Reconciling Non-malleability & Homomorphic Encryption

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

University of Maryland · January 16, 2012

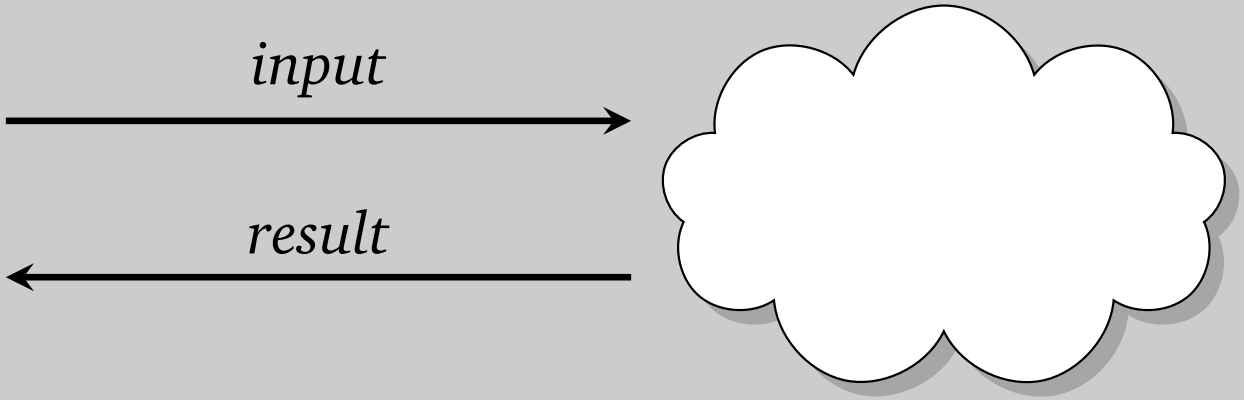
Discussing papers:

[PR07] Manoj Prabhakaran & Mike Rosulek. Rerandomizable RCCA Encryption. In *CRYPTO* 2007.

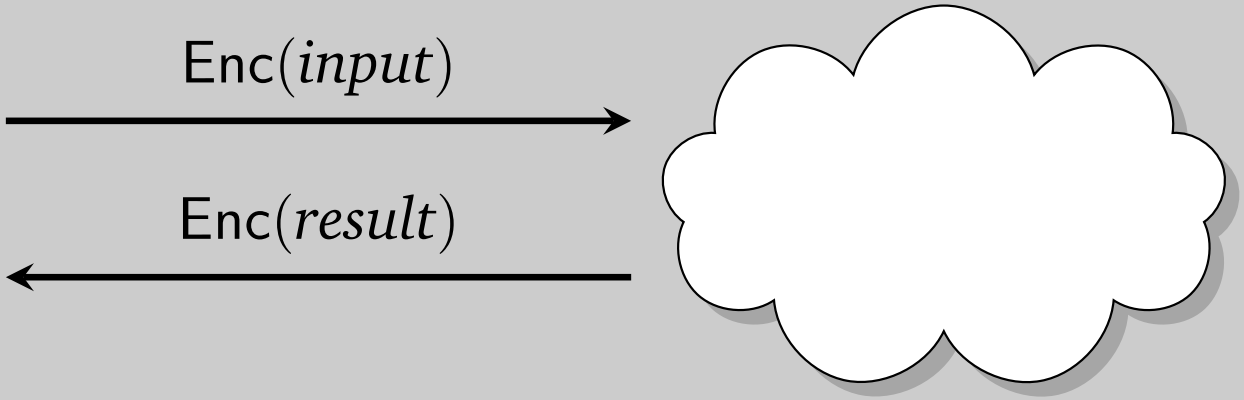
[PR08] Manoj Prabhakaran & Mike Rosulek. Homomorphic Encryption with CCA Security. In *ICALP* 2008.

[PR08b] Manoj Prabhakaran & Mike Rosulek. Towards Robust Computation on Encrypted Data. In *ASIACRYPT* 2008.

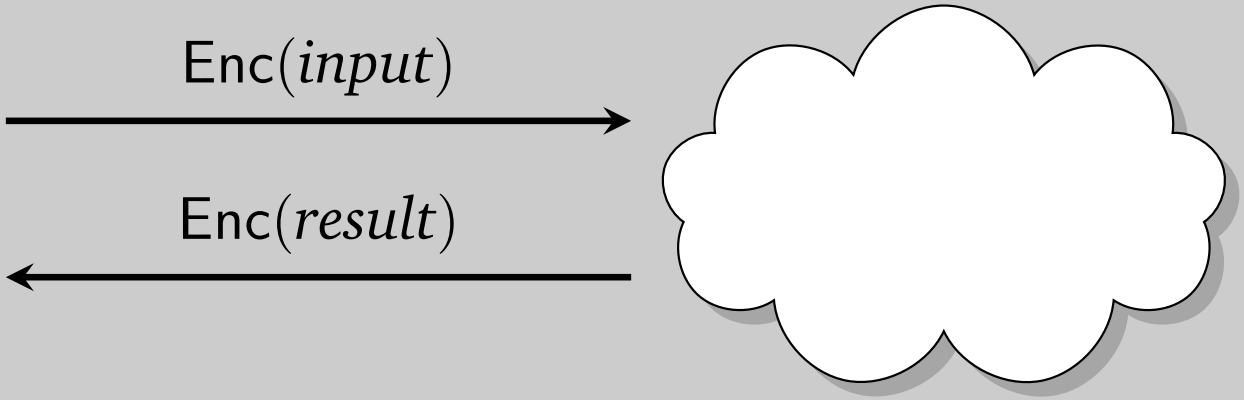
computing on encrypted data



computing on encrypted data



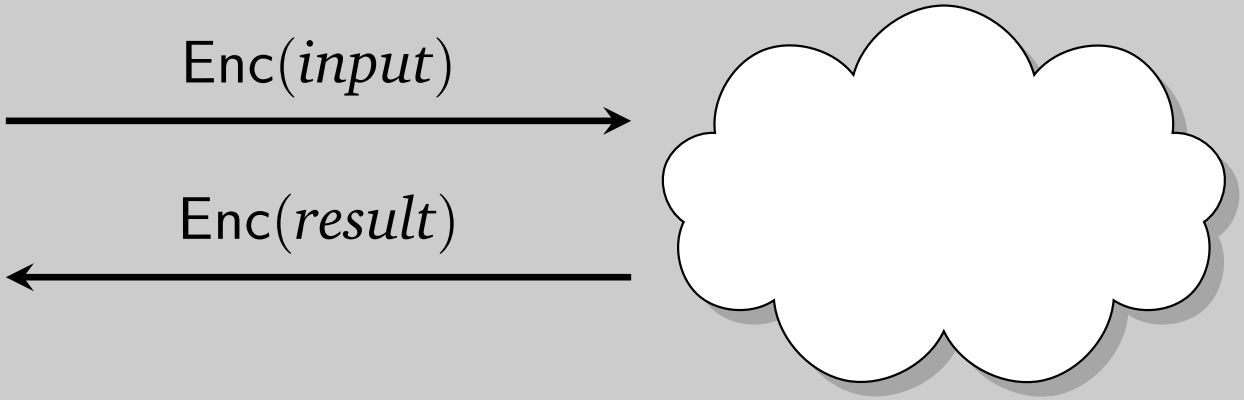
computing on encrypted data



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?

the problem

Desired security (informal)

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

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 $\mathsf{Enc}(m)$ 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

(unary) homomorphic encryption

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 $\alpha \mapsto \beta\alpha$ (known β):

- 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 $\mathsf{Enc}(\alpha\beta)$

contrast with FHE:

Fully homomorphic encryption [G09]:

- Focus: maximum expressivity
- No consideration of non-malleability
- Binary operations: $\mathsf{Enc}(m_1), \mathsf{Enc}(m_2) \rightsquigarrow \mathsf{Enc}(m_1 + m_2)$

contrast with FHE:

Fully homomorphic encryption [G09]:

- Focus: maximum expressivity
- No consideration of non-malleability
- Binary operations: $\mathsf{Enc}(m_1), \mathsf{Enc}(m_2) \rightsquigarrow \mathsf{Enc}(m_1 + m_2)$

This talk:

Focus: sharp tradeoff between malleability / non-malleability:

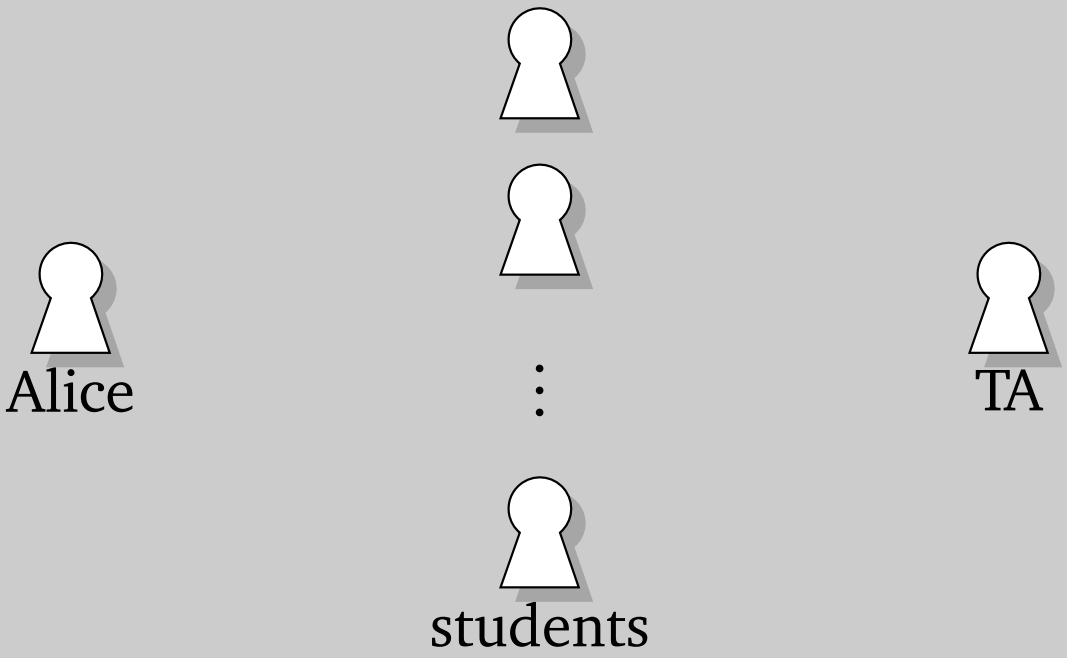
allowed f: available as highly expressive full feature

everything else: computationally infeasible

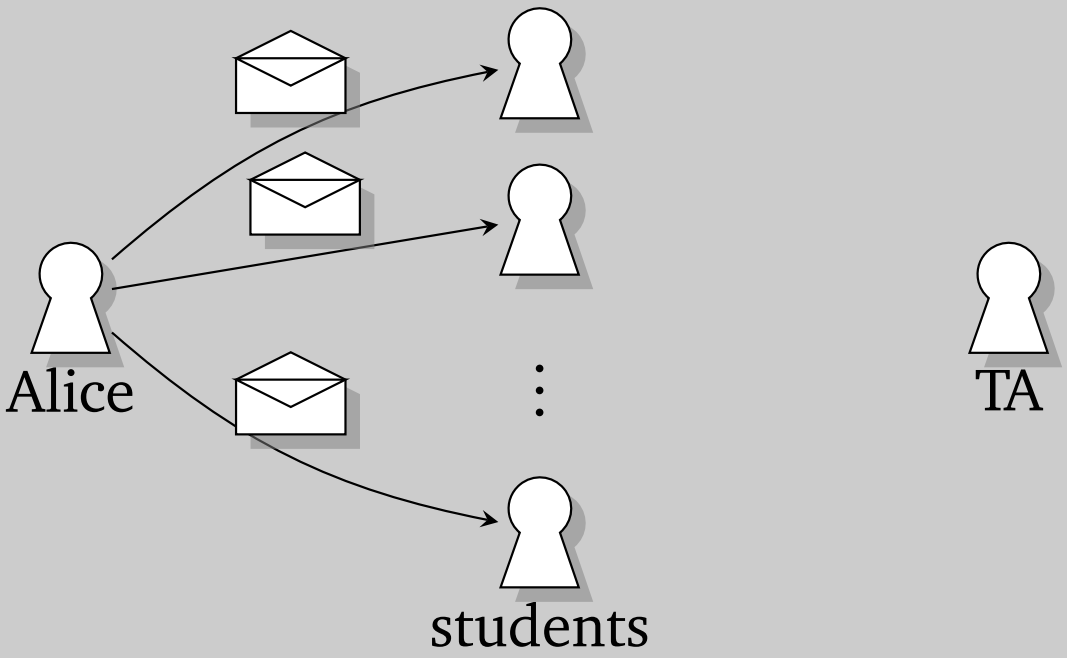
Challenging even when expressivity is low:

- e.g.: $\mathcal{F}$ contains only one operation

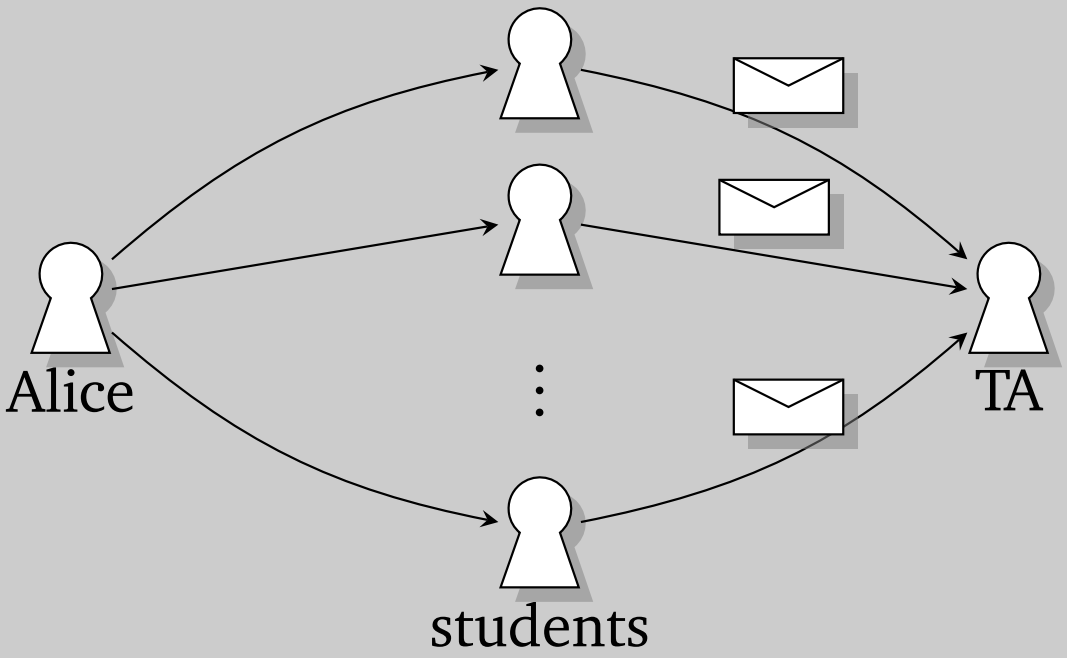
teaching evaluations [PR08b]



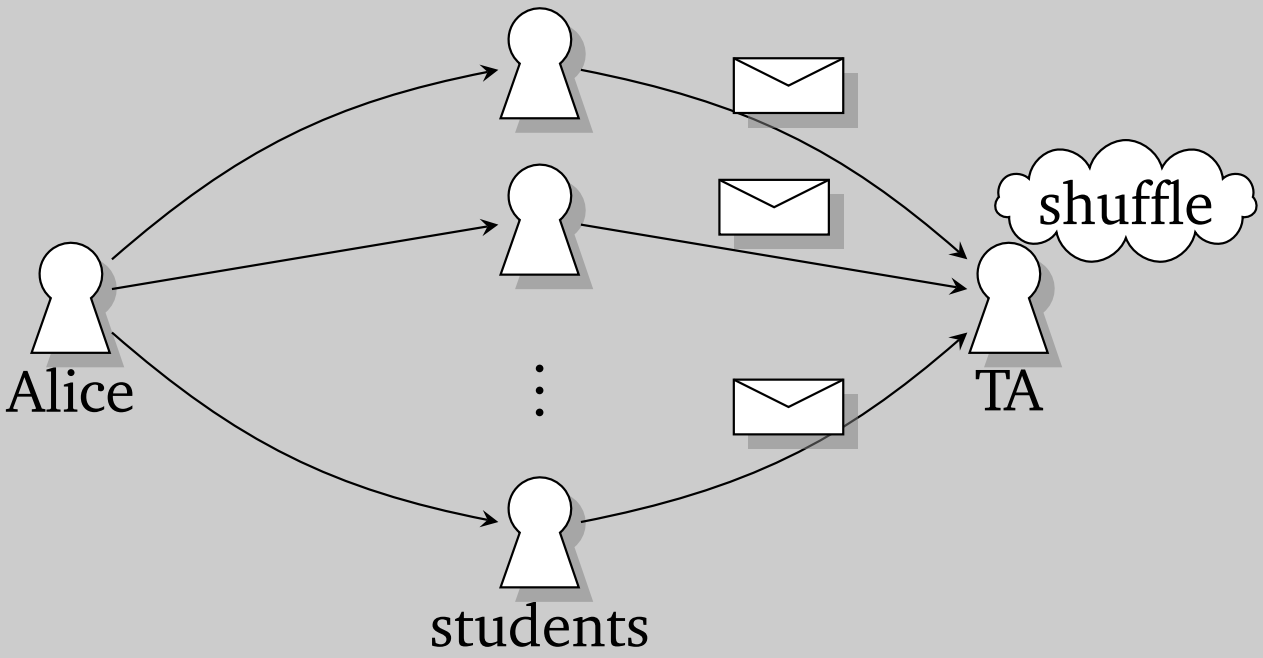
teaching evaluations [PR08b]



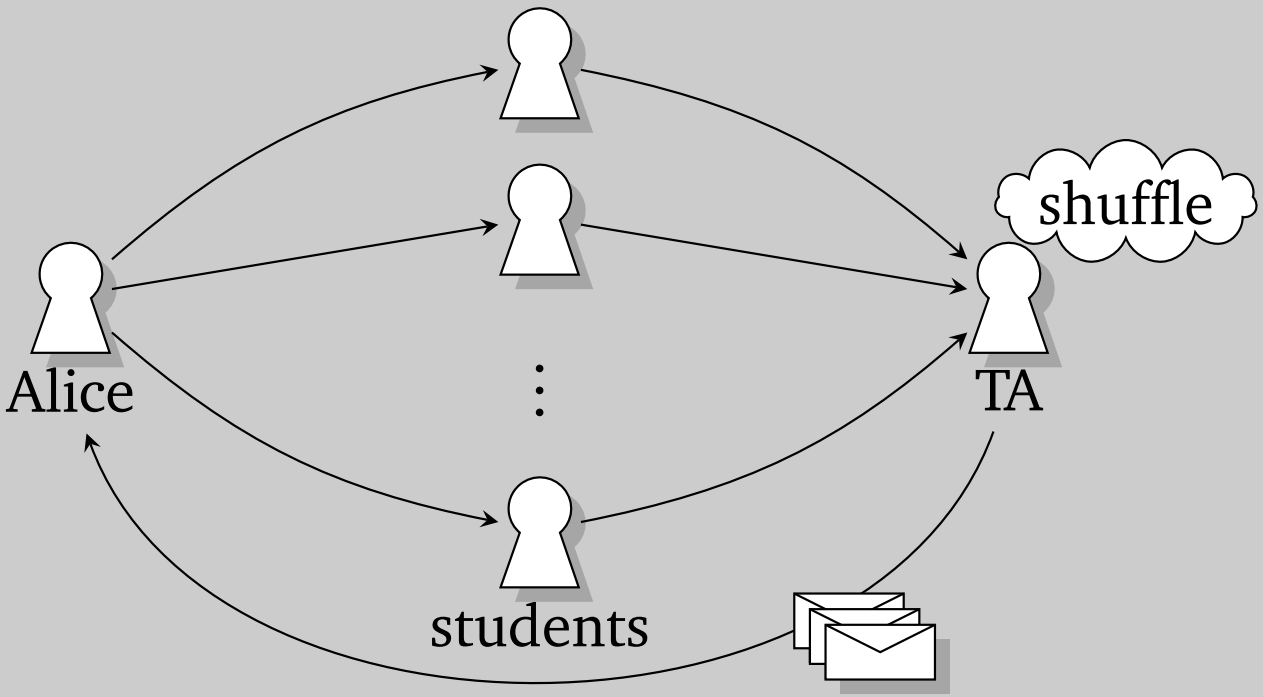
teaching evaluations [PR08b]



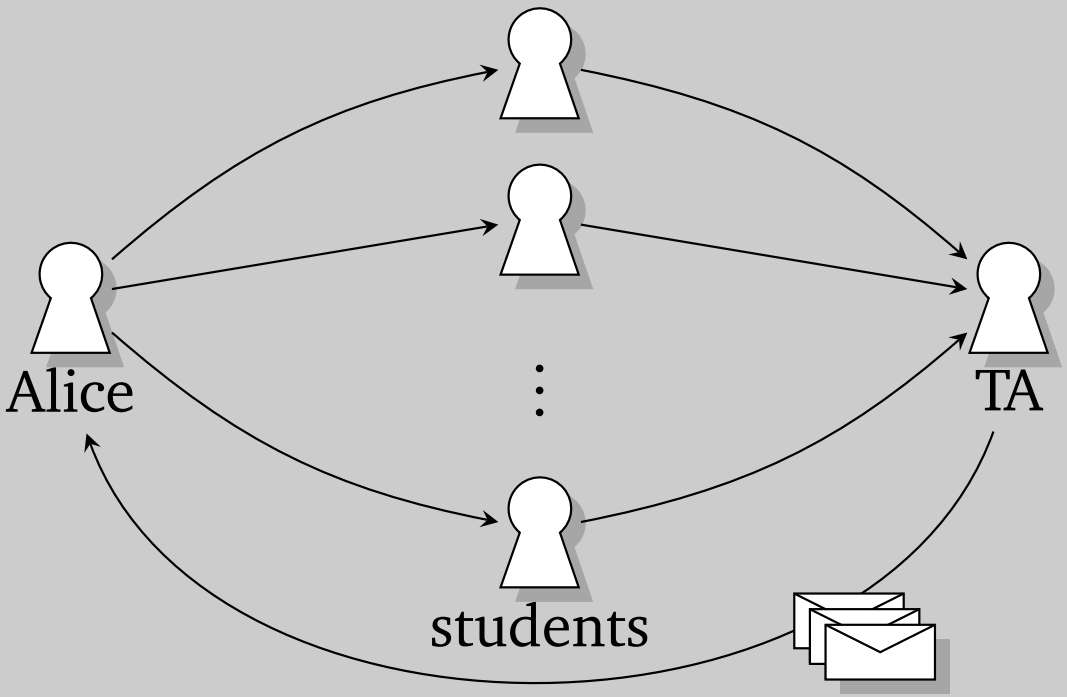
teaching evaluations [PR08b]



teaching evaluations [PR08b]



teaching evaluations [PR08b]

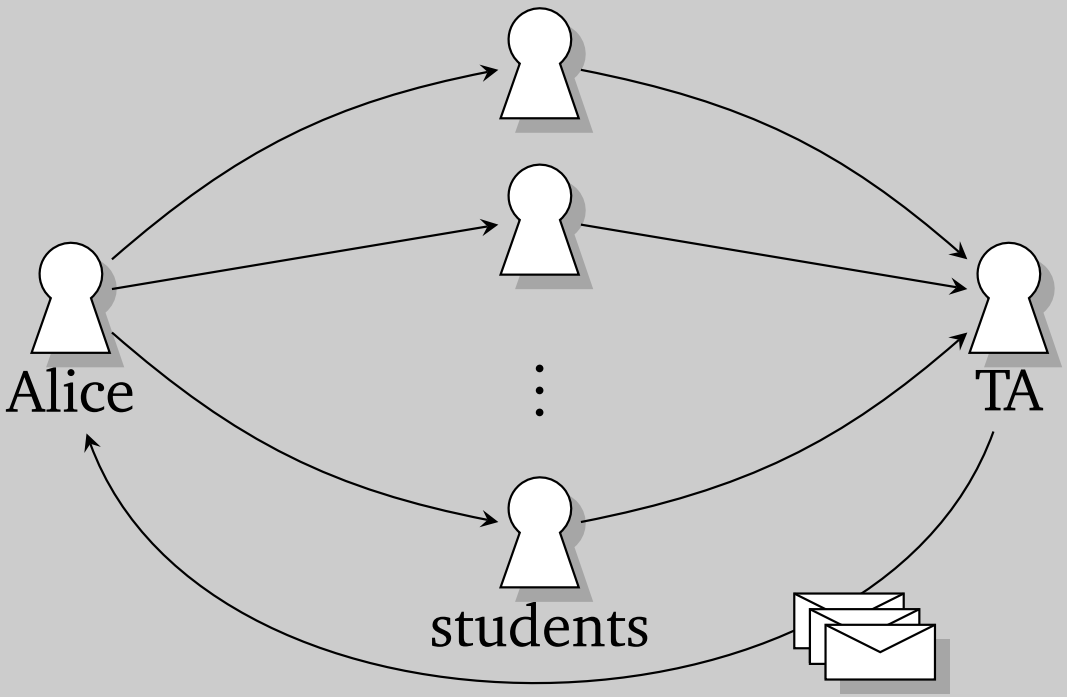


Privacy: TA can't see responses

Functionality: TA must be able to anonymize (shuffle)

Integrity: TA can't modify/replace responses

teaching evaluations [PR08b]



Privacy: TA can't see responses

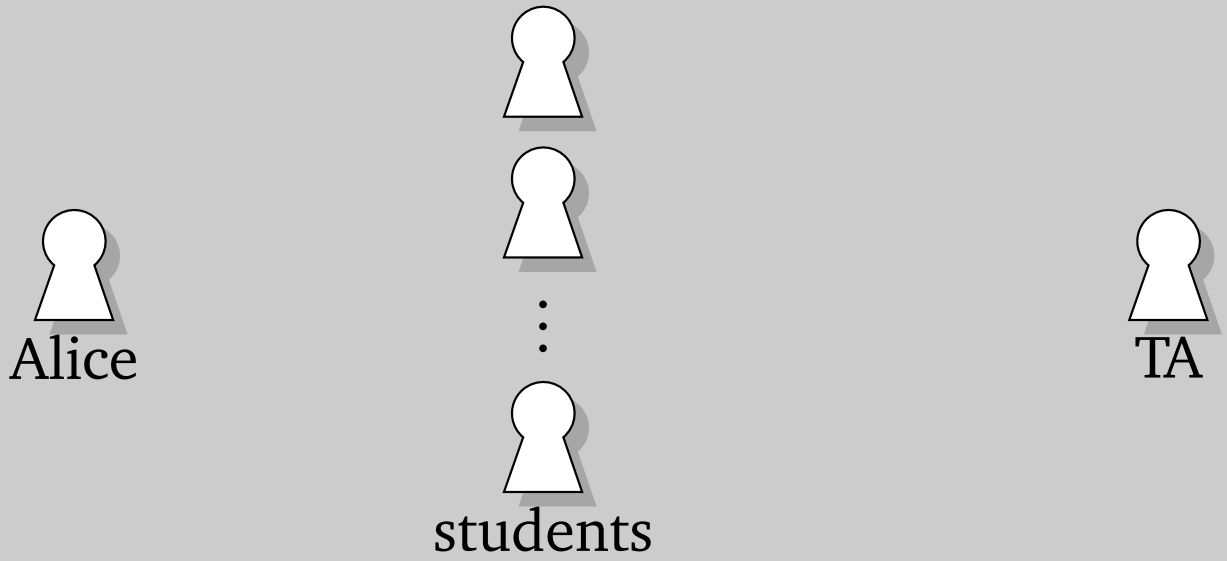
Functionality: TA must be able to anonymize (shuffle)

Integrity: TA can't modify/replace responses

Verifiable shuffle [G02,GL07a,GL07b]

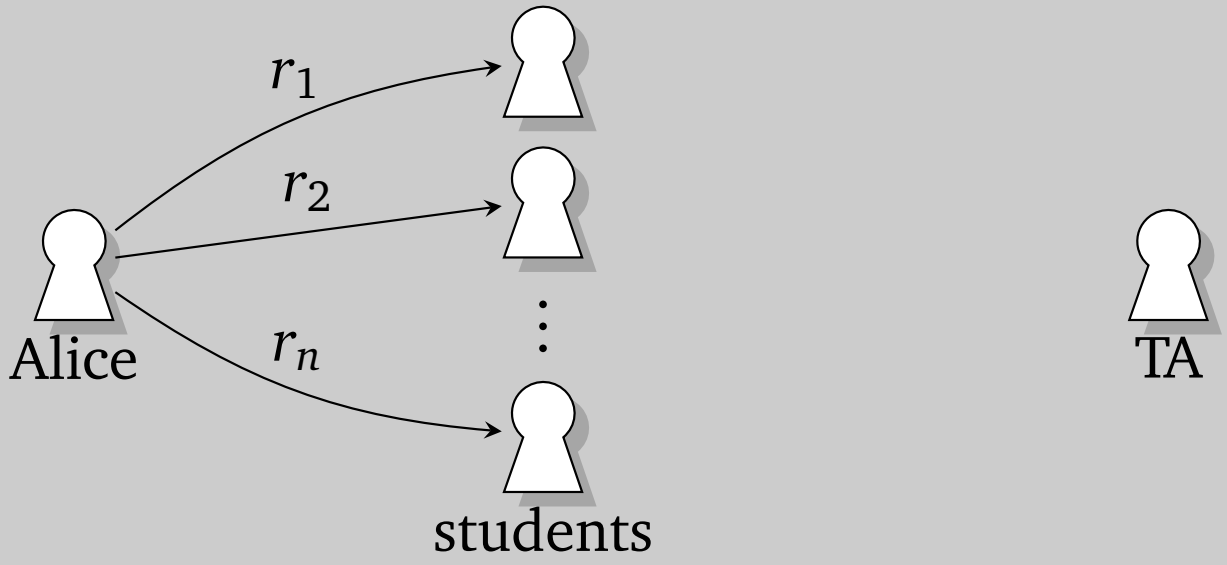
simple protocol

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



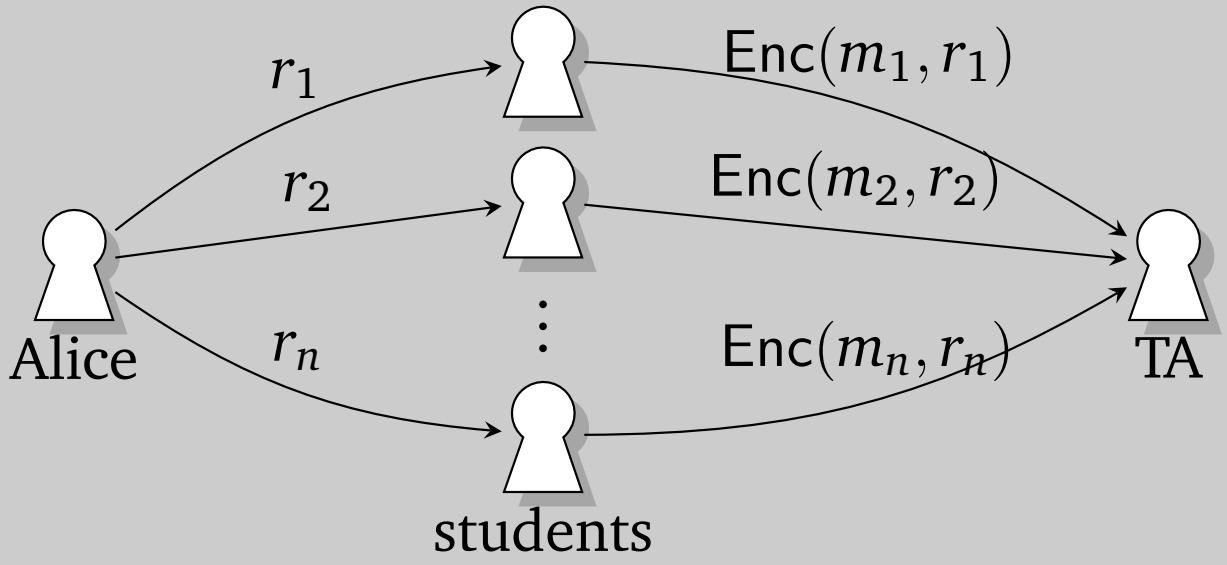
simple protocol

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



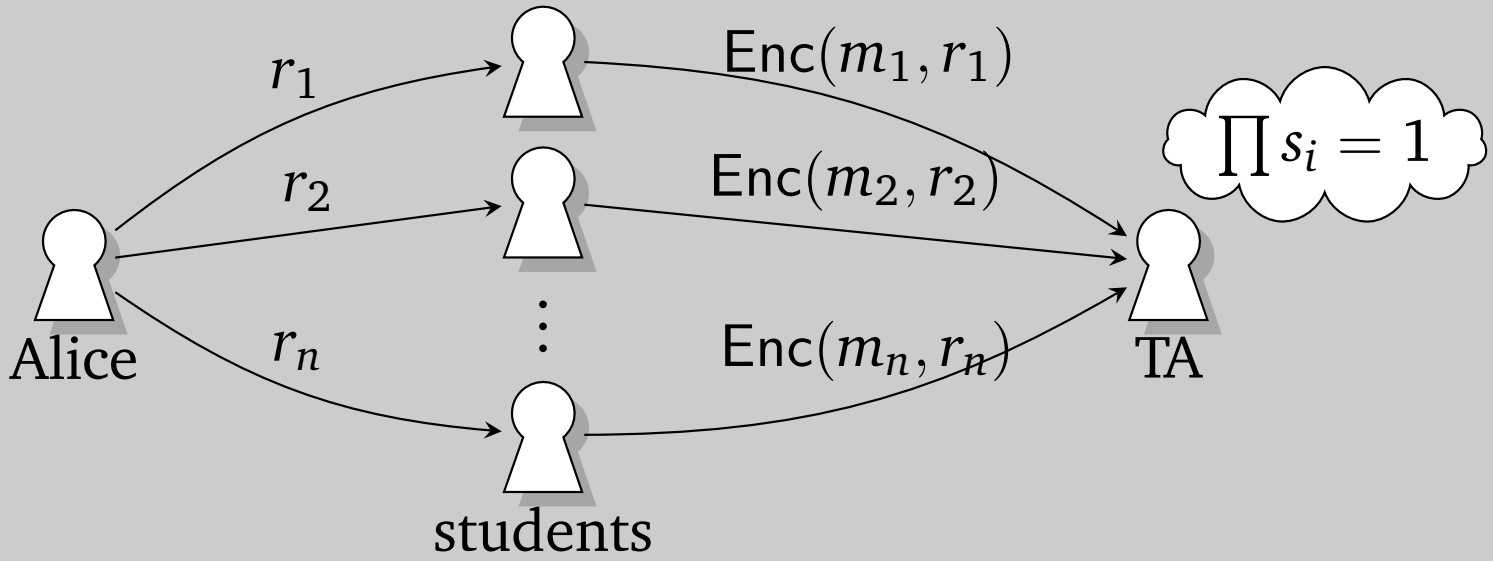
simple protocol

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



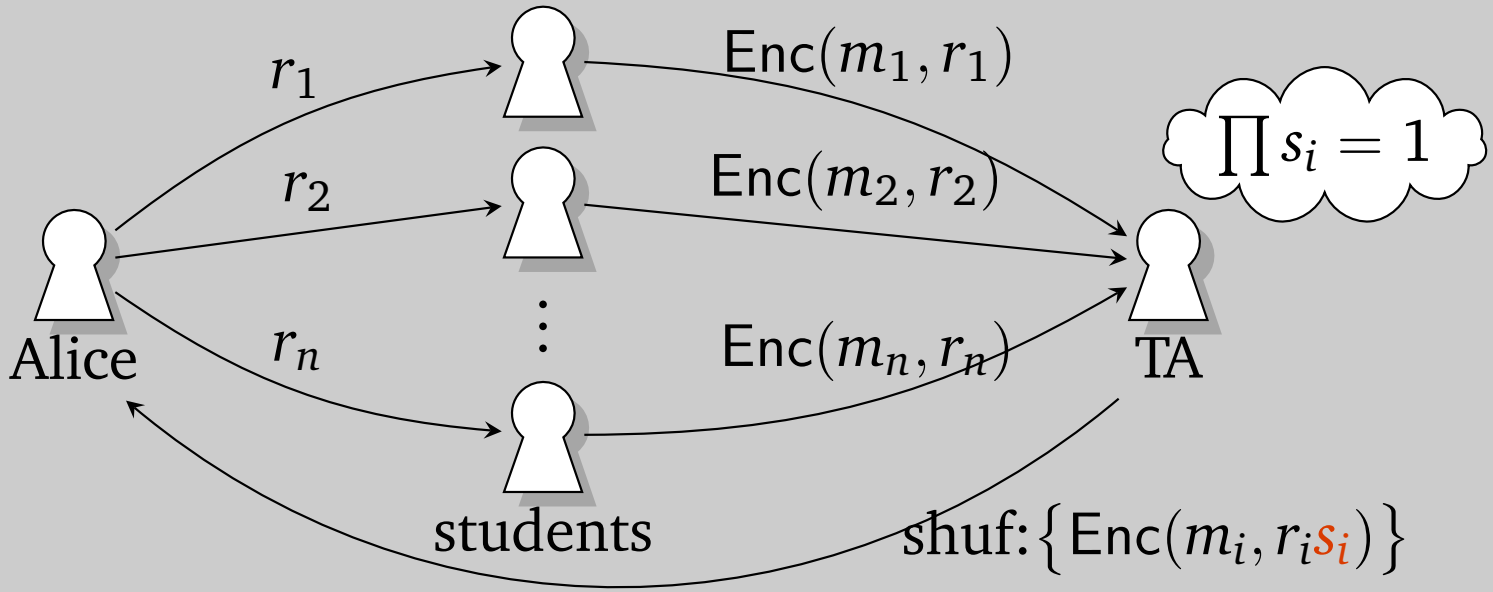
simple protocol

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



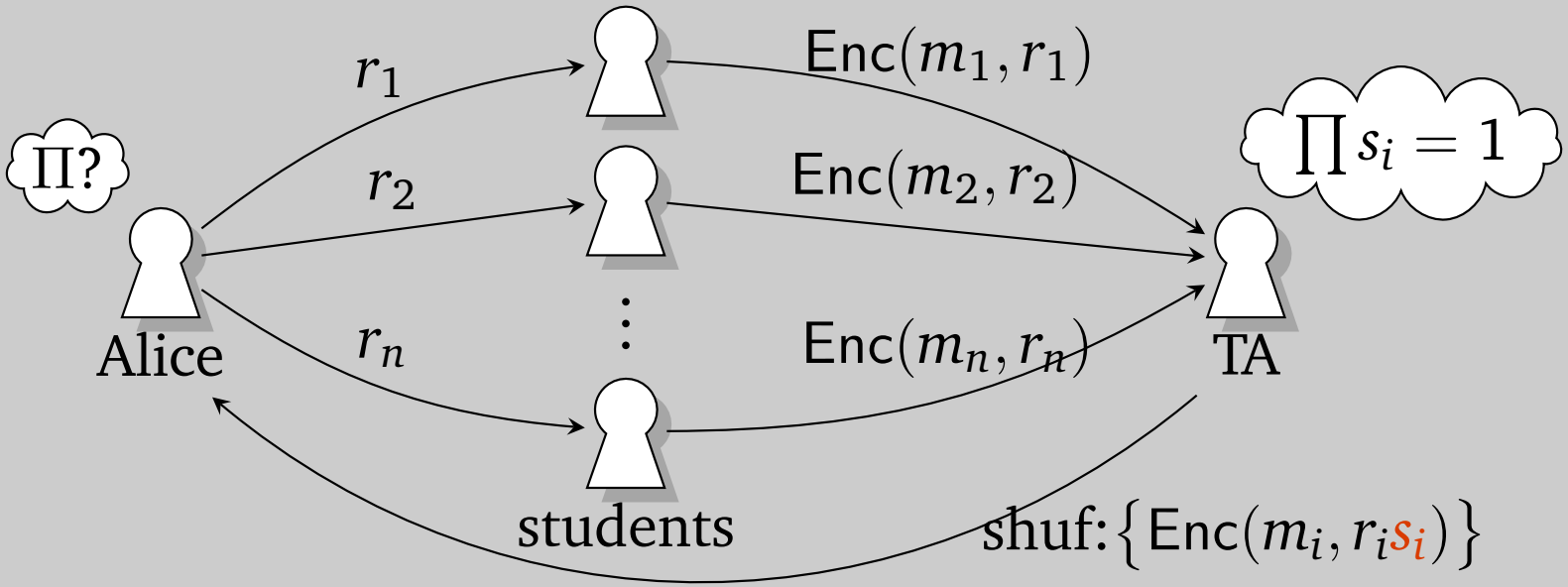
simple protocol

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



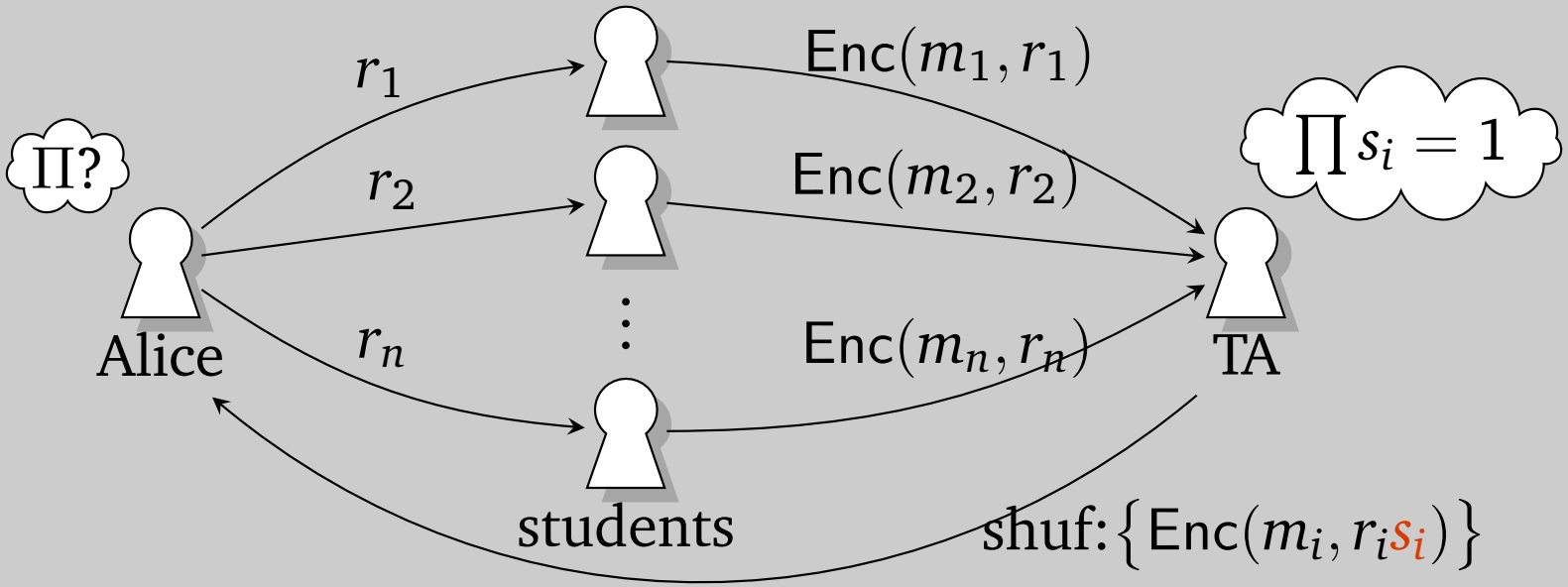
simple protocol

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



simple protocol

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

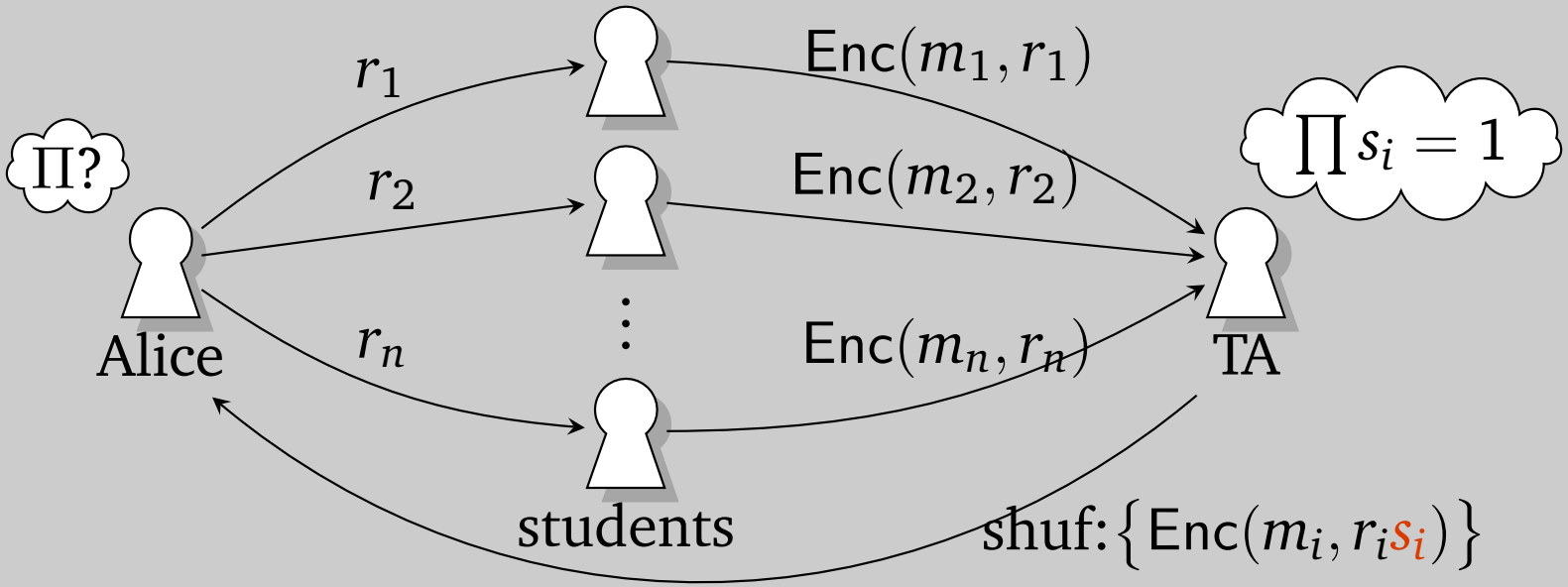


Security proof:

- TA must give $\{\mathsf{Enc}(m'_i, r'_i)\}$, where $\prod r'_i = \prod r_i$

simple protocol

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

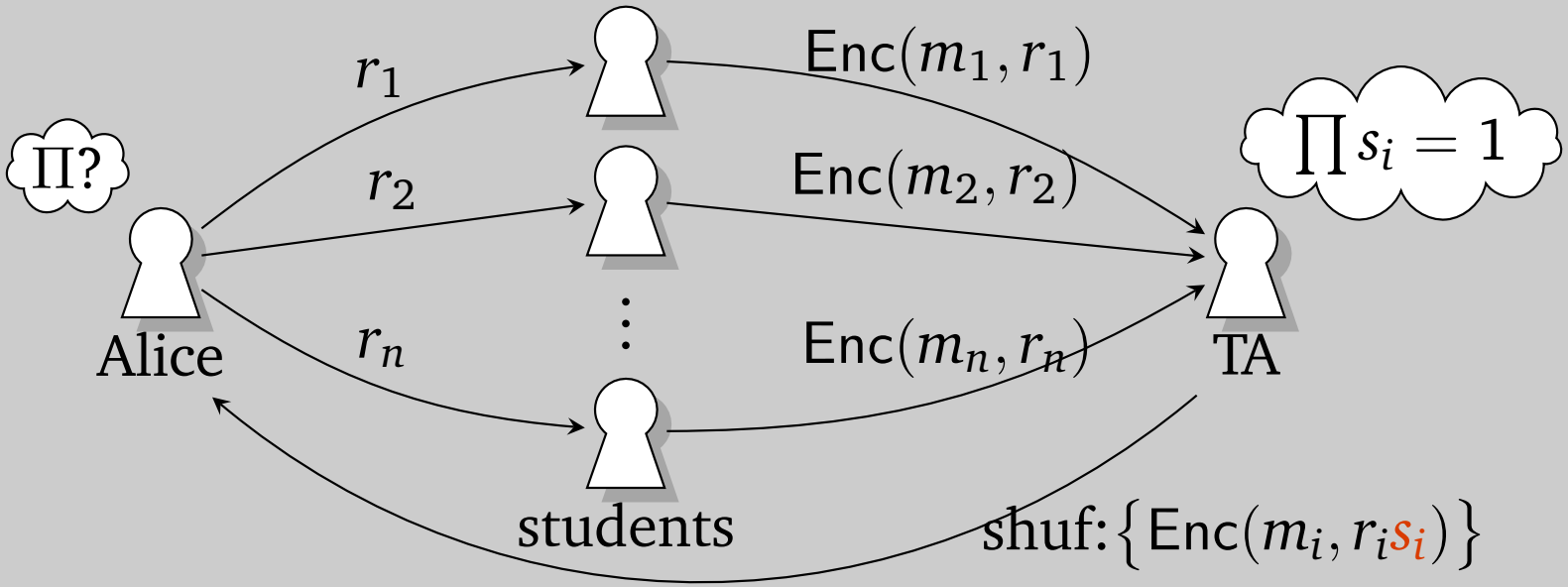


Security proof:

- TA must give $\{\mathsf{Enc}(m'_i, r'_i)\}$, 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 $\prod r_i$

simple protocol

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



Security proof:

- TA must give $\{\mathsf{Enc}(m'_i, r'_i)\}$, 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

$\mathcal{F}$-unlinkability

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

approach to definitions

$\mathcal{F}$-unlinkability

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

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

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

approach to definitions

$\mathcal{F}$-unlinkability

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

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

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

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

approach to definitions

$\mathcal{F}$-unlinkability

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

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

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

- ... unless the dependence is $\mathsf{Dec}(C) = f(m)$ 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 $C \leftarrow \mathsf{Enc}(m_0)$.

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 $C \leftarrow \mathsf{Enc}(m_1)$.

5. Provide Dec oracle, except:
   - Refuse if given C.

generalizing IND-CCA game

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

generalizing IND-CCA game

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathsf{Enc}(m_0)$.
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 $C \leftarrow \mathsf{Enc}(m_1)$.
   (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 $C \leftarrow \mathsf{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathsf{Enc}(m_1)$.
   (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 $C \leftarrow \mathsf{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathsf{Enc}(m_1)$.
   (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 $C \leftarrow \mathsf{Enc}(m_0)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m_0.
4. Give $C \leftarrow \mathsf{Enc}(m_1)$.
   (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

derivative ciphertexts

Derivatives of C

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

Examples:

CCA: C' is derivative $\Leftrightarrow C' = C$

Intuition: Dec oracle only allowed to compensate on derivatives!

derivative ciphertexts

Derivatives of C

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

Examples:

CCA: C' is derivative $\Leftrightarrow C' = C$

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

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

Intuition: Dec oracle only allowed to compensate on derivatives!

why this won't work

Example: $\mathcal{F} = \{$ group operations, $\alpha \mapsto \beta\alpha$ $\}$

These distributions are *identical*:

- $\mathsf{Enc}(\beta)$ derived by legitimately multiplying $\mathsf{Enc}(\alpha)$ by β/α
- $\mathsf{Enc}(\beta)$ obtained independently by encrypting β

why this won't work

Example: $\mathcal{F} = \{$ group operations, $\alpha \mapsto \beta\alpha$ $\}$

These distributions are *identical*:

- $\mathsf{Enc}(\beta)$ derived by legitimately multiplying $\mathsf{Enc}(\alpha)$ by β/α
- $\mathsf{Enc}(\beta)$ obtained independently by encrypting β

Problem:

- Dec oracle cannot identify derived ciphertexts
- Alternately, every ciphertext flagged as (possibly) derived from C
  - Game collapses to IND-CCA1.

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 $C \leftarrow \mathsf{Enc}(m^*)$.
5. Provide Dec oracle.

1. Generate keypair, give PK.
2. Provide Dec oracle.
3. Adversary chooses m^*.
4. Give $C \leftarrow \mathsf{Enc}(m_1)$.
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 $C \leftarrow \mathsf{Enc}(m^*)$.
5. Provide Dec oracle.

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

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

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

interpreting the security game

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

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

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

Suppose adversary can generate C' whose value depends on C

Submit C'; response is $\mathsf{Dec}(C')$

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 $C \leftarrow \mathsf{Enc}(m^*)$.
5. Provide Dec oracle.

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

Suppose adversary can generate C' whose value depends on C

Submit C'; response is $\mathsf{Dec}(C')$

Response is $f(m^*)$ for some f.

$\Longrightarrow$ Dependence is $\mathsf{Dec}(C') = f(\mathsf{Dec}(C))$, for some $f \in \mathsf{range}(\mathsf{RigExtract})$.

interpreting the security game

Suppose RigExtract never outputs f^*:

- Scheme cannot be malleable via f^* operation.

interpreting the security game

Suppose 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 $\exists$ RigEnc, RigExtract:

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

Fine print:

- Oracles for RigEnc and RigExtract should be provided, too.
- RigEnc, 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 [C01,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 [PR07,PR08]

our construction [PR07,PR08]

Simplest example: $\mathcal{F} = \{\text{identity function}\}$:

$\mathcal{F}$-unlinkability = rerandomizability

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

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

our construction [PR07,PR08]

Simplest example: $\mathcal{F} = \{\text{identity function}\}$:

$\mathcal{F}$-unlinkability = rerandomizability

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

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

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

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

our construction [PR07,PR08]

Simplest example: $\mathcal{F} = \{\text{identity function}\}$:

$\mathcal{F}$-unlinkability = rerandomizability

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

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

$\mathcal{F}$-HCCA = 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.

our construction [PR07]

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):

$$g^x, m A^x \qquad \text{for random } x$$

our construction [PR07]

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):

$$g^x, mA^x \qquad \text{for random } x$$

- "Double-Strand" ElGamal encryption of m:

$$g^x, mA^x, g^y, A^y \qquad \text{for random } x, y$$

our construction [PR07]

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):

$$g^x, m A^x \qquad \text{for random } x$$

- "Double-Strand" ElGamal encryption of m:

$$g^x, m A^x, g^y, A^y \qquad \text{for random } x, y$$

- Rerandomize $\qquad y \mapsto yt$ (multiplicative)

$$(g^y)^t, \quad (A^y)^t \qquad \text{for rand } t$$

$$= \qquad g^{yt}, \quad A^{yt}$$

our construction [PR07]

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):

$$g^x, mA^x \qquad \text{for random } x$$

- "Double-Strand" ElGamal encryption of m:

$$g^x, mA^x, g^y, A^y \qquad \text{for random } x, y$$

- Rerandomize $x \mapsto x + ys$ (additive), $y \mapsto yt$ (multiplicative)

$$\begin{aligned} & g^x(g^y)^s, \quad mA^x(A^y)^s, \quad (g^y)^t, \quad (A^y)^t \qquad \text{for rand } t, s \\ = \; & g^{x+ys}, \quad mA^{x+ys}, \quad g^{yt}, \quad A^{yt} \end{aligned}$$

double-strand Cramer-Shoup

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

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

double-strand Cramer-Shoup

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

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

Double-strand idea applied:

$$g_1^x, g_2^x, m\,B^x, (CD^\mu)^x, g_1^y, g_2^y, B^y, (CD^\mu)^y \qquad \text{for rand } x, y$$

double-strand Cramer-Shoup

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

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

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

$$g_1^x, g_2^x, m\,B^x, (CD^\mu)^x, g_1^y, g_2^y, B^y, (CD^\mu)^y \qquad \text{for rand } x, y$$

double-strand Cramer-Shoup

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

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

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

$$g_1^x, g_2^x, m\,B^x, (CD^\mu)^x, g_1^y, g_2^y, B^y, (CD^\mu)^y \qquad \text{for rand } x, y$$

Possible Attack!

"Mix-and-match" strands from 2 independent ciphertexts

- Result is valid $\Leftrightarrow$ same plaintext (μ)

tie the knot

Key Idea #2

"Tie strands together" with shared randomness u; give an additional encryption of u

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

tie the knot

Key Idea #2

"Tie strands together" with shared randomness u; give an additional encryption of u

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

Thwarts "mix-and-match" attack:

- Strands combine nicely, only if they share common u

tie the knot

Key Idea #2

"Tie strands together" with shared randomness u; give an additional encryption of u

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

Thwarts "mix-and-match" attack:

- Strands combine nicely, only if they share common u

What's that $\mathsf{Enc}(u)$ doing?

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

tie the knot

Key Idea #2

"Tie strands together" with shared randomness u; give an additional encryption of u

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

Thwarts "mix-and-match" attack:

- Strands combine nicely, only if they share common u

What's that $\mathsf{Enc}(u)$ doing?

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

"Cramer-Shoup lite" has these properties [fine print].

“twist” the first strand

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

Possible Attack

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

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

“twist” the first strand

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

Possible Attack

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

$$\begin{aligned} g_1^{xu}, g_2^{xu}, mB^x, (CD^\mu)^x \quad &\rightsquigarrow \quad (g_1^{xu})^2, \quad (g_2^{xu})^2, \quad \frac{(mB^x)^2}{m'}, \quad ((CD^\mu)^x)^2 \\ &= \quad g_1^{(2x)u}, \quad g_2^{(2x)u}, \quad \frac{m^2}{m'}B^{2x}, \quad (CD^\mu)^{2x} \end{aligned}$$

“twist” the first strand

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

Possible Attack

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

$$\begin{aligned} g_1^{xu}, g_2^{xu}, mB^x, (CD^\mu)^x \quad &\rightsquigarrow \quad (g_1^{xu})^2, \quad (g_2^{xu})^2, \quad \frac{(mB^x)^2}{m'}, \quad ((CD^\mu)^x)^2 \\ &= \quad g_1^{(2x)u}, \quad g_2^{(2x)u}, \quad \frac{m^2}{m'}B^{2x}, \quad (CD^\mu)^{2x} \end{aligned}$$

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.

$$\begin{array}{cccccccc} g_1^{xu}, & g_2^{xu}, & g_3^{xu}, & g_4^{(x+1)u}, & mB^x, & (CD^\mu)^x, \\ g_1^{yu}, & g_2^{yu}, & g_3^{yu}, & g_4^{yu}, & B^y, & (CD^\mu)^y, \\ & & & & & \mathsf{Enc}(u) \end{array}$$

thinking about u

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

thinking about u

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

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

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

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:

$$\mathsf{Enc}(u) \rightsquigarrow \mathsf{Enc}(ru)$$

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:

$$\mathsf{Enc}(u) \rightsquigarrow \mathsf{Enc}(ru)$$

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 $q, 2q+1, 4q+3$.

cunningham chains

Definition

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

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

security

Theorem [PR07]

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

generalization

Generalized construction [PR08] allows operations of the form:

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

... 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.

generalization

Generalized construction [PR08] allows operations of the form:

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

... 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:

- $\mathbb{H} = \{\mathbf{1}\}$: Cannot change plaintext, only rerandomize
- $\mathbb{H} = \mathbb{G}^n$: Can freely "multiply" coordinate-wise
- $\mathbb{H} = \{\mathbf{1}\} \times \mathbb{G}$: Only first component non-malleable
- $\mathbb{H} = \{(h, \ldots, h) \mid h \in \mathbb{G}\}$: "Scalar multiplication" of vector

beyond unary operations

What about binary operations?

$$\mathsf{Enc}(x) \otimes \mathsf{Enc}(y) \rightsquigarrow \mathsf{Enc}(x+y)$$

beyond unary operations

What about binary operations?

$$\mathsf{Enc}(x) \otimes \mathsf{Enc}(y) \rightsquigarrow \mathsf{Enc}(x+y)$$

Theorem [PR08]

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

beyond unary operations

What about binary operations?

$$\mathsf{Enc}(x) \otimes \mathsf{Enc}(y) \rightsquigarrow \mathsf{Enc}(x+y)$$

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:

$$\#\text{ possible histories} = 2^n \gg \#\text{ possible ciphertexts}$$

relaxed requirements

We can achieve relaxed requirements [PR08b]:

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

$$\mathsf{Enc}(\alpha;\ell), \mathsf{Enc}(\beta;\ell') \rightsquigarrow \mathsf{Enc}(\alpha * \beta;\ell+\ell')$$

relaxed requirements

We can achieve relaxed requirements [PR08b]:

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

$$\mathsf{Enc}(\alpha;\ell), \mathsf{Enc}(\beta;\ell') \rightsquigarrow \mathsf{Enc}(\alpha * \beta;\ell+\ell')$$

Uses unary HCCA as a building block.

- Similar to [SYY99] notion of "cryptocomputing" with ciphertexts of increasing length.

taming malleability

"Non-malleable except for prescribed [strong] operations:"

taming malleability

"Non-malleable except for prescribed [strong] operations:"

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

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

$$\text{“}\exists w : R(x, w)\text{”} \rightsquigarrow \text{“}\exists w' : R(f(x), w')\text{”}$$

Modular construction of unlinkable-HCCA-secure encryption (CPA + NIZK paradigm)

taming malleability (cont.)

Computing on Authenticated Data: Ahn et al. [ABCHSW12]:

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

$$\mathsf{Sign}(m) \rightsquigarrow \mathsf{Sign}(f(m))$$

Definitions support binary operations.

taming malleability (cont.)

Computing on Authenticated Data: Ahn et al. [ABCHSW12]:

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

$$\mathsf{Sign}(m) \rightsquigarrow \mathsf{Sign}(f(m))$$

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 $\mathsf{Enc}(b)$, 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 $\mathsf{Enc}(b)$, get $(g^a)^b$.

Anything else?

Thanks for your attention!

fin.