Towards Robust Computation on Encrypted Data

Manoj Prabhakaran & Mike Rosulek

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

ASIACRYPT 2008
December 9, 2008

Computing on Encrypted Data

Conflicting demands in crypto protocols:

Data Privacy

Parties should only see data they're allowed to see

Computing on Encrypted Data

Conflicting demands in crypto protocols:

Data Privacy

Parties should only see data they're allowed to see

Functionality

Data needs to be manipulated, used for computation

Computing on Encrypted Data

Conflicting demands in crypto protocols:

Data Privacy

Parties should only see data they're allowed to see

Functionality

Data needs to be manipulated, used for computation

Data Robustness

Data should only be manipulatable in allowed ways

Homomorphic Encryption

Definition: Homomorphic encryption

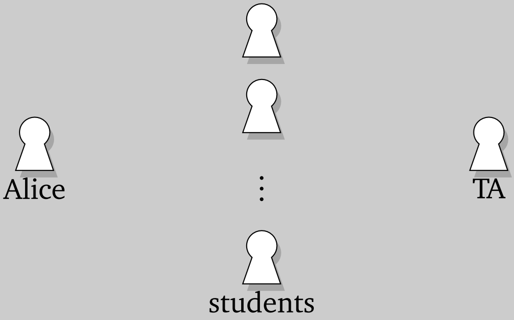
Scheme is homomorphic with respect to f if anyone can take $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n)$ and produce (fresh) $\mathsf{Enc}(f(\vec{x}))$.

- Example: f is addition

Natural ingredient to achieve demands of protocols?

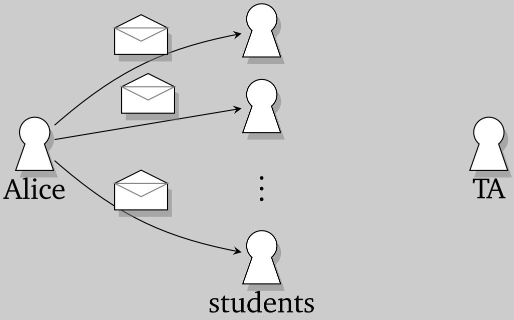
Motivating Example: Teaching evaluations

Alice teaches a wildly popular crypto course:



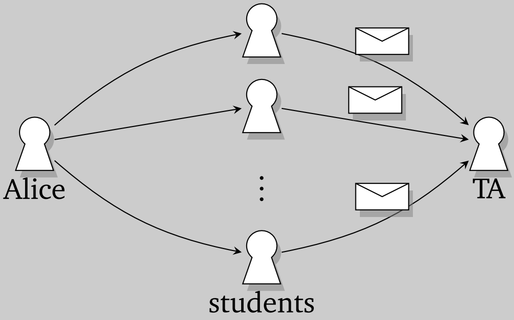
Motivating Example: Teaching evaluations

Alice teaches a wildly popular crypto course:



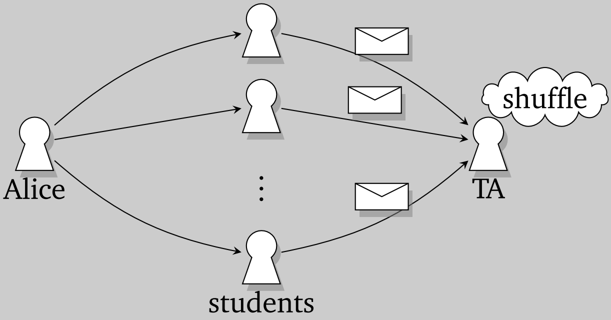
Motivating Example: Teaching evaluations

Alice teaches a wildly popular crypto course:



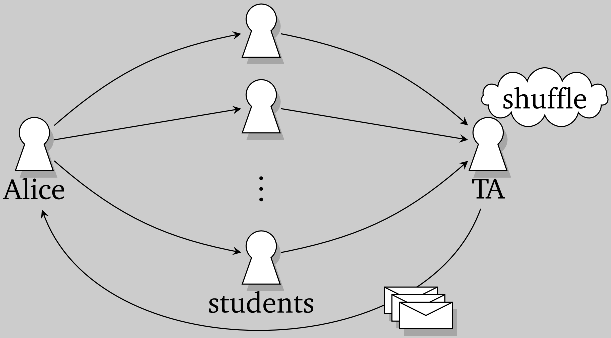
Motivating Example: Teaching evaluations

Alice teaches a wildly popular crypto course:



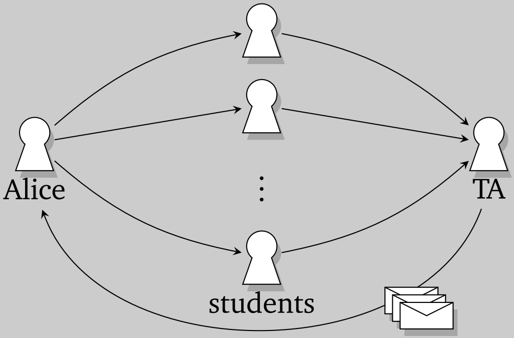
Motivating Example: Teaching evaluations

Alice teaches a wildly popular crypto course:



Motivating Example: Teaching evaluations

Alice teaches a wildly popular crypto course:

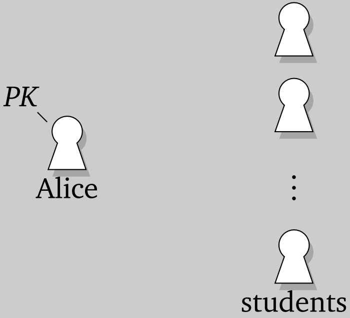


Privacy: TA can't see responses

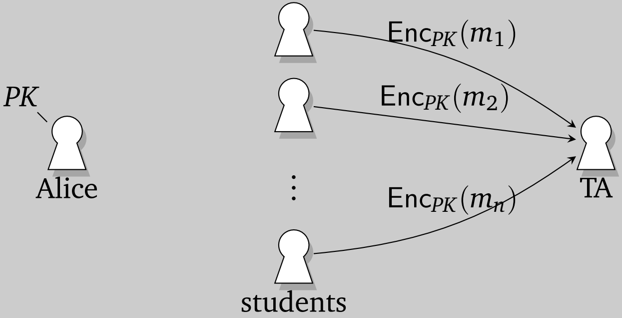
Functionality: TA must be able to anonymize (shuffle)

Robustness: TA can't modify/replace responses

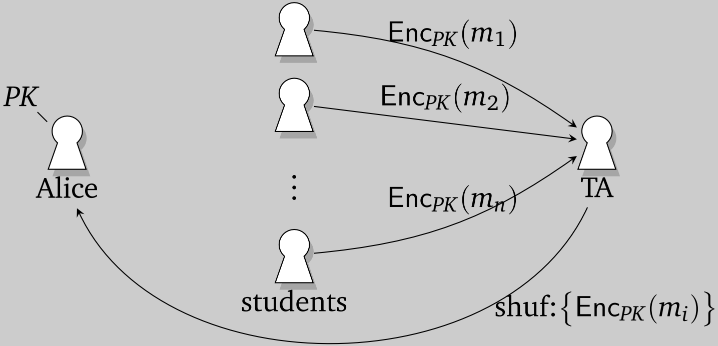
Towards a protocol



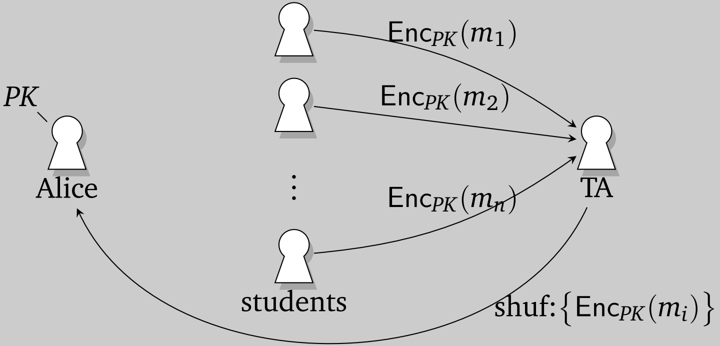
Towards a protocol



Towards a protocol



Towards a protocol

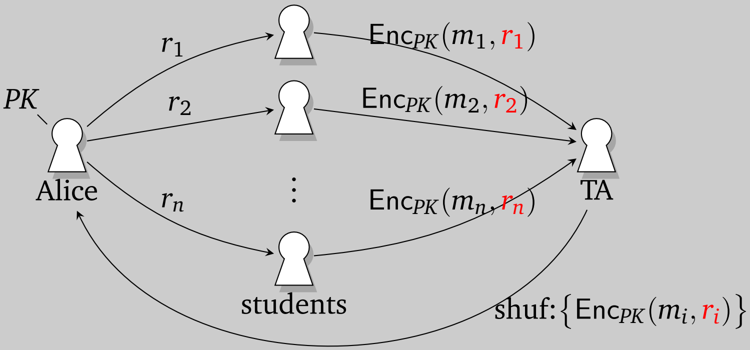


Problem:

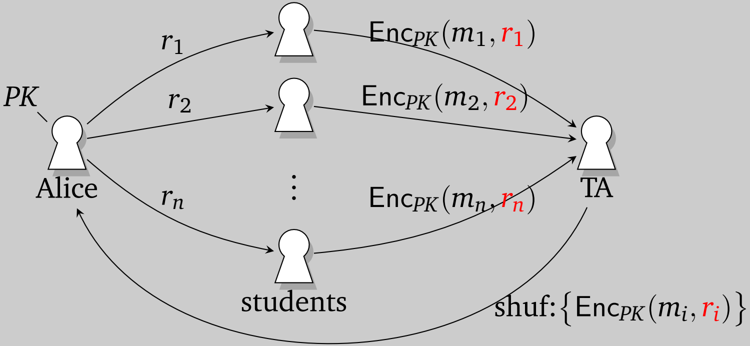
TA could omit someone's response & insert his own.

- Need some shared secret between Alice & students

Towards a protocol



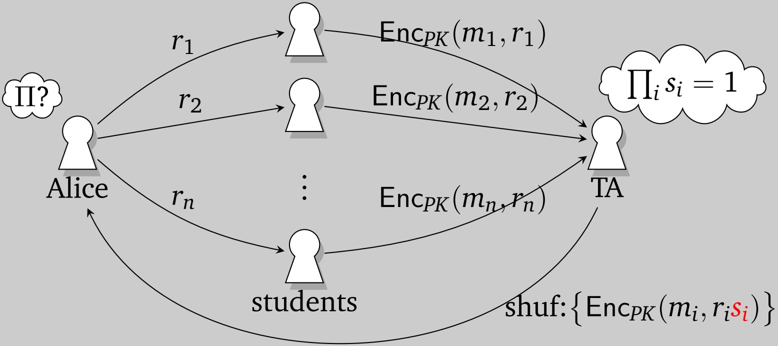
Towards a protocol



Problem:

Alice can associate m_i with the student to whom she sent r_i.

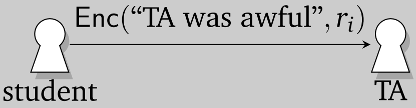
Towards a protocol



Solution:

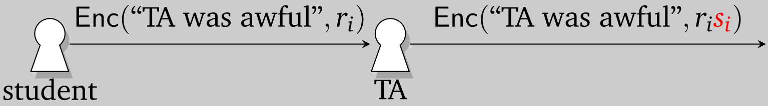
- Scheme is homomorphic via $\text{Enc}(m,r) \rightsquigarrow \text{Enc}(m, rs)$
- Alice verifies $\prod_i r_i = \prod_i (r_i s_i)$
- TA can't throw out anyone's response.

Not all homomorphic operations created equal



Two classes of homomorphic operations:

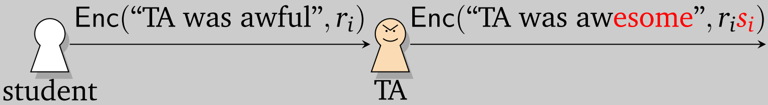
Not all homomorphic operations created equal



Two classes of homomorphic operations:

- Crucial within a protocol ("features")

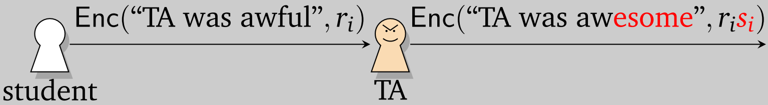
Not all homomorphic operations created equal



Two classes of homomorphic operations:

- Crucial within a protocol ("features")
- Problematic if possible by adversary ("vulnerabilities")

Not all homomorphic operations created equal



Two classes of homomorphic operations:

- Crucial within a protocol (“features”)
- Problematic if possible by adversary (“vulnerabilities”)

Problem of robustness against unwanted manipulation.

Insufficient Definitions

Homomorphic encryption provides:

Privacy: CPA security

Insufficient Definitions

Homomorphic encryption provides:

Privacy: CPA security

Functionality: Homomorphic operations/features

Insufficient Definitions

Homomorphic encryption provides:

Privacy: CPA security

Functionality: Homomorphic operations/features

Robustness: ???

Insufficient Definitions

Homomorphic encryption provides:

Privacy: CPA security

Functionality: Homomorphic operations/features

Robustness: ???

Existing work-arounds:

- TA gives zero-knowledge proof
  - Inefficient, complex, impossible in UC model
- Other ad-hoc band-aids [KKLZ'06]
- Avoid homomorphic encryption altogether?

Insufficient Definitions

Homomorphic encryption provides:

Privacy: CPA security

Functionality: Homomorphic operations/features

Robustness: ???

Existing work-arounds:

- TA gives zero-knowledge proof
  - Inefficient, complex, impossible in UC model
- Other ad-hoc band-aids [KKLZ'06]
- Avoid homomorphic encryption altogether?

Can't we do better?

- Encryption scheme itself should have some robustness mechanism

Previous work

"Non-malleable homomorphic encryption" [PR08]

Formal definitions demanding that scheme is simultaneously:

- homomorphic with respect to certain operations
  - $\mathsf{Enc}(x) \rightsquigarrow \mathsf{Enc}(f(x))$, for specific f's.
- but non-malleable with respect to *all other operations*
  - Infeasible to make ciphertexts related in any other way

Previous work

"Non-malleable homomorphic encryption" [PR08]

Formal definitions demanding that scheme is simultaneously:

- homomorphic with respect to certain operations
  - $\mathsf{Enc}(x) \rightsquigarrow \mathsf{Enc}(f(x))$, for specific f's.
- but non-malleable with respect to *all other operations*
  - Infeasible to make ciphertexts related in any other way

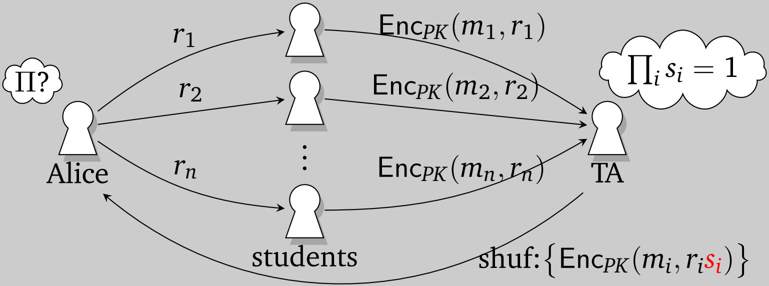
Parameterized Construction [PR08]

Non-malleable homomorphic scheme can be instantiated to support (only) operations:

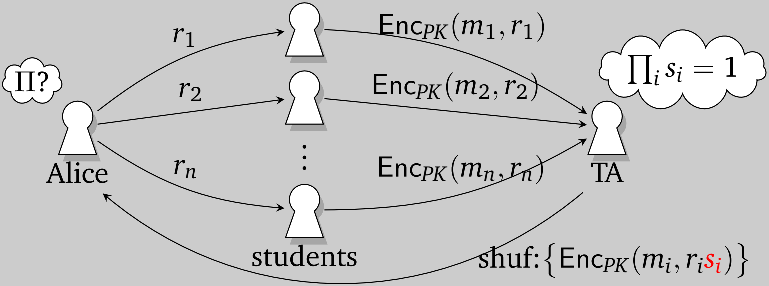
$$\mathsf{Enc}(m, r) \rightsquigarrow \mathsf{Enc}(m, rs)$$

where m, r, s are cyclic group elements.

Using construction



Using construction



Theorem

Teaching evaluations protocol is UC-secure when using the [PR08] encryption scheme, appropriately instantiated.

- [PR08] encryption secure under DDH assumption
- Protocol needs no additional hardness assumption

Implications

Encryption schemes can provide privacy, functionality, *and* robustness in a protocol.

- Intuitively simple protocols (minimal round complexity)
- Secure even in UC model (ZK proofs impossible)

Binary Homomorphic Operations

[PR08] scheme supports certain unary homomorphic operations

$$\mathsf{Enc}(x) \rightsquigarrow \mathsf{Enc}(f(x))$$

Binary/*n*-ary operations also useful, natural, e.g.:

$$\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n) \rightsquigarrow \mathsf{Enc}\left(\sum_i x_i\right)$$

Binary Homomorphic Operations

[PR08] scheme supports certain unary homomorphic operations

$$\mathsf{Enc}(x) \rightsquigarrow \mathsf{Enc}(f(x))$$

Binary/n-ary operations also useful, natural, e.g.:

$$\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n) \rightsquigarrow \mathsf{Enc}\left(\sum_i x_i\right)$$

Unfortunately,

Theorem (PR'08)

Impossible to achieve new notions of security, if any allowed f is a group operation.

What about relaxed requirements?

Relaxed requirements for homomorphic encryption:

- Ciphertexts have “length” parameter
- Length parameter increases when combining ciphertexts
- Ciphertexts leak length parameter but *nothing else*

What about relaxed requirements?

Relaxed requirements for homomorphic encryption:

- Ciphertexts have “length” parameter
- Length parameter increases when combining ciphertexts
- Ciphertexts leak length parameter but *nothing else*

Theorem (SYY99)

Given relaxed requirements, can construct homomorphic scheme supporting certain boolean operations:

- $\mathsf{Enc}(x;\ell), \mathsf{Enc}(y;\ell') \rightsquigarrow \mathsf{Enc}\Big(x \vee y; 8\max\{\ell,\ell'\}\Big)$
- $\mathsf{Enc}(x;\ell) \rightsquigarrow \mathsf{Enc}(\neg x;\ell)$

“Can evaluate any circuit on encrypted bits, with exponential blowup in length parameter”

Cryptocomputing

Cryptocomputing approach [SYY99]

- Encode X into several component ciphertexts:

$$\mathsf{Enc}(X;\ell) := (E(x_1),\ldots,E(x_\ell))$$

  where x_i's are some randomized encoding of X
- Use unary operations of E to manipulate encodings.

Cryptocomputing

Cryptocomputing approach [SYY99]

- Encode X into several component ciphertexts:

$$\mathsf{Enc}(X;\ell) := (E(x_1), \ldots, E(x_\ell))$$

  where x_i's are some randomized encoding of X
- Use unary operations of E to manipulate encodings.

"Use unary operations to achieve (relaxed) binary operations"

- [SYY99] considers only honest-but-curious adversaries
- Can we do something similar, but with robustness?

Overview of our result

SYY99	this work
any boolean operations on encrypted bits	group operation on encrypted group elements
length parameter increases exponentially	length parameter increases linearly
no "non-malleability" guarantee against malicious manipulation	malicious parties can manipulate ciphertexts no more than honest parties can

Recall: binary group operation impossible without some relaxation of requirements.

First attempt at binary group operation

First Idea

Encode X into random (multiplicative) sharing. I.e:

$$\mathsf{Enc}(X; n) := (E(x_1), \ldots, E(x_n))$$

where x_i random subject to $\prod_i x_i = X$.

First attempt at binary group operation

First Idea

Encode X into random (multiplicative) sharing. I.e:

$$\mathsf{Enc}(X; n) := (E(x_1), \ldots, E(x_n))$$

where x_i random subject to $\prod_i x_i = X$.

How to "multiply" $\mathsf{Enc}(X; n)$ and $\mathsf{Enc}(Y; m)$:

1. Concatenate encrypted shares

$$\Big(E(x_1), \quad \ldots, E(x_n), \quad E(y_1), \qquad \ldots, E(y_m) \qquad \Big)$$

First attempt at binary group operation

First Idea

Encode X into random (multiplicative) sharing. I.e:

$$\mathsf{Enc}(X; n) := (E(x_1), \ldots, E(x_n))$$

where x_i random subject to $\prod_i x_i = X$.

How to "multiply" $\mathsf{Enc}(X; n)$ and $\mathsf{Enc}(Y; m)$:

1. Concatenate encrypted shares
2. Use unary operations of E to re-randomize sharing

$$\Big(E(x_1 r_1), \ldots, E(x_n r_n), E(y_1 r_{n+1}), \ldots, E(y_m r_{n+m})\Big)$$

where r_i random subject to $\prod_i r_i = 1$.

First attempt at binary group operation

First Idea

Encode X into random (multiplicative) sharing. I.e:

$$\mathsf{Enc}(X; n) := (E(x_1), \ldots, E(x_n))$$

where x_i random subject to $\prod_i x_i = X$.

How to "multiply" $\mathsf{Enc}(X; n)$ and $\mathsf{Enc}(Y; m)$:

1. Concatenate encrypted shares
2. Use unary operations of E to re-randomize sharing

$$\Big(E(x_1 r_1), \ldots, E(x_n r_n), E(y_1 r_{n+1}), \ldots, E(y_m r_{n+m})\Big)$$

where r_i random subject to $\prod_i r_i = 1$.

Result is distributed as $\mathsf{Enc}(XY; n+m)$.

Binary group operation

Problem

Given $\mathsf{Enc}(X;\ell)$, can derive “shorter” related ciphertexts:

$$\mathsf{Enc}(X;\ell) := \Big(\underbrace{E(x_1),\ldots,E(x_k)}_{\text{legitimate } \mathsf{Enc}(S;k)},\underbrace{E(x_{k+1}),\ldots,E(x_\ell)}_{\text{legitimate } \mathsf{Enc}(T;\ell-k)}\Big)$$

where S, T unknown, but $ST = X$.

Want “non-malleability w.r.t. shorter ciphertexts”.

Binary group operation

Solution:

Encode X into two independent sharings (“top”, “bottom”)

$$\mathsf{Enc}(X; n) := \left(E\begin{pmatrix} x_1 \\ x'_1 \end{pmatrix}, \ldots, E\begin{pmatrix} x_n \\ x'_n \end{pmatrix} \right)$$

where x_i and x'_i random subject to $\prod_i x_i = \prod_i x'_i = X$.

Binary group operation

Solution:

Encode X into two independent sharings ("top", "bottom")

$$\mathsf{Enc}(X;n) := \left(E\binom{x_1}{x'_1}, \ldots, E\binom{x_n}{x'_n} \right)$$

where x_i and x'_i random subject to $\prod_i x_i = \prod_i x'_i = X$.

How to "multiply" $\mathsf{Enc}(X;n)$ and $\mathsf{Enc}(Y;m)$:

1. Concatenate encrypted share-pairs

$$\left(E\binom{x_1}{x'_1}, \quad \ldots, E\binom{x_n}{x'_n}, \quad E\binom{y_1}{y'_1}, \quad \ldots E\binom{y_m}{y'_m} \quad \right)$$

Binary group operation

Solution:

Encode X into two independent sharings ("top", "bottom")

$$\mathsf{Enc}(X;n) := \left(E\begin{pmatrix} x_1 \\ x'_1 \end{pmatrix}, \ldots, E\begin{pmatrix} x_n \\ x'_n \end{pmatrix} \right)$$

where x_i and x'_i random subject to $\prod_i x_i = \prod_i x'_i = X$.

How to "multiply" $\mathsf{Enc}(X;n)$ and $\mathsf{Enc}(Y;m)$:

1. Concatenate encrypted share-pairs
2. Use unary operations of E to re-randomize "top" and "bottom" sharings independently

$$\left(E\begin{pmatrix} x_1 r_1 \\ x'_1 r'_1 \end{pmatrix}, \ldots, E\begin{pmatrix} x_n r_n \\ x'_n r'_n \end{pmatrix}, E\begin{pmatrix} y_1 r_{n+1} \\ y'_1 r'_{n+1} \end{pmatrix}, \ldots E\begin{pmatrix} y_m r_{n+m} \\ y'_m r'_{n+m} \end{pmatrix} \right)$$

where r_i and r'_i random subject to $\prod_i r_i = \prod_i r'_i = 1$.

Binary operations

Previous "attack" thwarted:

$$\mathsf{Enc}(X;n) := \left(\underbrace{E\binom{x_1}{x'_1}, \ldots, E\binom{x_k}{x'_k}}_{\text{invalid}}, \underbrace{E\binom{x_{k+1}}{x'_{k+1}}, \ldots, E\binom{x_n}{x'_n}}_{\text{invalid}} \right)$$

"Top" and "bottom" sharings do not encode same value, with overwhelming probability.

Binary operations

Previous "attack" thwarted:

$$\mathsf{Enc}(X;n) := \left(\underbrace{E\binom{x_1}{x_1'}, \ldots, E\binom{x_k}{x_k'}}_{\text{invalid}}, \underbrace{E\binom{x_{k+1}}{x_{k+1}'}, \ldots, E\binom{x_n}{x_n'}}_{\text{invalid}} \right)$$

"Top" and "bottom" sharings do not encode same value, with overwhelming probability.

Still other unwanted malleabilities?

- maybe split each x_i from x_i'?

Our results

Theorem

If component scheme is non-malleable & homomorphic with respect to $E(x,x') \rightsquigarrow E(xr, x'r')$, then compound scheme is non-malleable & homomorphic with respect to operations:

- $\mathsf{Enc}(X;\ell), \mathsf{Enc}(Y;\ell') \rightsquigarrow \mathsf{Enc}(XY;\ell+\ell')$, *and*
- $\mathsf{Enc}(X;\ell) \rightsquigarrow \mathsf{Enc}(rX;\ell)$

Our results

Theorem

If component scheme is non-malleable & homomorphic with respect to $E(x, x') \rightsquigarrow E(xr, x'r')$, then compound scheme is non-malleable & homomorphic with respect to operations:

- $\mathsf{Enc}(X; \ell), \mathsf{Enc}(Y; \ell') \rightsquigarrow \mathsf{Enc}(XY; \ell + \ell')$, *and*
- $\mathsf{Enc}(X; \ell) \rightsquigarrow \mathsf{Enc}(rX; \ell)$

Observations:

- [PR08] scheme can be so instantiated under DDH assumption.

Our results

Theorem

If component scheme is non-malleable & homomorphic with respect to $E(x, x') \rightsquigarrow E(xr, x'r')$, then compound scheme is non-malleable & homomorphic with respect to operations:

- $\mathsf{Enc}(X;\ell), \mathsf{Enc}(Y;\ell') \rightsquigarrow \mathsf{Enc}(XY;\ell+\ell')$, *and*
- $\mathsf{Enc}(X;\ell) \rightsquigarrow \mathsf{Enc}(rX;\ell)$

Observations:

- [PR08] scheme can be so instantiated under DDH assumption.
- Crucially use non-malleability of component scheme
  - "top" share can't be separated from its "bottom" partner

Conclusion

Take-home message:

With appropriate requirements on an encryption scheme, can achieve robust computation on encrypted data

- Intuitively simple protocols (no ZK!)
- Security even in UC model against malicious adversaries

Open questions

Open questions:

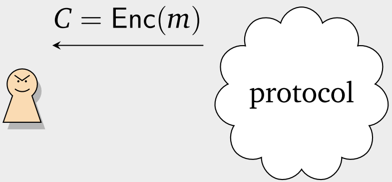
- Weaker requirements that provide robustness in protocols?
  - Existing definitions very strong, thus
  - Existing non-malleable homomorphic encryption construction is complicated
- Support/applications for other kinds of unary homomorphic operations?
- Can we achieve more robust cryptocomputing?
  - Both binary operations in ring (not just one group)?
  - [SYY99] do, but without robustness
  - Their encoding too fragile to immediately work, even with strong component encryption

Thanks for your attention! Any questions?

fin.

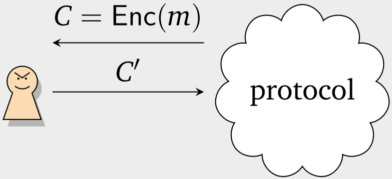
How to Define "Non-malleable Homomorphic"

Essence of a malleability attack:



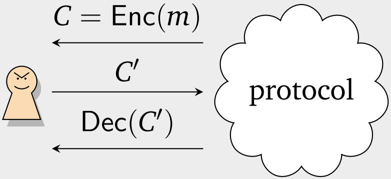
How to Define “Non-malleable Homomorphic”

Essence of a malleability attack:



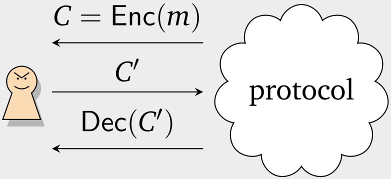
How to Define "Non-malleable Homomorphic"

Essence of a malleability attack:



How to Define "Non-malleable Homomorphic"

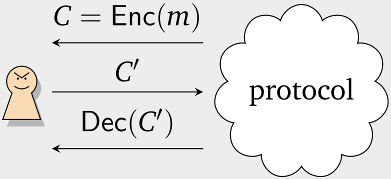
Essence of a malleability attack:



Non-malleability defined in terms of this kind of interaction

How to Define "Non-malleable Homomorphic"

Essence of a malleability attack:



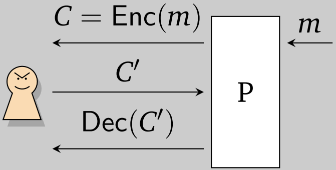
Non-malleability defined in terms of this kind of interaction

Now, it's legitimate if C' related to C in certain ways.

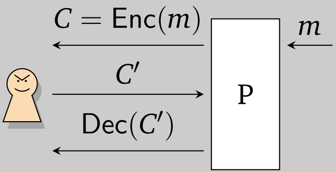
Idea:

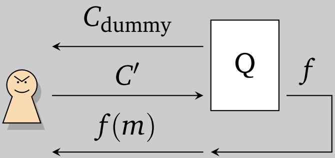
Design a game where adversary wins by making bad related ciphertexts.

Games Cryptographers Play (simplified)

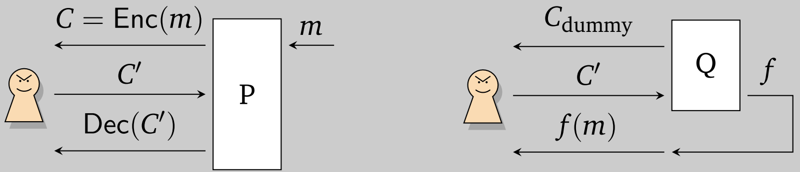


Games Cryptographers Play (simplified)





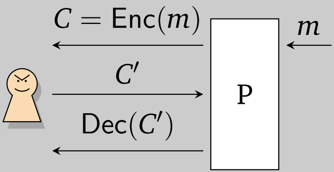
Games Cryptographers Play (simplified)

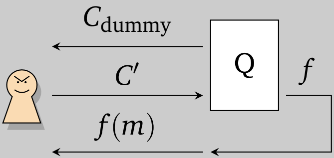


Adversary's goal: Determine whether talking to P or Q.

- Distinguish C_{dummy} from $\mathsf{Enc}(m)$
- Generate confounding C'

Games Cryptographers Play (simplified)





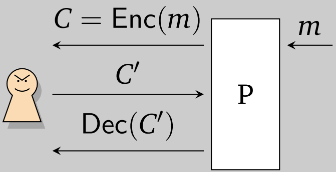
Adversary's goal: Determine whether talking to P or Q.

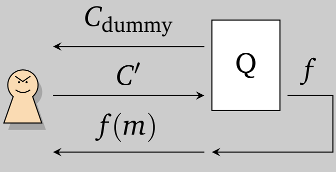
- Distinguish C_{dummy} from $\mathsf{Enc}(m)$
- Generate confounding C'

Q's goal: Make two worlds look indistinguishable.

- Generate C_{dummy} that looks like $\mathsf{Enc}(m)$
- Determine how C' related.

Games Cryptographers Play (simplified)





Adversary's goal: Determine whether talking to P or Q.

- Distinguish C_{dummy} from $\mathsf{Enc}(m)$
- Generate confounding C'

Q's goal: Make two worlds look indistinguishable.

- Generate C_{dummy} that looks like $\mathsf{Enc}(m)$
- Determine how C' related.

Observation

If Q restricted to output $\{f_1, \ldots, f_n\}$, then adversary can always win by making C' related to C/C_{dummy} in some other way.

The "Right" Definition

Contrapositive

If Q restricted to output $\{f_1, \ldots, f_n\}$, but adversary still can't win, then it must be impossible to make C' related to C/C_{dummy} in some other way.

The "Right" Definition

Contrapositive

If Q restricted to output $\{f_1, \ldots, f_n\}$, but adversary still can't win, then it must be impossible to make C' related to C/C_{dummy} in some other way.

New Security Definition [PR '08]

Scheme is non-malleable except for operations $\{f_1, \ldots, f_n\}$ if there is a strategy Q that only outputs $f \in \{f_1, \ldots, f_n\}$, and no adversary ever wins.