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

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 \( x \in X \), \( H(x) = Hash_L(hk; x) \)

For any \( x \in L \), \( H(x) = ProjHash_L(hp; x, w) \) \( w \) witness that \( x \in L \)

**Smoothness**

For any \( x \notin L \), \( H(x) \) 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:

\[
hp = ProjKG_L(hk)
\]

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

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

\[
hp = ProjKG_L(hk; x)
\]

**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

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

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

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

**Key-Generation Algorithms**

- Probabilistic hashing key algorithm:
  \[
hk \overset{\$}{\leftarrow} HashKG()
\]
- Deterministic projection key algorithm
  \[
  hp = ProjKG(hk; c)
  \]
  (where \( c \) is either a ciphertext or a commitment in \( X \))

**Examples**

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

\[
L \quad m = \{c\} \quad c \quad m \\
\quad r \quad c = (m; r)
\]

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

\[
L \quad (m) = \{c\} \quad c \quad m \\
\quad \ell \quad r \quad c = \mathcal{E}_{pk}(m; r)
\]
**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 \( g = \text{ProjHash}(hp; c, w) \in G \)

**Properties**

**Pseudorandomness**

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

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


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

**Correctness**

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

\( hk \leftarrow \text{HashKG()} \) and \( hp = \text{ProjKG}(hk; c) \) implies

\[
\text{Hash}(hk; c) = \text{ProjHash}(hp; c, w)
\]

**Smoothness**

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

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

with \( hk \leftarrow \text{HashKG()} \)

**ElGamal Encryption**

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

**ElGamal Encryption Schemes**

Let \( pk = h = g^x \) (public key), where \( sk = x \leftarrow \mathbb{Z}_q \) (private key)

- If \( M \in G \), the multiplicative ElGamal encryption is:
  \[
  \text{EG}^*_pk(M; r) = (u_t = g^r, e = h^t M)
  \]
  which can be decrypted by \( M = e/u_t^r \).

- If \( M \in \mathbb{Z}_q \), the additive ElGamal encryption is:
  \[
  \text{EG}^+_pk(M; r) = (u_t = g^r, e = h^t g^M)
  \]
  Note that \( \text{EG}^+_pk(g^M; r) = \text{EG}^+_pk(M; r) \)
  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 Family for ElGamal**

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

Language: \( L = L_{(EG^+, ρ), M} = \{ C = (u, e) = EG^+_{pk}(M; r), r \leftarrow Z_q \} \)
- \( L \) 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**
- \( \text{HashKG}(M) = \text{hk} = (\gamma_1, \gamma_3) \leftarrow Z_q \times Z_q \)
- \( \text{Hash}(hk; M, C) = (u_i)^{\gamma_1}(eg^{-M})^{\gamma_3} \)
- \( \text{ProjKG}(hk; M, C) = \text{hp} = (g)^{\gamma_1}(h)^{\gamma_3} \)
- \( \text{ProjHash}(hp; M, C; r) = (hp)r \)

**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 \( \text{SHS}_2 \), onto \( G \), 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:
- \( \text{hk}_1 = \text{HashKG}_1(r_1) \)
- \( \text{hk}_2 = \text{HashKG}_2(r_2) \)
- \( \text{hp}_1 = \text{ProjKG}_1(hk_1; c) \)
- \( \text{hp}_2 = \text{ProjKG}_2(hk_2; c) \)

**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 \( i = 1, 2 \):

\[
\begin{align*}
\text{HashKG}_L(r = r_1 \parallel r_2) &= \text{hk} = (\text{hk}_1, \text{hk}_2) \\
\text{ProjKG}_L(hk; c) &= \text{hp} = (\text{hp}_1, \text{hp}_2) \\
\text{Hash}_L(hk; c) &= \text{Hash}_1(hk_1; c) \oplus \text{Hash}_2(hk_2; c) \\
\text{ProjHash}_L(hp; c, (w_1, w_2)) &= \text{ProjHash}_1(hp_1; c, w_1) \\
&\quad \oplus \text{ProjHash}_2(hp_2; c, w_2)
\end{align*}
\]
- 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 \( i \in \{1, 2\} \):

\[
\begin{align*}
\text{HashKG}_L(r = r_1 \parallel r_2) &= \text{hk} = (\text{hk}_1, \text{hk}_2) \\
\text{ProjKG}_L(hk; c) &= \text{hp} = (\text{hp}_1, \text{hp}_2, \text{hp}_\Delta) \\
\text{Hash}_L(hk; c) &= \text{Hash}_1(hk_1; c) \oplus \text{Hash}_2(hk_2; c) \\
\text{ProjHash}_L(hp; c, w) &= \text{ProjHash}_1(hp_1; c, w) \text{ if } c \in L_1 \\
&\quad \text{or } \text{hp}_\Delta \oplus \text{ProjHash}_2(hp_2; c, w) \text{ if } c \in L_2
\end{align*}
\]
- \( \text{hp}_\Delta \) 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 \( aux \)).

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

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

Pedersen

- \( C = \text{comPed}(x; r) = g^x h^r, \text{ with } r \overset{\$}{\leftarrow} \mathbb{Z}_q; \)
- 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.

### Conditional Extractability

**Motivation**

**ElGamal Commitment**

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

**Example**

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

**Homomorphic Property**

Let us assume \( 2^{k-1} < q < 2^k \), then for any \( x = \sum_{i=0}^{k-1} x_i \times 2^i \in \mathbb{Z}_q \),

\[ C(x) = \{ C_i = \text{comEG}_{\text{pk}}(x_i; r_i) = \text{EG}_{\text{pk}}^+(x_i; r_i) \}_{i=0}^{k-1}, \text{ is extractable under} \]

the condition that \( (x_i)_i \in \{0, 1\}^k \).

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

\[ x = 0 \iff C(x) = \text{comEG}_{\text{pk}}(x; r) \in L(\text{EG}^+, \rho)_0 \]

\[ x = 1 \iff C(x) = \text{comEG}_{\text{pk}}(x; r) \in L(\text{EG}^+, \rho)_1 \]

We then define

\[ L(\text{EG}^+, \rho)_0 \mathbin{\lor} 1 = L(\text{EG}^+, \rho)_0 \cup L(\text{EG}^+, \rho)_1 \]

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

\[ L = \{(C_0, \ldots, C_{k-1}) \mid \forall i, C_i \in L(\text{EG}^+, \rho)_0 \mathbin{\lor} 1 \} \]

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: \textit{e.g.}, in the UC Framework).

**New Process**
- the user and the authority use a smooth projective hash system for \( L \): \( \text{HASH}(pk) = (\text{HashKG, ProjKG, Hash, ProjHash}) \);
- 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 \( \text{hk} \leftarrow \text{HashKG}() \),
  - the corresponding projected key on \( C \), \( \text{hp} = \text{ProjKG(hk, C)} \),
  - the hash value \( \text{Hash} = \text{Hash(hk; C)} \)
  - and sends \( \text{hp} \) 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 \( \text{hp} \) 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

[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

Random string:
\[ R = (x_{1,π_1}, (r_{1,0}[j], r_{1,1}[j]), \ldots, x_{m,π_m}, (r_{m,0}[j], r_{m,1}[j])) \]

Commitment: \( \text{comPed}(π; R) = (a, b) \)
where \( a = (a_i = \text{comPed}(π_i, x_{i,π_i})) \)
\( b = (b_i[j] = \text{EG}_{pk}(x_{i,δ}[j] \cdot 2^{j-1}, r_{i,δ}[j]))_{i,δ,j} \)

Witness: the values \( r_{i,π_i}[j] \) can be erased,
\[ w = (x_{1,π_1}, (r_{1,π_1}[j]), \ldots, x_{m,π_m}, (r_{m,π_m}[j])) \]

Opening: given the above witness, and the value \( π \)
\[ \forall i, j : b_{i,π_i}[j] \stackrel{?}{=} \text{EG}_{pk}^+(x_{i,δ}[j] \cdot 2^{j-1}, r_{i,δ}[j]) \]
\[ \forall i : a_i \stackrel{?}{=} \text{comPed}(π_i, x_{i,π_i}) \]
Properties

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

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

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 from 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:

\[ L_{\rho,\pi} = \{ C \mid \exists R \text{ such that } C = \text{com}_\rho(\pi, R) \]
\[ \text{and } \forall i \forall j b_{i,\pi_i}[j] \in L(\text{EG}^+_{\pi_i}, 0 \lor 1) \]
\[ \text{and } \forall i B_{i,\pi_i} \in L(\text{EG}^+_{\cdot,\rho}, a_i/x_i^\pi) \}\]

Non-Malleability

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

- Random string:
  \[ R = (x_i, r_{i,0}[j], r_{i,1}[j]) \]
  \[ \ldots, x_{m,\pi_m}, (r_{m,0}[j], r_{m,1}[j]) \]
- Commitment:
  \[ \text{com}_\rho(\pi; R) = (a, b) \]
  \[ \text{where } a = (a_i = \text{comPed}(\pi_i, x_{i,0}))_i \]
  \[ b = (b_{i,\delta}[j] = \text{CS}_{pk}^+(x_{i,\pi_i}[j] \cdot 2^{i-1}, r_{i,\pi_i}[j]))_{i,\delta,j} \]
- Opening:
  \[ \forall i, j : b_{i,\pi_i}[j] \overset{?}{=} \text{CS}_{pk}^+(x_{i,\pi_i}[j] \cdot 2^{i-1}, r_{i,\pi_i}[j]) \]
  \[ \forall i : a_i \overset{?}{=} \text{comPed}(\pi_i, x_{i,\pi_i}) \]

- 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).
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

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]

- Not UC secure!

Scheme II [Canetti-Halevi-Katz-Lindell-MacKenzie EC '05]

Add of a first commitment round: for non-adaptive UC security.

Not UC secure!
**Security 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
  $\implies$ perfect simulation

If $A$ plays the server role:
- $S$ does not know yet 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!
$\implies$ 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