 and SHS_2 with respect to these languages. We define $L = L_1 \cup L_2$ and construct a smooth projective hash function for this language as follows:

$$\begin{aligned}
\mathsf{HashKG}_L(\rho, aux, r = r_1 \| r_2) &= \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2) \\
\mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta = \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) \\
&\qquad \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, c)) \\
\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) \\
\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, c; w) &= \mathsf{ProjHash}_1(\mathsf{hp}_1; \rho, aux, c; w) \qquad \text{if } c \in L_1 \\
&\text{or } \mathsf{hp}_\Delta \ominus \mathsf{ProjHash}_2(\mathsf{hp}_2; \rho, aux, c; w) \quad \text{if } c \in L_2
\end{aligned}$$

where w is a witness of $c \in L_i$ for $i \in \{1, 2\}$. Then $\mathsf{ProjHash}_i(\mathsf{hp}_i; \rho, aux, c; w) = \mathsf{Hash}_i(\mathsf{hk}_i; \rho, aux, c)$. The player in charge of computing this value is supposed to know w, and in particular the language which c belongs to (the index i).

Uniformity and Independence. In the above definition of SPHF (contrarily to the original Cramer-Shoup [16] definition), the value of the projected key formally depends on the ciphertext/commitment c. However, in some cases, one may not want to reveal any information about this dependency. In fact, in certain cases such as in the construction of a SPHF for equivocable and extractable commitments in Section 5, one may not even want to leak any information about the auxiliary elements aux. When no information is revealed about aux, it means that the details about the exact language will be concealed.

We thus add a notion similar to the smoothness, but for the projected key: the projected key may or may not depend on c (and aux), but its distribution does not: Let us denote by $D_{\rho,aux,c}$ the distribution $\{\mathsf{hp} \mid \mathsf{hk} = \mathsf{HashKG}_L(\rho, aux, r)$ and $\mathsf{hp} = \mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c)\}$, on the projected keys. If, for any $c, c' \in X$, $D_{\rho,aux,c'}$ and $D_{\rho,aux,c}$ are indistinguishable, then we say that the smooth hash system has the *1-uniformity* property. If, for any $c, c' \in X$, and any auxiliary

elements aux, aux', $D_{\rho,aux',c'}$ and $D_{\rho,aux,c}$ are indistinguishable, we name it *2-uniformity* property.

More than indistinguishability of distributions, the actual projected key hp may not depend at all on c, as in the Cramer and Shoup's definition. Then, we say that the smooth hash system guarantees *1-independence* (resp. *2-independence* if it does not depend on aux either). Note that the latter independence notions immediately imply the respective uniformity notions.

As an example, the smooth hash system associated with the ElGamal cryptosystem (see Section 4 page 10) guarantees 2-independence. On the other hand, the analogous system associated with the Cramer-Shoup encryption (see the full version) guarantees 2-uniformity only. For smooth hash systems combinations, one can note that in the case of disjunctions, one can get, at best, the uniformity property, since hash computations on the commitment are needed for generating the projected key. Furthermore, this is satisfied under the condition that the two underlying smooth hash systems already satisfy this property (see the full version for more details and proofs).

Finally, one should note that, in the case of disjunction, the view of the projected hash value could leak some information about the sub-language in which the input lies, if an adversary sends a fake hp_Δ. The adversary could indeed check whether $\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, c; w)$ equals $\mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c)$ or $\mathsf{hp}_\Delta \ominus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, c)$. But first, it does not contradict any security notion for smooth hash systems; second, in all the applications below, the projected hash value is never revealed; and third, in the extractable commitments below, because of the global conjunction of the languages, an exponential exhaustive search would be needed to exploit this information, even if the committed value is a low-entropy one.

4 A Conditionally Extractable Commitment

ElGamal Commitment and Associated Smooth Hash. The ElGamal commitment is realized in the common reference string model, where the CRS ρ contains (G, pk), as defined in Section 2, for the ElGamal encryption scheme. In practice, sk should not be known by anybody, but in the security analysis, sk will be the extraction trapdoor. Let the input of the committing algorithm be a scalar $M \in \mathbb{Z}_q$. The commitment algorithm consists of choosing a random r and computing the following ElGamal encryption under random r: $C = \mathsf{EG}^+_{\mathsf{pk}}(M, r) = (u_1 = g_1^r, e = h^r g^M)$.

The smooth projective hashing, associated with this commitment scheme and the language $L = L_{(\mathbf{EG}^+,\rho),M} \subset X = G^2$ of the additive ElGamal ciphertexts C of M under the global parameters and public key defined by ρ, is the family based on the underlying ElGamal encryption scheme, as defined in [22]:

$$\begin{aligned}
&\mathsf{HashKG}((\mathbf{EG}^+, \rho), M) = \mathsf{hk} = (\gamma_1, \gamma_3) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q\\
&\mathsf{Hash}(\mathsf{hk}; (\mathbf{EG}^+, \rho), M, C) = (u_1)^{\gamma_1} (e g^{-M})^{\gamma_3}\\
&\mathsf{ProjKG}(\mathsf{hk}; (\mathbf{EG}^+, \rho), M, C) = \mathsf{hp} = (g_1)^{\gamma_1} (h)^{\gamma_3}\\
&\mathsf{ProjHash}(\mathsf{hp}; (\mathbf{EG}^+, \rho), M, C; r) = (\mathsf{hp})^r
\end{aligned}$$

First, under the DDH problem (semantic security of the ElGamal encryption scheme), L is a hard partitioned subset of $X = G^2$. Then, for $C = \mathsf{EG}^+_{\mathsf{pk}}(M, r)$, and thus with the witness r, the algorithms are defined as above using the same notations as in [22].

L-extractable Commitments. Note that the value g^M would be easily extractable from this commitment (seen as the multiplicative ElGamal encryption). However, one can extract M itself (the actual committed value) only if its size is small enough so that it can be found as a solution to the discrete logarithm problem. In order to obtain "extractability" (up to a certain point, see below), one should rather commit to it in a bit-by-bit way.

Let us denote $M \in \mathbb{Z}_q$ by $\sum_{i=1}^{m} M_i \cdot 2^{i-1}$, where $m \leq n$. Its commitment is $\mathsf{comEG}_{\mathsf{pk}}(M) = (b_1, \ldots, b_m)$, where $b_i = \mathsf{EG}^+_{\mathsf{pk}}(M_i \cdot 2^{i-1}, r_i) = (u_{1,i} = {g_1}^{r_i}, e_i = h^{r_i} g^{M_i \cdot 2^{i-1}})$, for $i = 1, \ldots, m$. The homomorphic property of the encryption scheme allows to obtain, from this tuple, the above simple commitment of M

$$C = \mathsf{EG}^+_{\mathsf{pk}}(M, r) = (u_1, e) = (\prod u_{1,i}, \prod e_i) = \prod b_i, \text{ for } r = \sum r_i.$$

We now precise what we mean by "extractability": Here, the commitment will be extractable if the messages M_i are bits (or at least small enough), but we cannot ensure that it will be extractable otherwise. More generally, this leads to a new notion of *$L-$extractable commitments*, which means that we allow the primitive not to be extractable if the message does not belong to a certain language L (*e.g.* the language of encryptions of 0 or 1), which is informally the language of all commitments valid and "of good shape", and is included into the set X of all commitments.

Smooth Hash Functions. For the above protocol, we need a smooth hash system on the language $L = L_1 \cap L_2$, where $L_1 = \{(b_1, \ldots, b_m) \mid \forall i, b_i \in L_{(\mathbf{EG}^+, \rho), 0 \vee 1}\}$, $L_2 = \{(b_1, \ldots, b_m) \mid C = \prod_i b_i \in L_{(\mathbf{EG}^\times, \rho), g^M}\}$, to within a factor (corresponding to the offest 2^{i-1}) with

$L_{(\mathbf{EG}^+, \rho), 0 \vee 1} = L_{(\mathbf{EG}^+, \rho), 0} \cup L_{(\mathbf{EG}^+, \rho), 1}$ $\quad L_{(\mathbf{EG}^+, \rho), 0} = \{C \mid \exists r\ C = \mathsf{EG}^+_{\mathsf{pk}}(0, r)\}$

$L_{(\mathbf{EG}^\times, \rho), g^M} = \{C \mid \exists r\ C = \mathsf{EG}^\times_{\mathsf{pk}}(g^M, r)\}$ $\quad L_{(\mathbf{EG}^+, \rho), 1} = \{C \mid \exists r\ C = \mathsf{EG}^+_{\mathsf{pk}}(1, r)\}$

It is easy to see that this boils down to constructing a smooth hash system corresponding to a conjunction and disjunction of languages, as presented in the previous section.

Certification of Public Keys.
Description. A classical application of extractable commitments is in the certification of public keys (when we want to be sure that a person joining the system actually knows the associated private key). Suppose that a user U owns a pair of secret and public keys, and would like to have the public key certified by the authority. A natural property is that the authority will not certify this public key unless it is sure that the user really owns the related private key, which is usually ensured by a zero-knowledge proof of knowledge: the user knows the private key if a successful extractor exists.

Here we present a construction that possesses the same property without requiring any explicit proof of knowledge, furthermore in a concurrent way since there is no need of any rewinding:

– First, the user sends his public key g^M, along with a bit-by-bit L-extractable commitment of the private key M, i.e. a tuple $\mathsf{comEG}_{\mathsf{pk}}(M) = (b_1, \ldots, b_m)$ as described above, from which one can derive $C = \prod b_i = \mathsf{EG}^{+}_{\mathsf{pk}}(M, r) = \mathsf{EG}^{\times}_{\mathsf{pk}}(g^M, r)$.
– We define the smooth hash system related to the language $L_1 \cap L_2$, where $L_1 = \cap_i L_{1,i}$, with $L_{1,i}$ the language of the tuples where the i-th component b_i is an encryption of 0 or 1, and L_2 is the language of the tuples where the derived $C = \prod b_i$ is an encryption of the public key g^M (under the multiplicative ElGamal, as in Section 4 page 10).
Note that when the tuple $(b_1, \ldots, b_m)$ lies in $L_1 \cap L_2$, it really corresponds to an extractable commitment of the private key M associated to the public key g^M: each b_i encrypts a bit, and can thus be decrypted, which provides the i-th bit of M.
– The authority computes a hash key hk, the corresponding projected key hp on $(b_1, \ldots, b_m)$ and the related hash value Hash on $(b_1, \ldots, b_m)$. It sends hp to U along with $\mathsf{Cert} \oplus \mathsf{Hash}$, where Cert is the expected certificate. Note that if Hash is not large enough, a pseudo-random generator can be used to expand it.
– The user is then able to recover his certificate if and only if he can compute Hash: this value can be computed with the algorithm $\mathsf{ProjHash}$ on $(b_1, \ldots, b_m)$, from hp. But it also requires a witness w proving that the tuple $(b_1, \ldots, b_m)$ lies in $L_1 \cap L_2$.

With the properties of the smooth hash system, if the user correctly computed the commitment, he knows the witness w, and can get the same mask Hash to extract the certificate. If the user cheated, the smoothness property makes Hash perfectly unpredictable: no information is leaked about the certificate.

Security Analysis. Let us outline the security proof of the above protocol. First, the security model is the following: no one can obtain a certificate on a public key if it does not know the associated private key (that is, if no simulator can extract the private key). In other words, the adversary wins if it is able to output (g^M, Cert) and no simulator can produce M.

The formal attack game can thus be described as follows: the adversary $\mathcal{A}$ interacts several times with the authority, by sending public keys and commitments, and asks for the corresponding certificates. It then outputs a pair (g^M, Cert) and wins if no simulator is able to extract M from the transcript.

The simulator works as follows: it is given access to a certification (signing) oracle, and generates a pair of public and private keys $(\mathsf{sk}, \mathsf{pk})$ for the ElGamal encryption. The public key is set as the CRS that defines the commitment scheme. The private key will thus be the extraction trapdoor.

When the simulator receives a certification request, with a public key and a commitment, it first tries to extract the associated private key, granted the extraction trapdoor. In case of success, the simulator asks the signing oracle to provide it with the corresponding certificate on the public key, and complete the process as described in the protocol. However, extraction may fail if the commitments are not well constructed (not in $L_1 \cap L_2$). In such a case, the simulator

sends back a random bit-string of appropriate length. In case of successful extraction, the answer received by the user is exactly the expected one. In case of failure, it is perfectly indistinguishable too since the smoothness property of the hash function would make a perfectly random mask Hash (since the input is not in the language).

After several interactions, $\mathcal{A}$ outputs a pair (g^M, Cert), which is forwarded by the simulator. Either g^M has been queried to the signing oracle, which means that the extraction had succeeded, the simulator knows M and the adversary did not win the attack game, or this is a valid signature on a new message: existential forgery under chosen-message attack.

5 A Conditionally Extractable Equivocable Commitment

In this section, we enhance the previous commitment schemes with equivocability, which is not a trivial task when one wants to keep the extraction property. Note that we first build a malleable extractable and equivocable commitment using the ElGamal-based commitment (see Section 4 page 10), but one can address the non-malleability property by simply building the commitment upon the Cramer-Shoup encryption scheme. All the details of this extension are given in the full version. In the following, if b is a bit, we denote its complement by $\overline{b}$ (i.e., $\overline{b} = 1 - b$). We furthermore denote by $x[i]$ the i^{th} bit of the bit-string x.

Equivocability. Commitments that are both extractable and equivocable seem to be very difficult to obtain. Canetti and Fischlin [12] proposed a solution but for one bit only. Damgård and Nielsen [17] proposed later another construction. But for efficiency reasons, in our specific context, we extend the former proposal. In this section, we thus enhance our previous commitment (that is already L-extractable) to make it equivocable, using the Canetti and Fischlin's approach. Section 5 page 16 will then apply a non-malleable variant of our new commitment together with the associated smooth hash function family in order to build a password-authenticated key exchange protocol with adaptive security in the UC framework [11]. The resulting protocol is reasonably efficient and, in particular, more efficient than the protocol by Barak *et al.* [3], which to our knowledge is the only one achieving the same level of security in the standard model.

Description of the Commitment. Our commitment scheme is a natural extension of Canetti-Fischlin commitment scheme [12], in a bit-by-bit way. It indeed uses the ElGamal public-key encryption scheme, for each bit of the bit-string. Let $(y_1, \ldots, y_m)$ be random elements in G. This commitment is realized in the common reference string model, the CRS ρ contains (G, pk), where pk is an ElGamal public key and the private key is unknown to anybody, except to the commitment extractor. It also includes this tuple $(y_1, \ldots, y_m)$, for which the discrete logarithms in basis g are unknown to anybody, except to the commitment equivocator. Let the input of the committing algorithm be a bit-string $\pi = \sum_{i=1}^{m} \pi_i \cdot 2^{i-1}$. The algorithm works as follows:

– For $i = 1, \ldots, m$, it chooses a random value $x_{i,\pi_i} = \sum_{j=1}^{n} x_{i,\pi_i}[j] \cdot 2^{j-1}$ and sets $x_{i,\overline{\pi}_i} = 0$.
– For $i = 1, \ldots, m$, the algorithm commits to π_i, using the random x_{i,π_i}: $a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i}) = g^{x_{i,\pi_i}} y_i^{\pi_i}$ and defining $\mathbf{a} = (a_1, \ldots, a_m)$.
– For $i = 1, \ldots, m$, it computes the ElGamal commitments (see the previous section) of $x_{i,\delta}$, for $\delta = 0, 1$: $(\mathbf{b}_{i,\delta} = (b_{i,\delta}[j])_j = \mathsf{comEG}_{\mathsf{pk}}(x_{i,\delta})$, where $b_{i,\delta}[j] = \mathsf{EG}^+_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j])$. One can directly extract from the computation of the $b_{i,\delta}[j]$ an encryption $B_{i,\delta}$ of $x_{i,\delta}$: $B_{i,\delta} = \prod_j b_{i,\delta}[j] = \mathsf{EG}^+_{\mathsf{pk}}(x_{i,\delta}, r_{i,\delta})$, where $r_{i,\delta}$ is the sum of the random coins $r_{i,\delta}[j]$.

The entire random string for this commitment is (where n is the bit-length of the prime order q of the group G) $R = (x_{1,\pi_1}, (r_{1,0}[1], r_{1,1}[1], \ldots, r_{1,0}[n], r_{1,1}[n]), \ldots, x_{m,\pi_m}, (r_{m,0}[1], \ldots, r_{m,1}[n]))$. From which, all the values $r_{i,\overline{\pi}_i}[j]$ can be erased, letting the opening data (witness of the committed value) become limited to $\mathsf{w} = (x_{1,\pi_1}, (r_{1,\pi_1}[1], \ldots, r_{1,\pi_1}[n]), \ldots, x_{m,\pi_m}, (r_{m,\pi_m}[1], \ldots, r_{m,\pi_m}[n]))$. The output of the committing algorithm, of the bit-string π, using the random R, is $\mathsf{com}_\rho(\pi; R) = (\mathbf{a}, \mathbf{b})$, where $\mathbf{a} = (a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i}))_i, \mathbf{b} = (b_{i,\delta}[j] = \mathsf{EG}^+_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j]))_{i,\delta,j}$.

Opening. In order to open this commitment to π, the above witness w (with the value π) is indeed enough: one can build again, for all i and j, $b_{i,\pi_i}[j] = \mathsf{EG}^+_{\mathsf{pk}}(x_{i,\pi_i}[j] \cdot 2^{j-1}, r_{i,\pi_i}[j])$, and check them with $\mathbf{b}$. One can then also compute again all the $a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i})$, and check them with $\mathbf{a}$. The erased random elements would help to check the encryptions of zeroes, what we do not want, since the equivocability property will exploit that.

Properties. Let us briefly check the security properties, which are formally proven in the full version. First, because of the perfectly hiding property of the Pedersen commitment, unless some information is leaked about the $x_{i,\delta}[j]$'s, no information is leaked about the π_i's. And granted the semantic security of the ElGamal encryption scheme, the former privacy is guaranteed. Since the Pedersen commitment is (computationally) binding, the a_i's cannot be opened in two ways, but only one pair (π_i, x_{i,π_i}) is possible. Let us now consider the new extended properties:

– (conditional) extractability is provided by the bit-by-bit encryption. With the decryption key sk, one can decrypt all the $b_{i,\delta}[j]$, and get the $x_{i,\delta}$ (unless the ciphertexts contain values different from 0 and 1, which will be one condition for extractability). Then, one can check, for $i = 1, \ldots, m$, whether $a_i = \mathsf{comPed}(0, x_{i,0})$ or $a_i = \mathsf{comPed}(1, x_{i,1})$, which provides π_i (unless none of the equalities is satisfied, which will be another condition for extractability).
– equivocability is possible using the Pedersen commitment trapdoor. Instead of taking a random x_{i,π_i} and then $x_{i,\overline{\pi}_i} = 0$, which specifies π_i as the committed bit, one takes a random $x_{i,0}$, computes $a_i = \mathsf{comPed}(0, x_{i,0})$, but also extracts $x_{i,1}$ so that $a_i = \mathsf{comPed}(1, x_{i,1})$ too (which is possible with the knowledge of discrete logarithm of y_i in basis g, the trapdoor). The rest of

the commitment procedure remains the same, but now, one can open any bit-string for π, using the appropriate x_{i,π_i} and the corresponding random elements (the simulator did not erase).

The Associated Smooth Projective Hash Function. As noticed above, our new commitment scheme is conditionally extractable (one can recover the $x_{i,\delta}$'s, and then the committed value π), under the conditions that all the ElGamal ciphertexts encrypt either 0 or 1, and the a_i is a commitment of either 0 or 1, with random $x_{i,0}$ or $x_{i,1}$.

As before, one wants to make the two hash values (direct computation and the one from the projected key) be the same if the two parties use the same input π and perfectly independent if they use different inputs (smoothness). One furthermore wants to control that each a_i is actually a Pedersen commitment of π_i using the encrypted random x_{i,π_i}, and thus $g^{x_{i,\pi_i}} = a_i/y_i^{\pi_i}$: the extracted x_{i,π_i} is really the private key M related to a given public key g^M that is $a_i/y_i^{\pi_i}$ in our case. Using the same notations as in Section 4 page 10, we want to define a smooth hash system showing that, for all i, δ, j, $b_{i,\delta}[j] \in L_{(\mathbf{EG}^+,\rho),0\vee1}$ and, for all i, $B_{i,\pi_i} \in L_{(\mathbf{EG}^\times,\rho),(a_i/y_i{}^{\pi_i})}$, where $B_{i,\pi_i} = \prod_j b_{i,\pi_i}[j]$.

Combinations of these smooth hashes. Let C be the above commitment of π using randomness R as defined in Section 5 page 13. We now precise the language $L_{\rho,\pi}$, consisting informally of all the valid commitments "of good shape":

$$L_{\rho,\pi} = \left\{ C \;\middle|\; \begin{array}{ll} \exists R \text{ s. t. } C = \mathsf{com}_\rho(\pi, R) & \text{and } \forall i\ \forall j \quad b_{i,\pi_i}[j] \in L_{(\mathbf{EG}^+,\rho),0\vee1} \\ & \text{and } \forall i \quad B_{i,\pi_i} \in L_{(\mathbf{EG}^\times,\rho),a_i/y_i{}^{\pi_i}} \end{array} \right\}$$

The smooth hash system for this language relies on the smooth hash systems described previously, using the generic construction for conjunctions and disjunctions as described in Section 3. The precise definition of this language (which is constructed from conjunctions and disjunctions of simple languages) can be found in the full version, omitting the labels and replacing the Cramer-Shoup encryption $\mathbf{CS}^+$ by the ElGamal one $\mathbf{EG}^+$.

Properties: Uniformity and Independence. With a non-malleable variant of such a commitment and smooth hash function, it is possible to improve the establishment of a secure channel between two players, from the one presented Section 4 page 11. More precisely, two parties can agree on a common key if they both share a common (low entropy) password π. However, a more involved protocol than the one proposed in Section 4 is needed to achieve all the required properties of a password-authenticated key exchange protocol, as it will be explained in Section 5 page 16 and proven in the full version.

Nevertheless, there may seem to be a leakage of information because of the language that depends on the input π: the projected key hp seems to contain some information about π, that can be used in another execution by an adversary. Hence the independence and uniformity notions presented Section 3 page 9, which ensure that hp does not contain any information about π. Proofs of these properties can be found in the full version.

Estimation of the Complexity. Globally, each operation (commitment, projected key, hashing and projected hashing) requires $\mathcal{O}(mn)$ exponentiations in G, with small constants (at most 16).

UC-Secure PAKE with Adaptive Security. The primitive presented above, but using the Cramer-Shoup encryption scheme (as described in the full version) is a non-malleable conditionally extractable and equivocable commitment. We now sketch how to use this new primitive in order to construct the first efficient adaptively-secure password-authenticated key exchange protocol in the UC framework with erasures. For lack of space, all the details can be found in the full version. The passwords are not known at the beginning of the simulation: $\mathcal{S}$ will manage to correct the errors (thanks to the equivocability) but without erasures there would remain clues on how the computations were held, which would give indications on the passwords used.

Our protocol is based on that of Gennaro and Lindell [22]. At a high level, the players in the KOY/GL protocol exchange CCA-secure encryptions of the password, under the public-key found in the common reference string, which are essentially commitments of the password. Then, they compute the session key by combining smooth projective hashes of the two password/ciphertext pairs. The security of this protocol relies on the properties of smoothness and pseudo-randomness of the smooth projective hash function. But as noted by Canetti *et al* in [13], the KOY/GL protocol is not known to achieve UC security: the main issue is that the ideal-model simulator must be able to extract the password used by the adversary before playing, which is impossible if the simulator is the initiator (on behalf of the client), leading to such situation in which the simulator is stuck with an incorrect ciphertext and will not be able to predict the value of the session key.

To overcome this problem, the authors of [13] made the client send a pre-flow which also contains an encryption of the password. The server then sends its own encryption, and finally the client sends another encryption, as well as a zero-knowledge proof showing that both ciphertexts are consistent and encrypt the same password. This time the simulator, playing as the client or the server, is able to use the correct password, recovered from the encrypted value sent earlier by the other party. The pre-flow is never used in the remaining of the protocol, hence the simulator can send a fake one, and simulate the zero-knowledge proof.

Unfortunately, the modification above does not seem to work when dealing with adaptive adversaries, which is the case in which we are interested. This is because the simulator cannot correctly open the commitment when the adversary corrupts the client after the pre-flow has been sent. A similar remark applies to the case in which the server gets corrupted after sending its first message. As a result, in addition to being extractable, the commitment scheme also needs to be equivocable for the simulator to be able to provide a consistent view to the adversary. Since the use of the equivocable and extractable commitment schemes also seems to solve the problem of proving the original Gennaro-Lindell protocol secure in the UC model, we opted to use that protocol as the starting point of our protocol.

These remarks are indeed enough (along with minor modifications) to obtain adaptive security. Thus, our solution essentially consists in using our non-malleable extractable and equivocable commitment scheme in the Gennaro-Lindell protocol when computing the first two flows. As presented in the previous subsections, extractability may be conditional: We include this condition in the language of the smooth hash function (note that the projected keys sent do not leak any information about the password). Additional technical modifications were also needed to make things work and can be found in the full version.

Acknowlegments

This work was supported in part by the French ANR-07-SESU-008-01 PAMPA Project, and the European ECRYPT Project.

References

1. 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, Dec. 2006.
2. W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT 2001*, *LNCS* 2045, pages 119–135. Springer, May 2001.
3. B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin. Secure computation without authentication. In *CRYPTO 2005*, *LNCS* 3621, pages 361–377. Springer, Aug. 2005.
4. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, Oct. 1997.
5. 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.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
7. 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.
8. 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.
9. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *PKC 2003*, *LNCS* 2567, pages 31–46. Springer, Jan. 2003.
10. 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.
11. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.
12. R. Canetti and M. Fischlin. Universally composable commitments. In *CRYPTO 2001*, *LNCS* 2139, pages 19–40. Springer, Aug. 2001.

13. 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.
14. R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *EUROCRYPT 2002*, *LNCS* 2332, pages 337–351. Springer, Apr. / May 2002.
15. 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, Aug. 1998.
16. 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, Apr. / May 2002.
17. I. Damgård and J. B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO 2002*, *LNCS* 2442, pages 581–596. Springer, Aug. 2002.
18. C. Delerablée and D. Pointcheval. Dynamic fully anonymous short group signatures. In *Progress in Cryptology - VIETCRYPT 06*, *LNCS* 4341, pages 193–210. Springer, Sept. 2006.
19. G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and non-interactive non-malleable commitment. In *EUROCRYPT 2001*, *LNCS* 2045, pages 40–59. Springer, May 2001.
20. Y. Dodis and J. Katz. Chosen-ciphertext security of multiple encryption. In *TCC 2005*, *LNCS* 3378, pages 188–209. Springer, Feb. 2005.
21. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*, *LNCS* 196, pages 10–18. Springer, Aug. 1985.
22. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT 2003*, *LNCS* 2656, pages 524–543. Springer, May 2003.
23. O. Goldreich and Y. Lindell. Session-key generation using human passwords only. In *CRYPTO 2001*, *LNCS* 2139, pages 408–432. Springer, Aug. 2001.
24. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
25. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.
26. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
27. Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. In *EUROCRYPT 2005*, *LNCS* 3494, pages 78–95. Springer, May 2005.
28. 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.
29. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT 2006*, *LNCS* 4004, pages 465–485. Springer, May / June 2006.
30. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *12th SODA*, pages 448–457. ACM-SIAM, Jan. 2001.
31. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, *LNCS* 1592, pages 223–238. Springer, May 1999.
32. V. Shoup. ISO 18033-2: An emerging standard for public-key encryption. Dec. 2004. Final Committee Draft.