

RUHR-UNIVERSITÄT BOCHUM

## On the Selective Opening Security of Practical Public-Key Encryption Schemes

PKC 2015, NIST, Maryland: March 30, 2015

**Felix Heuer**, Tibor Jager, Eike Kiltz, Sven Schäge  
Horst Görtz Institute for IT Security  
Ruhr University Bochum

- 1 Selective-Opening Security
- 2 A Generic Transformation for SIM-SO-CCA Security
- 3 Proof Idea
- 4 Results

# Selective-Opening Attacks

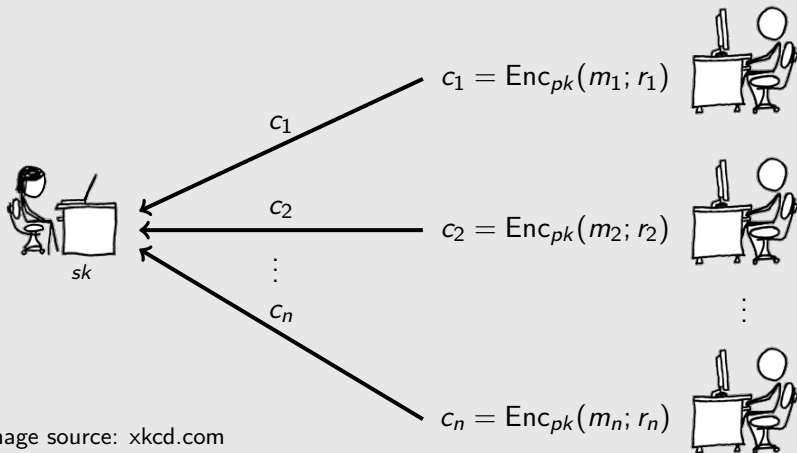


Image source: xkcd.com

# Selective-Opening Attacks

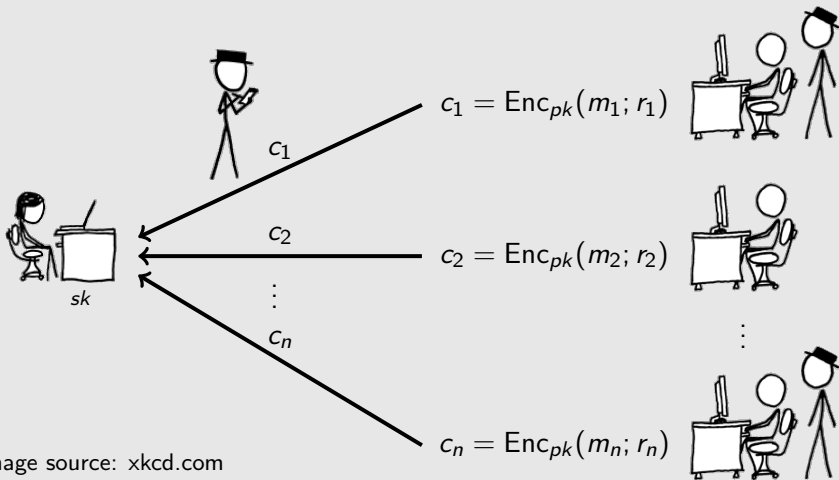
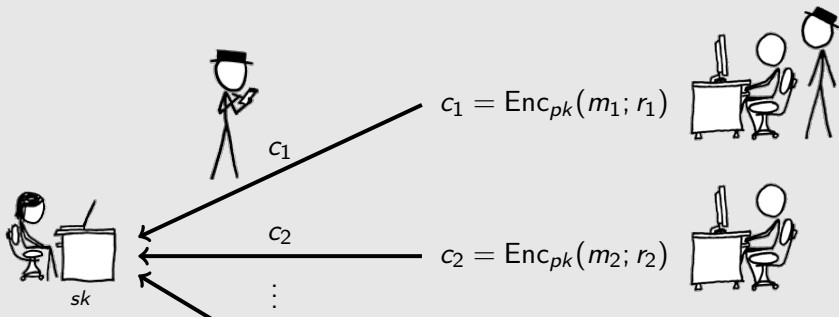


Image source: xkcd.com

# Selective-Opening Attacks

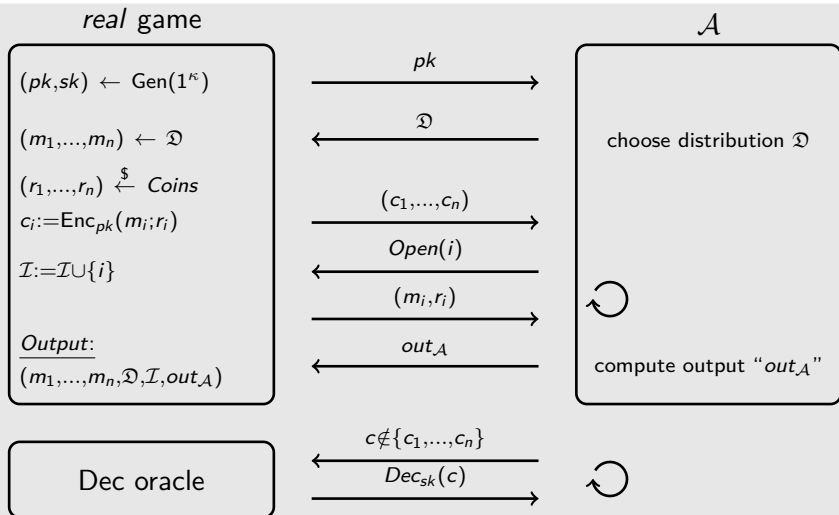


**Do the messages of uncorrupted parties remain confidential?**

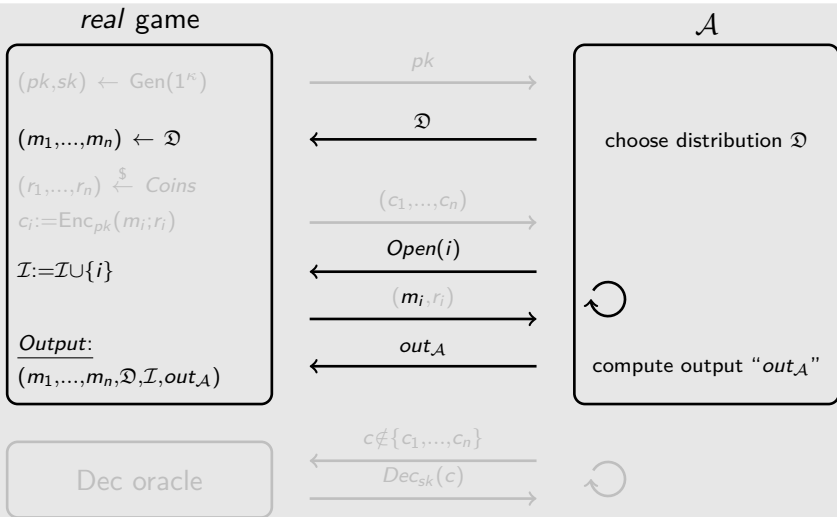
Image source: xkcd.com

$$c_n = \text{Enc}_{pk}(m_n; r_n)$$

## SIM-SO-CCA Security Definition [FHKW10]

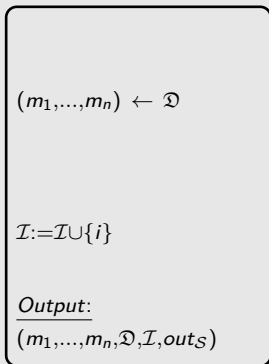


# SIM-SO-CCA Security Definition [FHKW10]

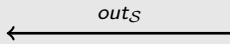
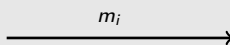
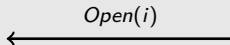
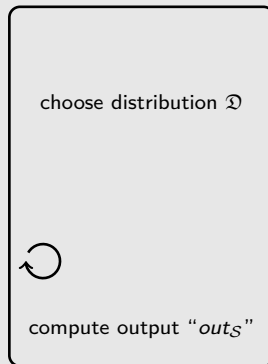


# SIM-SO-CCA Security Definition [FHKW10]

*ideal game*



$\mathcal{S}$





## Definition 1 (SIM-SO-CCA security)

A public key encryption scheme is SIM-SO-CCA secure if for every PPT adversary  $\mathcal{A}$  there exists a PPT simulator  $\mathcal{S} := \mathcal{S}(\mathcal{A})$  such that the distributions induced by

$\mathcal{A}$  run in the *real* game      and       $\mathcal{S}$  run in the *ideal* game

are computationally indistinguishable.

- two flavours: IND-based and SIM-based.  
SIM implies IND, not known to be equivalent.

- two flavours: IND-based and SIM-based.  
SIM implies IND, not known to be equivalent.
- IND-SO-CCA stronger security notion than IND-CCA security.  
[HR14]

- two flavours: IND-based and SIM-based.  
SIM implies IND, not known to be equivalent.
- IND-SO-CCA stronger security notion than IND-CCA security.  
[HR14]
- Existing SIM-SO-CCA schemes are very inefficient. [FHKW10],  
[Hof12], [LP15]

- two flavours: IND-based and SIM-based.  
SIM implies IND, not known to be equivalent.
- IND-SO-CCA stronger security notion than IND-CCA security.  
[HR14]
- Existing SIM-SO-CCA schemes are very inefficient. [FHKW10], [Hof12], [LP15]
- **This work:** Certain known constructions give SIM-SO-CCA security for free in the ROM.

Part I:

PKE from any one-way PCA secure KEM

Part II:

PKE from OAEP for any partial-domain TP

Part I:

PKE from any one-way PCA secure KEM

Part II:

PKE from OAEP for any partial-domain TP

## “Hashed KEM/DEM Approach”

Let  $KEM = (KGen, Encap, Decap)$  be a Key Encapsulation Mechanism,  $MAC = (Tag, Vrfy)$  a Message Authentication Code and  $H$  a hash function. Consider the following PKE:



## “Hashed KEM/DEM Approach”

Let  $KEM = (KGen, Encap, Decap)$  be a Key Encapsulation Mechanism,  $MAC = (Tag, Vrfy)$  a Message Authentication Code and  $H$  a hash function. Consider the following PKE:

$Gen(1^\lambda)$

$(pk, sk) \leftarrow KGen(1^\lambda)$

Return  $pk$

# “Hashed KEM/DEM Approach”

Let  $KEM = (KGen, Encap, Decap)$  be a Key Encapsulation Mechanism,  $MAC = (Tag, Vrfy)$  a Message Authentication Code and  $H$  a hash function. Consider the following PKE:

|  |   |
|--|---|
| $\underline{Gen(1^\lambda)}$ $(pk, sk) \leftarrow KGen(1^\lambda)$ $\text{Return } pk$ | $\underline{Enc_{pk}(m)}$ $r \xleftarrow{\$} Coins$ $(k, c^{(1)}) \leftarrow Encap_{pk}(r)$ $(k^{sym}, k^{mac}) \leftarrow H(k)$ $c^{(2)} := m \oplus k^{sym}$ $c^{(3)} := Tag_{k^{mac}}(c^{(2)})$ $\text{Return } (c^{(1)}, c^{(2)}, c^{(3)})$ |
|--|---|

# “Hashed KEM/DEM Approach”

Let  $KEM = (KGen, Encap, Decap)$  be a Key Encapsulation Mechanism,  $MAC = (Tag, Vrfy)$  a Message Authentication Code and  $H$  a hash function. Consider the following PKE:

Gen( $1^\lambda$ )

$(pk, sk) \leftarrow KGen(1^\lambda)$

Return  $pk$

Enc $_{pk}(m)$

$r \xleftarrow{\$} Coins$

$(k, c^{(1)}) \leftarrow Encap_{pk}(r)$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

$c^{(2)} := m \oplus k^{sym}$

$c^{(3)} := Tag_{k^{mac}}(c^{(2)})$

Return  $(c^{(1)}, c^{(2)}, c^{(3)})$

Dec $_{sk}(c^{(1)}, c^{(2)}, c^{(3)})$

$k \leftarrow Decap_{sk}(c^{(1)})$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

if  $Vrfy_{k^{mac}}(c^{(2)}, c^{(3)}) = 1$

Return  $c^{(2)} \oplus k^{sym}$

else

Return  $\perp$

# “Hashed KEM/DEM Approach”

## Theorem 2 (SBZ02)

*The given transformation achieves IND-CCA security in the ROM if KEM is OW-PCA secure and MAC is sUF-OT-CMA.*

Gen( $1^\lambda$ )

$(pk, sk) \leftarrow \text{KGen}(1^\lambda)$

Return  $pk$

Enc $_{pk}(m)$

$r \xleftarrow{\$} \text{Coins}$

$(k, c^{(1)}) \leftarrow \text{Encap}_{pk}(r)$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

$c^{(2)} := m \oplus k^{sym}$

$c^{(3)} := \text{Tag}_{k^{mac}}(c^{(2)})$

Return  $(c^{(1)}, c^{(2)}, c^{(3)})$

Dec $_{sk}(c^{(1)}, c^{(2)}, c^{(3)})$

$k \leftarrow \text{Decap}_{sk}(c^{(1)})$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

if  $\text{Vrfy}_{k^{mac}}(c^{(2)}, c^{(3)}) = 1$

Return  $c^{(2)} \oplus k^{sym}$

else

Return  $\perp$

# “Hashed KEM/DEM Approach”

## Theorem 2 (SBZ02)

*The given transformation achieves IND-CCA security in the ROM if KEM is OW-PCA secure and MAC is sUF-OT-CMA.*

Gen( $1^\lambda$ )

$(pk, sk) \leftarrow \text{KGen}(1^\lambda)$

Return  $pk$

Enc $_{pk}(m)$

$r \xleftarrow{\$} \text{Coins}$

$(k, c^{(1)}) \leftarrow \text{Encap}_{pk}(r)$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

$c^{(2)} := m \oplus k^{sym}$

$c^{(3)} := \text{Tag}_{k^{mac}}(c^{(2)})$

Return  $(c^{(1)}, c^{(2)}, c^{(3)})$

Dec $_{sk}(c^{(1)}, c^{(2)}, c^{(3)})$

$k \leftarrow \text{Decap}_{sk}(c^{(1)})$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

if  $\text{Vrfy}_{k^{mac}}(c^{(2)}, c^{(3)}) = 1$

Return  $c^{(2)} \oplus k^{sym}$

else

Return  $\perp$

– Use RSA KEM under RSA assumption

# “Hashed KEM/DEM Approach”

## Theorem 2 (SBZ02)

*The given transformation achieves IND-CCA security in the ROM if KEM is OW-PCA secure and MAC is sUF-OT-CMA.*

Gen( $1^\lambda$ )

$(pk, sk) \leftarrow \text{KGen}(1^\lambda)$

Return  $pk$

Enc $_{pk}(m)$

$r \xleftarrow{\$} \text{Coins}$

$(k, c^{(1)}) \leftarrow \text{Encap}_{pk}(r)$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

$c^{(2)} := m \oplus k^{sym}$

$c^{(3)} := \text{Tag}_{k^{mac}}(c^{(2)})$

Return  $(c^{(1)}, c^{(2)}, c^{(3)})$

Dec $_{sk}(c^{(1)}, c^{(2)}, c^{(3)})$

$k \leftarrow \text{Decap}_{sk}(c^{(1)})$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

if  $\text{Vrfy}_{k^{mac}}(c^{(2)}, c^{(3)}) = 1$

Return  $c^{(2)} \oplus k^{sym}$

else

Return  $\perp$

- Use RSA KEM under RSA assumption
- Use DH KEM under strong DH assumption

**Instantiation  
of DHIES**

# “Hashed KEM/DEM Approach”

## Theorem 3 (This work)

*The same transformation gives rise to a SIM-SO-CCA secure PKE in the ROM without additional assumptions.*

Gen( $1^\lambda$ )

$(pk, sk) \leftarrow \text{KGen}(1^\lambda)$

Return  $pk$

Enc $_{pk}(m)$

$r \xleftarrow{\$} \text{Coins}$

$(k, c^{(1)}) \leftarrow \text{Encap}_{pk}(r)$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

$c^{(2)} := m \oplus k^{sym}$

$c^{(3)} := \text{Tag}_{k^{mac}}(c^{(2)})$

Return  $(c^{(1)}, c^{(2)}, c^{(3)})$

Dec $_{sk}(c^{(1)}, c^{(2)}, c^{(3)})$

$k \leftarrow \text{Decap}_{sk}(c^{(1)})$

$(k^{sym}, k^{mac}) \leftarrow H(k)$

if  $\text{Vrfy}_{k^{mac}}(c^{(2)}, c^{(3)}) = 1$

Return  $c^{(2)} \oplus k^{sym}$

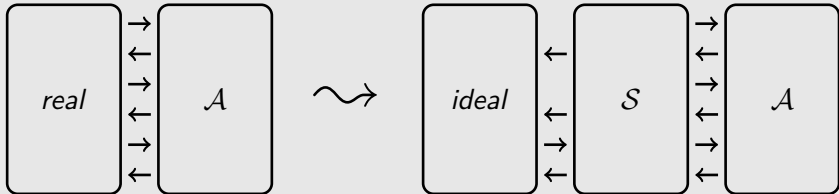
else

Return  $\perp$

- Use RSA KEM under RSA assumption
- Use DH KEM under strong DH assumption

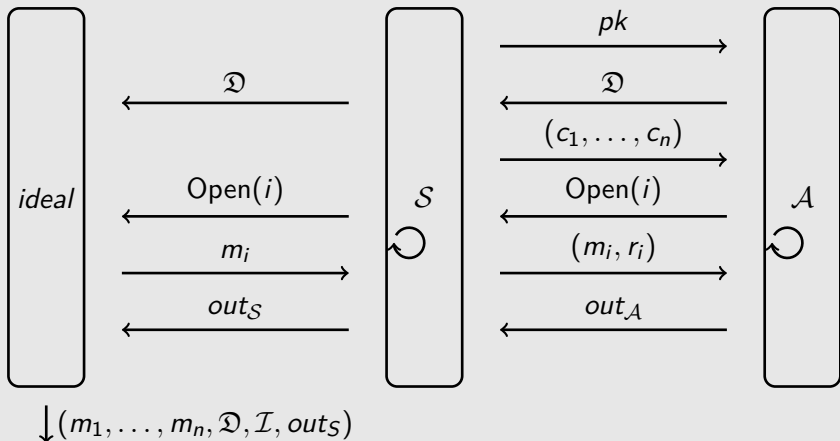
**Instantiation  
of DHIES**

# How to Prove SIM-SO-CCA Security

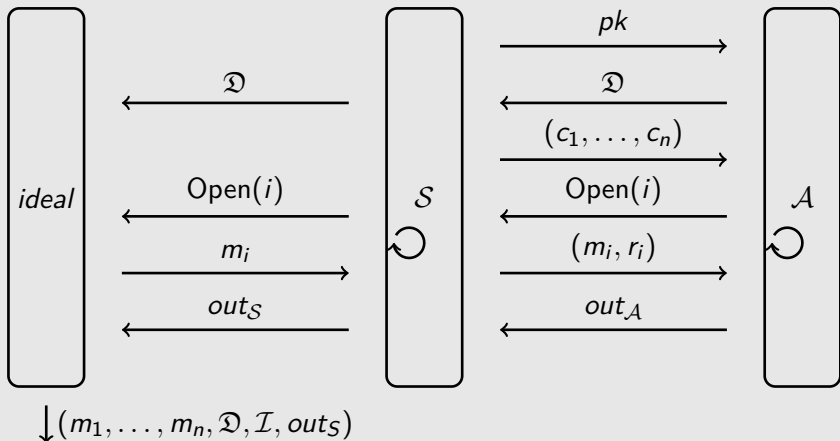




# How to Construct a Simulator



# How to Construct a Simulator



$A$  is allowed to make additional **Hash** or **Dec** queries at any time!

# Possible Tripping Hazards for a Simulator

- $\mathcal{S}$  must not make more opening queries than  $\mathcal{A}$  to learn  $m_i$ .

# Possible Tripping Hazards for a Simulator

- $\mathcal{S}$  must not make more opening queries than  $\mathcal{A}$  to learn  $m_i$ .
- $\mathcal{S}$  has to create non-committing “dummy”-encryptions that allow for later opening to any message.

# Possible Tripping Hazards for a Simulator

- $\mathcal{S}$  must not make more opening queries than  $\mathcal{A}$  to learn  $m_i$ .
- $\mathcal{S}$  has to create non-committing “dummy”-encryptions that allow for later opening to any message.
- Answering Hash or Decryption queries is easy if  $\mathcal{A}$  called  $\text{Open}(i)$  earlier.

## Possible Tripping Hazards for a Simulator

- $\mathcal{S}$  must not make more opening queries than  $\mathcal{A}$  to learn  $m_i$ .
- $\mathcal{S}$  has to create non-committing “dummy”-encryptions that allow for later opening to any message.
- Answering Hash or Decryption queries is easy if  $\mathcal{A}$  called  $\text{Open}(i)$  earlier.
- However,  $\mathcal{S}$  has to answer Hash and Decryption queries without committing to  $m_i$  if  $\mathcal{A}$  did not call  $\text{Open}(i)$  (yet).

# Tweaking $\mathcal{A}$ in a Sequence of Games

$G_0$ : *real* SIM-SO-CCA game.

$$\begin{aligned} & \text{Enc}_{pk}(m_i) \\ & r_i \xleftarrow{\$} \text{Coins} \\ & (k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i) \\ & (k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i) \\ & c_i^{(2)} := m_i \oplus k_i^{\text{sym}} \\ & c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)}) \\ & \text{Return } (c_i^{(1)}, c_i^{(2)}, c_i^{(3)}) \end{aligned}$$

$G_0$

# Tweaking $\mathcal{A}$ in a Sequence of Games

$G_0$ : *real* SIM-SO-CCA game.

$G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .

$$\text{Enc}_{pk}(m_i)$$

$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := m_i \oplus k_i^{\text{sym}}$$

$$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$$

$$\text{Return } (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$$
 $G_0$ 
 $G_1$



# Tweaking $\mathcal{A}$ in a Sequence of Games

$G_0$ : real SIM-SO-CCA game.

$G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .

$\text{Enc}_{pk}(m_i)$

$r_i \xleftarrow{\$} \text{Coins}$

$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$

$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$

$c_i^{(2)} := m_i \oplus k_i^{\text{sym}}$

$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$

Return  $(c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$

$G_0 \approx_s G_1$

statistical  
argument

# Tweaking $\mathcal{A}$ in a Sequence of Games

$G_0$ : *real* SIM-SO-CCA game.

$G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .

$G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.

$\text{Enc}_{pk}(m_i)$

$r_i \xleftarrow{\$} \text{Coins}$

$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$

$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$

$c_i^{(2)} := \text{uniform}$

$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$

Return  $(c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$

$G_0 \approx_s G_1 \quad G_2$

statistical  
argument

# Tweaking $\mathcal{A}$ in a Sequence of Games

$G_0$ : *real* SIM-SO-CCA game.

$G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .

$G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.

$\text{Enc}_{pk}(m_i)$

$r_i \xleftarrow{\$} \text{Coins}$

$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$

$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$

$c_i^{(2)} := \text{uniform}$

$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$

Return  $(c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$

$$G_0 \approx_s G_1 = G_2$$

statistical  
argument

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : real SIM-SO-CCA game.
- $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .
- $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.

$\text{Enc}_{pk}(m_i)$   
 $r_i \xleftarrow{\$} \text{Coins}$   
 $(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$   
 $(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$   
 $c_i^{(2)} := \text{uniform}$   
 $c_i^{(3)} := \text{Tag}_{mac}(c_i^{(2)})$   
 $(c_i^{(2)}, c_i^{(3)})$

How to answer Hash or Decryption queries?  
 Assume  $\mathcal{A}$  makes a valid  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  query:

$G_0$   
st  
an

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : real SIM-SO-CCA game.
- $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .
- $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.

$$\begin{aligned} & \text{Enc}_{pk}(m_i) \\ & r_i \xleftarrow{\$} \text{Coins} \\ & (k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i) \\ & (k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i) \\ & c_i^{(2)} := \text{uniform} \\ & c_i^{(3)} := \text{Tag}_{mac}(c_i^{(2)}) \end{aligned}$$

How to answer Hash or Decryption queries?  
Assume  $\mathcal{A}$  makes a valid  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  query:

Case 1:  
 $H(k_i)$  is not defined

$G_0$

st  
an

# Tweaking $\mathcal{A}$ in a Sequence of Games

$G_0$ : real SIM-SO-CCA game.

$G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .

$G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.

$$\text{Enc}_{pk}(m_i)$$

$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := \text{uniform}$$

$$c_i^{(3)} := \text{Tag}_{mac}(c_i^{(2)})$$

$$(c_i^{(2)}, c_i^{(3)})$$

How to answer Hash or Decryption queries?

Assume  $\mathcal{A}$  makes a valid  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  query:

Case 1:

$H(k_i)$  is not defined

Case 2:

$H(k_i)$  is defined

$G_0$

st  
an

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : real SIM-SO-CCA game.  
 $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .  
 $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.

$\text{Enc}_{pk}(m_i)$   
 $r_i \xleftarrow{\$} \text{Coins}$   
 $(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$   
 $(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$   
 $c_i^{(2)} := \text{uniform}$   
 $c_i^{(3)} := \text{Tag}_{mac}(c_i^{(2)})$

How to answer Hash or Decryption queries?  
 Assume  $\mathcal{A}$  makes a valid  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  query:

Case 1:  
 $H(k_i)$  is not defined  $\rightsquigarrow$   $k_i^{\text{mac}}$  still uniform,  
 use MAC security

Case 2:  
 $H(k_i)$  is defined

$G_3$

# Tweaking $\mathcal{A}$ in a Sequence of Games

$G_0$ : real SIM-SO-CCA game.

$G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .

$G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.

$$\text{Enc}_{pk}(m_i)$$

$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := \text{uniform}$$

$$c_i^{(3)} := \text{Tag}_{\text{mac}}(c_i^{(2)})$$

$$(c_i^{(2)}, c_i^{(3)})$$

How to answer Hash or Decryption queries?

Assume  $\mathcal{A}$  makes a valid  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  query:

Case 1:

$H(k_i)$  is not defined



$k_i^{\text{mac}}$  still uniform,  
use MAC security

Case 2:

$H(k_i)$  is defined



$\mathcal{A}$  decapsulated  $c_i^{(1)}$ ,  
use KEM security

$G_0$

st  
an

$G_3$

$G_4$



## Tweaking $\mathcal{A}$ in a Sequence of Games

$G_0$ : *real* SIM-SO-CCA game.

$G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .

$G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.

$\text{Enc}_{pk}(m_i)$

$r_i \xleftarrow{\$} \text{Coins}$

$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$

$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$

$c_i^{(2)} := \text{uniform}$

$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$

Return  $(c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$

$$G_0 \approx_s G_1 = G_2$$

statistical  
argument

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : real SIM-SO-CCA game.  
 $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .  
 $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.  
 $G_3$ : Abort if  $\mathcal{A}$  issues a valid decryption query  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  and  $H(k_i)$  is not yet defined.

$$\text{Enc}_{pk}(m_i)$$

$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := \text{uniform}$$

$$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$$

$$\text{Return } (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$$

$$G_0 \approx_s G_1 = G_2 = G_3$$

statistical  
argument

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : real SIM-SO-CCA game.  
 $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .  
 $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.  
 $G_3$ : Abort if  $\mathcal{A}$  issues a valid decryption query  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  and  $H(k_i)$  is not yet defined.

$$\text{Enc}_{pk}(m_i)$$

$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := \text{uniform}$$

$$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$$

$$\text{Return } (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$$

$$G_0 \approx_s G_1 = G_2 \approx_c G_3$$

statistical  
argument

MAC security

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : real SIM-SO-CCA game.  
 $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .  
 $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.  
 $G_3$ : Abort if  $\mathcal{A}$  issues a valid decryption query  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  and  $H(k_i)$  is not yet defined.  
 $G_4$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$ .

$$\text{Enc}_{pk}(m_i)$$

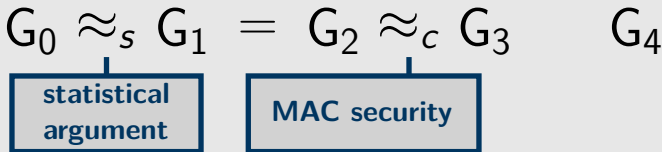
$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := \text{uniform}$$

$$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$$

$$\text{Return } (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$$


# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : real SIM-SO-CCA game.  
 $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .  
 $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.  
 $G_3$ : Abort if  $\mathcal{A}$  issues a valid decryption query  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  and  $H(k_i)$  is not yet defined.  
 $G_4$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$ .

$\text{Enc}_{pk}(m_i)$   
 $r_i \xleftarrow{\$} \text{Coins}$   
 $(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$   
 $(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$   
 $c_i^{(2)} := \text{uniform}$   
 $c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$   
 Return  $(c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$

**KEM security**

$$G_0 \approx_s G_1 = G_2 \approx_c G_3 \approx_c G_4$$

**statistical  
argument**

**MAC security**

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : real SIM-SO-CCA game.  
 $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .  
 $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.  
 $G_3$ : Abort if  $\mathcal{A}$  issues a valid decryption query  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  and  $H(k_i)$  is not yet defined.  
 $G_4$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$ .

$$\text{Enc}_{pk}(m_i)$$

$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := \text{uniform}$$

$$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$$

$$\text{Return } (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$$

**KEM security**

$$G_0 \approx_s G_1 = G_2 \approx_c G_3 \approx_c G_4$$

**statistical  
argument**

**MAC security**

**Oracle patching  
technique. [CS03]  
(Needs PCA KEM)**

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : *real* SIM-SO-CCA game.  
 $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .  
 $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.  
 $G_3$ : Abort if  $\mathcal{A}$  issues a valid decryption query  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  and  $H(k_i)$  is not yet defined.  
 $G_4$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$ .  
 $G_5$ : *ideal* SIM-SO-CCA game.

$$\text{Enc}_{pk}(m_i)$$

$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := \text{uniform}$$

$$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$$

 Return  $(c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$ 

**KEM security**

$G_0 \approx_s G_1 = G_2 \approx_c G_3 \approx_c G_4 = G_5$

**statistical argument**

**MAC security**

**Oracle patching technique. [CS03] (Needs PCA KEM)**

# Tweaking $\mathcal{A}$ in a Sequence of Games

- $G_0$ : *real* SIM-SO-CCA game.  
 $G_1$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$  or  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  before sending  $\mathcal{D}$ .  
 $G_2$ : Replace  $c_i^{(2)}$  with uniform randomness.  
 $G_3$ : Abort if  $\mathcal{A}$  issues a valid decryption query  $\text{Dec}(c_i^{(1)}, \cdot, \cdot)$  and  $H(k_i)$  is not yet defined.  
 $G_4$ : Abort if  $\mathcal{A}$  queries  $H(k_i)$ .  
 $G_5$ : *ideal* SIM-SO-CCA game.

$$\text{Enc}_{pk}(m_i)$$

$$r_i \xleftarrow{\$} \text{Coins}$$

$$(k_i, c_i^{(1)}) \leftarrow \text{Encap}_{pk}(r_i)$$

$$(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$$

$$c_i^{(2)} := \text{uniform}$$

$$c_i^{(3)} := \text{Tag}_{k_i^{\text{mac}}}(c_i^{(2)})$$

$$\text{Return } (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$$

**KEM security**

$$G_0 \approx_s G_1 = G_2 \approx_c G_3 \approx_c G_4 = G_5$$

**statistical argument**

**MAC security**

**Oracle patching technique. [CS03]  
(Needs PCA KEM)**



- SIM-SO-CCA secure PKE from generic construction in the ROM

- SIM-SO-CCA secure PKE from generic construction in the ROM
- No additional assumptions compared to proof for IND-CCA security

- SIM-SO-CCA secure PKE from generic construction in the ROM
- No additional assumptions compared to proof for IND-CCA security
- Employs OT-MAC – can be constructed information-theoretically

- SIM-SO-CCA secure PKE from generic construction in the ROM
- No additional assumptions compared to proof for IND-CCA security
- Employs OT-MAC – can be constructed information-theoretically
- Can be instantiated with any OW-PCA secure KEM, such as RSA KEM or DH KEM, latter gives instantiation of DHIES.

- SIM-SO-CCA secure PKE from generic construction in the ROM
- No additional assumptions compared to proof for IND-CCA security
- Employs OT-MAC – can be constructed information-theoretically
- Can be instantiated with any OW-PCA secure KEM, such as RSA KEM or DH KEM, latter gives instantiation of DHIES.
- Same proof-ideas can be applied to prove RSA-OAEP SIM-SO-CCA secure in the ROM.

- SIM-SO-CCA secure PKE from generic construction in the ROM
- No additional assumptions compared to proof for IND-CCA security
- Employs OT-MAC – can be constructed information-theoretically
- Can be instantiated with any OW-PCA secure KEM, such as RSA KEM or DH KEM, latter gives instantiation of DHIES.
- Same proof-ideas can be applied to prove RSA-OAEP SIM-SO-CCA secure in the ROM.



**Certain transformations give SIM-SO-CCA security for free in the ROM**



Image source: [xkcd.com](http://xkcd.com)

- SIM-SO-CCA secure PKE from generic construction in the ROM
- No additional assumptions compared to proof for IND-CCA security
- Employs OT-MAC – can be constructed information-theoretically
- Can be instantiated with any OW-PCA secure KEM, such as RSA KEM or DH KEM, latter gives instantiation of DHIES.
- Same proof-ideas can be applied to prove RSA-OAEP SIM-SO-CCA secure in the ROM.



Thank you!



**Certain transformations give SIM-SO-CCA security for free in the ROM**



Image source: [xkcd.com](http://xkcd.com)