Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

PART XIV

Outline

Homomorphic Encryption

Let $\Pi = (\text{KeyGen}, \text{Enc}, \text{Dec})$ a PKE scheme such that the message and ciphertext spaces $\mathcal{M}$, $\mathcal{C}$.

Definition (Weakly Homomorphic Encryption)

Π is weakly homomorphic with respect to the internal operation $(\mathcal{M}, \otimes)$ if there exists a PPTM HomEval such that $\forall (pk, sk) \leftarrow \text{KeyGen}(\ell); \forall m_1, m_2 \in \mathcal{M}_\ell$

$$\Pr[\text{Dec}(sk, \text{HomEval}(1^\ell, pk, \text{Enc}(pk, m_1), \text{Enc}(pk, m_2))) = m_1 \otimes m_2] = 1$$

Homomorphic Encryption

Let $\Pi = (\text{KeyGen}, \text{Enc}, \text{Dec})$ a PKE scheme such that the message and ciphertext spaces $\mathcal{M}$, $\mathcal{C}$.

Definition (Weakly Homomorphic Encryption)

Π is weakly homomorphic with respect to the internal operation $(\mathcal{M}, \otimes)$ if there exists a PPTM HomEval such that $\forall (pk, sk) \leftarrow \text{KeyGen}(\ell); \forall m_1, m_2 \in \mathcal{M}_\ell$

$$\Pr[\text{Dec}(sk, \text{HomEval}(1^\ell, pk, \text{Enc}(pk, m_1), \text{Enc}(pk, m_2))) = m_1 \otimes m_2] = 1$$

Definition (Strongly Homomorphic Encryption)

Π is strongly homomorphic if in addition the random variables $\text{HomEval}(1^\ell, pk, \text{Enc}(pk, m_1), \text{Enc}(pk, m_2))$ and $\text{Enc}(pk, m_1 \otimes m_2)$ are independent and identically distributed

Examples

In a typical homomorphic encryption scheme there are two efficiently computable internal (group) operations $(\mathcal{M}, \otimes)$, $(\mathcal{C}, \odot)$ and $\mathrm{Dec}(sk, \cdot)$ is a (group) homomorphism

Examples

In a typical homomorphic encryption scheme there are two efficiently computable internal (group) operations $(\mathcal{M}, \otimes)$, $(\mathcal{C}, \odot)$ and $\mathrm{Dec}(sk, \cdot)$ is a (group) homomorphism

- **RSA:** strongly homomorphic $\mathrm{Dec}(sk, \cdot) : \mathbb{Z}^\times \to \mathbb{Z}^\times$
  $\mathrm{Dec}(d, c_1 c_2) = (c_1 c_2)^d = c_1^d c_2^d = \mathrm{Dec}(d, c_1)\mathrm{Dec}(d, c_2)$ or
  $\mathrm{Enc}(e, m_1)\mathrm{Enc}(e, m_2) = m_1^e m_2^e = (m_1 m_2)^e = \mathrm{Enc}(e, m_1 m_2)$

Examples

In a typical homomorphic encryption scheme there are two efficiently computable internal (group) operations $(\mathcal{M}, \otimes)$, $(\mathcal{C}, \odot)$ and $\mathrm{Dec}(sk, \cdot)$ is a (group) homomorphism

- **RSA:** strongly homomorphic $\mathrm{Dec}(sk, \cdot) : \mathbb{Z}^\times \to \mathbb{Z}^\times$
  $\mathrm{Dec}(d, c_1 c_2) = (c_1 c_2)^d = c_1^d c_2^d = \mathrm{Dec}(d, c_1)\mathrm{Dec}(d, c_2)$ or
  $\mathrm{Enc}(e, m_1)\mathrm{Enc}(e, m_2) = m_1^e m_2^e = (m_1 m_2)^e = \mathrm{Enc}(e, m_1 m_2)$
- **ElGamal:** strongly homomorphic $\mathrm{Dec}(sk, \cdot) : G \times G \to G$
  $\mathrm{Enc}(y, m_1) \cdot \mathrm{Enc}(y, m_2) = (g^{r_1}, y^{r_1} m_1) \cdot (g^{r_2}, y^{r_2} m_2) =$
  $(g^{r_1+r_2}, y^{r_1+r_2} m_1 m_2) \in \mathrm{Enc}(e, m_1 m_2)$ then
  $\mathrm{HomEval}(y, c_1, c_2) = c_1 \cdot c_2 \cdot (g^{r_3}, y^{r_3}) = (g^r, y^r m_1 m_2)$ for $r_3 \leftarrow \mathbb{Z}_q$.
  Therefore $r = r_1 + r_2 + r_3$ is independent of r_1 and r_2

Examples

- **Paillier:** strongly homomorphic $\mathrm{Dec}(sk, \cdot) : \mathbb{Z}_{n^2}^{\times} \to \mathbb{Z}_n$
  $\mathrm{Enc}(n, m_1)\mathrm{Enc}(n, m_2) = (1 + m_1 n) r_1^n (1 + m_2 n) r_2^n =$
  $(1 + (m_1 + m_2)n)(r_1 r_2)^n \in \mathrm{Enc}(n, m_1 + m_2)$ then
  $\mathrm{HomEval}(n, c_1, c_2) = c_1 c_2 r_3^n = (1 + (m_1 + m_2)n) r^n$ for $r_3 \leftarrow \mathbb{Z}_n^{\times}$.
  Therefore $r = r_1 r_2 r_3$ is independent of r_1 and r_2

Examples

- **Paillier:** strongly homomorphic $\mathrm{Dec}(sk, \cdot) : \mathbb{Z}_{n^2}^{\times} \to \mathbb{Z}_n$
  $\mathrm{Enc}(n, m_1)\mathrm{Enc}(n, m_2) = (1 + m_1 n) r_1^n (1 + m_2 n) r_2^n =$
  $(1 + (m_1 + m_2)n)(r_1 r_2)^n \in \mathrm{Enc}(n, m_1 + m_2)$ then
  $\mathrm{HomEval}(n, c_1, c_2) = c_1 c_2 r_3^n = (1 + (m_1 + m_2)n) r^n$ for $r_3 \leftarrow \mathbb{Z}_n^{\times}$.
  Therefore $r = r_1 r_2 r_3$ is independent of r_1 and r_2

RSA and ElGamal PKE schemes are multiplicatively homomorphic while Paillier's is **additively** homomorphic

Examples

- **Paillier:** strongly homomorphic $\mathrm{Dec}(sk, \cdot) : \mathbb{Z}_{n^2}^{\times} \to \mathbb{Z}_n$
  $\mathrm{Enc}(n, m_1)\mathrm{Enc}(n, m_2) = (1 + m_1 n) r_1^n (1 + m_2 n) r_2^n =$
  $(1 + (m_1 + m_2)n)(r_1 r_2)^n \in \mathrm{Enc}(n, m_1 + m_2)$ then
  $\mathrm{HomEval}(n, c_1, c_2) = c_1 c_2 r_3^n = (1 + (m_1 + m_2)n) r^n$ for $r_3 \leftarrow \mathbb{Z}_n^{\times}$.
  Therefore $r = r_1 r_2 r_3$ is independent of r_1 and r_2

RSA and ElGamal PKE schemes are multiplicatively homomorphic while Paillier's is **additively** homomorphic

Additively homomorphic PKE schemes have interesting applications (e.g., electronic election systems):

Examples

- **Paillier:** strongly homomorphic $\mathrm{Dec}(sk, \cdot) : \mathbb{Z}_{n^2}^{\times} \to \mathbb{Z}_n$
  $\mathrm{Enc}(n, m_1)\mathrm{Enc}(n, m_2) = (1 + m_1 n) r_1^n (1 + m_2 n) r_2^n =$
  $(1 + (m_1 + m_2)n)(r_1 r_2)^n \in \mathrm{Enc}(n, m_1 + m_2)$ then
  $\mathrm{HomEval}(n, c_1, c_2) = c_1 c_2 r_3^n = (1 + (m_1 + m_2)n) r^n$ for $r_3 \leftarrow \mathbb{Z}_n^{\times}$.
  Therefore $r = r_1 r_2 r_3$ is independent of r_1 and r_2

RSA and ElGamal PKE schemes are multiplicatively homomorphic while Paillier's is **additively** homomorphic

Additively homomorphic PKE schemes have interesting applications (e.g., electronic election systems):

- $c_i = \mathrm{Enc}(pk, vote_i)$ where $vote_i \in \{0, 1\}$
- $c \leftarrow \mathrm{HomEval}(pk, c_1, \cdots, c_n)$
- $tally = \mathrm{Dec}(sk, c)$, where $tally = \sum_i vote_i = \#\{vote_i = 1\}$

Fully Homomorphic Encryption

In a **fully homomorphic** encryption scheme $\mathrm{Dec}(sk, \cdot)$ is a ring homomorphism and both $(\mathcal{M})$ and $(\mathcal{C})$ are rings with efficiently computable operations

Fully Homomorphic Encryption

In a **fully homomorphic** encryption scheme $\mathrm{Dec}(sk, \cdot)$ is a ring homomorphism and both $(\mathcal{M})$ and $(\mathcal{C})$ are rings with efficiently computable operations

No efficient fully homomorphic encryption scheme is known to date

Fully Homomorphic Encryption

In a **fully homomorphic** encryption scheme $\mathrm{Dec}(sk, \cdot)$ is a ring homomorphism and both $(\mathcal{M})$ and $(\mathcal{C})$ are rings with efficiently computable operations

No efficient fully homomorphic encryption scheme is known to date. . . but if one exists then one can securely evaluate polynomial functions (arithmetic circuits) over encrypted data.

Fully Homomorphic Encryption

In a **fully homomorphic** encryption scheme $\mathrm{Dec}(sk, \cdot)$ is a ring homomorphism and both $(\mathcal{M})$ and $(\mathcal{C})$ are rings with efficiently computable operations

No efficient fully homomorphic encryption scheme is known to date. . . but if one exists then one can securely evaluate polynomial functions (arithmetic circuits) over encrypted data.

Arithmetic circuits over any ring $\mathbb{Z}_q$ include boolean circuits, as if one encodes logical '0' and '1' as arithmetic '0' and '1', then

- x_1 **AND** $x_2 = x_1 x_2$
- x_1 **OR** $x_2 = x_1 + x_2 - x_1 x_2$
- x_1 **XOR** $x_2 = x_1 + x_2 - 2x_1 x_2$
- **NOT** $x = 1 - x$

Outline

Chosen Ciphertext Security Again

Experiment Exp-PKE-IND-CCA$(\Pi, \mathcal{A}_1, \mathcal{A}_2, \ell)$:
$(pk, sk) \leftarrow \text{KeyGen}(\ell)$;
$(m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{Dec}}}(1^\ell, pk)$;
$b_* \leftarrow \{0,1\}$;
$c_* \leftarrow \text{Enc}(pk, m_{b_*})$;
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{Dec}}}(1^\ell, c_*, s)$;
if $b' = b_*$ **and** $|m_0| = |m_1|$ **output** 1; //$\mathcal{A}$ wins
else output 0;

Oracle $\mathcal{O}_{\text{Dec}}(c)$:
if $c = c_*$ **output** $\perp$; //Illegal oracle query
else output $\text{Dec}(sk, c)$;

Definition (PKE-IND-CCA)

The public key encryption scheme Π is PKE-IND-CCA secure if for all **Two-Stages** PPOTM, $(\mathcal{A}_1, \mathcal{A}_2)$,

$$|\Pr[\text{Exp-PKE-IND-CCA}(\Pi, \mathcal{A}_1, \mathcal{A}_2, \ell) = 1] - 1/2| \in \textbf{negl}(\ell)$$

Why Chosen Ciphertext Security

Actual attacks beyond the Chosen Plaitext Attack model were reported:

- Some attacks observe the reaction of a system when unauthorized access is attempted (e.g., the error messages can leak some crucial information)

Why Chosen Ciphertext Security

Actual attacks beyond the Chosen Plaitext Attack model were reported:

- Some attacks observe the reaction of a system when unauthorized access is attempted (e.g., the error messages can leak some crucial information)
- Real systems requiring only one user authentication per session (e.g., those web browsers remembering the passwords) potentially give access to the decryption functionality

Why Chosen Ciphertext Security

Moreover, PKE-IND-CCA security has been shown equivalent to other interesting security notion like **Non-Malleability** of Encryptions: "A ciphertext cannot be efficiently manipulated to obtain the encryption of a message related in a predictable way to the original one"

Why Chosen Ciphertext Security

Moreover, PKE-IND-CCA security has been shown equivalent to other interesting security notion like **Non-Malleability** of Encryptions: "A ciphertext cannot be efficiently manipulated to obtain the encryption of a message related in a predictable way to the original one"

However, CCA security is unachievable if a PKE has homomorphic properties:

Why Chosen Ciphertext Security

Moreover, PKE-IND-CCA security has been shown equivalent to other interesting security notion like **Non-Malleability** of Encryptions: "A ciphertext cannot be efficiently manipulated to obtain the encryption of a message related in a predictable way to the original one"

However, CCA security is unachievable if a PKE has homomorphic properties:

- After receiving $c_* \leftarrow \mathsf{Enc}(pk, m_{b_*})$, $\mathcal{A}_2$ computes an encryption of $m' = m_{b_*} \otimes \mu$, for some arbitrary $\mu \in \mathcal{M}_\ell$, as $c' \leftarrow \mathsf{HomEval}(pk, c_*, \mathsf{Enc}(pk, \mu))$
- Then $\mathcal{A}_2$ obtains $m' = \mathcal{O}_{\mathsf{Dec}}(c')$
- Finally, $\mathcal{A}_2$ guesses b_* by comparing m' to $m_1 \otimes \mu$.

Achieving CCA Security

It is a very challenging task: In a reduction $\mathcal{O}_{\mathsf{Dec}}$ has to be simulated **without** knowing *sk*!

Achieving CCA Security

It is a very challenging task: In a reduction $\mathcal{O}_{\mathsf{Dec}}$ has to be simulated **without** knowing *sk*!

Several approaches have been used:

- Use an idealized model, the **Random Oracle Model**, to ensure that $\mathcal{O}_{\mathsf{Dec}}$ gives no information to the adversary.
- Use a two-key trick, **Twin Encryption**, to give one of the two secret keys to the reduction.
- Use a combinatorial object, a **Hash Proof System**, to indistinguishably switch between to operation modes: real and simulated.
- Use the sophisticated key generation of **Identity-Based** encryption schemes to make different encryptions "independent".

Achieving CCA Security

It is a very challenging task: In a reduction $\mathcal{O}_{\text{Dec}}$ has to be simulated **without** knowing *sk*!

Several approaches have been used:

- Use an idealized model, the **Random Oracle Model**, to ensure that $\mathcal{O}_{\text{Dec}}$ gives no information to the adversary. No actual security proof is given
- Use a two-key trick, **Twin Encryption**, to give one of the two secret keys to the reduction. Involves non-efficient proofs of correctness of the ciphertexts
- Use a combinatorial object, a **Hash Proof System**, to indistinguishably switch between to operation modes: real and simulated. The first efficient CCA scheme!
- Use the sophisticated key generation of **Identity-Based** encryption schemes to make different encryptions "independent". Also leads to efficient PKE-IND-CCA

Achieving CCA Security

It is a very challenging task: In a reduction $\mathcal{O}_{\mathsf{Dec}}$ has to be simulated **without** knowing sk!

Several approaches have been used:

- Use an idealized model, the **Random Oracle Model**, to ensure that $\mathcal{O}_{\mathsf{Dec}}$ gives no information to the adversary.
- Use a two-key trick, **Twin Encryption**, to give one of the two secret keys to the reduction.
- Use a combinatorial object, a **Hash Proof System**, to indistinguishably switch between to operation modes: real and simulated.
- Use the sophisticated key generation of **Identity-Based** encryption schemes to make different encryptions "independent".

The Random Oracle Model (ROM)

Introduced as an **idealized model of computation** in which Truly Random Functions are made efficiently computable (by oracle access)

The Random Oracle Model (ROM)

Introduced as an **idealized model of computation** in which Truly Random Functions are made efficiently computable (by oracle access)

Cryptosystems proven secure under the ROM **may not** be secure when the Random Oracle is instantiated with any real efficient function (typically, with a hash function), but security proofs in the ROM can be interpreted as heuristic security arguments:

The Random Oracle Model (ROM)

Introduced as an **idealized model of computation** in which Truly Random Functions are made efficiently computable (by oracle access)

Cryptosystems proven secure under the ROM **may not** be secure when the Random Oracle is instantiated with any real efficient function (typically, with a hash function), but security proofs in the ROM can be interpreted as heuristic security arguments: "Any successful attack must use specific information of the function instantiating the Random Oracle"

The Random Oracle Model (ROM)

In a reduction in the ROM, a random oracle $H : \{0,1\} \to \mathcal{Y}$ is simulated by a table of random values:

- Any "new" call to H is answered with a random value in $\mathcal{Y}$, which is stored in the table
- Any "repeated" call to H is answered with the value stored in the table

The Random Oracle Model (ROM)

In a reduction in the ROM, a random oracle $H : \{0,1\} \to \mathcal{Y}$ is simulated by a table of random values:

- Any "new" call to H is answered with a random value in $\mathcal{Y}$, which is stored in the table
- Any "repeated" call to H is answered with the value stored in the table
- Although a truly random function requires exponential space to be described, the table can only have polynomial size, due to the polynomial bound in the running time of the adversary

The Random Oracle Model (ROM)

In a reduction in the ROM, a random oracle $H : \{0,1\} \to \mathcal{Y}$ is simulated by a table of random values:

- Any "new" call to H is answered with a random value in $\mathcal{Y}$, which is stored in the table
- Any "repeated" call to H is answered with the value stored in the table
- Although a truly random function requires exponential space to be described, the table can only have polynomial size, due to the polynomial bound in the running time of the adversary

Random oracles provide automatic privacy amplification, as "If an adversary **learns something** about $H(x)$ is because it **known** x"

Fujisaki-Okamoto Transformation

It builds a IND-CCA secure PKE in the Random Oracle Model from any **Trapdoor** Partial One-Way Injective Function Family $\mathcal{F}$ for set families $\mathcal{X} \times \mathcal{Y} = \{\mathcal{X}_\ell \times \mathcal{Y}_\ell\}_{\ell \in \mathbb{Z}^+}$ and $\mathcal{Z}$, and two random oracles G and H.

KeyGen(ℓ) :
$(k, t) \leftarrow$ Sample(ℓ);
output (k, t);

Enc(k, m) :
$x \leftarrow \mathcal{X}_k$;
output $(f_k(x, H(m, x)), m \oplus G(x))$;

Dec(t, c) :
$(c_1, c_2) = c$;
$x \leftarrow$ Inv(c_1, t);
$m \leftarrow c_2 \oplus G(x)$;
if $c_1 = f_k(x, H(m, x))$ **output** m; **else output** $\perp$;

Achieving CCA Security

It is a very challenging task: In a reduction $\mathcal{O}_{\mathsf{Dec}}$ has to be simulated **without** knowing sk!

Several approaches have been used:

- Use an idealized model, the **Random Oracle Model**, to ensure that $\mathcal{O}_{\mathsf{Dec}}$ gives no information to the adversary.
- Use a two-key trick, **Twin Encryption**, to give one of the two secret keys to the reduction.
- Use a combinatorial object, a **Hash Proof System**, to indistinguishably switch between to operation modes: real and simulated.
- Use the sophisticated key generation of **Identity-Based** encryption schemes to make different encryptions "independent".

Hash Proof Systems (I)

Hard Subset Membership Problem (Family): $\mathcal{L} \subset \mathcal{X}$:
$\Lambda = \bigcup_{\ell \in \mathbb{Z}^+} \Lambda_\ell$, $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \Lambda}$, $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda \in \Lambda}$, $\mathcal{L}_\lambda \subset \mathcal{X}_\lambda$,
$\overline{\mathcal{L}}_\lambda = \mathcal{X}_\lambda \setminus \mathcal{L}_\lambda$, $\mathcal{W}_\lambda$ set of witnesses for $x \in \mathcal{L}_\lambda$

Definition

$\mathcal{L} \subset \mathcal{X}$ is hard if $U_{\mathcal{L}_\lambda} \approx_c U_{\overline{\mathcal{L}}_\lambda}$

Hash Proof Systems (I)

Hard Subset Membership Problem (Family): $\mathcal{L} \subset \mathcal{X}$:
$\Lambda = \bigcup_{\ell \in \mathbb{Z}^+} \Lambda_\ell$, $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \Lambda}$, $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda \in \Lambda}$, $\mathcal{L}_\lambda \subset \mathcal{X}_\lambda$,
$\overline{\mathcal{L}}_\lambda = \mathcal{X}_\lambda \setminus \mathcal{L}_\lambda$, $\mathcal{W}_\lambda$ set of witnesses for $x \in \mathcal{L}_\lambda$

Definition

$\mathcal{L} \subset \mathcal{X}$ is hard if $U_{\mathcal{L}_\lambda} \approx_c U_{\overline{\mathcal{L}}_\lambda}$

Examples:

- DDH: $G = \langle g_1 \rangle = \langle g_2 \rangle$ prime order q
  $\mathcal{X}_{(G,q,g_1,g_2)} = G \times G$,
  $\mathcal{L}_{(G,q,g_1,g_2)} = \langle (g_1, g_2) \rangle = \{(g_1^w, g_2^w) \ : \ w \in \mathbb{Z}_q\}$
- DCR: $n = pq$ for two $\ell/2$-bit long primes, p and q
  $\mathcal{X}_n = \mathbb{Z}_{n^2}^\times$, $\mathcal{L}_n = \left(\mathbb{Z}_{n^2}^\times\right)^n = \{w^n \ : \ w \in \mathbb{Z}_{n^2}^\times\}$

Hash Proof Systems (II)

Projective Hash Families: $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\Lambda}$,
$\mathcal{H}_\lambda = \{h_k : \mathcal{X}_\lambda \to \mathcal{Z}_\lambda\}_{k\in\mathcal{K}_\lambda}$, $\alpha_\lambda : \mathcal{K}_\lambda \to \mathcal{S}_\lambda$ surjective

Definition

$(\mathcal{H} : \mathcal{X} \to \mathcal{Z}, \alpha : \mathcal{K} \to \mathcal{S})$ is a **projective hash family** for $\mathcal{L} \subset \mathcal{X}$ if $\forall k_1, k_2 \in \mathcal{K}_\lambda;\ \alpha(k_1) = \alpha(k_2) \Rightarrow h_{k_1}|_{\mathcal{L}_\lambda} = h_{k_2}|_{\mathcal{L}_\lambda}$

Hash Proof Systems (II)

Projective Hash Families: $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\Lambda}$,
$\mathcal{H}_\lambda = \{h_k : \mathcal{X}_\lambda \to \mathcal{Z}_\lambda\}_{k\in\mathcal{K}_\lambda}$, $\alpha_\lambda : \mathcal{K}_\lambda \to \mathcal{S}_\lambda$ surjective

Definition

$(\mathcal{H} : \mathcal{X} \to \mathcal{Z}, \alpha : \mathcal{K} \to \mathcal{S})$ is a **projective hash family** for $\mathcal{L} \subset \mathcal{X}$ if $\forall k_1, k_2 \in \mathcal{K}_\lambda;\ \alpha(k_1) = \alpha(k_2) \Rightarrow h_{k_1}|_{\mathcal{L}_\lambda} = h_{k_2}|_{\mathcal{L}_\lambda}$

Definition

$(\mathcal{H}, \alpha)$ is **smooth** if $\forall x \in \overline{\mathcal{L}}_\lambda,\ \forall s \in \mathcal{S}_\lambda;\ h_{U_{\alpha^{-1}(s)}}(x) = U_{\mathcal{Z}_\lambda}$

Hash Proof Systems (II)

Projective Hash Families: $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\Lambda}$,
$\mathcal{H}_\lambda = \{h_k : \mathcal{X}_\lambda \to \mathcal{Z}_\lambda\}_{k\in\mathcal{K}_\lambda}$, $\alpha_\lambda : \mathcal{K}_\lambda \to \mathcal{S}_\lambda$ surjective

Definition

$(\mathcal{H} : \mathcal{X} \to \mathcal{Z}, \alpha : \mathcal{K} \to \mathcal{S})$ is a **projective hash family** for $\mathcal{L} \subset \mathcal{X}$ if $\forall k_1, k_2 \in \mathcal{K}_\lambda;\ \alpha(k_1) = \alpha(k_2) \Rightarrow h_{k_1}|_{\mathcal{L}_\lambda} = h_{k_2}|_{\mathcal{L}_\lambda}$

Definition

$(\mathcal{H}, \alpha)$ is **smooth** if $\forall x \in \overline{\mathcal{L}}_\lambda,\ \forall s \in \mathcal{S}_\lambda;\ h_{U_{\alpha^{-1}(s)}}(x) = U_{\mathcal{Z}_\lambda}$

Definition

$(\mathcal{H}, \alpha)$ is **pairwise independent** if $\forall x_1, x_2 \in \overline{\mathcal{L}}_\lambda,\ x_1 \neq x_2,\ \forall s \in \mathcal{S}_\lambda;\ (h_{U_{\alpha^{-1}(s)}}(x_1), h_{U_{\alpha^{-1}(s)}}(x_2)) = U_{\mathcal{Z}_\lambda \times \mathcal{Z}_\lambda}$

Hash Proof Systems (III)

Example of Smooth Projective Hash Family for DDH:

$\mathcal{K}_{(G,q,g_1,g_2)} = \mathbb{Z}_q \times \mathbb{Z}_q$; $\mathcal{Z}_{(G,q,g_1,g_2)} = G$

$h_{k_1,k_2}(x_1, x_2) = x_1^{k_1} x_2^{k_2}$; $\mathcal{S}_{(G,q,g_1,g_2)} = G$

$\alpha(k_1, k_2) = h_{k_1,k_2}(g_1, g_2) = g_1^{k_1} g_2^{k_2}$

Hash Proof Systems (III)

Example of Smooth Projective Hash Family for DDH:

$\mathcal{K}_{(G,q,g_1,g_2)} = \mathbb{Z}_q \times \mathbb{Z}_q$; $\mathcal{Z}_{(G,q,g_1,g_2)} = G$

$h_{k_1,k_2}(x_1, x_2) = x_1^{k_1} x_2^{k_2}$; $\mathcal{S}_{(G,q,g_1,g_2)} = G$

$\alpha(k_1, k_2) = h_{k_1,k_2}(g_1, g_2) = g_1^{k_1} g_2^{k_2}$

Projective: $h_{k_1,k_2}(g_1^w, g_2^w) = g_1^{wk_1} g_2^{wk_2} = \alpha(k_1, k_2)^w$

Hash Proof Systems (III)

Example of Smooth Projective Hash Family for DDH:

$\mathcal{K}_{(G,q,g_1,g_2)} = \mathbb{Z}_q \times \mathbb{Z}_q$; $\mathcal{Z}_{(G,q,g_1,g_2)} = G$
$h_{k_1,k_2}(x_1, x_2) = x_1^{k_1} x_2^{k_2}$; $\mathcal{S}_{(G,q,g_1,g_2)} = G$
$\alpha(k_1, k_2) = h_{k_1,k_2}(g_1, g_2) = g_1^{k_1} g_2^{k_2}$

Projective: $h_{k_1,k_2}(g_1^w, g_2^w) = g_1^{wk_1} g_2^{wk_2} = \alpha(k_1, k_2)^w$

Smooth:
$h_{k_1,k_2}(g_1^{w_1}, g_2^{w_2}) = g_1^{w_1 k_1} g_2^{w_2 k_2} = \alpha(k_1, k_2)^{w_1} g_2^{(w_2 - w_1)k_2} =$
$s^{w_1} g_2^{(w_2 - w_1)k_2}$, that is uniformly distributed in G if $w_2 \neq w_1$

Hash Proof Systems (III)

Example of Smooth Projective Hash Family for DDH:

$\mathcal{K}_{(G,q,g_1,g_2)} = \mathbb{Z}_q \times \mathbb{Z}_q$; $\mathcal{Z}_{(G,q,g_1,g_2)} = G$
$h_{k_1,k_2}(x_1, x_2) = x_1^{k_1} x_2^{k_2}$; $\mathcal{S}_{(G,q,g_1,g_2)} = G$
$\alpha(k_1, k_2) = h_{k_1,k_2}(g_1, g_2) = g_1^{k_1} g_2^{k_2}$

Projective: $h_{k_1,k_2}(g_1^w, g_2^w) = g_1^{wk_1} g_2^{wk_2} = \alpha(k_1, k_2)^w$

Smooth:
$h_{k_1,k_2}(g_1^{w_1}, g_2^{w_2}) = g_1^{w_1 k_1} g_2^{w_2 k_2} = \alpha(k_1, k_2)^{w_1} g_2^{(w_2-w_1)k_2} =$
$s^{w_1} g_2^{(w_2-w_1)k_2}$, that is uniformly distributed in G if $w_2 \neq w_1$

▸ A pairwise indep. PHF...

Hash Proof Systems (IV)

A (smooth / strongly smooth) Hash Proof System consists of

- a hard subset membership problem $\mathcal{L} \subset \mathcal{X}$ with witness set $\mathcal{W}$
- a corresponding (smooth / pairwise independent) projective hash family $(\mathcal{H} : \mathcal{X} \to \mathcal{Z}, \alpha : \mathcal{K} \to \mathcal{S})$
- efficient sampling algorithm for $\lambda \leftarrow \Lambda_\ell$, $k \leftarrow \mathcal{K}_\lambda$, $x \leftarrow \mathcal{X}_\lambda$ and $(x, w) \leftarrow \mathcal{L}_\lambda \times \mathcal{W}_\lambda$
- an efficient algorithm $h_k(x) = \text{PrivEval}(k, x)$
- an efficient algorithm $h_k(x) = \text{PubEval}(s, x, w)$ where w is a witness for $x \in \mathcal{L}_\lambda$ and $s = \alpha(k)$

Cramer-Shoup CCA Encryption

$(\mathcal{L} \subset \mathcal{X}, \mathcal{H} : \mathcal{X} \to \mathcal{Z}, \alpha : \mathcal{K} \to \mathcal{S})$ is a smooth HPS, $(\mathcal{Z}, +)$ is a group and $(\mathcal{L} \times \mathcal{Z} \subset \mathcal{X} \times \mathcal{Z}, \widehat{\mathcal{H}} : \mathcal{X} \times \mathcal{Z} \to \mathcal{Z}, \widehat{\alpha} : \widehat{\mathcal{K}} \to \widehat{\mathcal{S}})$ is a strongly smooth HPS

KeyGen(ℓ) :
$\lambda \leftarrow \Lambda_\ell$; $k \leftarrow \mathcal{K}_\lambda$; $k \leftarrow \widehat{\mathcal{K}}_\lambda$; $s \leftarrow \alpha(k)$; $\widehat{s} \leftarrow \widehat{\alpha}(\widehat{k})$;
output $((s, \widehat{s}), (k, \widehat{k}))$;

Enc$(s, \widehat{s}, m)$:
$(x, w) \leftarrow \mathcal{L}_\lambda \times \mathcal{W}_\lambda$;
$e \leftarrow$ PubEval$(s, x, w) + m$; // $e = h_k(x) + m$
$\widehat{e} \leftarrow \widehat{\text{PubEval}}(\widehat{s}, (x, e), w)$; // $\widehat{e} = \widehat{h}_{\widehat{k}}(x, e)$;
output $(x, e, \widehat{e})$;

Dec$(k, \widehat{k}, c)$:
$(x, e, \widehat{e}) = c$;
if $\widehat{e} \neq \widehat{\text{PrivEval}}(\widehat{k}, (x, e))$; **output** $\perp$
else output $e -$ PrivEval(k, x);

Cramer-Shoup CCA Encryption (II)

Why is it CCA secure?

Cramer-Shoup CCA Encryption (II)

Why is it CCA secure?

Game hopping:

- **Game 0:** The adversary plays the original PKE-IND-CCA experiment
- **Game 1:** c_* is computed from the secret keys by using $e_* \leftarrow \text{PrivEval}(k, x_*) + m_{b_*}$ and $\widehat{e}_* \leftarrow \widehat{\text{PrivEval}}(\widehat{k}, x_*, e_*)$
- **Game 2:** c_* is now computed from $x_* \leftarrow \overline{\mathcal{L}}_\lambda$
- **Game 3:** The decryption oracle now rejects all queries with $x \notin \mathcal{L}_\lambda$
- **Game 4:** e_* is taken at random

Cramer-Shoup CCA Encryption (III)

Game hops analysis:

- **Game 0↔Game 1:** Games are identical
- **Game 1↔Game 2:** Any noticeable difference solves the decision membership problem
- **Game 2↔Game 3:** There is a negligible difference between games. Observe that the challenger in Game 3 is not efficient!
- **Game 3↔Game 4:** Games are identical, but now the adversary's advantage in Game 4 is trivially 0

Cramer-Shoup CCA Encryption (III)

Game hops analysis:

- **Game 0↔Game 1:** Games are identical
- **Game 1↔Game 2:** Any noticeable difference solves the decision membership problem
- **Game 2↔Game 3:** There is a negligible difference between games. Observe that the challenger in Game 3 is not efficient!
- **Game 3↔Game 4:** Games are identical, but now the adversary's advantage in Game 4 is trivially 0

As a consequence, the adversary's advantage in Game 0 is at least negligibly close to the advantage of another adversary solving the decision membership problem

Codes and Cryptography

Jorge L. Villar

MAMME, Fall 2015

END OF PART XIV

Example of Pairwise Independent PHF for DDH:

Extended subset membership problem:

$\widehat{\mathcal{X}}_{(G,q,g_1,g_2,n)} = G \times G \times G$

$\widehat{\mathcal{L}}_{(G,q,g_1,g_2,n)} = \langle (g_1, g_2) \rangle \times G$

$\gamma : G \times G \times G \to \mathbb{Z}_q^n$ injective. Let $\mathbf{t} = (1, \gamma(x_1, x_2, e)) \in \mathbb{Z}_q^{n+1}$

The projective hash family:

$\widehat{\mathcal{K}}_{(G,q,g_1,g_2,n)} = \mathbb{Z}_q^{n+1} \times \mathbb{Z}_q^{n+1}$

$\mathcal{Z}_{(G,q,g_1,g_2)} = G$

$\widehat{h}_{\mathbf{k_1},\mathbf{k_2}}(x_1, x_2, e) = x_1^{\mathbf{k_1}\cdot\mathbf{t}} x_2^{\mathbf{k_2}\cdot\mathbf{t}}$

$\mathcal{S}_{(G,q,g_1,g_2)} = G^{n+1}$

$\widehat{\alpha}(\mathbf{k_1}, \mathbf{k_2}) = (g_1^{k_{10}} g_2^{k_{20}}, \ldots, g_1^{k_{1n}} g_2^{k_{2n}}) = (s_0, \ldots, s_n)$

◂ go back...

▸ more details...

Example of Pairwise Independent PHF for DDH:

Pairwise Independence:

$$h_{\mathbf{k_1},\mathbf{k_2}}(g_1^{w_1}, g_2^{w_2}, e) = g_1^{w_1 \mathbf{k_1}\cdot\mathbf{t}} g_2^{w_2 \mathbf{k_2}\cdot\mathbf{t}} = \left(s_0 s_1^{t_1} \cdots s_n^{t_n}\right)^{w_1} g_2^{(w_2 - w_1)\mathbf{k_2}\cdot\mathbf{t}}$$

For any $(w_1, w_2, e) \neq (w_1', w_2', e')$, $\mathbf{t}$ and $\mathbf{t}'$ are linearly independent, and for a uniformly distributed $\mathbf{k_2}$, $\mathbf{k_2}\cdot\mathbf{t}$ and $\mathbf{k_2}\cdot\mathbf{t}'$ are uniformly distributed and independent.

As a consequence, if $w_2 \neq w_1$ and $w_2' \neq w_1'$ then $h_{\mathbf{k_1},\mathbf{k_2}}(g_1^{w_1}, g_2^{w_2}, e)$ and $h_{\mathbf{k_1},\mathbf{k_2}}(g_1^{w_1'}, g_2^{w_2'}, e')$ are uniformly distributed and independent.

go back...