Smooth Projective Hashing for Conditionally Extractable Commitments

Michel Abdalla, Céline Chevalier, and David Pointcheval

Ecole normale supérieure, CNRS & INRIA

CRYPTO 2009 – Santa Barbara – USA
August 20th, 2009

Outline

Commitments

Definition

A commitment scheme is defined by two algorithms:

- the committing algorithm, $C = \mathrm{com}(x; r)$ with randomness r, on input x, to commit on this input;
- the decommitting algorithm, $(x, D) = \mathrm{decom}(C, x, r)$, where x is the claimed committed value, and D the proof

Properties

The commitment $C = \mathrm{com}(x; r)$

- 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 ρ is $(G, q, g, \mathrm{pk} = h)$,
and $(x, D = r) = \mathrm{decom}(C, x, r)$

ElGamal

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

Pedersen

- $C = \mathrm{comPed}_{\mathrm{pk}}(x; r) = g^x h^r$, with $r \stackrel{\$}{\leftarrow} \mathbb{Z}_q$;
- 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, extraction key = decryption key

Equivocability

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

This is possible for computationally binding commitments only

Motivation

ElGamal Commitment

$\text{comEG}_{\text{pk}}(x; r)$ is extractable for small x 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 $x = \sum_{i=0}^{k-1} x_i \times 2^i \in \mathbb{Z}_q$, $C(x) = (C_i = \text{comEG}_{\text{pk}}(x_i; r_i))_i$, is extractable if $(x_i)_i \in \{0, 1\}^k$
Furthermore, $\text{comEG}_{\text{pk}}(x; r) = \prod C_i^{2^i}$, for $r = \sum_{i=0}^{k-1} r_i \times 2^i$

Extractable Languages

$$x = 0 \iff C(x) = \mathsf{comEG}_{\mathsf{pk}}(x; r) \in L_0$$
$$x = 1 \iff C(x) = \mathsf{comEG}_{\mathsf{pk}}(x; r) \in L_1$$

We then define

$$L_{0\vee 1} = L_0 \cup L_1$$

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

$$L = \{(C_0, \ldots, C_{k-1}) \mid \forall i, C_i \in L_{0\vee 1}\}$$

A conjunction of disjunctions of basic languages

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

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 pk: $\mathsf{pk} = \mathsf{ProjKG}_L(\mathsf{hk})$

Properties

For any $x \in X$, $H(x) = \mathsf{Hash}_L(\mathsf{hk}; x)$
For any $x \in L$, $H(x) = \mathsf{ProjHash}_L(\mathsf{pk}; x, w)$ w witness that $x \in L$

Smoothness

For any $x \notin L$, $H(x)$ and pk are independent

Pseudo-Randomness

For any $x \in L$, $H(x)$ is pseudo-random, given pk, 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 pk depends on the hashing key hk only:
$\mathsf{pk} = \mathsf{ProjKG}_L(\mathsf{hk})$

New Definition [Gennaro-Lindell EC ’03]

The projected key pk depends on the hashing key hk, and x:
$\mathsf{pk} = \mathsf{ProjKG}_L(\mathsf{hk}; x)$

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 HF Family for ElGamal

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

Language: $L = L_M = \{C = (u_1 = g^r, e = h^r g^M), r \stackrel{\$}{\leftarrow} \mathbb{Z}_q\}$

- 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

- $\mathsf{HashKG}_M(\$) = \mathsf{hk} = (\gamma_1, \gamma_3) \stackrel{\$}{\leftarrow} \mathbb{Z}_q \times \mathbb{Z}_q$
- $\mathsf{Hash}_M(\mathsf{hk}; C) = (u_1)^{\gamma_1}(eg^{-M})^{\gamma_3}$
- $\mathsf{ProjKG}_M(\mathsf{hk}; C) = \mathsf{pk} = (g)^{\gamma_1}(h)^{\gamma_3}$
- $\mathsf{ProjHash}_M(\mathsf{pk}; C; r) = (\mathsf{pk})^r$

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 SHS_1 and SHS_2, onto G, corresponding to the languages L_1 and L_2 respectively:

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

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

$$\begin{aligned} \mathsf{hk}_1 &= \mathsf{HashKG}_1(r_1) \\ \mathsf{hk}_2 &= \mathsf{HashKG}_2(r_2) \\ \mathsf{pk}_1 &= \mathsf{ProjKG}_1(\mathsf{hk}_1; c) \\ \mathsf{pk}_2 &= \mathsf{ProjKG}_2(\mathsf{hk}_2; c) \end{aligned}$$

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 $i = 1, 2$:

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

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

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

pk_Δ helps to compute the missing hash value,
if and only if at least one can be computed

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)

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:

New Process

Use of **HASH**(pk) $=$ (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 $\stackrel{\$}{\leftarrow}$ HashKG(),
  - the corresponding projected key on C, pk $=$ ProjKG(hk, C)
  - the hash value Hash $=$ Hash(hk; C)

  and sends pk along with Cert $\oplus$ Hash;
- The user computes Hash $=$ ProjHash(pk; C, r), and gets Cert.

Commitment and Smooth Projective HF

The authority sends pk along with Cert $\oplus$ 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 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 Hash perfectly unpredictable: no information is leaked about the certificate.

Conclusion

Smooth Projective Hash Functions for Complex Languages

Various Applications

- in place of some ZK proofs
- conditional secure-channels
- adaptive security in UC for PAKE
  - Gennaro-Lindell's approach [EC '03]
  - with a smooth hash system
  - for an equivocable, extractable and non-malleable commitment