Reconciling Non-malleability & Homomorphic Encryption

Manoj Prabhakaran (U Illinois) · Mike Rosulek (U Montana)

University of Maryland · January 16, 2012

Discussing papers:


computing on encrypted data

\[ \text{Enc} (\text{input}) \]

\[ \text{Enc} (\text{result}) \]

Cloud should be able to manipulate encrypted data.

- **Non-malleability!**
  - "Can't generate certain related ciphertexts"

Mike Rosulek (U Montana)  Non-Malleability, Homomorphic Encryption  January 16, 2012 @ U Maryland
computing on encrypted data

\[ \text{Enc}(\text{input}) \]

\[ \text{Enc}(\text{result}) \]

Cloud should be able to manipulate encrypted data.

▶ Homomorphic features (malleability)

Cloud shouldn't be able to perform incorrect computation.

▶ Non-malleability!

▶ “Can't generate certain related ciphertexts”
Cloud should be able to manipulate encrypted data.

- Homomorphic features (**malleability**)
computing on encrypted data

Cloud should be able to manipulate encrypted data.

- Homomorphic features (malleability)

Cloud shouldn’t be able to perform incorrect computation.

- Non-malleability!
- “Can’t generate certain related ciphertexts”
malleability: all-or-nothing

**Homomorphic encryption?**

- Can’t say anything about non-malleability
- Resort to ZK proofs for integrity
malleability: all-or-nothing

**Homomorphic encryption?**
- Can’t say anything about non-malleability
- Resort to ZK proofs for integrity

**Non-malleable encryption?**
- Doesn’t allow any useful features
- Need additional tools (MPC?) to manipulate encrypted data
malleability: all-or-nothing

### Homomorphic encryption?
- Can’t say anything about non-malleability
- Resort to ZK proofs for integrity

### Non-malleable encryption?
- Doesn’t allow any useful features
- Need additional tools (MPC?) to manipulate encrypted data

Can we address both concerns in a single encryption scheme?
Desired security (informal)

“Scheme is non-malleable, except for explicitly allowed features.”
Desired security (informal)

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

.. or ..

“Only way to make a ciphertext depend on others is via explicitly approved features.”
the problem

Desired security (informal)

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

.. or..

“Only way to make a ciphertext depend on others is via explicitly approved features.”

Challenges:

- Convincingly formalize intuitive definition
- Achieve definition with new constructions
the problem

Desired security (informal)

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

.. or..

“Only way to make a ciphertext depend on others is via explicitly approved features.”

Challenges:

- Convincingly formalize intuitive definition
- Achieve definition with new constructions

This talk: “allowed features” = homomorphic encryption

1. New general-purpose non-malleability definition
2. New family of constructions
3. Some impossibility results
(unary) homomorphic encryption

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

- 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

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

- Scheme parameterized by set of allowed $f$’s

**Example:**

Rerandomizable Replayable-CCA (RCCA) \[\text{[CKN03, G04, PR07]}\]:

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

**Example:**

Only allowed $f$’s are group operations $\alpha \mapsto \beta\alpha$ (known $\beta$):

- Possible to change any message to any other message
- Infeasible to change $\text{Enc}(\alpha)$ into $\text{Enc}(\alpha^k)$
- Infeasible to change $\text{Enc}(\alpha), \text{Enc}(\beta)$ into $\text{Enc}(\alpha\beta)$
contrast with FHE:

**Fully homomorphic encryption [G09]:**

- **Focus:** maximum expressivity
- No consideration of non-malleability
- **Binary operations:** $\text{Enc}(m_1), \text{Enc}(m_2) \rightarrow \text{Enc}(m_1 + m_2)$
contrast with FHE:

**Fully homomorphic encryption [G09]:**
- Focus: maximum expressivity
- No consideration of non-malleability
- Binary operations: $\text{Enc}(m_1), \text{Enc}(m_2) \leadsto \text{Enc}(m_1 + m_2)$

**This talk:**
Focus: sharp tradeoff between malleability / non-malleability:
- allowed $f$: available as highly expressive full feature
- everything else: computationally infeasible

Challenging even when expressivity is low:
- e.g.: $\mathcal{F}$ contains only one operation
teaching evaluations [PR08b]

- Alice
- : 
- students
- TA

Privacy: TA can't see responses
Functionality: TA must be able to anonymize (shuffle)
Integrity: TA can't modify/replace responses
Verifiable shuffle [G02, GL07a, GL07b]
teaching evaluations \[PR08b]\]

Alice

\[
\begin{array}{c}
\text{students} \\
\vdots \\
\end{array}
\]

Mike Rosulek (U Montana) Non-Malleability, Homomorphic Encryption January 16, 2012 @ U Maryland
Alice

students

TA

Privacy: TA can’t see responses

Functionality: TA must be able to anonymize (shuffle)

Integrity: TA can’t modify/replace responses

Verifiable shuffle [PR08b]
teaching evaluations [PR08b]

Privacy: TA can't see responses
Functionality: TA must be able to anonymize (shuffle)
Integrity: TA can't modify/replace responses

Verifiable shuffle [G02, GL07a, GL07b]

Mike Rosulek (U Montana)    Non-Malleability, Homomorphic Encryption    January 16, 2012 @ U Maryland
teaching evaluations [PR08b]
teaching evaluations [PR08b]

Privacy: TA can’t see responses
Functionality: TA must be able to anonymize (shuffle)
Integrity: TA can’t modify/replace responses
teaching evaluations [PR08b]

Privacy: TA can’t see responses
Functionality: TA must be able to anonymize (shuffle)
Integrity: TA can’t modify/replace responses

Verifiable shuffle [G02, GL07a, GL07b]
Use encryption that is *non-malleable except for* $\text{Enc}(m, r) \not\sim \text{Enc}(m, rs)$.
simple protocol

Use encryption that is *non-malleable except for* \(\text{Enc}(m, r) \sim \text{Enc}(m, rs)\).

---

Security proof:

▶ TA must give \(\{\text{Enc}(m'_i, r'_i)\}\), where \(\prod r'_i = \prod r_i\)

▶ Each \(r'_i\) is multiple of a single \(r_j\), or independent of all \(r_i\)’s

▶ Must depend on each \(r_i\) once, else \(\prod r'_i\) independent of \(\prod r_i\)

▶ Can't get dependence on an \(r_i\) without its \(m_i\) intact
Use encryption that is **non-malleable except for** $\text{Enc}(m, r) \rightsquigarrow \text{Enc}(m, rs)$.
Use encryption that is *non-malleable except for* $\text{Enc}(m, r) \rightsquigarrow \text{Enc}(m, rs)$.

$$\prod s_i = 1$$

Mike Rosulek (U Montana)  Non-Malleability, Homomorphic Encryption  January 16, 2012 @ U Maryland  7 / 31
Use encryption that is *non-malleable except for* $\text{Enc}(m, r) \leadsto \text{Enc}(m, rs)$.

$\text{Alice}$

$\text{students}$

$\prod s_i = 1$

$\text{TA}$

$\text{Enc}(m_1, r_1)$

$\text{Enc}(m_2, r_2)$

$\text{Enc}(m_n, r_n)$

$\text{shuf:}\{\text{Enc}(m_i, r_is_i)\}$
Use encryption that is *non-malleable except for* \( \text{Enc}(m, r) \leadsto \text{Enc}(m, rs) \).

\[
\Pi\text{-?} \quad r_1 \quad \text{Enc}(m_1, r_1) \quad \Pi s_i = 1 \\
\text{Alice} \quad r_2 \quad \text{Enc}(m_2, r_2) \\
\quad \vdots \\
\quad r_n \quad \text{Enc}(m_n, r_n) \\
\text{students} \quad \text{shuf:}\{ \text{Enc}(m_i, r_is_i) \}\} \\
\quad \text{TA}
\]
Use encryption that is non-malleable except for $\text{Enc}(m, r) \sim \text{Enc}(m, rs)$.

Security proof:
- TA must give $\{\text{Enc}(m'_i, r'_i)\}$, where $\prod r'_i = \prod r_i$
Use encryption that is *non-malleable except for* \( \text{Enc}(m, r) \leadsto \text{Enc}(m, rs) \).

Security proof:
- TA must give \( \{\text{Enc}(m'_i, r'_i)\} \), where \( \prod r'_i = \prod r_i \)
- Each \( r'_i \) is multiple of a single \( r_j \), or independent of all \( r_i \)'s
- Must depend on each \( r_i \) once, else \( \prod r'_i \) independent of \( \prod r_i \)
Use encryption that is *non-malleable except for* \( \text{Enc}(m, r) \leadsto \text{Enc}(m, rs) \).

Security proof:
- TA must give \( \{\text{Enc}(m'_i, r'_i)\} \), where \( \prod r'_i = \prod r_i \)
- Each \( r'_i \) is multiple of a single \( r_j \), or independent of all \( r_i \)’s
- Must depend on each \( r_i \) once, else \( \prod r'_i \) independent of \( \prod r_i \)
- Can’t get dependence on an \( r_i \) without its \( m_i \) intact
approach to definitions

\[\mathcal{F}\text{-unlinkability}\]

Given unknown \(\text{Enc}(m)\), anyone can generate fresh \(\text{Enc}(f(m))\), for any \(f \in \mathcal{F}\).
approach to definitions

\( \mathcal{F} \)-unlinkability

Given unknown \( \text{Enc}(m) \), anyone can generate fresh \( \text{Enc}(f(m)) \), for any \( f \in \mathcal{F} \).

\( \mathcal{F} \)-Homomorphic-CCA (HCCA)

Given unknown \( \text{Enc}(m) \), cannot generate \( C \) such that \( \text{Dec}(C) \) depends on \( m \)...
**\(\mathcal{F}\)-unlinkability**

Given unknown \(\text{Enc}(m)\), anyone can generate fresh \(\text{Enc}(f(m))\), for any \(f \in \mathcal{F}\).

**\(\mathcal{F}\)-Homomorphic-CCA (HCCA)**

Given unknown \(\text{Enc}(m)\), cannot generate \(C\) such that \(\text{Dec}(C)\) depends on \(m\)...

\[\text{... unless the dependence is } \text{Dec}(C) = f(m) \text{ for some } f \in \mathcal{F}\]
approach to definitions

\( \mathcal{F} \)-unlinkability

Given unknown \( \text{Enc}(m) \), anyone can generate fresh \( \text{Enc}(f(m)) \), for any \( f \in \mathcal{F} \).

\( \mathcal{F} \)-Homomorphic-CCA (HCCA)

Given unknown \( \text{Enc}(m) \), cannot generate \( C \) such that \( \text{Dec}(C) \) depends on \( m \)...  

- ... unless the dependence is \( \text{Dec}(C) = f(m) \) for some \( f \in \mathcal{F} \)

Goal: Simultaneously achieve \( \mathcal{F} \)-unlinkability and \( \mathcal{F} \)-HCCA.
generalizing IND-CCA game

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 $C$. 

What if scheme has homomorphic features?

RHS Dec oracle must compensate for derivatives of $C$.

- Anything that could be legitimately derived from $C$. 

Mike Rosulek (U Montana)  Non-Malleability, Homomorphic Encryption  January 16, 2012 @ U Maryland
generalizing IND-CCA game

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 $C$.

What if scheme has homomorphic features?
RHS Dec oracle must compensate for derivatives of $C$.
▶ Anything that could be legitimately derived from $C$.
generalizing IND-CCA game

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 $C$. 

What if scheme has homomorphic features?
RHS Dec oracle must compensate for derivatives of $C$.

▶ Anything that could be legitimately derived from $C$.
generalizing IND-CCA game

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

1. Generate keypair, give $PK$.
2. Provide Dec oracle.
3. Adversary chooses $m_0$.
4. Give $C \leftarrow Enc(m_1)$.
   ($m_1$ public, fixed)
5. Provide Dec oracle, except:
   - Respond “$m_0$” if given $C$.
generalizing IND-CCA game

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 $C$.

What if scheme has homomorphic features?

RHS Dec oracle must compensate for derivatives of $C$.
▶ Anything that could be legitimately derived from $C$
generalizing IND-CCA game

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:
   
   ▶ Compensate if given a derivative of $C$

What if scheme has homomorphic features?

RHS Dec oracle must compensate for derivatives of $C$.

▶ Anything that could be legitimately derived from $C$
Derivatives of $C$

Ciphertexts that could have been \textit{legitimately} derived from $C$ (i.e., via scheme’s allowed features).

Examples:

\begin{itemize}
\item \textbf{CCA}: $C'$ is derivative $\iff C' = C$
\end{itemize}

Intuition: Dec oracle only allowed to compensate on derivatives!
Derivatives of $C$

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

Examples:

- **CCA:** $C'$ is derivative $\iff C' = C$
- **gCCA:** $C'$ is derivative $\iff R(C', C) = 1$ \[S01,ADR02\]
- **RCCA:** $C'$ is derivative $\iff \text{Dec}(C') = \text{Dec}(C)$ \[CKN03\]

Intuition: Dec oracle only allowed to compensate on derivatives!
Example: $\mathcal{F} = \{ \text{group operations, } \alpha \mapsto \beta \alpha \}$

These distributions are identical:

- $\text{Enc}(\beta)$ derived by legitimately multiplying $\text{Enc}(\alpha)$ by $\beta/\alpha$
- $\text{Enc}(\beta)$ obtained independently by encrypting $\beta$
why this won’t work

Example: \( \mathcal{F} = \{ \text{group operations, } \alpha \mapsto \beta \alpha \} \)

These distributions are identical:
- \( \text{Enc}(\beta) \) derived by legitimately multiplying \( \text{Enc}(\alpha) \) by \( \beta/\alpha \)
- \( \text{Enc}(\beta) \) obtained independently by encrypting \( \beta \)

Problem:
- \( \text{Dec} \) oracle cannot identify derived ciphertexts
- Alternately, every ciphertext flagged as (possibly) derived from \( C \)
  - Game collapses to IND-CCA1.
Key idea: \( C \) need not be actual encryption of some \( m_1 \):

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

1. Generate keypair, give \( PK \).
2. Provide Dec oracle.
3. Adversary chooses \( m^* \).
4. Give \( C \leftarrow \text{Enc}(m_1) \).
5. Provide Dec oracle, except:
   - “compensate for derivatives of \( C \)”
Key idea: C need not be actual encryption of some $m_1$:

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

---

1. Generate keypair, give $PK$.
2. Provide Dec oracle.
3. Adversary chooses $m^*$.
4. Give $C \leftarrow \text{RigEnc}(\cdot)$.
5. Provide Dec oracle, except:
   - “compensate for derivatives of $C$”
rigging the game

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

1. Generate keypair, give $PK$.
2. Provide Dec oracle.
3. Adversary chooses $m^*$.  
4. Give $C ← Enc(m^*)$.
5. Provide Dec oracle.

1. Generate keypair, give $PK$.
2. Provide Dec oracle.
3. Adversary chooses $m^*$.
4. Give $C ← RigEnc( )$.
5. Provide Dec oracle, except:
   ▶ If $f ← RigExtract(C')$, then answer $f(m^*)$.

“Rigged” Ciphertexts

Challenge “ciphertext” can have embedded tracking information.
Extraction procedure determines how $C'$ derived from $C$. 
interpreting the security game

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

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

Suppose adversary can generate $C'$ whose value depends on $C$.
Submit $C'$; response is $\text{Dec}(C')$.
Response is $f(m^*)$ for some $f \in \text{range}(\text{RigExtract})$. 

Mike Rosulek (U Montana)  Non-Malleability, Homomorphic Encryption  January 16, 2012 @ U Maryland
interpreting the security game

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

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

Suppose adversary can generate $C'$ whose value depends on $C$
interpreting the security game

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

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

Suppose adversary can generate $C'$ whose value depends on $C$

Submit $C'$; response is $\text{Dec}(C')$ 
Response is $f(m^*)$ for some $f$. 
interpreting the security game

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

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

Suppose adversary can generate $C'$ whose value depends on $C$

Submit $C'$; response is $\text{Dec}(C')$  Response is $f(m^*)$ for some $f$.

$\Rightarrow$ Dependence is $\text{Dec}(C') = f(\text{Dec}(C))$, for some $f \in \text{range}(\text{RigExtract})$. 
Suppose RigExtract never outputs $f^*$:

- Scheme cannot be malleable via $f^*$ operation.
Suppose $\text{RigExtract}$ never outputs $f^*$:

- Scheme cannot be malleable via $f^*$ operation.

**Homomorphic-CCA (HCCA) Security [PR08]**

Scheme is **non-malleable except for $\mathcal{F}$** if $\exists$ $\text{RigEnc}, \text{RigExtract}$:

1. two worlds indistinguishable in the HCCA game
2. $\text{range(RigExtract)} \subseteq \mathcal{F}$

**Fine print:**

- Oracles for $\text{RigEnc}$ and $\text{RigExtract}$ should be provided, too.
- $\text{RigEnc}, \text{RigExtract}$ needed only for security analysis.
interpreting HCCA

**Theorem [PR08]**

CCA, gCCA, RCCA are all special cases of HCCA

In each case:

- The only allowed transformation is identity function
  - “non-malleable in ways that change the plaintext”
- RigEnc simply uses Enc honestly
interpreting HCCA

**Theorem [PR08]**

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

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

**Message privacy:** Message handles reveal nothing; only recipient can obtain underlying message

**Homomorphism:** Anyone can generate a “derived message” by giving \( f \) and existing handle

**Unlinkability:** Same internal behavior for both kinds of posts

**Non-malleability:** No one can use unauthorized \( f \)
our construction \([\text{PR07,PR08}]\)
Simplest example: $\mathcal{F} = \{\text{identity function}\}$:

$\mathcal{F}$-unlinkability $= \text{rerandomizability}$

For any $C \leftarrow \text{Enc}_{PK}(m)$, two distributions identical:

$$\{\text{Rerand}(C)\} \equiv \{\text{Enc}_{PK}(m)\}$$
Simplest example: $\mathcal{F} = \{\text{identity function}\}$:

$\mathcal{F}$-unlinkability $\Rightarrow$ rerandomizability

For any $C \leftarrow \text{Enc}_{PK}(m)$, two distributions identical:

$$\{\text{Rerand}(C)\} \equiv \{\text{Enc}_{PK}(m)\}$$

$\mathcal{F}$-HCCA $\Rightarrow$ Replayable-CCA (RCCA) [CKN03]

Scheme is non-malleable in any ways that alter the plaintext
Simplest example: $F = \{\text{identity function}\}$:

$F$-unlinkability $= \text{rerandomizability}$

For any $C \leftarrow \text{Enc}_{PK}(m)$, two distributions identical:

$$\{\text{Rerand}(C)\} \equiv \{\text{Enc}_{PK}(m)\}$$

$F$-HCCA $= \text{Replayable-CCA (RCCA)}$ [CKN03]

Scheme is non-malleable in any ways that alter the plaintext

Question [CKN03]

Are there rerandomizable, RCCA-secure encryption schemes?
our construction [PR07]

Key Idea #1

Start with Cramer-Shoup; add second “strand” to allow rerandomization.
Key Idea #1

Start with Cramer-Shoup; add second “strand” to allow rerandomization.

Warm-up, “double-strand” ElGamal [GJJS04]:

- Normal ElGamal encryption of $m$ (public key is $A$):
  
  $$g^x, mA^x$$

  for random $x$
Key Idea #1

Start with Cramer-Shoup; add second “strand” to allow rerandomization.

Warm-up, “double-strand” ElGamal [GJJS04]:

- Normal ElGamal encryption of $m$ (public key is $A$):
  $$g^x, mA^x$$ for random $x$

- “Double-Strand” ElGamal encryption of $m$:
  $$g^x, mA^x, g^y, A^y$$ for random $x, y$$
Key Idea #1

Start with Cramer-Shoup; add second “strand” to allow rerandomization.

Warm-up, “double-strand” ElGamal [GJJS04]:

- Normal ElGamal encryption of $m$ (public key is $A$):
  \[ g^x, mA^x \text{ for random } x \]
- “Double-Strand” ElGamal encryption of $m$:
  \[ g^x, mA^x, g^y, A^y \text{ for random } x, y \]
- Rerandomize $y \mapsto yt$ (multiplicative)
  \[ (g^y)^t, (A^y)^t \text{ for rand } t \]
  \[ g^{yt}, A^{yt} \]
Key Idea #1

Start with Cramer-Shoup; add second “strand” to allow rerandomization.

Warm-up, “double-strand” ElGamal [GJJS04]:

- Normal ElGamal encryption of $m$ (public key is $A$):
  \[ g^x, mA^x \text{ for random } x \]

- “Double-Strand” ElGamal encryption of $m$:
  \[ g^x, mA^x, g^y, Ay^y \text{ for random } x, y \]

- Rerandomize $x \mapsto x + ys$ (additive), $y \mapsto yt$ (multiplicative)
  \[ g^x(g^y)^s, \quad mA^x(A^y)^s, \quad (g^y)^t, \quad (A^y)^t \text{ for rand } t, s \]
  \[ = \quad g^{x+ys}, \quad mA^{x+ys}, \quad g^{yt}, \quad A^{yt} \]
Normal Cramer-Shoup: ($\mu$ is hash of first 3 components)

$$g_1^x, g_2^x, m B^x, (CD^\mu)^x \quad \text{for random } x$$
Normal Cramer-Shoup: ($\mu$ is hash of first 3 components)

$$g_1^x, g_2^x, m B^x, (CD^\mu)^x \quad \text{for random } x$$

Double-strand idea applied:

$$g_1^x, g_2^x, m B^x, (CD^\mu)^x, g_1^y, g_2^y, B^y, (CD^\mu)^y \quad \text{for rand } x, y$$
double-strand Cramer-Shoup

Normal Cramer-Shoup: ($\mu$ is hash of first 3 components)

\[ g_1^x, g_2^x, mB^x, (CD\mu)^x \text{ for random } x \]

Double-strand idea applied: ($\mu$ is encoding of message)

\[ g_1^x, g_2^x, mB^x, (CD\mu)^x, g_1^y, g_2^y, B^y, (CD\mu)^y \text{ for rand } x, y \]
double-strand Cramer-Shoup

Normal Cramer-Shoup: \((\mu)\) is hash of first 3 components)

\[ g_1^x, g_2^x, m B^x, (CD^\mu)^x \quad \text{for random } x \]

Double-strand idea applied: \((\mu)\) is encoding of message)

\[ g_1^x, g_2^x, m B^x, (CD^\mu)^x, g_1^y, g_2^y, B^y, (CD^\mu)^y \quad \text{for rand } x, y \]

Possible Attack!

“Mix-and-match” strands from 2 independent ciphertexts

- Result is valid \(\iff\) same plaintext \((\mu)\)
Key Idea #2

“Tie strands together” with shared randomness $u$; give an additional encryption of $u$

$$(g_1^x)^u, (g_2^x)^u, mB^x, (CD^\mu)^x, (g_1^y)^u, (g_2^y)^u, B^y, (CD^\mu)^y, \text{Enc}(u)$$
Key Idea #2

“Tie strands together” with shared randomness $u$; give an additional encryption of $u$

$$(g_1^x)^u, (g_2^x)^u, mB^x, (CD^\mu)^x, (g_1^y)^u, (g_2^y)^u, B^y, (CD^\mu)^y, \text{Enc}(u)$$

Thwarts “mix-and-match” attack:

- Strands combine nicely, only if they share common $u$
Key Idea #2

“Tie strands together” with shared randomness $u$; give an additional encryption of $u$

\[(g_1^x)^u, (g_2^x)^u, mB^x, (CD^\mu)^x, (g_1^y)^u, (g_2^y)^u, B^y, (CD^\mu)^y, \text{Enc}(u)\]

Thwarts “mix-and-match” attack:

- Strands combine nicely, only if they share common $u$

What’s that $\text{Enc}(u)$ doing?

- Receiver must remove $u$ to decrypt
- $\text{Enc}(u)$ malleable, to allow rerandomization of $u$ ($u \mapsto ru$)
- $\text{Enc}(u)$ itself rerandomizable
Key Idea #2

“Tie strands together” with shared randomness $u$; give an additional encryption of $u$

$$(g_1^x)^u, (g_2^x)^u, mB^x, (CD^\mu)^x, (g_1^y)^u, (g_2^y)^u, B^y, (CD^\mu)^y, \text{Enc}(u)$$

Thwarts “mix-and-match” attack:

- Strands combine nicely, only if they share common $u$

What’s that $\text{Enc}(u)$ doing?

- Receiver must remove $u$ to decrypt
- $\text{Enc}(u)$ malleable, to allow rerandomization of $u$ ($u \mapsto ru$)
- $\text{Enc}(u)$ itself rerandomizable

“Cramer-Shoup lite” has these properties [fine print].
“twist” the first strand

\[ g_1^{xu}, g_2^{xu}, mB^x, (CD^\mu)^x, g_1^{yu}, g_2^{yu}, B^y, (CD^\mu)^y, \text{Enc}(u) \]

Possible Attack

Can test candidate plaintext by “rerandomizing” first strand multiplicatively:

\[ g_1^{xu}, g_2^{xu}, mB^x, (CD^\mu)^x \xrightarrow{\sim} (g_1^{xu})^2, (g_2^{xu})^2, \frac{(mB^x)^2}{m'}, ((CD^\mu)^x)^2 \]
“twist” the first strand

\[ g_1^{xu}, g_2^{xu}, mB^x, (CD^\mu)^x, g_1^{yu}, g_2^{yu}, B^y, (CD^\mu)^y, \text{Enc}(u) \]

**Possible Attack**

Can test candidate plaintext by “rerandomizing” first strand multiplicatively:

\[ g_1^{xu}, g_2^{xu}, mB^x, (CD^\mu)^x \rightsquigarrow (g_1^{xu})^2, (g_2^{xu})^2, \frac{(mB^x)^2}{m'}, (CD^\mu)^x \]

\[ = g_1^{(2x)u}, g_2^{(2x)u}, \frac{m^2}{m'}B^{2x}, (CD^\mu)^{2x} \]
“twist” the first strand

\[ g_1^{xu}, g_2^{xu}, mB^x, (CD^\mu)^x, g_1^{yu}, g_2^{yu}, B^y, (CD^\mu)^y, \text{Enc}(u) \]

### Possible Attack

Can test candidate plaintext by “rerandomizing” first strand multiplicatively:

\[
\begin{align*}
  g_1^{xu}, g_2^{xu}, mB^x, (CD^\mu)^x & \leadsto (g_1^{xu})^2, (g_2^{xu})^2, \frac{(mB^x)^2}{m'}, (CD^\mu)^x)^2 \\
  &= g_1^{(2x)u}, g_2^{(2x)u}, \frac{m^2}{m'} B^{2x}, (CD^\mu)^{2x}
\end{align*}
\]

Decryption succeeds iff \( m = m' \).
“twist” the first strand (cont.)

**Key Idea #3**

Make first strand fundamentally different, so it can only be rerandomized in the prescribed way
“twist” the first strand (cont.)

Key Idea #3

Make first strand fundamentally different, so it can only be rerandomized in the prescribed way

Analysis uses linear-algebraic interpretation:

- First strand’s randomness $x$ additively perturbed by fixed vector
- More components $(g_1, \ldots, g_4)$ needed for linear independence.

\[ g_{xu}^1, g_{xu}^2, g_{xu}^3, g_{(x+1)u}^4, mB^x, (CD\mu)^x, \]
\[ g_{yu}^1, g_{yu}^2, g_{yu}^3, g_{yu}^4, B^y, (CD\mu)^y, \]
\[ \text{Enc}(u) \]
thinking about $u$

When rerandomizing $u$ value (say, $u \leadsto ru$):

$g^u \leadsto g^{(tu)(ru)}$

... and as the homomorphic operation of CS-lite:

$Enc(u) \leadsto Enc(ru)$

For 2 operations to coincide, must have:

- CS-lite group is subgroup of $\mathbb{Z}_p^*$.
- DDH in both groups (that of CS and CS-lite)

Is it an unreasonable relationship between 2 groups?
When rerandomizing $u$ value (say, $u \rightsquigarrow ru$):

- Multiplication in *exponent* of Cramer-Shoup group ($\mathbb{Z}_p^*$):

$$g_1^{yu} \rightsquigarrow g_1^{(ty)(ru)}$$
thinking about $u$

When rerandomizing $u$ value (say, $u \rightsquigarrow ru$):

- Multiplication in *exponent* of Cramer-Shoup group ($\mathbb{Z}_p^*$):

  $$g_1^{yu} \rightsquigarrow g_1^{(ty)(ru)}$$

- ... and as the homomorphic operation of CS-lite:

  $$\text{Enc}(u) \rightsquigarrow \text{Enc}(ru)$$
thinking about $u$

When rerandomizing $u$ value (say, $u \rightsquigarrow ru$):

- Multiplication in exponent of Cramer-Shoup group ($\mathbb{Z}_p^*$):
  \[ g_1^{yu} \rightsquigarrow g_1^{(ty)(ru)} \]

- ... and as the homomorphic operation of CS-lite:
  \[ \text{Enc}(u) \rightsquigarrow \text{Enc}(ru) \]

For 2 operations to coincide, must have:

- CS-lite group is subgroup of $\mathbb{Z}_p^*$.
- DDH in both groups (that of CS and CS-lite)

Is it an unreasonable relationship between 2 groups?
cunningham chains

**Definition**

A Cunningham chain is 3 primes of the form $q, 2q + 1, 4q + 3$. 

Mike Rosulek (U Montana)  Non-Malleability, Homomorphic Encryption  January 16, 2012 @ U Maryland  24 / 31
**Definition**

A **Cunningham chain** is 3 primes of the form $q, 2q + 1, 4q + 3$.

- $\text{QR}^*_{4q+3}$ and $\text{QR}^*_{2q+1}$ have desired subgroup relationship
- DDH believed to hold ($4q + 3$ and $2q + 1$ are **safe primes**).
- Cunningham chains known to exist for $q \sim 2^{20,000}$. 
Theorem [PR07]

Our construction is unlinkable & RCCA-secure under DDH assumption in 2 groups of related size.
Generalized construction [PR08] allows operations of the form:

$$\text{Enc}(x_1, \ldots, x_n) \rightsquigarrow \text{Enc}(x_1 h_1, \ldots, x_n h_n) \text{ for any } \vec{h} \in \mathbb{H}$$

... where parameter $\mathbb{H}$ can be any subgroup of $\mathbb{G}^n$

**Theorem [PR08]**

Construction is unlinkable & HCCA-secure (w.r.t. “$\mathbb{H}$-restricted group operation”), under DDH assumption in 2 groups of related size.
Generalized construction [PR08] allows operations of the form:

\[ \text{Enc}(x_1, \ldots, x_n) \rightsquigarrow \text{Enc}(x_1 h_1, \ldots, x_n h_n) \text{ for any } \vec{h} \in \mathbb{H} \]

... where parameter \( \mathbb{H} \) can be any subgroup of \( \mathbb{G}^n \)

**Theorem [PR08]**

Construction is unlinkable & HCCA-secure (w.r.t. “\( \mathbb{H} \)-restricted group operation”), under DDH assumption in 2 groups of related size.

Example instantiations:

- \( \mathbb{H} = \{1\} \): Cannot change plaintext, only rerandomize
- \( \mathbb{H} = \mathbb{G}^n \): Can freely “multiply” coordinate-wise
- \( \mathbb{H} = \{1\} \times \mathbb{G} \): Only first component non-malleable
- \( \mathbb{H} = \{(h, \ldots, h) \mid h \in \mathbb{G}\} \): “Scalar multiplication” of vector
What about binary operations?

\[ \text{Enc}(x) \otimes \text{Enc}(y) \sim \text{Enc}(x + y) \]
beyond unary operations

What about binary operations?

\[ Enc(x) \otimes Enc(y) \rightsquigarrow Enc(x + y) \]

**Theorem [PR08]**

Impossible to achieve unlinkable-HCCA-security, for *binary group operation*. 

Mike Rosulek (U Montana)  Non-Malleability, Homomorphic Encryption  January 16, 2012 @ U Maryland
What about binary operations?

\[ \text{Enc}(x) \otimes \text{Enc}(y) \rightsquigarrow \text{Enc}(x + y) \]

**Theorem [PR08]**

Impossible to achieve unlinkable-HCCA-security, for binary group operation.

Proof sketch:

1. Give \( n \) “rigged” ciphertexts to adversary: \( C_1, \ldots, C_n \)
2. Adversary picks \( S \subseteq [n] \), requests decryption of \( C^* = \bigotimes_{i \in S} C_i \)
3. Simulator must extract “derivation history” for \( C^* \) (i.e., \( S \))
4. For \( n \) sufficiently large:

\[
\# \text{ possible histories} = 2^n \gg \# \text{ possible ciphertexts}
\]
We can achieve relaxed requirements \([PR08b]\): 

- Ciphertexts have “length” parameter
- Ciphertexts leak length parameter but *nothing else*
- Non-malleable except for unlinkable operation:

\[
\text{Enc}(\alpha; \ell), \text{Enc}(\beta; \ell') \leadsto \text{Enc}(\alpha \ast \beta; \ell + \ell')
\]
relaxed requirements

We can achieve relaxed requirements [PR08b]:

- Ciphertexts have “length” parameter
- Ciphertexts leak length parameter but \textit{nothing else}
- Non-malleable except for unlinkable operation:

\[
\text{Enc}(\alpha; \ell), \text{Enc}(\beta; \ell') \leadsto \text{Enc}(\alpha \times \beta; \ell + \ell')
\]

Uses unary HCCA as a building block.

- Similar to [SYY99] notion of “cryptocomputing” with ciphertexts of increasing length.
“Non-malleable except for prescribed [strong] operations:”
“Non-malleable except for prescribed [strong] operations:”

**Controlled-non-malleable NIZK:** Chase et al. [CKLM12]:

NIZK non-malleable except unlinkable operations (for $f \in \mathcal{F}$):

$$\exists w : R(x, w) \Rightarrow \exists w' : R(f(x), w')$$

Modular construction of unlinkable-HCCA-secure encryption (CPA + NIZK paradigm)
Computing on Authenticated Data: Ahn et al. [ABCHSW12]:

Signature non-malleable except unlinkable operations (for \( f \in \mathcal{F} \)):

\[
\text{Sign}(m) \leadsto \text{Sign}(f(m))
\]

Definitions support binary operations.
Computing on Authenticated Data: Ahn et al. [ABCHSW12]:
Signature non-malleable except unlinkable operations (for $f \in \mathcal{F}$):
\[
\text{Sign}(m) \rightsquigarrow \text{Sign}(f(m))
\]
Definitions support binary operations.

Targeted-malleable encryption: Boneh et al. [BSW12]:
- Relaxation of HCCA security
- Supports binary operations, ciphertext size grows
open problems

More of the same:

- Improved constructions; Hybrid encryption with HCCA-secure KEM?
- Extend “HCCA-style” ideas to other primitives
open problems

More of the same:

- Improved constructions; Hybrid encryption with HCCA-secure KEM?
- Extend “HCCA-style” ideas to other primitives

Applications:

- Have a hammer; nails sought
- Relaxations of HCCA security for natural applications
- Secret-key? Special features of the scheme *key-activated*?
open problems

More of the same:

- Improved constructions; Hybrid encryption with HCCA-secure KEM?
- Extend “HCCA-style” ideas to other primitives

Applications:

- Have a hammer; nails sought
- Relaxations of HCCA security for natural applications
- Secret-key? Special features of the scheme key-activated?

Technical: avoid pairings via “nested Cunningham groups”?

- Pairings: have $g^a, g^b$, get $e(g^a, g^b) = e(g, g)^{ab}$.
- Nested DDH: have $g^a$ and malleable Enc($b$), get $(g^a)^b$. 
open problems

More of the same:

- Improved constructions; Hybrid encryption with HCCA-secure KEM?
- Extend “HCCA-style” ideas to other primitives

Applications:

- Have a hammer; nails sought
- Relaxations of HCCA security for natural applications
- Secret-key? Special features of the scheme key-activated?

Technical: avoid pairings via “nested Cunningham groups” ?

- Pairings: have $g^a$, $g^b$, get $e(g^a, g^b) = e(g, g)^{ab}$.
- Nested DDH: have $g^a$ and malleable $\text{Enc}(b)$, get $(g^a)^b$.

Anything else?
Thanks for your attention!

fin.