Chosen-Ciphertext Security

CS 601.442/642 Modern Cryptography

Fall 2020

Recall: Public-Key Encryption

- **Syntax:**
  - $\mathsf{Gen}(1^n) \to (pk, sk)$
  - $\mathsf{Enc}(pk, m) \to c$
  - $\mathsf{Dec}(sk, c) \to m'$ or $\perp$

  All algorithms are polynomial time

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

Recall: IND-CPA Security

Definition (IND-CPA Security)

A public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ 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.:

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

Recall: IND-CPA Security

Definition (IND-CPA Security)

A public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ 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.:

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

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!

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

Security against Chosen-Ciphertext Attacks (CCA)

- Augment the IND-CPA security experiment
- Adversary can make decryption queries over ciphertexts of its choice

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:

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

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

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

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 rule out trivial attacks, decryption queries c made by the adversary in IND-CCA-2 should be different from the challenge ciphertext c^*!

CCA-1 Security

$\mathbf{Expt}_{\mathcal{A}}^{\mathsf{CCA1}}(b)$:

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

CCA-1 Security (contd.)

Definition (IND-CCA-1 Security)

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

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

CCA-2 Security

$\mathbf{Expt}_{\mathcal{A}}^{\mathsf{CCA2}}(b)$:

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

CCA-2 Security (contd.)

Definition (IND-CCA-2 Security)

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

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

How to Construct CCA-1 Secure PKE?

Main Challenge: How should the reduction answer decryption queries?

- Suppose we want to build IND-CCA-1 secure PKE starting from IND-CPA secure PKE

How to Construct CCA-1 Secure PKE?

Main Challenge: How should the reduction 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 reduction

How to Construct CCA-1 Secure PKE?

Main Challenge: How should the reduction 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 reduction
- However, in order to answer decryption queries of the adversary, we need the secret key!

How to Construct CCA-1 Secure PKE?

Main Challenge: How should the reduction 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 reduction
- 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

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), the reduction only needs to decrypt one of the two ciphertexts. Therefore, the reduction only need to know one of the secret keys

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), the reduction only needs to decrypt one of the two ciphertexts. Therefore, the reduction only need to know one of the secret keys
- Use IND-CPA security of the second encryption scheme when the reduction uses sk_1 in the experiment (but not sk_2)

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), the reduction only needs to decrypt one of the two ciphertexts. Therefore, the reduction only need to know one of the secret keys
- Use IND-CPA security of the second encryption scheme when the reduction uses 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

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), the reduction only needs to decrypt one of the two ciphertexts. Therefore, the reduction only need to know one of the secret keys
- Use IND-CPA security of the second encryption scheme when the reduction uses 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?

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), the reduction only needs to decrypt one of the two ciphertexts. Therefore, the reduction only need to know one of the secret keys
- Use IND-CPA security of the second encryption scheme when the reduction uses 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 and IND-CPA secure public-key encryption, there exists IND-CCA-1 secure public-key encryption

CCA-1 Secure Public-Key Encryption

Theorem (Naor-Yung)

Assuming NIZKs and IND-CPA secure public-key encryption, there exists IND-CCA-1 secure public-key encryption

- *Random Oracle model*: If we use NIZKs in the random oracle (RO) model, the resulting encryption scheme is also in the RO model.

CCA-1 Secure Public-Key Encryption

Theorem (Naor-Yung)

Assuming NIZKs and IND-CPA secure public-key encryption, there exists IND-CCA-1 secure public-key encryption

- *Random Oracle model*: If we use NIZKs in the random oracle (RO) model, the resulting encryption scheme is also in the RO model.
- *Standard model*: If we use NIZKs in the *common random string* (CRS) model, we can obtain an IND-CCA-1 encryption scheme in the standard model. The CRS of the NIZK is generated by the key generation algorithm of the encryption scheme.

How to Construct CCA-2 secure Encryption?

- Why doesn't a CCA-1 secure scheme also achieve CCA-2 security?

How to Construct CCA-2 secure Encryption?

- Why doesn't a CCA-1 secure scheme also achieve CCA-2 security?
- **Main problem:** An adversary may be able to modify the challenge ciphertext to obtain a new ciphertext of a *related* plaintext and then request its decryption in the second decryption query phase of IND-CCA-2. E.g., the adversary may be able to "maul" an encryption of x into an encryption of $x \oplus 1$ without knowing x. This is called *malleability attack*

  Think: Is the IND-CPA PKE scheme based on trapdoor permutations that we studied in the class *malleable*?

How to Construct CCA-2 secure Encryption?

- Why doesn't a CCA-1 secure scheme also achieve CCA-2 security?
- **Main problem:** An adversary may be able to modify the challenge ciphertext to obtain a new ciphertext of a *related* plaintext and then request its decryption in the second decryption query phase of IND-CCA-2. E.g., the adversary may be able to "maul" an encryption of x into an encryption of $x \oplus 1$ without knowing x. This is called *malleability attack*

  Think: Is the IND-CPA PKE scheme based on trapdoor permutations that we studied in the class *malleable*?
- **Solution Strategy:** Ensure that adversary's decryption query is "independent" of (and not just different from) the challenge ciphertext. That is, make the encryption *non-malleable*

CCA-2 Secure Public-Key Encryption

The first construction of CCA-2 secure encryption scheme was given by Dolev, Dwork and Naor.

Ingredients:

- An IND-CPA secure encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$
- A NIZK proof (P, V) (for simplicity of notation, we use NIZK in Random oracle model, but the construction also works if we use NIZKs in CRS model)
- A *strongly unforgeable* one-time signature (OTS) scheme $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$, where adversary cannot output a new forgery (i.e., a new signature) even on a message for which he has already seen a signature. Assume, wlog, that verification keys in OTS scheme are of length n.

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

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

- Compute $2n$ key pairs of IND-CPA encryption scheme: $\left(pk_i^j, sk_i^j\right) \leftarrow \mathsf{Gen}(1^n)$, where $j \in \{0,1\}$, $i \in [n]$.
- Output $pk' = \left(\{pk_i^0, pk_i^1\}\right)$, $sk' = \left(sk_1^0, sk_1^1\right)$.

Construction (contd.)

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

- Compute key pair for OTS scheme: $(SK, VK) \leftarrow \mathsf{Setup}(1^n)$.
- Let $VK = VK_1, \ldots, VK_n$. For every $i \in [n]$, encrypt m using $pk_i^{VK_i}$ and randomness r_i: $c_i \leftarrow \mathsf{Enc}\left(pk_i^{VK_i}, m; r_i\right)$
- Compute proof that each c_i encrypts the same message: $\pi \leftarrow \mathsf{P}(x, w)$ where $x = \left(\left\{pk_i^{VK_i}\right\}, \{c_i\}\right)$, $w = (m, \{r_i\})$ and $R(x, w) = 1$ iff every c_i encrypts the same message m.
- Sign everything: $\Phi \leftarrow \mathsf{Sign}(SK, M)$ where $M = (\{c_i\}, \pi)$
- Output $c' = (VK, \{c_i\}, \pi, \Phi)$

Construction (contd.)

$\mathsf{Dec}'(sk', c')$: Execute the following steps

- Parse $c' = (VK, \{c_i\}, \pi, \Phi)$
- Let $M = (\{c_i\}, \pi)$
- Verify the signature: Output $\perp$ if $\mathsf{Verify}(VK, M, \Phi) = 0$
- Verify the NIZK proof: Output $\perp$ if $\mathsf{V}(x, \pi) = 0$ where $x = \left(\left\{pk_i^{VK_i}\right\}, \{c_i\}\right)$
- Else, decrypt the first ciphertext component: $m' \leftarrow \mathsf{Dec}\left(sk_1^{VK_1}, c_1\right)$
- Output m'

Security (Intuition)

Consider decryption queries after adversary receives challenge ciphertext C^*:

- Let $C \neq C^*$ be a decryption query

Security (Intuition)

Consider decryption queries after adversary receives challenge ciphertext C^*:

- Let $C \neq C^*$ be a decryption query
- If verification key VK in C and verification key VK^* in challenge ciphertext C^* are same, then we can break the strong unforgeability of OTS

Security (Intuition)

Consider decryption queries after adversary receives challenge ciphertext C^*:

- Let $C \neq C^*$ be a decryption query
- If verification key VK in C and verification key VK^* in challenge ciphertext C^* are same, then we can break the strong unforgeability of OTS
- If different, then VK and VK^* differ in at least one position $\ell \in [n]$:

Security (Intuition)

Consider decryption queries after adversary receives challenge ciphertext C^*:

- Let $C \neq C^*$ be a decryption query
- If verification key VK in C and verification key VK^* in challenge ciphertext C^* are same, then we can break the strong unforgeability of OTS
- If different, then VK and VK^* differ in at least one position $\ell \in [n]$:
  - Answer decryption query using the secret key $sk_\ell^{VK_i}$.

Security (Intuition)

Consider decryption queries after adversary receives challenge ciphertext C^*:

- Let $C \neq C^*$ be a decryption query
- If verification key VK in C and verification key VK^* in challenge ciphertext C^* are same, then we can break the strong unforgeability of OTS
- If different, then VK and VK^* differ in at least one position $\ell \in [n]$:
  - Answer decryption query using the secret key $sk_\ell^{VK_i}$.
  - Don't need to know the secret keys $sk_i^{VK_i^*}$ for $i \in [n]$

Security (Intuition)

Consider decryption queries after adversary receives challenge ciphertext C^*:

- Let $C \neq C^*$ be a decryption query
- If verification key VK in C and verification key VK^* in challenge ciphertext C^* are same, then we can break the strong unforgeability of OTS
- If different, then VK and VK^* differ in at least one position $\ell \in [n]$:
  - Answer decryption query using the secret key $sk_\ell^{VK_i}$.
  - Don't need to know the secret keys $sk_i^{VK_i^*}$ for $i \in [n]$
  - Reduce to IND-CPA security of underlying encryption scheme

Security (Hybrids)

- H_0: (Honest) Encryption of m_0

Security (Hybrids)

- H_0: (Honest) Encryption of m_0
- H_1: Compute proof π in challenge ciphertext using NIZK simulator

Security (Hybrids)

- H_0: (Honest) Encryption of m_0
- H_1: Compute proof π in challenge ciphertext using NIZK simulator
- H_2: Choose VK^* in the beginning during Gen'

Security (Hybrids)

- H_0: (Honest) Encryption of m_0
- H_1: Compute proof π in challenge ciphertext using NIZK simulator
- H_2: Choose VK^* in the beginning during Gen'
- H_3: For any decryption query $C = (VK, \{c_i\}, \pi, \Phi)$:

Security (Hybrids)

- H_0: (Honest) Encryption of m_0
- H_1: Compute proof π in challenge ciphertext using NIZK simulator
- H_2: Choose VK^* in the beginning during Gen'
- H_3: For any decryption query $C = (VK, \{c_i\}, \pi, \Phi)$:
  - If $VK = VK^*$ and $\mathsf{Verify}(VK, (\{c_i\}, \pi), \Phi) = 1$, then abort

Security (Hybrids)

- H_0: (Honest) Encryption of m_0
- H_1: Compute proof π in challenge ciphertext using NIZK simulator
- H_2: Choose VK^* in the beginning during Gen'
- H_3: For any decryption query $C = (VK, \{c_i\}, \pi, \Phi)$:
  - If $VK = VK^*$ and $\mathsf{Verify}(VK, (\{c_i\}, \pi), \Phi) = 1$, then abort
  - Else, let $\ell \in [n]$ be such that VK^* and VK in C differ at position ℓ. Set $sk' = \left\{ sk_i^{\overline{VK}_i^*} \right\}$, $i \in [n]$, where $\overline{VK}_i^* = 1 - VK_i^*$. Decrypt C by decrypting c_ℓ (instead of c_1) using $sk_\ell^{\overline{VK}_\ell^*}$.

Security (Hybrids)

- H_0: (Honest) Encryption of m_0
- H_1: Compute proof π in challenge ciphertext using NIZK simulator
- H_2: Choose VK^* in the beginning during Gen'
- H_3: For any decryption query $C = (VK, \{c_i\}, \pi, \Phi)$:
  - If $VK = VK^*$ and $\mathsf{Verify}(VK, (\{c_i\}, \pi), \Phi) = 1$, then abort
  - Else, let $\ell \in [n]$ be such that VK^* and VK in C differ at position ℓ. Set $sk' = \left\{ sk_i^{\overline{VK}_i^*} \right\}$, $i \in [n]$, where $\overline{VK}_i^* = 1 - VK_i^*$. Decrypt C by decrypting c_ℓ (instead of c_1) using $sk_\ell^{\overline{VK}_\ell^*}$.
- H_4: Change every c_i^* in C^* to encryption of m_1

Security (Hybrids)

- H_0: (Honest) Encryption of m_0
- H_1: Compute proof π in challenge ciphertext using NIZK simulator
- H_2: Choose VK^* in the beginning during Gen'
- H_3: For any decryption query $C = (VK, \{c_i\}, \pi, \Phi)$:
  - If $VK = VK^*$ and $\mathsf{Verify}(VK, (\{c_i\}, \pi), \Phi) = 1$, then abort
  - Else, let $\ell \in [n]$ be such that VK^* and VK in C differ at position ℓ. Set $sk' = \left\{ sk_i^{\overline{VK}_i^*} \right\}$, $i \in [n]$, where $\overline{VK}_i^* = 1 - VK_i^*$. Decrypt C by decrypting c_ℓ (instead of c_1) using $sk_\ell^{\overline{VK}_\ell^*}$.
- H_4: Change every c_i^* in C^* to encryption of m_1
- H_5: Compute proof π in challenge ciphertext honestly. This experiment is same as (honest) encryption of m_1.

Indistinguishability of Hybrids

- $H_0 \approx H_1$: ZK property of NIZK

Indistinguishability of Hybrids

- $H_0 \approx H_1$: ZK property of NIZK
- $H_1 \approx H_2$: Generating VK^* early or later does not change the distribution

Indistinguishability of Hybrids

- $H_0 \approx H_1$: ZK property of NIZK
- $H_1 \approx H_2$: Generating VK^* early or later does not change the distribution
- $H_2 \approx H_3$: We argue indistinguishability as follows:

Indistinguishability of Hybrids

- $H_0 \approx H_1$: ZK property of NIZK
- $H_1 \approx H_2$: Generating VK^* early or later does not change the distribution
- $H_2 \approx H_3$: We argue indistinguishability as follows:
  - First, we argue that probability of aborting is negligible. Recall that $C \neq C^*$ by the definition of CCA-2. Then, if $VK = VK^*$, it must be that $(\{c_i\}, \pi, \Phi) \neq (\{c_i^*\}, \pi^*, \Phi^*)$. Now, if $\mathsf{Verify}(VK, (\{c_i\}, \pi), \Phi) = 1$, then we can break strong unforgeability of the OTS scheme.

Indistinguishability of Hybrids

- $H_0 \approx H_1$: ZK property of NIZK
- $H_1 \approx H_2$: Generating VK^* early or later does not change the distribution
- $H_2 \approx H_3$: We argue indistinguishability as follows:
  - First, we argue that probability of aborting is negligible. Recall that $C \neq C^*$ by the definition of CCA-2. Then, if $VK = VK^*$, it must be that $(\{c_i\}, \pi, \Phi) \neq (\{c_i^*\}, \pi^*, \Phi^*)$. Now, if $\mathsf{Verify}(VK, (\{c_i\}, \pi), \Phi) = 1$, then we can break strong unforgeability of the OTS scheme.
  - Now, conditioned on not aborting, let ℓ be the position s.t. $VK_\ell \neq VK_\ell^*$. Note that the only difference in H_2 and H_3 in this case might be the answers to the decryption queries of adversary. In particular, in H_2, we decrypt c_1 in C using $sk_1^{VK_1}$. In contrast, in H_3, we decrypt c_ℓ in C using $sk_\ell^{\overline{VK_\ell^*}}$. Now, from soundness of NIZK, it follows that except with negligible probability, all the c_i's in C encrypt the same message. Therefore decrypting c_ℓ instead of c_1 does not change the answer.

Indistinguishability of Hybrids (contd.)

- $H_3 \approx H_4$: IND-CPA security of underlying PKE
- $H_4 \approx H_5$: ZK property of NIZK

Indistinguishability of Hybrids (contd.)

- $H_3 \approx H_4$: IND-CPA security of underlying PKE
- $H_4 \approx H_5$: ZK property of NIZK

Combining the above, we get $H_0 \approx H_5$.