

Chosen-Ciphertext Security (I)

CS 601.442/642 Modern Cryptography

Fall 2017

Recall: Public-Key Encryption

- **Syntax:**

- $\text{Gen}(1^n) \rightarrow (pk, sk)$
- $\text{Enc}(pk, m) \rightarrow c$
- $\text{Dec}(sk, c) \rightarrow m'$ or \perp

All algorithms are polynomial time

- **Correctness:** For every m , $\text{Dec}(sk, \text{Enc}(pk, m)) = m$, where $(pk, sk) \leftarrow \text{Gen}(1^n)$

Recall: IND-CPA Security

Definition (IND-CPA Security)

A public-key encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is indistinguishably secure under chosen plaintext attack (IND-CPA) if for all n.u. PPT adversaries \mathcal{A} , there exists a negligible function $\mu(\cdot)$ s.t.:

$$\Pr \left[\begin{array}{l} (pk, sk) \xleftarrow{\$} \text{Gen}(1^n), \\ (m_0, m_1) \leftarrow \mathcal{A}(1^n, pk), \\ b \xleftarrow{\$} \{0, 1\} \end{array} : \mathcal{A}(pk, \text{Enc}(m_b)) = b \right] \leq \frac{1}{2} + \mu(n)$$

- 1 IND-CPA for one-message implies IND-CPA for multiple messages

Need for Stronger Security

Motivation:

- An adversary may be able to find an oracle that decrypts ciphertexts. In this case, IND-CPA security may break down!
- Real-world attacks possible, e.g., chosen ciphertext attacks on Apple iMessage [Garman-Green-Kaptchuk-Miers-Rushanan'16]

Security against Chosen-Ciphertext Attacks (CCA)

- Augment the IND-CPA security experiment
- Adversary can make decryption queries over ciphertexts of its choice
- Two security definitions:
 - CCA-1: Decryption queries only before challenge ciphertext query
 - CCA-2: Decryption queries before and after challenge ciphertext query

Note: To prevent trivial attacks, decryption queries c made by the adversary should be different from the challenge ciphertext c^* !

CCA-1 Security

Expt $_{\mathcal{A}}^{\text{CCA1}}(b, z)$:

- $\text{st} = z$
- $(pk, sk) \leftarrow \text{Gen}(1^n)$
- Decryption query phase (repeated poly times):
 - $c \leftarrow \mathcal{A}(pk, \text{st})$
 - $m \leftarrow \text{Dec}(sk, c)$
 - $\text{st} = (\text{st}, m)$
- $(m_0, m_1) \leftarrow \mathcal{A}(pk, \text{st})$
- $c^* \leftarrow \text{Enc}(pk, m_b)$
- Output $b' \leftarrow \mathcal{A}(pk, c^*, \text{st})$

CCA-1 Security (contd.)

Definition (IND-CCA-1 Security)

A public-key encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is IND-CCA-1 secure if for all n.u. PPT adversaries \mathcal{A} , there exists a negligible function $\mu(\cdot)$ s.t. for all auxiliary inputs $z \in \{0, 1\}^*$:

$$\left| \Pr \left[\mathbf{Expt}_{\mathcal{A}}^{\text{CCA1}}(1, z) = 1 \right] - \Pr \left[\mathbf{Expt}_{\mathcal{A}}^{\text{CCA1}}(0, z) = 1 \right] \right| \leq \mu(n)$$

CCA-2 Security

Expt $_{\mathcal{A}}^{\text{CCA2}}(b, z)$:

- $\text{st} = z$
- $(pk, sk) \leftarrow \text{Gen}(1^n)$
- Decryption query phase 1 (repeated poly times):
 - $c \leftarrow \mathcal{A}(pk, \text{st})$
 - $m \leftarrow \text{Dec}(sk, c)$
 - $\text{st} = (\text{st}, m)$
- $(m_0, m_1) \leftarrow \mathcal{A}(pk, \text{st})$
- $c^* \leftarrow \text{Enc}(pk, m_b)$
- Decryption query phase 2 (repeated poly times):
 - $c \leftarrow \mathcal{A}(pk, c^*, \text{st})$
 - If $c = c^*$, output reject
 - $m \leftarrow \text{Dec}(sk, c)$
 - $\text{st} = (\text{st}, m)$
- Output $b' \leftarrow \mathcal{A}(pk, c^*, \text{st})$

CCA-2 Security (contd.)

Definition (IND-CCA-2 Security)

A public-key encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is IND-CCA-1 secure if for all n.u. PPT adversaries \mathcal{A} , there exists a negligible function $\nu(\cdot)$ s.t. for all auxiliary inputs $z \in \{0, 1\}^*$:

$$\left| \Pr \left[\mathbf{Expt}_{\mathcal{A}}^{\text{CCA2}}(1, z) = 1 \right] - \Pr \left[\mathbf{Expt}_{\mathcal{A}}^{\text{CCA2}}(0, z) = 1 \right] \right| \leq \nu(n)$$

How to Construct CCA-1 Secure PKE?

Main Challenge: How to answer decryption queries?

- Suppose we want to build IND-CCA-1 secure PKE starting from IND-CPA secure PKE
- In order to rely on IND-CPA security of underlying PKE, we should not use secret key in the security experiment
- However, in order to answer decryption queries of the adversary, we need the secret key!
- How to resolve this seeming paradox?

How to Construct CCA-1 Secure PKE?

Main Idea: Use *two copies* of the encryption scheme

- Encrypt a message twice, using each of the two copies of the encryption scheme
- To answer a decryption query (c_1, c_2) , we only need to decrypt one of the two ciphertexts. Therefore, we only need to know one of the secret keys
- Use IND-CPA security of the second encryption scheme when we use sk_1 in the experiment (but not sk_2)
- Then, switch the secret key to sk_2 and use IND-CPA security of the first encryption scheme
- **Problem:** What if adversary sends decryption queries (c_1, c_2) such that c_1 and c_2 decrypt different messages?
- **Solution:** Modify the scheme so that encryption of message m also contains a NIZK proof that proves that c_1 and c_2 encrypt the same message m

CCA-1 Secure Public-Key Encryption

Theorem (Naor-Yung)

Assuming NIZKs in the CRS model and IND-CPA secure public-key encryption, there exists IND-CCA-1 secure public-key encryption

Construction

Let $(\text{Gen}, \text{Enc}, \text{Dec})$ be an IND-CPA encryption scheme.

Let $(\text{K}, \text{P}, \text{V})$ be an adaptive NIZK with Simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$.

Construction of $(\text{Gen}', \text{Enc}', \text{Dec}')$:

$\text{Gen}'(1^n)$: Execute the following steps:

- Compute $(pk_i, sk_i) \leftarrow \text{Gen}(1^n)$ for $i \in [2]$
- Compute $\sigma \leftarrow \text{K}(1^n)$
- Output $pk' = (pk_1, pk_2, \sigma)$, $sk' = sk_1$

$\text{Enc}'(pk', m)$: Execute the following steps:

- Compute $c_i \leftarrow \text{Enc}(pk_i, m; r_i)$ for $i \in [2]$
- Compute $\pi \leftarrow \text{P}(\sigma, x, w)$ where $x = (pk_1, pk_2, c_1, c_2)$, $w = (m, r_1, r_2)$ and $R(x, w) = 1$ iff c_1 and c_2 encrypt the same message m
- Output $C = (c_1, c_2, \pi)$

$\text{Dec}'(sk', c')$: If $\text{V}(\sigma, \pi) = 0$, output \perp . Else output $\text{Dec}(sk_1, c_1)$.

Hybrids

Hybrid $H_0 := \text{Expt}_{\mathcal{A}}^{\text{CCA1}}(0, z)$:

- $(pk_i, sk_i) \leftarrow \text{Gen}(1^n)$ for $i \in [2]$
- $\sigma \leftarrow \text{K}(1^n)$
- $pk' = (pk_1, pk_2, \sigma)$, $sk' = sk_1$
- On receiving a decryption query $c = (c_1, c_2, \pi)$ from $\mathcal{A}(z, pk')$, if $\text{V}(\sigma, x = (c_1, c_2), \pi) = 1$, return $\text{Dec}(sk' = sk_1, c_1)$
- $(m_0, m_1) \leftarrow \mathcal{A}(z, pk')$,
- $c_1^* \leftarrow \text{Enc}(pk_1, m_0; r_1^*)$, $c_2^* \leftarrow \text{Enc}(pk_2, m_0; r_2^*)$
- $\pi^* \leftarrow \text{P}\left(\sigma, x^* = (c_1^*, c_2^*), w^* = (r_1, r_2)\right)$
- Output $\mathcal{A}(z, pk', C = (c_1^*, c_2^*, \pi^*))$

Hybrids (contd.)

Hybrid H_1 : Simulate the CRS in public-key and simulate the proof in challenge ciphertext

- $(pk_i, sk_i) \leftarrow \text{Gen}(1^n)$ for $i \in [2]$
- $(\sigma, \tau) \leftarrow \mathcal{S}_0(1^n)$
- $pk' = (pk_1, pk_2, \sigma), sk' = sk_1$
- For every decryption query $c = (c_1, c_2, \pi)$ from $\mathcal{A}(z, pk')$, if $\mathbb{V}(\sigma, x = (c_1, c_2), \pi) = 1$, return $\text{Dec}(sk' = sk_1, c_1)$
- $(m_0, m_1) \leftarrow \mathcal{A}(z, pk')$,
- $c_1^* \leftarrow \text{Enc}(pk_1, m_0; r_1^*), c_2^* \leftarrow \text{Enc}(pk_2, m_0; r_2^*)$
- $\pi^* \leftarrow \mathcal{S}_1(\sigma, \tau, x^* = (c_1^*, c_2^*))$
- Output $\mathcal{A}(z, pk', C = (c_1^*, c_2^*, \pi^*))$

$H_0 \approx_c H_1$: Follows from ZK property of NIZK

Hybrids (contd.)

Hybrid H_2 : Modify c_2^* in challenge ciphertext to be an encryption of m_1

- $(pk_i, sk_i) \leftarrow \text{Gen}(1^n)$ for $i \in [2]$
- $(\sigma, \tau) \leftarrow \mathcal{S}_0(1^n)$
- $pk' = (pk_1, pk_2, \sigma)$, $sk' = sk_1$
- For every decryption query $c = (c_1, c_2, \pi)$ from $\mathcal{A}(z, pk')$, if $\mathbb{V}(\sigma, x = (c_1, c_2), \pi) = 1$, return $\text{Dec}(sk' = sk_1, c_1)$
- $(m_0, m_1) \leftarrow \mathcal{A}(z, pk')$,
- $c_1^* \leftarrow \text{Enc}(pk_1, m_0; r_1^*)$, $c_2^* \leftarrow \text{Enc}(pk_2, m_1; r_2^*)$
- $\pi^* \leftarrow \mathcal{S}_1(\sigma, \tau, x^* = (c_1^*, c_2^*))$
- Output $\mathcal{A}(z, pk', C = (c_1^*, c_2^*, \pi^*))$

$H_1 \approx_c H_2$: Follows from IND-CPA security of the PKE

Hybrids (contd.)

Hybrid H_3 : Change decryption key to sk_2

- $(pk_i, sk_i) \leftarrow \text{Gen}(1^n)$ for $i \in [2]$
- $(\sigma, \tau) \leftarrow \mathcal{S}_0(1^n)$
- $pk' = (pk_1, pk_2, \sigma)$, $sk' = sk_2$
- For every decryption query $c = (c_1, c_2, \pi)$ from $\mathcal{A}(z, pk')$, if $\mathcal{V}(\sigma, x = (c_1, c_2), \pi) = 1$, return $\text{Dec}(sk' = sk_2, c_2)$
- $(m_0, m_1) \leftarrow \mathcal{A}(z, pk')$,
- $c_1^* \leftarrow \text{Enc}(pk_1, m_0; r_1^*)$, $c_2^* \leftarrow \text{Enc}(pk_2, m_1; r_2^*)$
- $\pi^* \leftarrow \mathcal{S}_1(\sigma, \tau, x^* = (c_1^*, c_2^*))$
- Output $\mathcal{A}(z, pk', C = (c_1^*, c_2^*, \pi^*))$

$H_2 \approx H_3$: From soundness of NIZK, it follows that in every decryption query (c_1, c_2, π) made by \mathcal{A} , c_1 and c_2 encrypt the *same* message.

Therefore, decrypting c_1 or c_2 yields the same answer

Hybrids (contd.)

Hybrid H_4 : Modify c_1^* in challenge ciphertext to be an encryption of m_1

- $(pk_i, sk_i) \leftarrow \text{Gen}(1^n)$ for $i \in [2]$
- $(\sigma, \tau) \leftarrow \mathcal{S}_0(1^n)$
- $pk' = (pk_1, pk_2, \sigma)$, $sk' = sk_2$
- For every decryption query $c = (c_1, c_2, \pi)$ from $\mathcal{A}(z, pk')$, if $\mathbb{V}(\sigma, x = (c_1, c_2), \pi) = 1$, return $\text{Dec}(sk' = sk_2, c_2)$
- $(m_0, m_1) \leftarrow \mathcal{A}(z, pk')$,
- $c_1^* \leftarrow \text{Enc}(pk_1, m_1; r_1^*)$, $c_2^* \leftarrow \text{Enc}(pk_2, m_1; r_2^*)$
- $\pi^* \leftarrow \mathcal{S}_1(\sigma, \tau, x^* = (c_1^*, c_2^*))$
- Output $\mathcal{A}(z, pk', C = (c_1^*, c_2^*, \pi^*))$

$H_3 \approx_c H_4$: Follows from IND-CPA security of the PKE

Hybrids (contd.)

Hybrid H_5 : Change decryption key back to sk_1

- $(pk_i, sk_i) \leftarrow \text{Gen}(1^n)$ for $i \in [2]$
- $(\sigma, \tau) \leftarrow \mathcal{S}_0(1^n)$
- $pk' = (pk_1, pk_2, \sigma)$, $sk' = sk_1$
- For every decryption query $c = (c_1, c_2, \pi)$ from $\mathcal{A}(z, pk')$, if $\forall(\sigma, x = (c_1, c_2), \pi) = 1$, return $\text{Dec}(sk' = sk_1, c_1)$
- $(m_0, m_1) \leftarrow \mathcal{A}(z, pk')$,
- $c_1^* \leftarrow \text{Enc}(pk_1, m_1; r_1^*)$, $c_2^* \leftarrow \text{Enc}(pk_2, m_1; r_2^*)$
- $\pi^* \leftarrow \mathcal{S}_1(\sigma, \tau, x^* = (c_1^*, c_2^*))$
- Output $\mathcal{A}(z, pk', C = (c_1^*, c_2^*, \pi^*))$

$H_4 \approx H_5$: Follows in the same manner as $H_2 \approx H_3$

Hybrids (contd.)

Hybrid H_6 := $\text{Expt}_{\mathcal{A}}^{\text{CCA1}}(1, z)$: Honestly compute the CRS in public-key and the proof in challenge ciphertext

- $(pk_i, sk_i) \leftarrow \text{Gen}(1^n)$ for $i \in [2]$
- $\sigma \leftarrow \mathcal{K}(1^n)$
- $pk' = (pk_1, pk_2, \sigma)$, $sk' = sk_1$
- For every decryption query $c = (c_1, c_2, \pi)$ from $\mathcal{A}(z, pk')$, if $\mathcal{V}(\sigma, x = (c_1, c_2), \pi) = 1$, return $\text{Dec}(sk' = sk_1, c_1)$
- $(m_0, m_1) \leftarrow \mathcal{A}(z, pk')$,
- $c_1^* \leftarrow \text{Enc}(pk_1, m_1; r_1^*)$, $c_2^* \leftarrow \text{Enc}(pk_2, m_1; r_2^*)$
- $\pi^* \leftarrow \mathcal{P}(\sigma, \tau, x^* = (c_1^*, c_2^*), w^* = (r_1^*, r_2^*))$
- Output $\mathcal{A}(z, pk', C = (c_1^*, c_2^*, \pi^*))$

$H_5 \approx_c H_6$: Follows from the ZK property of NIZK