

Homomorphic Encryption with Chosen-Ciphertext Security

Manoj Prabhakaran (UIUC)

Mike Rosulek (UIUC)

Encryption: Security vs. Features

Computational
Features



Non-malleability
(lack of "undesired features")

Encryption: Security vs. Features

Computational
Features



CPA

Non-malleability
(lack of "undesired features")

Encryption: Security vs. Features

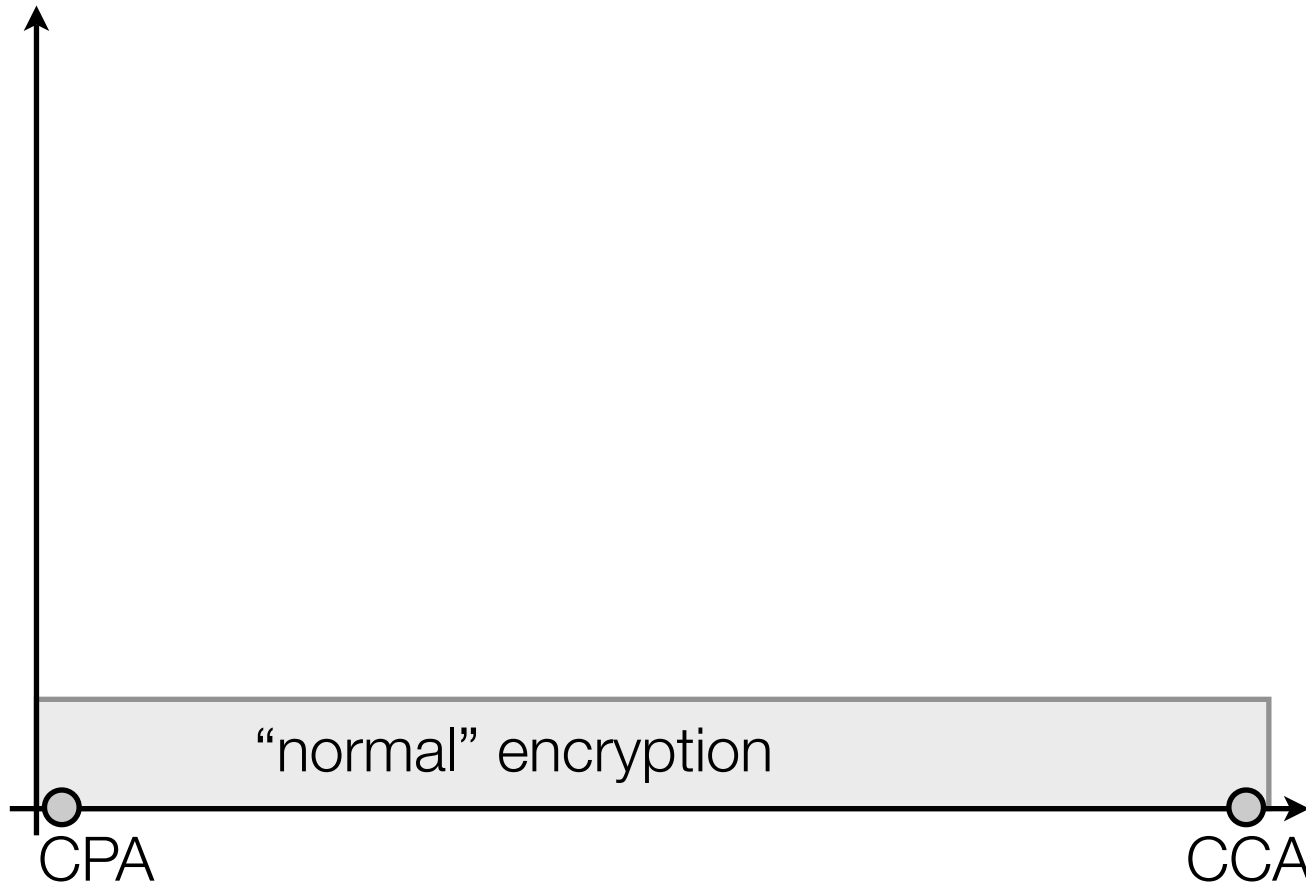
Computational
Features



Non-malleability
(lack of "undesired features")

Encryption: Security vs. Features

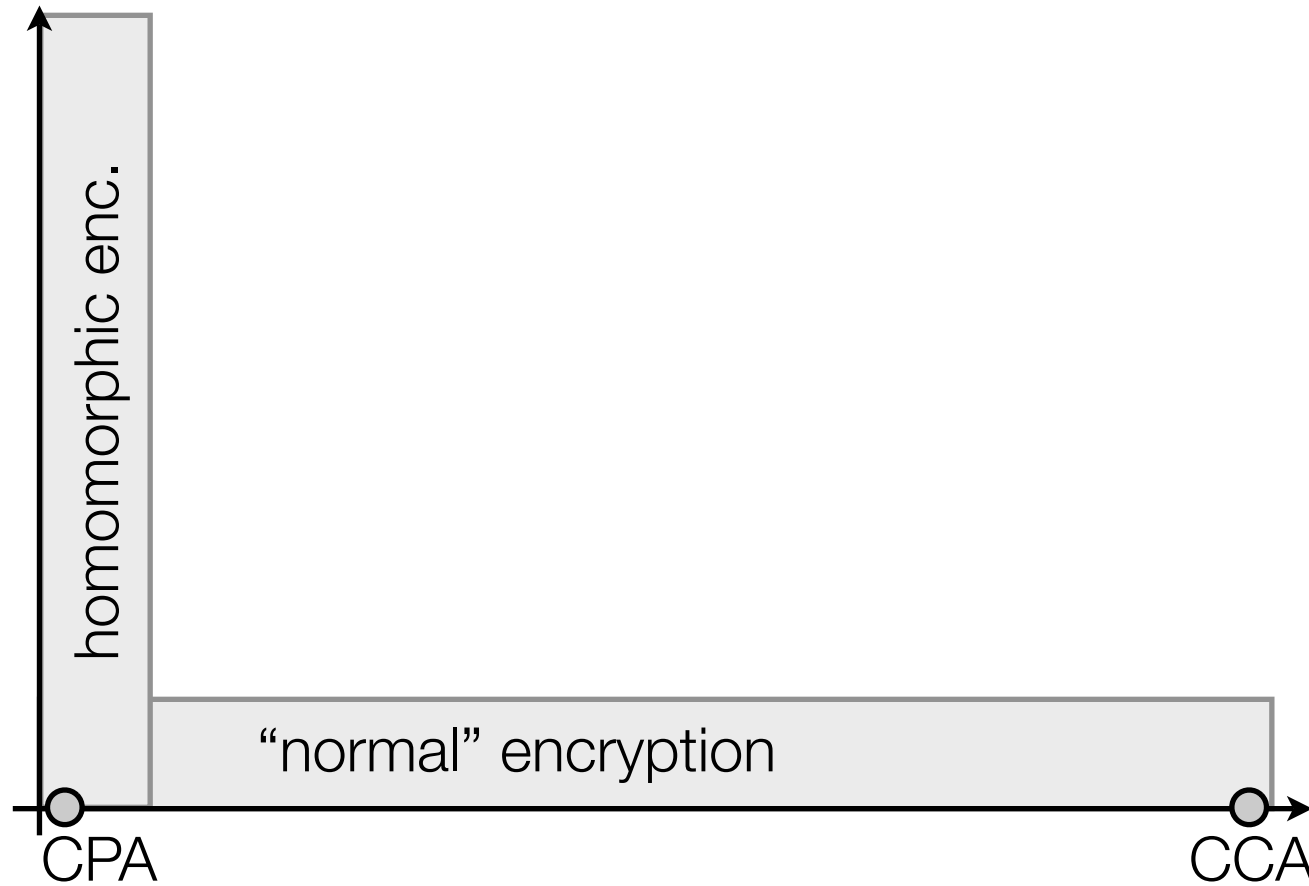
Computational
Features



Non-malleability
(lack of “undesired features”)

Encryption: Security vs. Features

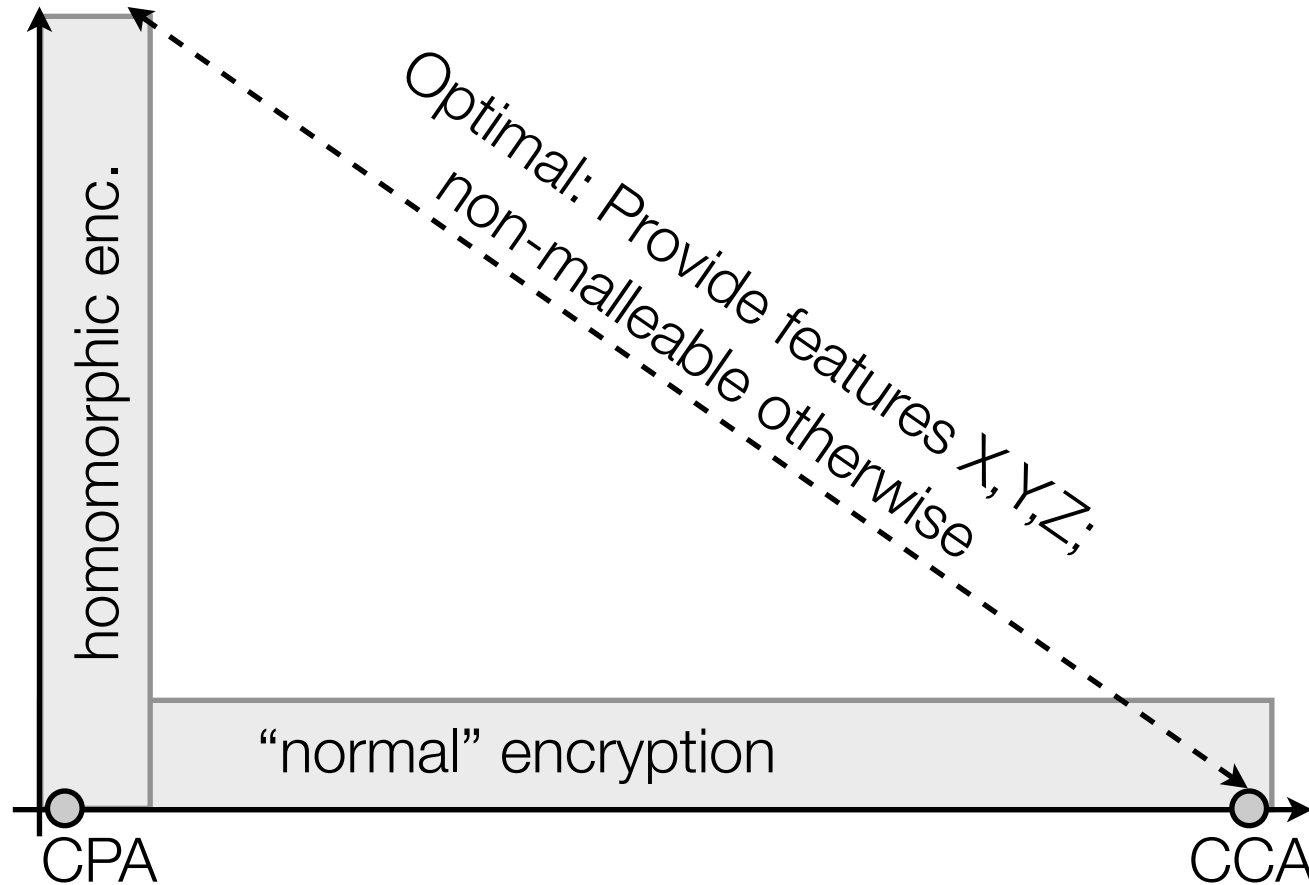
Computational
Features



Non-malleability
(lack of "undesired features")

Encryption: Security vs. Features

Computational
Features



Non-malleability
(lack of “undesired features”)

Unlinkable unary homomorphism

Anyone can change $\text{Enc}(m)$ into *fresh* $\text{Enc}(f(m))$

- Only for allowed functions f
- Cannot use ciphertexts in *any other way*
- example: only allow group operation; $f(m) = c \cdot m$

Need security definition formalizing “non-malleability except for allowed f ’s”

Defining security: Relaxing from CCA

1. Generate key, provide Dec oracle

2. Adv chooses m_0, m_1

3. Adv gets $C = \text{Enc}(m_0)$

4. Provide Dec oracle, except:
- refuses to decrypt C

1. Generate key, provide Dec oracle

2. Adv chooses m_0, m_1

3. Adv gets $C = \text{Enc}(m_1)$

4. Provide Dec oracle, except:
- refuses to decrypt C

Defining security: Relaxing from CCA

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_0)$

4. Provide Dec oracle, except:
- refuses to decrypt C

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_1)$
- m_1 known & fixed

4. Provide Dec oracle, except:
- refuses to decrypt C

Defining security: Relaxing from CCA

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_0)$

4. Provide Dec oracle, except:

- answer " m_0 " for C

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_1)$

- m_1 known & fixed

4. Provide Dec oracle, except:

- respond " m_0 " for C

Defining security: Relaxing from CCA

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_0)$

4. Provide Dec oracle

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_1)$
- m_1 known & fixed

4. Provide Dec oracle, except:
- respond " m_0 " for C

Defining security: Relaxing from CCA

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_0)$

4. Provide Dec

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_1)$
- m_1 known & fixed

4. Provide Dec oracle, except:
- should compensate if given a
“legitimate derivative” of C

Defining security: Relaxing from CCA

1. Generate key, provide Dec oracle
2. Adv chooses m_0
3. Adv gets $C = \text{Enc}(m_0)$
4. Provide Dec

1. Generate key, provide Dec oracle
2. Adv chooses m_0
3. Adv gets $C = \text{Enc}(m_1)$
 - m_1 known & fixed
4. Provide Dec oracle, except:
 - should compensate if given a *“legitimate derivative”* of C

Problem:

How to tell derivatives of C
from non-derivatives?

HCCA: trackable “pseudo-cipherexts”

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_0)$

4. Provide Dec

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{RigEnc}()$

4. Provide Dec oracle, except:

- if $\text{RigExtract}(C') = f$, respond: $f(m_0)$

HCCA: trackable “pseudo-cipherexts”

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_0)$


4. Provide Dec

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{RigEnc}()$

4. Provide Dec oracle, except:
- if $\text{RigExtract}(C') = f$, respond: $f(m_0)$



RigExtract's job:
determine how C'
was derived from C

HCCA: trackable “pseudo-cipherexts”

1. Generate key, provide Dec oracle

2. Adv chooses m_0

3. Adv gets $C = \text{Enc}(m_0)$


4. Provide Dec

1. Generate key, provide Dec oracle

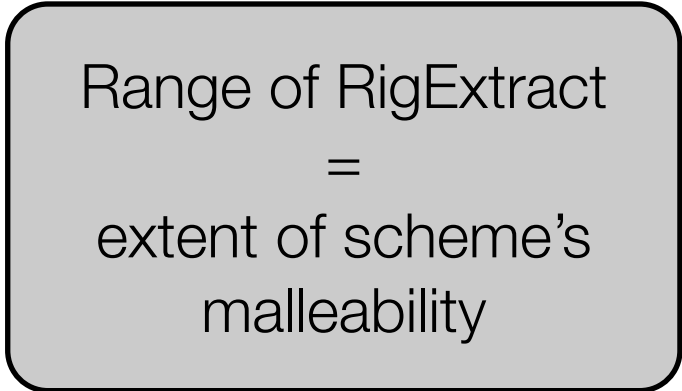
2. Adv chooses m_0

3. Adv gets $C = \text{RigEnc}()$

4. Provide Dec oracle, except:
- if $\text{RigExtract}(C') = f$, respond: $f(m_0)$



RigExtract's job:
determine how C'
was derived from C



Range of RigExtract
=
extent of scheme's
malleability

Results:

New definition called Homomorphic-CCA (HCCA)

- “scheme has no unexpected malleability”

Additional unlinkability requirement:

- “desired homomorphic operations available”
- “ciphertext doesn’t leak its history”

Definitions are justified:

- Above 2 imply natural UC definition
- HCCA subsumes CCA, gCCA, RCCA

Achieve definitions via construction (allow group operation)

Homomorphic operations that combine multiple messages?

- Show impossibility of getting binary group operation feature

The End

For more information, consult paper (or me):

- <http://eprint.iacr.org/2008/079>

Thank you!