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 Adaptive Security and UC PAKE
   - Universal Composability
   - Previous Schemes
   - Our Scheme

Smooth Projective Hashing for Conditionally Extractable Commitments

David Pointcheval

Joint work with Michel Abdalla and Céline Chevalier
Ecole normale supérieure, CNRS & INRIA

NTT – Tokyo – Japan
April 10th, 2009

Definitions
Smooth Projective Hash Functions

\[ \{ H \} \]

- 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 \( h_k \): \( H(x) = \text{Hash}_L(h_k; x) \);
    - or a public projected key \( h_p \): \( H(x) = \text{ProjHash}_L(h_p; x, w) \)

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 \( h_k \) into the projected key \( h_p \): \( h_p = \text{ProjKG}_L(h_k) \).
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 $X \setminus L$

Examples

Commitment

$L_{pk,m} = \{c\}$ 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

$L_{pk,(\ell,m)} = \{c\}$ 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

A family of smooth projective hash functions $\text{HASH}(pk)$,
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:

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

Key-Generation Algorithms

- Probabilistic hashing key algorithm:
  \[ hk \xleftarrow{\$} \text{HashKG}(pk, aux) \]
- Deterministic projection key algorithm
  \[ hp = \text{ProjKG}(hk, pk, aux, c) \]
  (where $c$ is either a ciphertext or a commitment in $X$)
Smooth Projective Hash Functions

**Definitions**

- **HASH**(pk) = (HashKG, ProjKG, Hash, ProjHash)

**Properties**

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

\[
\{pk, aux, c, hp = ProjKG(hk; pk, aux, c), g = Hash(hk; pk, aux, c)\} \not\approx \{pk, aux, c, hp = ProjKG(hk; pk, aux, c), g \leftarrow G\}
\]

This requires \( L_{pk,aux} \) to be a hard partitioned subset of \( X \):
- the uniform distributions in \( L_{pk,aux} \) and in \( X \backslash L_{pk,aux} \)
- are computationally indistinguishable

- **Correctness**
  Let \( c \in L_{pk,aux} \) and \( w \) a witness of this membership.

\[
hk \leftarrow HashKG(pk, aux) \text{ and } hp = ProjKG(hk; pk, aux, c) \implies \text{Hash}(hk; pk, aux, c) = \text{ProjHash}(hp; pk, aux, c; w)
\]

- **Smoothness**

If \( c \not\in L_{pk,aux} \), the two distributions are statistically indistinguishable:

\[
\{pk, aux, c, hp = ProjKG(hk; pk, aux, c), g = Hash(hk; pk, aux, c)\} \not\approx \{pk, aux, c, hp = ProjKG(hk; pk, aux, c), g \leftarrow G\}
\]

**ElGamal Encryption**

**Definitions**

- \( G = \langle g \rangle \), a cyclic group of prime order \( q \).

**ElGamal Encryption Schemes**

- Let \( pk = h = g^x \) (public key), where \( sk = x \leftarrow \mathbb{Z}_q \) (private key)
  
  - If \( M \in G \), the multiplicative ElGamal encryption is:
    - \( EG^+_p(k)(M; r) = (u_1 = g^r, e = h^r M) \)
    - which can be decrypted by \( M = e / u_1^x \).
  
  - If \( M \in \mathbb{Z}_q \), the additive ElGamal encryption is:
    - \( EG^-_p(k)(M; r) = (u_1 = g^r, e = h^r g^M) \)
    - Note that \( EG^+_p(g^M; r) = EG^-_p(M; r) \)
    - It can thus be decrypted as above, but after an additional discrete logarithm computation: \( M \) must be small enough.

**Properties**

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

The CRS: $\rho = (G, q, g, pk = h)$
Language: $L = L_{(EG^{+}, \rho), M} = \{ C = (u, e) = EG_{pk}^{+}(M; r), r \leftarrow \mathbb{Z}_{q} \}$

- $L$ 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

Definitions

Conjunctions and Disjunctions

Notations

We assume that $G$ possesses a group structure, and we denote by $\oplus$ the commutative law of the group (and by $\ominus$ the opposite operation)
We assume to be given two smooth hash systems $SHS_{1}$ and $SHS_{2}$, on the sets $G_{1}$ and $G_{2}$ (included in $G$) corresponding to the languages $L_{1}$ and $L_{2}$ respectively:

\[ SHS_{i} = \{ HashKG_{i}, ProjKG_{i}, Hash_{i}, ProjHash_{i} \} \]

Let $c \in X$, and $r_{1}$ and $r_{2}$ two random elements:

- $hk_{1} = HashKG_{1}(\rho, aux, r_{1})$
- $hk_{2} = HashKG_{2}(\rho, aux, r_{2})$
- $hp_{1} = ProjKG_{1}(hk_{1}; \rho, aux, c)$
- $hp_{2} = ProjKG_{2}(hk_{2}; \rho, aux, c)$

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 $i = 1, 2$:

\[
\text{HashKG}_{L}(\rho, aux, r = r_{1} \parallel r_{2}) = hk = (hk_{1}, hk_{2}) \\
\text{ProjKG}_{L}(hk; \rho, aux, c) = hp = (hp_{1}, hp_{2}) \\
\text{Hash}_{L}(hk; \rho, aux, c) = \text{Hash}_{1}(hk_{1}; \rho, aux, c) \\
\oplus \text{Hash}_{2}(hk_{2}; \rho, aux, c) \\
\text{ProjHash}_{L}(hp; \rho, aux, c; (w_{1}, w_{2})) = \text{ProjHash}_{1}(hp_{1}; \rho, aux, c; w_{1}) \\
\oplus \text{ProjHash}_{2}(hp_{2}; \rho, aux, c; w_{2})
\]

- 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 $i \in \{ 1, 2 \}$:

\[
\text{HashKG}_{L}(\rho, aux, r = r_{1} \parallel r_{2}) = hk = (hk_{1}, hk_{2}) \\
\text{ProjKG}_{L}(hk; \rho, aux, c) = hp = (hp_{1}, hp_{2}, hp_{\Delta}) \\
\text{where } hp_{\Delta} = \text{Hash}_{1}(hk_{1}; \rho, aux, c) \\
\oplus \text{Hash}_{2}(hk_{2}; \rho, aux, c) \\
\text{Hash}_{L}(hk; \rho, aux, c) = \text{Hash}_{1}(hk_{1}; \rho, aux, c) \\
\text{ProjHash}_{L}(hp; \rho, aux, c; w) = \text{ProjHash}_{1}(hp_{1}; \rho, aux, c; w) \text{ if } c \in L_{1} \\
\text{or } hp_{\Delta} \oplus \text{ProjHash}_{2}(hp_{2}; \rho, aux, c; w) \text{ if } c \in L_{2}
\]

$hp_{\Delta}$ 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 Adaptive Security and UC PAKE
   - Universal Composability
   - Previous Schemes
   - Our Scheme

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 $(G, q, g, pk = h)$, and $(x, D = r) = \text{decom}(C, x, r)$.

**ElGamal**

- $C = \text{comEG}(pk)(x; r) = (u, e) = E_{pk}^+(x; r)$, with $r \leftarrow Z_q$;
- 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 $r \leftarrow Z_q$;
- This commitment is perfectly hiding and computationally binding, (DL assumption)
Additional Properties

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.

Conditional Extractability

Motivation

ElGamal Commitment
\[ \text{com}_E^G_{pk}(x; r) = E^G_{pk}(x; r) \], is extractable for small \(x\) only.

Example
If \(x \in \{0, 1\}\), any \(C(x) = \text{com}_E^G_{pk}(x; r)\) is extractable.

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

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

Extended Languages

Extended Commitments
If \(x \in \{0, 1\}\), any \(C(x) = \text{com}_E^G_{pk}(x; r)\) is extractable.

We then define
\[ L_{(E^G^+, \rho), 0} \cup L_{(E^G^+, \rho), 1} = L_{(E^G^+, \rho), 0} \cup L_{(E^G^+, \rho), 1} \]

To be extractable, \(C = (C_i)\) has to lie in
\[ L = \{ (C_0, \ldots, C_{k-1}) \mid \forall i, C_i \in L_{(E^G^+, \rho), 0} \} \]
Certification of Public Keys

For the certification 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 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 $L$:
  - $\text{HASH}(\text{pk}) = (\text{HashKG}, \text{ProjKG}, \text{Hash}, \text{ProjHash})$
- 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 $hk \leftarrow \text{HashKG}()$
  - 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 ($C \notin L$)
  - 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. Adaptive Security and UC PAKE
   - Universal Composability
   - Previous Schemes
   - Our Scheme
A First Approach [Canetti-Fischlin C ’01]

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 \( x \)

Extractable and Equivocable Commitment

Common Reference String Model

The commitment is realized in the common reference string model: the CRS \( \rho \) contains

- \((G, pk)\), where \( pk \) is an ElGamal public key and the private key is unknown to anybody (except to the commitment extractor)
- the tuple \((y_1, \ldots, y_m) \in G^m\), 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

\[
\pi = \sum_{i=1}^{m} \pi_i \cdot 2^{i-1}
\]

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

- one chooses a random value \( x_{i,\pi_i} = \sum_{j=1}^{n} x_{i,\pi_i[j]} \cdot 2^{j-1} \in \mathbb{Z}_q \)
  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 \( \delta = 0, 1 \): \( b_{i,\delta} = (b_{i,\delta}[j]) = \text{comEG}_{pk}(x_{i,\delta}) \)
  where \( b_{i,\delta}[j] = \text{EG}_{pk}^+(x_{i,\delta} \cdot 2^{j-1}, r_{i,\delta}[j]) \)

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

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: \( \text{com}_{pk}(\pi; R) = (a, b) \)

where
\[
\begin{align*}
  a &= (a_i = \text{comPed}(\pi_i, x_{i,\pi_i}))_i \\
  b &= (b_{i,0}[j] = \text{EG}_{pk}^+(x_{i,0}[j] \cdot 2^{j-1}, r_{i,0}[j]), b_{i,1}[j] = \text{EG}_{pk}^+(x_{i,1}[j] \cdot 2^{j-1}, r_{i,1}[j]))_{i,\delta,j}
\end{align*}
\]

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 \( \pi \)

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

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

comₚ(π; R) = (a, b) : a = comPed(πᵢ, xᵢ,πᵢ))ᵢ
b = (bᵢ,δᵢ[j] = EG⁺ₚk(xᵢ,δᵢ[j] · 2⁻ᵢ, rᵢ,δᵢ[j]))ᵢ,δ,j

Intuition
- Granted the perfectly hiding property of the Pedersen commitment, without any information on the xᵢ,δᵢ[j]'s, no information is leaked about the πᵢ's
- Granted the semantic security of the ElGamal encryption scheme, the former privacy on the xᵢ,δᵢ[j]'s is guaranteed
- Granted the computationally binding property of the Pedersen commitment, the aᵢ's cannot be open in two ways

Conditional Extractability

Constraints
- bit-by-bit encryption of the xᵢ,δ[i]: with the ElGamal decryption key, one decrypts all the bᵢ,δ[i], and gets the xᵢ,πᵢ (unless the plaintexts are different to 0 and 2⁻ᵢ)
- then, one can confirm, for i = 1, . . . , m, whether aᵢ = comPed(0, xᵢ,0) or aᵢ = comPed(1, xᵢ,1), which provides πᵢ (unless none of the equalities is satisfied)

The above conditions define the language for extractability:

Lₚ,π = \{ C \mid \exists R such that C = comₚ(π, R) and \forall i \vee bᵢ,δᵢ[i] ∈ L(EG⁺, πk),0 \vee 1 and \forall i Bᵢ,πᵢ \in L(EG⁺, δk) \}

Non-Malleability

Using a non-malleable encryption scheme (Cramer-Shoup), one can make the commitment non-malleable:
- Random string:
  \[ R = (xᵢ,πᵢ, (rᵢ,0[i], rᵢ,1[i]), . . . , xᵢ,m,πm, (rᵢ,m,0[i], rᵢ,m,1[i])) \]
- Commitment: comₚ(π; R) = (a, b)

where
  \[ aᵢ = \text{comPed}(πᵢ, xᵢ,πᵢ) \]
  \[ bᵢ,δᵢ[j] = \text{CS⁺ₚk}(xᵢ,δᵢ[j] · 2⁻ᵢ, rᵢ,δᵢ[j])ᵢ,δ,j \]

Opening: given the above witness, and the value π

\[ \forall i, j : bᵢ,πᵢ[j] \stackrel{?}{=} \text{CS⁺ₚk}(xᵢ,πᵢ[j] · 2⁻ᵢ, rᵢ,πᵢ[j]) \]
\[ \forall i : aᵢ \stackrel{?}{=} \text{comPed}(πᵢ, xᵢ,πᵢ) \]
The protocol $\Pi$ securely realizes $\mathcal{F}$, if $\forall$ adversary $A$, $\exists$ a simulator $S$ such that no environment $Z$ can tell whether it interacts with a run of $\Pi$ with $A$ or with an ideal run with $\mathcal{F}$ and $S$.

$S$ has to simulate the view generated by the honest users without the private inputs.
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

- **on-line dictionary attack**: Elimination of one candidate per attack. This is unavoidable.
- **off-line 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.

TestPwd to model on-line dictionary attacks (once per session)

Ideal Functionality

**Functionality $F_{\text{pwd}}$**

The functionality $F_{\text{pwd}}$ is parameterized by a security parameter $k$. It interacts with an adversary $S$ and a set of parties via the following queries:

- **Upon receiving a query** $(\text{NewSession}, \text{sid}, P_i, P_j, \text{pw}, \text{role})$ **from party** $P_i$:
  
  Send $(\text{NewSession}, \text{sid}, P_i, P_j, \text{role})$ to $S$. In addition, if this is the first NewSession query or if this is the second NewSession query and there is a record $(P_j, P_i, \text{pw}')$ then record $(P_i, P_j, \text{pw})$ and mark this record fresh.

- **Upon receiving a query** $(\text{TestPwd}, \text{sid}, P_i, \text{pw}')$ **from the adversary** $S$:
  
  If there is a record of the form $(P_i, P_j, \text{pw})$ which is fresh, then do: If $\text{pw} = \text{pw}'$, mark the record compromised and reply to $S$ with “correct guess”. If $\text{pw} \neq \text{pw}'$, mark the record interrupted and reply with “wrong guess”.

- **Upon receiving a query** $(\text{NewKey}, \text{sid}, P_i, sk)$ **from $S$, where** $(\text{sid}) = k$:

  If there is a record of the form $(P_i, P_j, \text{pw})$, and this is the first NewKey query for $P_i$, then:
  
  - If this record is compromised, or either $P_i$ or $P_j$ is corrupted, then output $(\text{sid}, \text{sk})$ to player $P_i$.
  - If this record is fresh, and there is a record $(P_j, P_i, \text{pw}')$ with $\text{pw}' = \text{pw}$, and a key $sk$ was sent to $P_j$, and $(P_j, P_i, \text{pw})$ was fresh at the time, then output $(\text{sid}, sk')$ to $P_j$.

  In any other case, pick a new random key $sk'$ of length $k$ and send $(\text{sid}, sk')$ to $P_j$.

  Either way, mark the record $(P_i, P_j, \text{pw})$ as completed.

**Figure 2**: The password-based key-exchange functionality $F_{\text{pwd}}$

Analysis

Security in the classical framework:

- **Commitment to an incorrect password**: smoothness leads to a perfectly random session key.
- **Replay of a commitment**: pseudo-randomness leads to a computationally random session key (witness unknown).

Simulation of the honest players: use of a dummy password

- indistinguishable, unless $A$ committed to the correct password: $S$ cannot compute the correct key $\implies S$ aborts
- in the UC framework, $Z$ sees the difference between a real-execution and the simulation: when $A$ wins, $S$ aborts

Because of the short password, this is not negligible.
**Analysis**

If $A$ plays the server role:
- $S$ can extract the committed password, and check it granted the TestPwd query
- password valid: $S$ uses it
- else: dummy password

$\implies$ perfect simulation

If $A$ plays the client role:
- $S$ does not know yet the password sent by $A$: dummy password
- when $A$ sends its commitment, $S$ extracts the password and checks it granted the TestPwd query
- if the password is invalid, $S$ follows with the dummy password
- else, $S$ is stuck

**Scheme II**

**Previous Schemes**

**Adaptive Corruption**

If $A$ plays the server role:
- $S$ does not know the password: dummy password in $c_0$
- $S$ extracts the password from $c_1$, checks it (TestPwd query)
- if invalid: $S$ follows with the dummy password in $c_2$
- 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!

$\implies$ security against static-corruptions only (before the session starts)

**Non-malleable, L-extractable, equivocable commitment provides adaptive security**
Adaptively Secure UC-PAKE

**Our Scheme**

(U1) $(VK_I, SK_I) \leftarrow SKG$
$\ell_I = I \circ \text{ssid} \circ VK_I$
$com_I = \text{com}_{\ell_I}(\ell_I, pw_I; R_I)$

(S2) (publicly) checks the validity of $com_I$
$(VK_I, SK_I) \leftarrow SKG$
$\ell_I = I \circ \text{ssid} \circ VK_I$
$com_I = \text{com}_{\ell_I}(\ell_I, pw_I; R_I)$
$hp_I = \text{ProjKG}(hk_I; \ell_I, pw_I)$
$sk_I = \text{ProjHash}(hp_I; \ell_I, pw_I, com_I)$
$\sigma_I = \text{Sign}(SK_I; (com_I, com_J, hp_I, hp_J), \ell_I)$
$\text{if } \text{Ver}(VK_I, (com_I, com_J, hp_I, hp_J), \sigma_I) = 0$
outputs $(sid, ssid, sk_I)$
erases everything
sets the session as **accepted**

(U3) (publicly) checks the validity of $com_J$
$hk_J = \text{HashKG}(\rho; (\ell_J, pw_J), r_J)$
$hp_J = \text{ProjKG}(hk_J; \ell_J, pw_J)$
$sk_J = \text{ProjHash}(hp_J; \ell_J, pw_J, com_J)$
$\sigma_J = \text{Sign}(SK_J; (com_I, com_J, hp_I, hp_J), \ell_J)$
$\text{if } \text{Ver}(VK_J, (com_I, com_J, hp_I, hp_J), \sigma_J) = 0$
outputs $(sid, ssid, sk_J)$
erases everything
sets the session as **accepted**

(S4) aborts if
$\text{Ver}(VK_I, (com_I, com_J, hp_I, hp_J), \sigma_I) = 0$
$\sigma_J = \text{Sign}(SK_J; (com_I, com_J, hp_I, hp_J), \ell_J)$
$sk_J = \text{ProjHash}(hp_J; \ell_J, pw_J, com_J)$
$\text{if } \text{Ver}(VK_J, (com_I, com_J, hp_I, hp_J), \sigma_J) = 0$
outputs $(sid, ssid, sk_I)$
erases everything
sets the session as **accepted**

**Conclusion**

Smooth Projective Hash Functions for Complex Languages

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