Smooth Projective Hashing for Conditionally Extractable Commitments

David Pointcheval
Joint work with Michel Abdalla and Céline Chevalier
Ecole normale supérieure, CNRS & INRIA
EPFL – Lausanne – Switzerland
June 18th, 2009

Outline
1 Smooth Projective Hash Functions
   • Definitions
   • Conjunctions and Disjunctions
2 Extractable Commitments
   • Properties
   • Conditional Extractability
   • Application: Certification of Public Keys
3 Equivocable and Extractable Commitments
   • Description
   • Analysis
4 Password-Authenticated Key Exchange

Smooth Projective Hash Functions

Family of Hash Function $H$
Let $\{H\}$ be a family of functions:
- $X$, domain of these functions
- $L$, subset (a language) of this domain such that, for any point $x$ in $L$, $H(x)$ can be computed by using
  - either a secret hashing key $hk$: $H(x) = Hash_L(hk; x)$;
  - or a public projected key $hp$: $H(x) = ProjHash_L(hp; x, w)$

While the former works for all points in the domain $X$, the latter works for $x \in L$ only, and requires a witness $w$ to this fact. There is a public mapping that converts the hashing key $hk$ into the projected key $hp$: $hp = ProjKG_L(hk)$
Properties

For any \( x \in X \), \( H(x) = \text{Hash}_L(hk; x) \)

For any \( x \in L \), \( H(x) = \text{ProjHash}_L(hp; x, w) \) \( w \) witness that \( x \in L \)

Smoothness

For any \( x \not\in L \), \( H(x) \) and \( hp \) are independent

Pseudo-Randomness

For any \( x \in L \), \( H(x) \) is pseudo-random, without a witness \( w \)

The latter property requires \( L \) to be a hard partitioned subset of \( X \):

Hard-Partitioned Subset

\( L \) is a hard-partitioned subset of \( X \) if it is computationally hard to distinguish a random element in \( L \) from a random element in \( X \setminus L \)

Element-Based Projection

**Initial Definition** [Cramer-Shoup EC ’02]

The projected key \( hp \) depends on the hashing key \( hk \) only:

\[
hp = \text{ProjKG}_L(hk)
\]

**New Definition** [Gennaro-Lindell EC ’03]

The projected key \( hp \) depends on the hashing key \( hk \), and \( x \):

\[
hp = \text{ProjKG}_L(hk; x)
\]

Applications: Encryption and Commitments

The input \( x \) can be a ciphertext or a commitment, where the indistinguishability for the hard partitioned subset relies

- either on the semantic security of the encryption scheme
- or the hiding property of the commitment scheme

Examples

**Commitment** [Gennaro-Lindell EC ’02]

\( L_{pk,m} = \{ c \} \) such that \( c \) is a commitment of \( m \)

using public parameter \( pk \):

- there exists \( r \) such that \( c = \text{com}_{pk}(m; r) \)

where \( \text{com} \) is the committing algorithm

**Labeled Encryption** [Canetti-Halevi-Katz-Lindell-MacKenzie EC ’05]

\( L_{pk,(\ell,m)} = \{ c \} \) such that \( c \) is an encryption of \( m \)

with label \( \ell \), under the public key \( pk \):

- there exists \( r \) such that \( c = \mathcal{E}_{pk}^\ell(m; r) \)

where \( \mathcal{E} \) is the encryption algorithm

Smooth Projective Hash Functions [Gennaro-Lindell EC ’03]

A family of smooth projective hash functions \( \text{HASH}(L_{pk,aux}) \), for a language \( L_{pk,aux} \subset X \), onto the set \( G \), based on

- either a labeled encryption scheme with public key \( pk \)
- or on a commitment scheme with public parameters \( pk \)

consists of four algorithms:

\[
\text{HASH}(L_{pk,aux}) = (\text{HashKG}, \text{ProjKG}, \text{Hash}, \text{ProjHash})
\]

Key-Generation Algorithms

- Probabilistic hashing key algorithm:

\[
hk \overset{\$}{\leftarrow} \text{HashKG}()
\]

- Deterministic projection key algorithm

\[
hp = \text{ProjKG}(hk; c)
\]

(where \( c \) is either a ciphertext or a commitment in \( X \))
### Smooth Projective Hash Functions

$$\text{HASH}(L_{pk,aux}) = (\text{HashKG}, \text{ProjKG}, \text{Hash}, \text{ProjHash})$$

#### Hashing Algorithms
- The hashing algorithm $\text{Hash}$ computes,
  - on $c \in X$
  - using the secret hashing key $hk$
  - the value $g = \text{Hash}(hk; c) \in G$
- The projected hashing algorithm $\text{ProjHash}$ computes,
  - on $c \in X$
  - using the projection key $hp$
  - and a witness $w$ to the fact that $c \in L_{pk,aux}$
  - the value $g = \text{ProjHash}(hp; c, w) \in G$

#### Pseudorandomness
If $c \in L_{pk,aux}$, without a witness $w$ of this membership, the two distributions are **computationally** indistinguishable:

$$\{c, hp = \text{ProjKG}(hk; c), g = \text{Hash}(hk; c)\}$$
$$\{c, hp = \text{ProjKG}(hk; c), g \xleftarrow{\$} G\}$$

This requires $L_{pk,aux}$ to be a **hard partitioned subset** of $X$:
- the uniform distributions in $L_{pk,aux}$ and in $X \setminus L_{pk,aux}$ are computationally indistinguishable.

### Correctness
Let $c \in L_{pk,aux}$ and $w$ a witness of this membership.

$$hk \xleftarrow{\$} \text{HashKG}()$$
$$\text{ProjKG}(hk; c) \implies \text{ProjHash}(hp; c, w)$$

### Smoothness
If $c \notin L_{pk,aux}$, the two distributions are **statistically** indistinguishable:

$$\{c, hp = \text{ProjKG}(hk; c), g = \text{Hash}(hk; c)\}$$
$$\{c, hp = \text{ProjKG}(hk; c), g \xleftarrow{\$} G\}$$

with $hk \xleftarrow{\$} \text{HashKG}()$

### ElGamal Encryption

$G = \langle g \rangle$, a cyclic group of prime order $q$.

#### ElGamal Encryption Schemes
Let $pk = h = g^x$ (public key), where $sk = x \xleftarrow{\$} \mathbb{Z}_q$ (private key)

- If $M \in G$, the multiplicative ElGamal encryption is:
  $$\text{EG}_{pk}^x(M; r) = (u_t = g^r, e = h^r M)$$
  which can be decrypted by $M = e/u_t^x$.
- If $M \in \mathbb{Z}_q$, the additive ElGamal encryption is:
  $$\text{EG}_{pk}^x(M; r) = (u_t = g^r, e = h^r g^M)$$
  Note that $\text{EG}_{pk}^x(g^M; r) = \text{EG}_{pk}^x(M; r)$
  It can thus be decrypted as above, but after an additional discrete logarithm computation: $M$ must be small enough.

IND-CPA security = DDH assumption.
Smooth Projective HF Family for ElGamal

The CRS: \( \rho = (G, q, g, pk = h) \)

Language: \( L = \{ (EG_{g^m}, M), C \} = E_{g^m}^G(M; r), r \leftarrow \mathbb{Z}_q \}

- \( L \) is a hard partitioned subset of \( X = G^2 \), under the semantic security of the ElGamal encryption scheme (DDH assumption)
- the random \( r \) is the witness to \( L \)-membership

Algorithms

- \( \text{HashKG}(M) = hk = (\gamma_1, \gamma_3) \leftarrow \mathbb{Z}_q \times \mathbb{Z}_q \)
- \( \text{Hash}(hk; M, C) = (u_i, e) = E_{g^m}^G(M; r) \)
- \( \text{ProjKG}(hk; M, C) = hp = (g)^{\gamma_1} (h)^{\gamma_3} \)
- \( \text{ProjHash}(hp; M, C, r) = (hp)^r \)

Conjunctions and Disjunctions

Conjunction of Languages

A 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 \( i = 1, 2 \):

- \( \text{HashKG}_L(r = r_1 \| r_2) = hk = (hk_1, hk_2) \)
- \( \text{ProjKG}_L(hk; c) = hp = (hp_1, hp_2) \)
- \( \text{Hash}_L(hk; c) = \text{Hash}_1(hk_1; c) \oplus \text{Hash}_2(hk_2; c) \)
- \( \text{ProjHash}_L(hp; c, (w_1, w_2)) = \text{ProjHash}_1(hp_1; c, w_1) \oplus \text{ProjHash}_2(hp_2; c, w_2) \)

- if \( c \) is not in one of the languages, then the corresponding hash value is perfectly random: smoothness
- without one of the witnesses, then the corresponding hash value is computationally unpredictable: pseudo-randomness

Disjunction of Languages

A hash system for the language \( L = L_1 \cup L_2 \) is then defined as follows, if \( c \in L_1 \cup L_2 \) and \( w \) is a witness that \( c \in L_i \) for \( i \in \{1, 2\} \):

- \( \text{HashKG}_L(r = r_1 \| r_2) = hk = (hk_1, hk_2) \)
- \( \text{ProjKG}_L(hk; c) = hp = (hp_1, hp_2, hp) \)
- \( \text{Hash}_L(hk; c) = \text{Hash}_1(hk_1; c) \oplus \text{Hash}_2(hk_2; c) \)
- \( \text{ProjHash}_L(hp; c, w) = \text{ProjHash}_1(hp_1; c, w) \oplus \text{ProjHash}_2(hp_2; c, w) \)

- if \( c \in L_2 \)

\( hp \) helps to compute the missing hash value, if and only if at least one can be computed.
Properties

Contrarily to the original Cramer-Shoup definition, the value of the projected key formally depends on the word $c$. But this dependence maybe invisible.

Uniformity
The projected key may or may not depend on $c$ (and aux), but its distribution does not.

Independence
The projected key does not depend at all on $c$ (and aux).

Outline

1. Smooth Projective Hash Functions
   - Definitions
   - Conjunctions and Disjunctions

2. Extractable Commitments
   - Properties
   - Conditional Extractability
   - Application: Certification of Public Keys

3. Equivocable and Extractable Commitments
   - Description
   - Analysis

4. Password-Authenticated Key Exchange

Commitments

Definition
A commitment scheme is defined by two algorithms:
- the committing algorithm, $C = \text{com}(x; r)$ with randomness $r$, on input $x$, to commit on this input;
- the decommitting algorithm, $(x, D) = \text{decom}(C, x, r)$, where $x$ is the claimed committed value, and $D$ the proof.

Properties
The commitment $C = \text{com}(x; r)$
- reveals nothing about the input $x$: the hiding property
- nobody can open $C$ in two different ways: the binding property

Examples

In both cases, the CRS $\rho$ is $(G, q, g, \text{pk} = h)$, and $(x, D = r) = \text{decom}(C, x, r)$

ElGamal
- $C = \text{comEG}_{\text{pk}}(x; r) = (u_1, e) = \text{EG}^+_\text{pk}(x; r)$, with $r \overset{\$}{\leftarrow} \mathbb{Z}_q$;
- As any IND-CPA encryption scheme, this commitment is perfectly binding and computationally hiding, (DDH assumption)

Pedersen
- $C = \text{comPed}(x; r) = g^x h^r$, with $r \overset{\$}{\leftarrow} \mathbb{Z}_q$;
- This commitment is perfectly hiding and computationally binding, (DL assumption)
**Extractability**

A commitment is extractable if there exists an efficient algorithm, called extractor, capable of generating a new CRS (with similar distribution) such that it can extract $x$ from any $C = \text{com}(x, r)$.

This is possible for computationally hiding commitments only: with an encryption scheme, the decryption key is the extraction key.

**Equivocability**

A commitment is equivocable if there exists an efficient algorithm, called equivocator, capable of generating a new CRS and a commitment (with similar distributions) such that the commitment can be opened in different ways.

This is possible for computationally binding commitments only.

**Non-Malleability**

A commitment is non-malleable if, for any adversary receiving a commitment $C$ of some unknown value $x$ that can generate a valid commitment for a related value $y$, then a simulator could perform the same without seeing the commitment $C$.

This is meaningful for perfectly binding commitments only: with an encryption scheme, IND-CCA2 security level guarantees non-malleability.

**Extended Languages**

**Motivation**

**ElGamal Commitment**

$\text{comEG}_{pk}(x; r) = \text{EG}^{+}_{pk}(x; r)$, is extractable for small $x$ only.

**Example**

If $x \in \{0, 1\}$, any $C(x) = \text{comEG}_{pk}(x; r)$ is extractable.

**Homomorphic Property**

Let us assume $2^k - 1 < q < 2^k$, then for any $x = \sum_{i=0}^{k-1} x_i \times 2^i \in \mathbb{Z}_q$, $C(x) = \{C_i = \text{comEG}_{pk}(x_i; r_i) = \text{EG}^{+}_{pk}(x_i; r_i)\}_{i=0}^{k-1}$, is extractable under the condition that $(x_i)_i \in \{0, 1\}^k$.

Furthermore, $\text{comEG}_{pk}(x; r) = \prod C_i^{2^i}$, for $r = \sum_{i=0}^{k-1} r_i \times 2^i$.

We then define

$L_{(\text{EG}^{+}_{}, \rho), 0 \lor 1} = L_{(\text{EG}^{+}_{}, \rho), 0} \cup L_{(\text{EG}^{+}_{}, \rho), 1}$

To be extractable, $C = (C_i)_i$ has to lie in

$L = \{(C_0, \ldots, C_{k-1}) \mid \forall i, C_i \in L_{(\text{EG}^{+}_{}, \rho), 0 \lor 1}\}$

A conjunction of disjunctions.
Certification of Public Keys

For the certification $\text{Cert}$ of an ElGamal public key $y = g^x$, in most of the protocols, the simulator needs to be able to extract the secret key:

### Classical Process
- the user sends his public key $y = g^x$;
- the user and the authority run a ZK proof of knowledge of $x$;
- if convinced, the authority generates and sends the certificate $\text{Cert}$ for $y$.

But for extracting $x$ in the simulation, the reduction requires a rewinding (that is not always allowed: e.g., in the UC Framework).

### New Process
the user and the authority use a smooth projective hash system for $L$: $\text{HASH}(\text{pk}) = (\text{HashKG, ProjKG, Hash, ProjHash})$
- the user sends his public key $y = g^x$, together with an $L$-extractable commitment $C$ of $x$, with random $r$;
- the authority generates
  - a hashing key $hk \xleftarrow{} \text{HashKG}()$,
  - the corresponding projected key on $C$, $hp = \text{ProjKG}(hk, C)$
  - the hash value $\text{Hash} = \text{Hash}(hk; C)$
- and sends $hp$ along with $\text{Cert} \oplus \text{Hash}$;
- The user computes $\text{Hash} = \text{ProjHash}(hp; C, r)$, and gets $\text{Cert}$.

Commitment and Smooth Projective HF

The authority sends $hp$ along with $\text{Cert} \oplus \text{Hash}$

### Analysis: Correct Commitment
- If the user correctly computed the commitment ($C \in L$)
  - he knows the witness $r$, and can get the same mask $\text{Hash}$;
  - the simulator can extract $x$, granted the $L$-extractability.

### Analysis: Incorrect Commitment
- If the user cheated ($C \not\in L$)
  - the simulator is not guaranteed to extract anything;
  - but, the smoothness property makes $\text{Hash}$ perfectly unpredictable: no information is leaked about the certificate.

Outline

1. Smooth Projective Hash Functions
   - Definitions
   - Conjunctions and Disjunctions

2. Extractable Commitments
   - Properties
   - Conditional Extractability
   - Application: Certification of Public Keys

3. Equivocable and Extractable Commitments
   - Description
   - Analysis

4. Password-Authenticated Key Exchange
A First Approach

To get both extractability and equivocability (at the same time), one can combine perfectly hiding and perfectly binding commitments:

- Pedersen's commitment is perfectly hiding
- ElGamal's commitment is perfectly binding

**Notations**

If \( b \) is a bit, we denote its complement by \( \overline{b} \).

\( x[i] \) denotes the \( i \)th bit of the bit-string \( x \).

**Description**

Extractable and Equivocable Commitment

In order to commit to \( \pi \), for \( i = 1, \ldots, m \),

- one chooses a random value \( x_{i,\pi_i} = \sum_{j=1}^{\pi_i} x_{i,j} \cdot 2^{j-1} \in \mathbb{Z}_q \)
  and sets \( x_{i,\pi_i} = 0 \)
- one commits to \( \pi_i \), using the random \( x_{i,\pi_i} \):
  \[
  a_i = \text{comPed}(\pi_i, x_{i,\pi_i}) = g^{x_{i,\pi_i}} y_i^{\pi_i}
  \]

This defines \( a = (a_1, \ldots, a_m) \)

- one commits to \( x_{i,\delta_i} \), for \( \delta = 0, 1 \): \( b_{i,\delta} = (b_{i,\delta}[j]) = \text{comEG}_p(x_{i,\delta}), \)
  where \( b_{i,\delta}[j] = \text{EG}_p(x_{i,[j]} \cdot 2^{j-1}, r_{i,[j]}) \)

Then, \( B_{i,\delta} = \prod_j b_{i,\delta}[j] = \text{EG}_p^+(x_{i,\delta}, r_{i,\delta}) \), where \( r_{i,\delta} = \sum_j r_{i,[j]} \).

**Extractable**

Random string:

\[
R = (x_{1,\pi_1}, (r_{1,0}[j], r_{1,1}[j]), \ldots, x_{m,\pi_m}, (r_{m,0}[j], r_{m,1}[j]))
\]

Commitment: \( \text{comPed}(\pi; R) = (a, b) \)

where:

- \( a = (a_i = \text{comPed}(\pi_i, x_{i,\pi_i})) \)
- \( b = (b_{i,\delta}[j] = \text{EG}_p^+(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j]))_{i,\delta,j} \)

**Equivocable**

Witness: the values \( r_{i,\pi_i}[j] \) can be erased,

\[
w = (x_{1,\pi_1}, (r_{1,\pi_1}[j]), \ldots, x_{m,\pi_m}, (r_{m,\pi_m}[j]))
\]

Opening: given the above witness, and the value \( \pi \)

\[
\forall i, j : b_{i,\pi_i}[j] \overset?= \text{EG}_p^{+}(x_{i,\pi_i}[j] \cdot 2^{j-1}, r_{i,\pi_i}[j])
\]

\[
\forall i : a_i \overset?= \text{comPed}(\pi_i, x_{i,\pi_i})
\]
Properties

\[ \text{com}_\rho(\pi; R) = (a, b) : \quad a = (a_i = \text{comPed}(\pi_i, x_i, \pi_i))_{i} \]
\[ b = (b_{i,\delta}[j] = \text{EG}_{\text{pk}}^{+}(x_{i,\delta}[j], 2^{i-1}, r_{i,\delta}[j]))_{i,\delta,j} \]

Intuition

- Granted the perfectly hiding property of the Pedersen commitment, without any information on the \(x_{i,\delta}[j]\)'s, no information is leaked about the \(\pi_i\)'s.
- Granted the semantic security of the ElGamal encryption scheme, the former privacy on the \(x_{i,\delta}[j]\)'s is guaranteed.
- Granted the computationally binding property of the Pedersen commitment, the \(a_i\)'s cannot be open in two ways.

Equivocability

Normal Procedure

- One takes a random \(x_{i,\pi_i}\) and then \(x_{i,\pi_i} = 0\), which specifies \(\pi_i\).
- One commits on \(\pi_i\) using randomness \(x_{i,\pi_i}\).
- One encrypts both \(x_{i,\pi_i}\) and \(x_{i,\pi_i}\), bit-by-bit.

Equivocable Procedure

- Granted the Pedersen commitment trapdoor
  - one takes a random \(x_{i,0}\) and extracts \(x_{i,1}\) such that
    - \(a_i = \text{comPed}(0, x_{i,0}) = \text{comPed}(1, x_{i,1})\)
  - the rest of the commitment procedure remains the same.
- One can open any bit-string for \(\pi_i\), using the appropriate \(x_{i,\pi_i}\) and the corresponding random elements (no erasure).

Conditional Extractability

Constraints

- bit-by-bit encryption of the \(x_{i,\delta}[j]\):
  - with the ElGamal decryption key, one decrypts all the \(b_{i,\delta}[j]\), and gets the \(x_{i,\pi_i}\) (unless the plaintexts are different to 0 and \(2^{i-1}\)).
- then, one can confirm, for \(i = 1, \ldots, m\), whether
  - \(a_i = \text{comPed}(0, x_{i,0})\) or \(a_i = \text{comPed}(1, x_{i,1})\), which provides \(\pi_i\) (unless none of the equalities is satisfied).

The above conditions define the language for extractability:

\[
L_{\rho,\pi} = \left\{ C \left| \begin{array}{l}
\exists R \text{ such that } C = \text{com}_\rho(\pi, R) \\
\text{and } \forall i \lor b_{i,\pi_i}[j] \in L_{(\text{EG}^+,\rho),0\lor1} \\
\text{and } \forall i \lor B_{i,\pi_i} \in L_{(\text{EG}^-,\rho),1\lor1}
\end{array} \right. \right\}
\]

Non-Malleability

Using a non-malleable encryption scheme (Cramer-Shoup), one can make the commitment non-malleable:

- Random string:
  \[
  R = (x_{1,\pi_1}, (r_{1,0}[j], r_{1,1}[j]), \ldots, x_{m,\pi_m}, (r_{m,0}[j], r_{m,1}[j]))
  \]
- Commitment: \(\text{com}_\rho(\pi; R) = (a, b)\)
  
  where
  \[
  a = (a_i = \text{comPed}(\pi_i, x_{i,\pi_i}))_{i}
  \]
  \[
  b = (b_{i,\delta}[j] = \text{CS}_{\text{pk}}^{+}(x_{i,\delta}[j], 2^{i-1}, r_{i,\delta}[j]))_{i,\delta,j}
  \]
- Opening: given the above witness, and the value \(\pi\)
  \[
  \forall i,j : b_{i,\pi_i}[j] \equiv \text{CS}_{\text{pk}}^{+}(x_{i,\pi_i}[j], 2^{i-1}, r_{i,\pi_i}[j])
  \]
  \[
  \forall i : a_i \equiv \text{comPed}(\pi_i, x_{i,\pi_i})
  \]
**Outline**

1. **Smooth Projective Hash Functions**
   - Definitions
   - Conjunctions and Disjunctions

2. **Extractable Commitments**
   - Properties
   - Conditional Extractability
   - Application: Certification of Public Keys

3. **Equivocable and Extractable Commitments**
   - Description
   - Analysis

4. **Password-Authenticated Key Exchange**

---

**Introduction**

Password-Authenticated Key Exchange (PAKE) involves two players who want to establish a common secret key, using a short password as authentication means. Exhaustive search is possible:

- **Online dictionary attack:** Elimination of one candidate per attack. This is unavoidable.
- **Offline dictionary attack:** The transcript of a communication helps to eliminate one or a few candidates. This is avoidable, and should be avoided.

One wants to prove that eliminating one candidate per active attempt is the best attack.

---

**Previous Schemes**

- **Scheme I**
  - [Katz-Ostrovsky-Yung EC '01, Gennaro-Lindell C '03]
  - Not UC secure!

- **Scheme II**
  - [Canetti-Halevi-Katz-Lindell-MacKenzie EC '05]
  - Add of a first commitment round: for non-adaptive UC security.

---

**Password-Authenticated Key Exchange**

**Definition**

Two players want to establish a common secret key, using a short password as authentication means: exhaustive search is possible:

- **Online dictionary attack:** Elimination of one candidate per attack. This is unavoidable.
- **Offline dictionary attack:** The transcript of a communication helps to eliminate one or a few candidates. This is avoidable, and should be avoided.

One wants to prove that eliminating one candidate per active attempt is the best attack.

---

**Scheme I**

<table>
<thead>
<tr>
<th>$P_1$ (client)</th>
<th>CRS: $pke$</th>
<th>$P_2$ (server)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_2 = E_{pke}(pw, r_2)$</td>
<td>$c_1, vk$</td>
<td>$(sk, vk) \rightarrow \text{sigKey}($)$</td>
</tr>
<tr>
<td>$hk \leftarrow H$</td>
<td>$c_1 = E_{pke}(pw, r_1)$</td>
<td></td>
</tr>
<tr>
<td>$hp \leftarrow \alpha(hk; c_1)$</td>
<td>$c_2, hp$</td>
<td></td>
</tr>
<tr>
<td>$h_{hp'} \leftarrow H$</td>
<td>$\sigma \leftarrow \text{Sign}_{k_1}(c_2, hp, hp')$</td>
<td></td>
</tr>
<tr>
<td>if (Verify$_{sk}$ ($(c_2, hp, hp'), \sigma = 1$)</td>
<td>session-key $\leftarrow H_{sk}(c_1, pw)$</td>
<td>session-key $\leftarrow h_{hp}(c_1, pw, r_1)$</td>
</tr>
<tr>
<td>$+$ $h_{hp}(c_2, pw, r_2)$</td>
<td>$+$ $H_{sk}(c_2, pw)$</td>
<td></td>
</tr>
</tbody>
</table>

Not UC secure!

**Scheme II**

<table>
<thead>
<tr>
<th>$P_1$ (client)</th>
<th>CRS: $pke$</th>
<th>$P_2$ (server)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_0 \rightarrow E_{pke}(pw, r_0)$</td>
<td>$c_0$</td>
<td>$(sk, vk) \rightarrow \text{sigKey}($)$</td>
</tr>
<tr>
<td>$c_1, ek$</td>
<td>$c_1 = E_{pke}(pw, r_1)$</td>
<td></td>
</tr>
<tr>
<td>$c_2 = E_{pke}(pw, r_2)$</td>
<td>$c_2, hp$</td>
<td></td>
</tr>
<tr>
<td>$hk \leftarrow H$</td>
<td>$h_{hp'} \leftarrow \alpha(hk', c_2)$</td>
<td></td>
</tr>
<tr>
<td>$hp \leftarrow \alpha(hk; c_1)$</td>
<td>$\sigma \leftarrow \text{Sign}_{k_1}(c_2, hp, hp')$</td>
<td></td>
</tr>
<tr>
<td>ZKP($c_0 \equiv c_2$)</td>
<td>session-key $\leftarrow H_{sk}(c_1, pw)$</td>
<td>session-key $\leftarrow h_{hp}(c_1, pw, r_1)$</td>
</tr>
<tr>
<td>$+$ $h_{hp}(c_2, pw, r_2)$</td>
<td>$+$ $H_{sk}(c_2, pw)$</td>
<td></td>
</tr>
</tbody>
</table>

Add of a first commitment round: for non-adaptive UC security.
If \( A \) plays the client role:
- \( S \) can extract the committed password, and check it granted the TestPwd query
- password valid: \( S \) uses it
- else: dummy password

\[ \implies \text{perfect simulation} \]

If \( A \) plays the server role:
- \( S \) does not know the password: dummy password in \( c_0 \)
- when \( A \) sends its commitment \( c_1 \), \( S \) extracts the password and checks it granted the TestPwd query
- if the password is invalid, \( S \) follows with the dummy password
- else, \( S \) uses the correct password in \( c_2 \) and simulates the ZKP

What about if \( A \) corrupts the client right after \( c_0 \)?
\( S \) gets the correct password, but cannot open \( c_0 \) correctly!
\[ \implies \text{security against static-corruptions only (before the session starts)} \]

Non-malleable, L-extractable, equivocable commitment provides adaptive security to the KOY/GL construction

---

### Conclusion

Smooth Projective Hash Functions for Complex Languages

Various Applications
- in place of some ZK proofs
- conditional secure channels
- adaptive security in UC PAKE