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

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

Definitions

Smooth Projective Hash Functions [Cramer-Shoup EC ‘02]

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

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 $hp$: $hp = \text{ProjKG}_L(hk)$
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 $X \setminus L$

Examples

**Commitment**

$L_{pk,m} = \{ c \}$ such that $c$ is a commitment of $m$
using public parameter $pk$:
\[
\text{there exists } r \text{ such that } c = \text{com}_{pk}(m; r) \\
\text{where com is the committing algorithm}
\]

**Labeled Encryption**

$L_{pk,(\ell,m)} = \{ c \}$ such that $c$ is an encryption of $m$
with label $\ell$, under the public key $pk$:
\[
\text{there exists } r \text{ such that } c = \mathcal{E}_{pk}^\ell(m; r) \\
\text{where } \mathcal{E} \text{ is the encryption algorithm}
\]

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

Definitions

Smooth Projective Hash Functions

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

Hashing Algorithms

- The hashing algorithm Hash computes,
  - on \( c \in X \)
  - using the secret hashing key \( hk \)
  - the value \( g = \text{Hash}(hk; pk, aux, c) \in G \)
- The projected hashing algorithm 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 \( g = \text{ProjHash}(hp; pk, aux, c; w) \in G \)

Correctness

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

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

Smoothness

If \( c \notin L_{pk, aux} \), the two distributions are statistically indistinguishable:

\[
\{pk, aux, c, hp = \text{ProjKG}(hk; pk, aux, c), g = \text{Hash}(hk; pk, aux, c)\} \quad \{pk, aux, c, hp = \text{ProjKG}(hk; pk, aux, c), g \leftarrow G\}
\]

ElGamal Encryption

Definitions

ElGamal Encryption

\[
G = \langle g \rangle, \text{ a cyclic group of prime order } q.
\]

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:
  - \( \text{EG}^x_{pk}(M; r) = (u_1 = g^r, e = h^r M) \)
  - which can be decrypted by \( M = e / u_1^x \).
- If \( M \in \mathbb{Z}_q \), the additive ElGamal encryption is:
  - \( \text{EG}^+_{pk}(M; r) = (u_1 = g^r, e = h^r g^M) \)
  - Note that \( \text{EG}^x_{pk}(g^M; r) = \text{EG}^+_{pk}(M; r) \)
  - 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 Ext. Commitments Equivocability UC PAKE
Definitions

Smooth Projective HF Family for ElGamal

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

Language: $L = L_{(EG^+, \rho), M} = \{ C = (u, e) = EG_{pk}^+(M; r), r \overset{\$}{\leftarrow} \mathbb{Z}_q \}$

- $L$ 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

- $\text{HashKG}((EG^+, \rho), M) = hk = (\gamma_1, \gamma_3) \overset{\$}{\leftarrow} \mathbb{Z}_q \times \mathbb{Z}_q$
- $\text{Hash}(hk; (EG^+, \rho), M, C) = (u_1)^{\gamma_1}(eg^{-M})^{\gamma_3}$
- $\text{ProjKG}(hk; (EG^+, \rho), M, C) = hp = (g)^{\gamma_1}(h)^{\gamma_3}$
- $\text{ProjHash}(hp; (EG^+, \rho), M, C; r) = (hp)^r$

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 $\text{SHS}_1$ and $\text{SHS}_2$, on the sets $G_1$ and $G_2$ (included in $G$) corresponding to the languages $L_1$ and $L_2$ respectively:

$$\text{SHS}_i = \{\text{HashKG}_i, \text{ProjKG}_i, \text{Hash}_i, \text{ProjHash}_i\}$$

Let $c \in X$, and $r_1$ and $r_2$ two random elements:

$$\begin{align*}
    hk_1 &= \text{HashKG}_1(\rho, aux, r_1) \\
    hk_2 &= \text{HashKG}_2(\rho, aux, r_2) \\
    hp_1 &= \text{ProjKG}_1(hk_1; \rho, aux, c) \\
    hp_2 &= \text{ProjKG}_2(hk_2; \rho, aux, c)
\end{align*}$$

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 $i = 1, 2$:

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

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

- $\text{HashKG}_L(\rho, aux, r = r_1 || r_2) = hk = (hk_1, hk_2)$
- $\text{ProjKG}_L(hk; \rho, aux, c) = hp = (hp_1, hp_2, h_{\Delta})$
- where $h_{\Delta} = \text{Hash}_1(hk_1; \rho, aux, c)$
- $\oplus \text{Hash}_2(hk_2; \rho, aux, c)$
- $\text{Hash}_L(hk; \rho, aux, c) = \text{Hash}_1(hk_1; \rho, aux, c)$
- $\text{ProjHash}_L(hp; \rho, aux, c; w) = \text{ProjHash}_1(hp_1; \rho, aux, c; w)$ if $c \in L_1$
- or $h_{\Delta} \oplus \text{ProjHash}_2(hp_2; \rho, aux, c; w)$ 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 might be 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 $aux$).

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

ElGamal

- $C = \text{comEG}_{pk}(x; r) = (u, e) = \text{EG}_{pk}^+(x; r)$, 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: with an encryption scheme, IND-CCA2 security level guarantees non-malleability.

**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:

$\text{com}_{\text{EG}}(x; r)$ is extractable for small $x$ only.

**Example**

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

**Homomorphic Property**

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

Furthermore, $\text{com}_{\text{EG}}(x; r) = \prod C_i^{2^i}$, 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

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

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

- 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 $hk \xleftarrow{\$} \text{HashKG}$,
  - 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 ($C \notin L$)
- 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 Equivocal and Extractable Commitments
   - Description
   - Analysis
4 Adaptive Security and UC PAKE
   - Universal Composability
   - Previous Schemes
   - Our Scheme
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

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 $x_{i,\pi_i}$:

  $$a_i = \text{comPed}(\pi_i, x_{i,\pi_i}) = g^{x_{i,\pi_i}}_{\pi_i} y_i^{\pi_i}$$

This defines $\mathbf{a} = (a_1, \ldots, a_m)$

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

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

### Common Reference String Model

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

- $(G, \mathbf{p}k)$, where $\mathbf{p}k$ 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

$$\pi = \sum_{i=1}^{m} \pi_i \cdot 2^{i-1}$$
Properties

\[ \text{com}_R(\pi; R) = (a, b) : a = \text{comPed}(\pi_i, x_i) \]
\[ b = (b_i, [j] = \text{EG}_x^+(x_i, [j] \cdot 2^{i-1}, r_i, [j])) ] \]

Intuition

- Granted the perfectly hiding property of the Pedersen commitment, without any information on the \( x_i, [j] \)'s, no information is leaked about the \( \pi_i \)’s
- Granted the semantic security of the ElGamal encryption scheme, the former privacy on the \( x_i, [j] \)'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 \( x_i \), and then \( x_i = 0 \), which specifies \( \pi_i \)
- One commits on \( \pi_i \) using randomness \( x_i \)
- One encrypts both \( x_i \) and \( x_i, [j] \), bit-by-bit

Equivocal Procedure

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

Conditional Extractability

Constraints

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

The above conditions define the language for extractability:

\[ L_{\text{pi}, \pi} = \left\{ C \mid \exists \delta \text{ such that } C = \text{com}_R(\pi, R) \right\}
\]

Non-Malleability

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

- Random string:
  \[ R = (x_i, [j], r_i, [j]), \ldots, x_m, r_m, [j]) \]
- Commitment:
  \[ \text{com}_R(\pi; R) = (a, b) \]
  \[ a = (a_i = \text{comPed}(\pi_i, x_i)) \]
  \[ b = (b_i, [j] = \text{CS}^+_x(x_i, [j] \cdot 2^{i-1}, r_i, [j])) ] \]
- Opening:
  given the above witness, and the value \( \pi \)
  \[ \forall i, j : b_i, [j] = \text{CS}^+_x(x_i, [j] \cdot 2^{i-1}, r_i, [j])) \]
  \[ \forall i : a_i = \text{comPed}(\pi_i, x_i) \]
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. Adaptive Security and UC PAKE
   - Universal Composability
   - Previous Schemes
   - Our Scheme

Universal Composability

Ideal process:

The protocol $\Pi$ securely realizes $F$, 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 $F$ and $S$.

Real-life Execution

Ideal Execution

$S$ 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.

**Previous Schemes**

- **Scheme I** [Katz-Ostrovsky-Yung EC '01, Gennaro-Lindell C '03]

**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 $\mathcal{A}$ 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 $\mathcal{A}$ wins, $S$ aborts. Because of the short password, this is not negligible.
Smooth Projective HF Ext. Commitments Equivocability UC PAKE

### 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

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

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

Add of a first commitment round

### Scheme II

#### Previous Schemes

- **Canetti-Halevi-Katz-Lindell-MacKenzie EC ’05**
  - Add of a first commitment round

#### Analysis

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

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

If \(A\) plays the server role:
- \(S\) does not know yet the password: dummy password in \(c_0\)
- \(S\) extracts the password from \(c_1\) and 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!

\[ \Rightarrow \text{security against static-corruptions only (before the session starts)} \]

- Non-malleable, \(L\)-extractable, equivocable commitment provides adaptive security
Adaptively Secure UC-PAKE

Our Scheme

(U1) \((VK_I, SK_I) \leftarrow SKG\)
\(\ell_I \leftarrow 1\) o ssid o \(VK_I\)
\(com_I = com_J(\ell_I, pw_I; R_I)\)

(S2) (publicly) checks the validity of \(com_I\)
\((VK_J, SK_J) \leftarrow SKG\)
\(\ell_J \leftarrow 1\) o ssid o \(VK_J\)
\(hk_J = HashKG(\rho, (\ell_J, pw_J); r_J)\)
\(com_J = com_I(\ell_J, pw_J; R_J)\)
\(hp_J = ProjKG(hk_J; \rho, (\ell_J, pw_J), com_I)\)
\(Hash_J = Hash(hk_J; \rho, (\ell_J, pw_J), com_I)\)
\((com_J, VK_J, hp_J)\)
\((\sigma_J, hp_J)\)
\((\sigma_J, hp_J)\)

(U3) (publicly) checks the validity of \(com_J\)
\(hk_I = HashKG(\rho, (\ell_I, pw_I); r_I)\)
\(hp_I = ProjKG(hk_I; (\ell_I, pw_I), com_I)\)
\(\sigma_I = Sign(SK_I, (com_I, com_J, hp_I, hp_J))\)
\(sk_I = ProjHash(hp_I; \rho, (\ell_I, pw_I), com_J; w_I)\)
\(Hash(hk_I; \rho, (\ell_I, pw_I), com_J)\)

(S4) aborts if
\(Ver(VK_I, (com_I, com_J, hp_I, hp_J), \sigma_I) = 0\)
\(\sigma_I = Sign(SK_I, (com_I, com_J, hp_I, hp_J))\)
\(sk_I = ProjHash(hp_I; \rho, (\ell_I, pw_I), com_J; w_I)\)
\(Hash(hk_I; \rho, (\ell_I, pw_I), com_J)\)

(U5) aborts if
\(Ver(VK_I, (com_I, com_J, hp_I, hp_J), \sigma_I) = 0\)
outputs (sid, ssid, sk_I)
erases everything
sets the session as accepted

David Pointcheval – 49/50

Conclusion

Smooth Projective Hash Functions for Complex Languages

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

David Pointcheval – 50/50