Homomorphic Encryption with CCA Security

Manoj Prabhakaran & Mike Rosulek

ICALP 2008 – July 9, 2008

Thanks to IFIP for travel support

Opposing Demands for Encryption

Computational Features

Ciphertexts are active objects:

- Message homomorphism
- Proxy re-encryption
- Keyword search
- Attribute-/identity-based

Opposing Demands for Encryption

Computational Features

Ciphertexts are active objects:

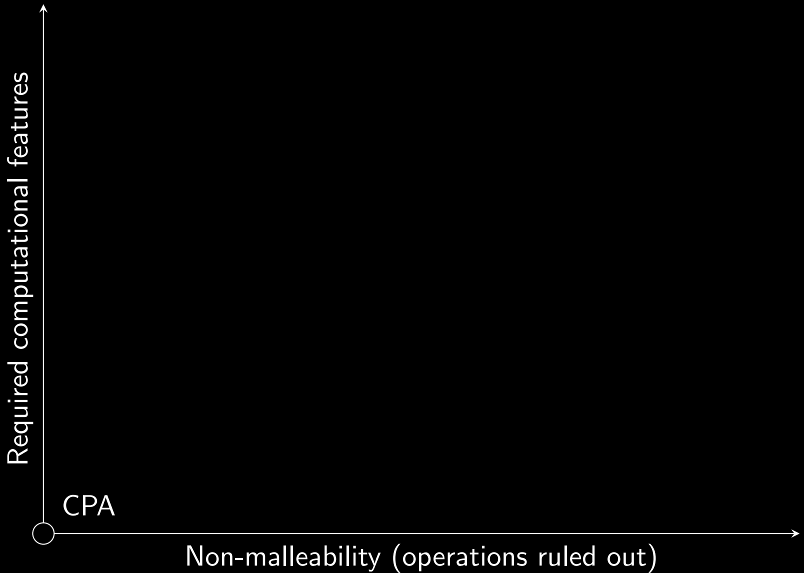
- Message homomorphism
- Proxy re-encryption
- Keyword search
- Attribute-/identity-based

Non-malleability

Require lack "unexpected operations" an adversary may exploit

A Map of Encryption Requirements

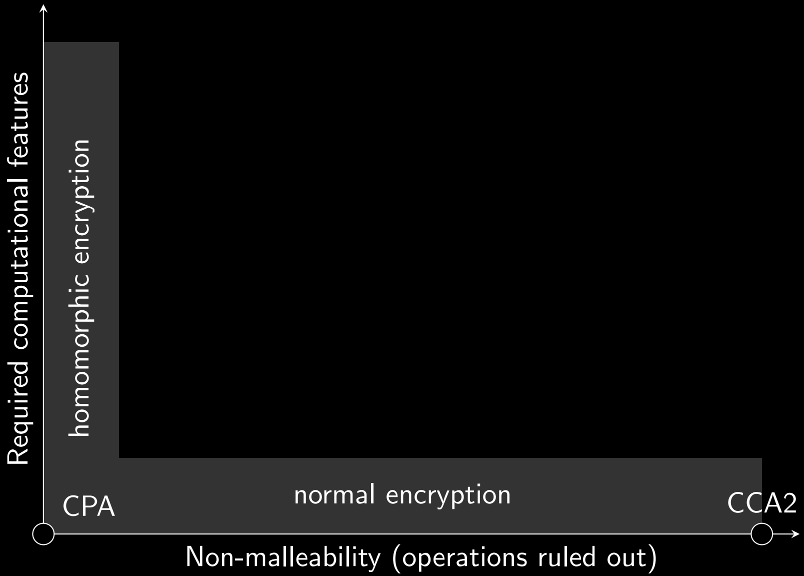
A Map of Encryption Requirements



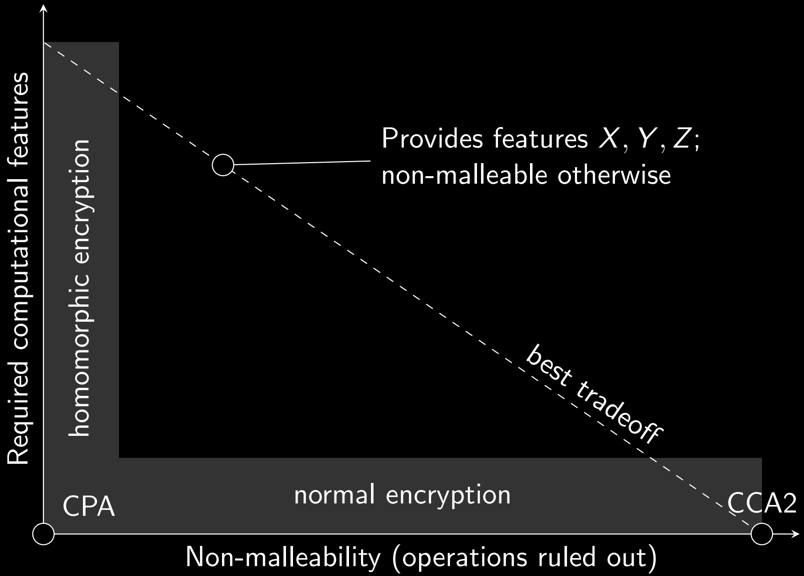
A Map of Encryption Requirements

A Map of Encryption Requirements

A Map of Encryption Requirements



A Map of Encryption Requirements



The Problem

Non-malleability is traditionally all (CCA) or nothing (CPA)

Desired Security Requirement

Scheme is *non-malleable, except for explicitly allowed features.*

The Problem

Non-malleability is traditionally all (CCA) or nothing (CPA)

Desired Security Requirement

Scheme is *non-malleable, except for explicitly allowed features.*

Challenges:

- Rigorously, convincingly define "partial non-malleability"
- Achieve definition via construction

The Problem

Non-malleability is traditionally all (CCA) or nothing (CPA)

Desired Security Requirement

Scheme is *non-malleable, except for explicitly allowed features.*

Challenges:

- Rigorously, convincingly define "partial non-malleability"
- Achieve definition via construction

In this work:

- Address problem in context of homomorphic encryption
- New general-purpose non-malleability definition
- New family of constructions

Unary Homomorphic Encryption

Desired features:

- Anyone can change Enc(m) into fresh Enc($f(m)$).
- Scheme parameterized by set of allowed f's

Example: Rerandomizable Replayable-CCA (RCCA) [CKN03,G04,PR07]:

- Only allowed f is identity function
- Non-malleable in any ways that alter message

Unary Homomorphic Encryption

Desired features:

- Anyone can change $\mathsf{Enc}(m)$ into fresh $\mathsf{Enc}(f(m))$.
- Scheme parameterized by set of allowed f's

Example: Rerandomizable Replayable-CCA (RCCA) [CKN03,G04,PR07]:

- Only allowed f is identity function
- Non-malleable in any ways that alter message

Example: Only allowed f's are group operations $\alpha \rightsquigarrow \beta\alpha$:

- Possible to change any message to any other message
- Infeasible to change $\mathsf{Enc}(\alpha)$ into $\mathsf{Enc}(\alpha^k)$
- Infeasible to change $\mathsf{Enc}(\alpha), \mathsf{Enc}(\beta)$ into $\mathsf{Enc}(\alpha\beta)$

Outline

Security Definitions – Approach

We define security with two complementary definitions:

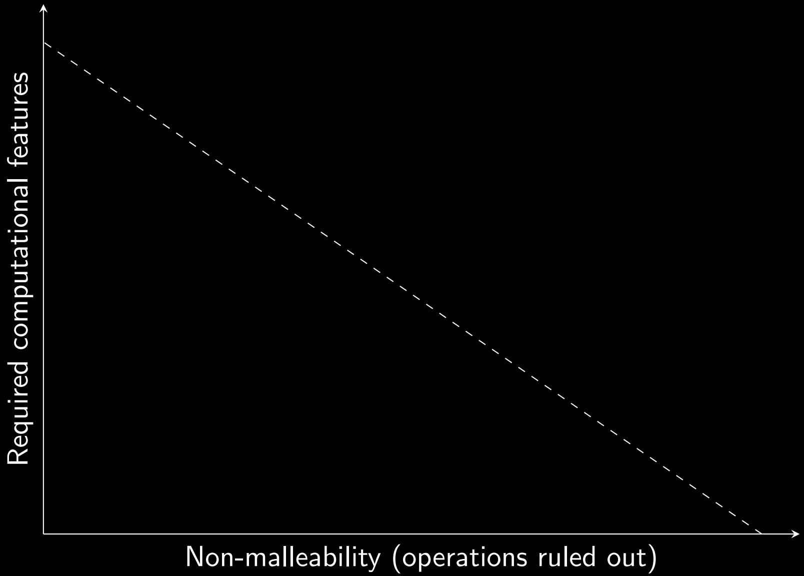
Homomorphic-CCA (HCCA) security

Scheme is non-malleable, except possibly via unary operations $f \in \mathcal{F}$

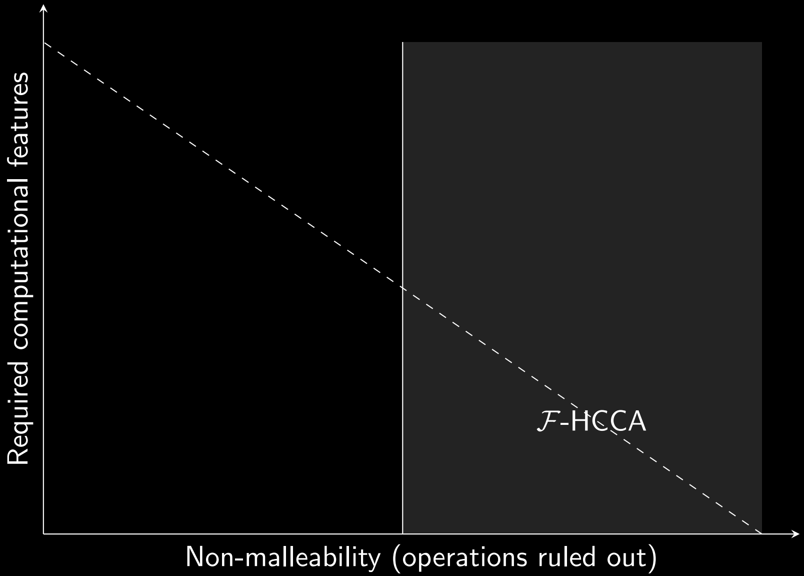
Unlinkability

One can transform $\mathrm{Enc}(m)$ to "fresh" $\mathrm{Enc}(f(m))$ for any $f \in \mathcal{F}$, as a feature of the scheme.

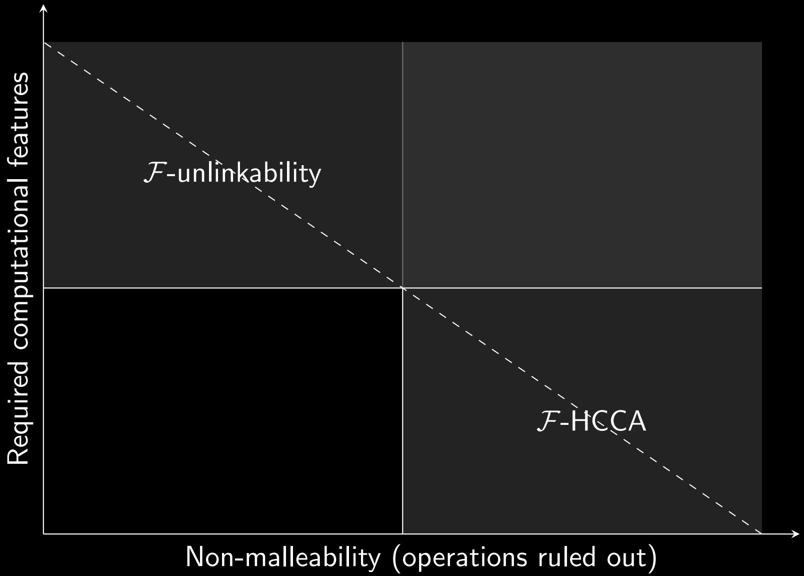
Security Definitions – Approach



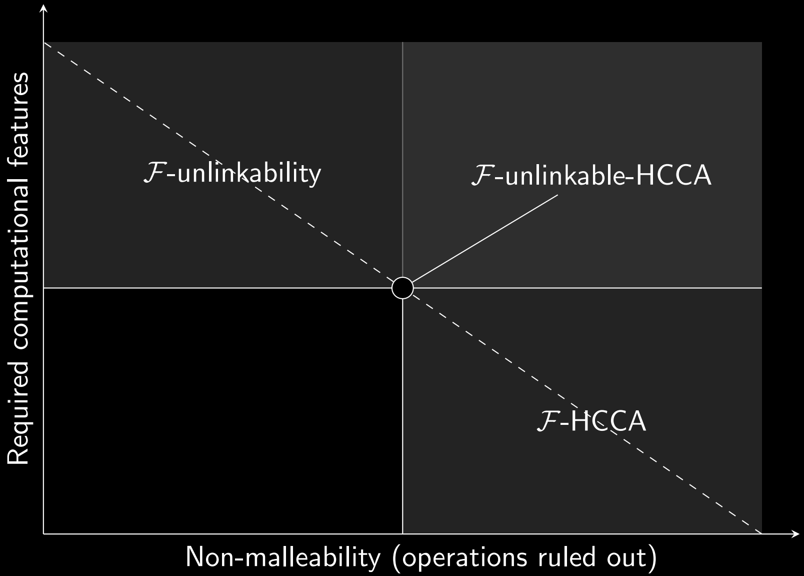
Security Definitions – Approach



Security Definitions – Approach



Security Definitions – Approach



Generalizing CCA to HCCA

Start by modifying CCA experiment:

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0, m_1.
4. Give $C \leftarrow \text{Enc}(m_0)$.
5. Provide Dec oracle, except:
   - Refuse if given C.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0, m_1.
4. Give $C \leftarrow \text{Enc}(m_1)$.
5. Provide Dec oracle, except:
   - Refuse if given C.

Generalizing CCA to HCCA

Start by modifying CCA experiment:

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_0)$.
5. Provide Dec oracle, except:
   - Refuse if given C.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_1)$.
   (m_1 public, fixed)
5. Provide Dec oracle, except:
   - Refuse if given C.

Generalizing CCA to HCCA

Start by modifying CCA experiment:

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_0)$.
5. Provide Dec oracle, except:
   - Respond "m_0" if given C.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_1)$.
   (m_1 public, fixed)
5. Provide Dec oracle, except:
   - Respond "m_0" if given C.

Generalizing CCA to HCCA

Start by modifying CCA experiment:

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_1)$.
   (m_1 public, fixed)
5. Provide Dec oracle, except:
   - Respond "m_0" if given C.

Generalizing CCA to HCCA

Start by modifying CCA experiment:

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_0)$.
5. Provide Dec oracle

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_1)$.
   (m_1 public, fixed)
5. Provide Dec oracle, except:
   - Respond "m_0" if given C.

Idea for Generalization

Dec oracle should compensate for derivatives of C.

Derivative Ciphertexts

Derivatives of C

Ciphertexts that could have been *legitimately* derived from C (i.e., via scheme's allowed features).

Different security levels for different derivative condition:

CCA: C' is derivative iff $C' = C$

Derivative Ciphertexts

Derivatives of C

Ciphertexts that could have been *legitimately* derived from C (i.e., via scheme's allowed features).

Different security levels for different derivative condition:

CCA: C' is derivative iff $C' = C$

gCCA: C' is derivative iff $R(C', C) = 1$ [S01,ADR02]

RCCA: C' is derivative iff $\mathrm{Dec}(C') = \mathrm{Dec}(C)$ [CKN03]

Can We Always Identify Derivative Ciphertexts?

For certain $\mathcal{F}$, these distributions could be identical:

- Enc(β) obtained by encrypting known β
- Enc(β) derived by legitimately multiplying Enc(α) by β/α

Problem:

- Strong homomorphic operation *demands* identical distributions
- Impossible to identify derived ciphertexts

Can We Always Identify Derivative Ciphertexts?

For certain $\mathcal{F}$, these distributions could be identical:

- Enc(β) obtained by encrypting known β
- Enc(β) derived by legitimately multiplying Enc(α) by β/α

Problem:

- Strong homomorphic operation *demands* identical distributions
- Impossible to identify derived ciphertexts

What we want:

- Ciphertexts derived from C have different distribution than independently encrypted ciphertexts

Rigged Ciphertexts

Key idea: C need not be actual encryption of some m_1:

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{Enc}(m_1)$.
5. Provide Dec oracle, except:
   - "compensate for derivatives of C"

Rigged Ciphertexts

Key idea: C need not be actual encryption of some m_1:

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{RigEnc}(\ )$.
5. Provide Dec oracle, except:
   - "compensate for derivatives of C"

Rigged Ciphertexts

Key idea: C need not be actual encryption of some m_1:

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{RigEnc}(\ )$.
5. Provide Dec oracle, except:
   - If $f \leftarrow \text{RigExtract}(C')$, then answer $f(m_0)$.

"Rigged" Ciphertexts

Challenge "ciphertext" can have embedded tracking information.
Extraction procedure determines how C' derived from C.

Interpreting Security Guarantee

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{RigEnc}(\ )$.
5. Provide Dec oracle, except:
   - If $f \leftarrow \mathrm{RigExtract}(C')$, then answer $f(m_0)$.

Interpreting Security Guarantee

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{RigEnc}(\ )$.
5. Provide Dec oracle, except:
   - If $f \leftarrow \mathrm{RigExtract}(C')$, then answer $f(m_0)$.

Suppose scheme is "malleable": can generate C' whose value depends on (possibly many) other ciphertexts.

Interpreting Security Guarantee

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \text{RigEnc}(\ )$.
5. Provide Dec oracle, except:
   - If $f \leftarrow \text{RigExtract}(C')$, then answer $f(m_0)$.

Suppose scheme is "malleable": can generate C' whose value depends on (possibly many) other ciphertexts.

Submit C' to get $\text{Dec}(C')$

Response is $f(m_0)$ for some f.

Interpreting Security Guarantee

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathrm{RigEnc}(\ )$.
5. Provide Dec oracle, except:
   - If $f \leftarrow \mathrm{RigExtract}(C')$, then answer $f(m_0)$.

Suppose scheme is "malleable": can generate C' whose value depends on (possibly many) other ciphertexts.

Submit C' to get $\mathrm{Dec}(C')$

Response is $f(m_0)$ for some f.

$\implies$ this malleability "looks like" $m \rightsquigarrow f(m)$

A Limit on Malleability

Suppose RigExtract never outputs f':

- Scheme must not be malleable via f' operation.
- In particular, operations that nontrivially *combine multiple* ciphertexts.

A Limit on Malleability

Suppose RigExtract never outputs f':

- Scheme must not be malleable via f' operation.
- In particular, operations that nontrivially *combine multiple* ciphertexts.

⋆ Homomorphic-CCA (HCCA) Security

Scheme is non-malleable except for unary operations $f \in \mathcal{F}$ if there is a good $(\mathsf{RigEnc}, \mathsf{RigExtract})$, where $\mathsf{range}(\mathsf{RigExtract}) \subseteq \mathcal{F}$.

Disclaimer:

- Oracles for RigEnc and RigExtract should be provided, too.

Outline

Relationships with other definitions

Theorem

CCA, gCCA, RCCA are all special cases of HCCA

In each of these cases:

- The only allowed transformation is identity function
- RigEnc simply uses Enc honestly

HCCA more expressive when its full power is used.

Natural UC Security Definition

Theorem

HCCA and unlinkability imply UC-secure protocol for "natural" ideal functionality

Analogous to [C01,CKN03], use UC model to define encryption security.

Natural UC Security Definition

Theorem

HCCA and unlinkability imply UC-secure protocol for "natural" ideal functionality

Analogous to [C01,CKN03], use UC model to define encryption security.

In our UC functionality:, parties post messages, represented as "formal ciphertexts"

Message privacy:	Formal ciphertexts reveal nothing; only recipient can obtain underlying message
Homomorphic feature:	Anyone can generate a "derived post" by giving f and existing ciphertext
Unlinkability:	Same internal behavior for both kinds of posts
Non-malleability:	No one can use unauthorized f

Encapsulation Theorem

Theorem

Any unlinkable-HCCA + (plain) CCA = rerandomizable RCCA

- RCCA demands: identity function is only legal operation
- HCCA scheme could have *any* set of allowed operations.

Encapsulation Theorem

Theorem

Any unlinkable-HCCA + (plain) CCA = rerandomizable RCCA

- RCCA demands: identity function is only legal operation
- HCCA scheme could have *any* set of allowed operations.

Proof.

Encapsulate CCA scheme inside *any* unlinkable HCCA scheme

- New scheme inherits outer unlinkability
- Inner CCA scheme "cancels" everything except identity function

$\square$

Outline

Construction

Parameterized family of constructions achieving our definitions:

- Message space: $\mathcal{G}^n$, where $\mathcal{G}$ is cyclic group.
- $\mathcal{H}$ is any subgroup of $\mathcal{G}^n$.
- Allowed transformations: $m \mapsto f * m$, for all $f \in \mathcal{H}$.

Construction

Parameterized family of constructions achieving our definitions:

- Message space: $\mathcal{G}^n$, where $\mathcal{G}$ is cyclic group.
- $\mathcal{H}$ is any subgroup of $\mathcal{G}^n$.
- Allowed transformations: $m \mapsto f * m$, for all $f \in \mathcal{H}$.

Example instantiations:

- Allow all group operations in $\mathcal{G}^n$
- Allow only "scalar multiplication" of vectors:

$$(m_1, \ldots, m_n) \mapsto (f \cdot m_1, \ldots, f \cdot m_n)$$

- Allow group operations only on particular components – other components non-malleable
- Allow only identity function (Rerandomizable RCCA)

Construction

Our construction significantly generalizes rerandomizable RCCA scheme of [PR07].

- Obtain [PR07] scheme as special case
- Uses techniques from [G+04,CS01].

Theorem

Our construction is unlinkable & HCCA-secure under DDH assumption in 2 groups of related size.

Outline

Summary

The problem: Encryption schemes with both:

- Robust computational features, and
- Non-malleability guarantee
- “Non-malleable except for explicitly allowed features”

Summary

The problem: Encryption schemes with both:

- Robust computational features, and
- Non-malleability guarantee
- “Non-malleable except for explicitly allowed features”

Our contributions:

- New definitions for case of unary homomorphic encryption
- Justify definitions by relating to existing ones
- Family of constructions that achieve definitions

Open problems

Extend to binary operations: $\mathsf{Enc}(\alpha), \mathsf{Enc}(\beta) \rightsquigarrow \mathsf{Enc}\big(f(\alpha, \beta)\big)$

- We show that natural generalization is impossible!
- Some slight relaxation possible (work in progress)
- Even new *security definitions* would be non-trivial.

Open problems

Extend to binary operations: $\mathsf{Enc}(\alpha), \mathsf{Enc}(\beta) \rightsquigarrow \mathsf{Enc}\big(f(\alpha, \beta)\big)$

- We show that natural generalization is impossible!
- Some slight relaxation possible (work in progress)
- Even new *security definitions* would be non-trivial.

"Key-activated" homomorphic encryption:

- Scheme is CCA secure . . .
- . . . unless you have a token that "activates" only selected homomorphic features.

takk fyrir.*

$\star$: *Thank you* (Icelandic)