CRYPTOGRAPHIC PROTOCOLS
2016, LECTURE 5

TWO-MESSAGE HOMOMORPHIC PROTOCOLS

Helger Lipmaa
University of Tartu, Estonia
UP TO NOW

- Assumptions, reductions
  - DL, CDH, DDH
- Key exchange
- Idea of secure computation
- Modularization, Elgamal, malleability
This time:

- secure computation
- Concrete two-message protocols
  - Some of them are toy, some are useful

Note: here Enc will denote lifted Elgamal
FUNCTIONALITY

\[ a \in S_a \quad \text{and} \quad b \in S_b \]

\[ f_a(a, b) \]

Goal
IDEAL MODEL

\[ a \in S_a \quad TTP \quad b \in S_b \]

\[ f_a(a, b) \]

Goal
REAL MODEL

$\forall a \in S_a \quad \forall b \in S_b$

$f_a(a, b)$

Protocol

Goal

Tool

Usually correct output only guaranteed when inputs come from correct sets.
IDEAL MODEL: VETO

\[ a \in \{0,1\} \]

\[ b \in \{0,1\} \]

TTP

\[ a \& b \]
REAL MODEL: VETO

\[ a \in \{0,1\} \]

\[ b \in \{0,1\} \]

\[ a \land b \]

Output undefined when \( a \) or \( b \) is not Boolean
FUNCTIONALITY VS PROTOCOL

- $(a, a \& b)$ leaks something about $b$
  - If $a = 1$ then $a \& b = b$
- Problem of **functionality**, not of protocol
- If leaking not desired, redefine functionality!
- Some functionalities just do not exist
- **Goal**: functionality
- **Tool**: cryptography

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$a &amp; b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
IDEA: LINEARIZATION

- Lifted Elgamal enables to compute linear functions

- **Linearization:**
  - transform given non-linear function $f$ into a linear function $f^*$ that agrees with $f$ on the restricted input set

- **Example:** $a \& b = a \cdot b$ for $a, b \in \{0, 1\}$
VETO PROTOCOL

\[ r \leftarrow \mathbb{Z}_q \]
\[ c \leftarrow \text{Enc}(a; r) \]
\[ d \leftarrow c^b \]
\[ M \leftarrow \text{Dec}(d) \]
**VETO: CORRECTNESS**

- **pk, sk**
- $a \in \{0, 1\}$
- $M \leftarrow \text{Dec}(d)$
- Correctness: $ok$
- $d \leftarrow c^b$
- $d = \text{Enc}(ab; rb)$
- $M = ab$
- Decryption succeeds, since $ab$ is small when $a, b \in \{0, 1\}$
VETO: ALICE'S PRIVACY

VETO: ALICE'S PRIVACY

Alice's privacy: ok, Bob sees only random ciphertext

pk, sk \( a \in \{0, 1\} \)

\[ c \leftarrow \text{Enc}(a; r) \]

\[ d \leftarrow c^b \]

\[ M \leftarrow \text{Dec}(d) \]

\[ d = \text{Enc}(ab; rb) \]

\[ M = ab \]
QUIZ: BOB’S PRIVACY

\[ \begin{align*}
\text{pk, sk} & \quad a \in \{0, 1\} \\
\text{M} & \leftarrow \text{Dec}(d) \\

r & \leftarrow \mathbb{Z}_q \\
c & \leftarrow \text{Enc}(a; r) \\
d & \leftarrow c^b \\
M & \leftarrow \text{Dec}(d) \\
d & = \text{Enc}(ab; rb) \\
M & = ab
\end{align*} \]

Bob's privacy: Ok?

Might leak information (consider \( b = 0 \))
BLINDING PROPERTY

- \( \text{Enc}(m; r) \cdot \text{Enc}(0; r') = \text{Enc}(m; r + r') \)

- follows from the definition of (lifted) Elgamal

- \( r' \) random => \( r + r' \) random for any \( r \)

- **Blinding property:**

  - given **any** encryption of (unknown) \( m \) => can compute **random** encryption of \( m \)
The protocol proceeds as follows:

1. Alice chooses a random $r \in \mathbb{Z}_q$ and computes $c = \text{Enc}(a; r)$.
2. Bob chooses a random $r' \in \mathbb{Z}_q$ and computes $d = c^b \cdot \text{Enc}(0; r')$.
3. Alice decodes $d$ to obtain $M = \text{Dec}(d)$.

Correctness: $d = \text{Enc}(ab; \ldots + r')$ and $M = ab$.

Random since $r'$ is random.

Alice's privacy: Ok, Bob sees only ciphertext.

Bob's privacy: Ok, Alice sees random encryption of intended output.

pk, sk

pk

a

b

Random since $r'$ is random

Correctness: ok
FUNCTIONALITY: SCALAR PRODUCT

\[ a \in \{0,1\}^L \]
\[ a = \{a_i : i \in [n]\} \]

\[ b \in \{0,1\}^L \]
\[ b = \{b_i : i \in [n]\} \]

\[ <a,b> = \sum a_i b_i \]
SCALAR PRODUCT PROTOCOL

\[ pk, sk \]
\[ a_i \in \{0, 1\}^L \]
\[ a = \{a_i : i \in [n]\} \]

\[ M \leftarrow \text{Dec}(d) \]

\[ \text{for } i = 1 \text{ to } n \]
\[ r_i \leftarrow \mathbb{Z}_q \]
\[ c_i \leftarrow \text{Enc}(a_i; r_i) \]

\[ \{c_i\} \]

\[ r' \leftarrow \mathbb{Z}_q \]
\[ d \leftarrow \prod c_i b_i \cdot \text{Enc}(0; r') \]

\[ \text{pk} \]
\[ b_i \in \{0, 1\}^L \]
\[ b = \{b_i : i \in [n]\} \]
SCALAR PRODUCT PROTOCOL

pk, sk

\( a_i \in \{0,1\}^L \)
\( a = \{a_i : i \in [n]\} \)

pk

\( b_i \in \{0,1\}^L \)
\( b = \{b_i : i \in [n]\} \)

Alice's privacy: ok, Bob sees only ciphertexts
Bob's privacy: ok, Alice sees random encryption of intended output

Correctness: ok

for \( i = 1 \) to \( n \)
\( r_i \leftarrow \mathbb{Z}_q \)
\( c_i \leftarrow \text{Enc}(a_i ; r_i) \)

\( r' \leftarrow \mathbb{Z}_q \)
\( d \leftarrow \prod c_i b_i \cdot \text{Enc}(0 ; r') \)

\( M \leftarrow \text{Dec}(d) \)

\( d = \text{Enc}(\Sigma a_i b_i, \ldots + r') \)
\( M = \Sigma a_i b_i \)

Decryption succeeds when \( \Sigma a_i b_i \leq n2^{2L} \) is small, e.g., \( n2^{2L} < 2^{40} \)
MORE FUN...

- Veto and scalar product are "linear" functions
  - ... thus straightforward to implement by using lifted Elgamal
- It comes out we can also implement less straightforward functionalities
FUNCTIONALITY: HAMMING DISTANCE

\[ a \in \{0,1\}^n \]

\[ b \in \{0,1\}^n \]

\[ \delta(a, b) := |\{i : a_i \neq b_i\}| \]

Does not seem to be "linear" at all???
QUIZ: LINEARIZATION OF HD

Note $a_i$ and $b_i$ are Boolean!

$a_i \neq b_i$ iff $a_i \text{XOR} b_i = 1$

Moreover, Bob knows $b_i$

$x \text{ XOR } 0 = x = 0 + (1 - 2 \cdot 0) x$

$x \text{ XOR } 1 = 1 - x = 1 + (1 - 2 \cdot 1) x$

$x \text{ XOR } y = y + (1 - 2 \cdot y) x$

$\delta (a, b) = \Sigma_i (b_i + (1 - 2b_i) a_i) = \Sigma_i (1 - 2b_i) a_i + \Sigma_i b_i$

$\delta$ is “linear” for correct inputs and thus we can construct efficient 2MAH protocol for Hamming distance
pk, sk \ a \in \{0, 1\}^n

for i = 1 to n

\[ r_i \leftarrow \mathbb{Z}_q \]
\[ c_i \leftarrow \text{Enc}(a_i; r_i) \]

\( \{c_i\} \)

\[ r' \leftarrow \mathbb{Z}_q \]
\[ d \leftarrow \prod_i c_i^{1 - 2b_i} \cdot \text{Enc}(\Sigma_i b_i; r') \]

M \leftarrow \text{Dec}(d)

pk \ b \in \{0, 1\}^n
Protocol

\begin{align*}
\text{pk, sk} & \quad a \in \{0, 1\}^n \\
M & \leftarrow \text{Dec}(d) \\
\text{Correctness: ok}
\end{align*}

for \( i = 1 \) to \( n \)
\begin{align*}
\text{ri} & \leftarrow \mathbb{Z}_q \\
c_i & \leftarrow \text{Enc}(a_i; r_i)
\end{align*}

\begin{align*}
\text{pk} & \quad b \in \{0, 1\}^n \\
M' & \leftarrow \text{Dec}(d) \\
\text{Decryption succeeds when } & \delta(a, b) \in \{0, \ldots, n\} \text{ is small, e.g., } n < 2^{40}
\end{align*}

Bob's privacy: ok, Alice sees random encryption of intended output

Alice's privacy: ok, Bob sees only ciphertexts
REAL PROTOCOL: CPIR

- Assume any privacy-preserving database application

- **Most trivial operation**: Alice queries one element from Bob's database

  - subprotocol in myriad other protocols

- How to do it so that Bob has no clue which element was obtained?

- **simplest case**: Bob's database has two elements
(2,1)-CPIR

Does not seem to be "linear" at all???
Note $x \in \{0, 1\}$

Moreover, Bob knows $f_0$ and $f_1$

$f_x = f_0 \cdot [x = 0] + f_1 \cdot [x = 1]$  

$f_x = f_0 \cdot (1 - x) + f_1 \cdot x = f_0 + (f_1 - f_0) \cdot x$

"linear" and thus we can construct 2MAH for (2, 1)-CPIR
\((2, 1)\)-CPIR PROTOCOL

- \(pk, sk\)
- \(x \in \{0, 1\}\)
- \(r \leftarrow \mathbb{Z}_q\)
- \(c \leftarrow \text{Enc}(x; r)\)
- \(M \leftarrow \text{Dec}(d)\)
- \(f = (f_0, f_1)\)
- \(f_i \in \{0, 1\}^L\)
**Protocol:**

- **pk, sk**
- \( x \in \{0, 1\} \)
- **M** ← Dec(\( d \))
- Correctness: ok

**CPIR: SECURITY**

- Bob's privacy: ok, Alice sees random encryption of intended output
- Alice's privacy: ok, Bob sees only ciphertext

**pk, sk**

- \( r \leftarrow \mathbb{Z}_q \)
- \( c \leftarrow \text{Enc}(x; r) \)

**pk**

- \( f = (f_0, f_1) \)
- \( f_i \in \{0, 1\}^L \)

**Protocol**

- \( r' \leftarrow \mathbb{Z}_q \)
- \( d \leftarrow c f_1 - f_0 \cdot \text{Enc}(f_0; r') \)

**d**

- \( d = \text{Enc}(f_x; ... + r') \)
- \( M = f_x \)

**Decryption succeeds when** \( f_x \leq 2^L \) **is small, e.g.,** \( 2^L < 2^{40} \)
Query($a$):
for $i = 1$ to $n$
\[ r_i \leftarrow \mathbb{Z}_q \]
\[ a_i \leftarrow f_i(a,...) \]
\[ c_i \leftarrow \text{Enc}(a_i; r_i) \]

Choose random $r'$
// $n'$ values $d_j$
\[ \{d_j\} \leftarrow \text{Reply}(b, \{c_i\}; r') \]

for $j = 1$ to $n'$
\[ M_j \leftarrow \text{Dec}(d_j) \]
\[ M \leftarrow \text{Answer}(a, \{M_j\},...) \]
ON EFFICIENCY

❖ **Alice:**
- \(n\) encryptions + \(n'\) decryptions
- that is, \((3n + \Theta(n'))\) exponentiations + \(n'\) DL-s

❖ **Bob:**
- depends very much on protocol
- Usually \(\Theta(n + n')\) exponentiations

Since exponentiation takes \(\Theta(L^{2.58})\) bit-ops by using Karatsuba multiplication, and DL takes \(\Theta(2^{L/2})\) bit-ops, for large \(L\), DL is the bottleneck
"Bitwise\" Tricks

- **DL timing:** $\Theta(n' 2^{L/2})$ bit-ops /* linear in $n'$ but exponential in $L*/

- **Common trick:**
  - let Alice encrypt every bit separately, and then construct protocol so that every final ciphertext also encrypts a bit
  
  - $\Theta(L)$ times more DL-s, but each DL gets "1-bit\" input

- **Pro:** DL-s dominated by $\Theta(n'L)$ bit-ops

- **Con:** comm. and Bob's comp. increased by factor of $\Theta(L)$

- Similarly, can handle Bob's inputs bitwise
BITWISE $(2, 1)$-CPIR

**Protocol**

\[ \{d_i\} \ (i = 1 \text{ to } L) \]

**for** \(i = 1 \text{ to } L:\)

\[ r_i' \leftarrow \mathbb{Z}_q \]

\[ y_i \leftarrow f_{i_1} - f_{i_0} \]

\[ d_i \leftarrow c y_i \cdot \text{Enc} (f_{i_0}; r_i') \]

**pk, sk**

\( x \in \{0, 1\} \)

**for** \(i = 1 \text{ to } L:\)

\[ M_i \leftarrow \text{Dec} (d_i) \]

\( M \leftarrow (M_1, \ldots, M_L) \)

**pk**

\( f = (f_0, f_1) \)

\( f_i \in \{0, 1\}^L \)
## CPIR Complexity

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Communication (group elem.)</th>
<th>Alice's comput. (exp, DL)</th>
<th>Bob's comput. (exp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>First protocol</td>
<td>4</td>
<td>3 exp + 1 · DL (L bits)</td>
<td>5</td>
</tr>
<tr>
<td>Bitwise protocol</td>
<td>2L + 2</td>
<td>(L + 2) exp + L·DL(1 bits)</td>
<td>2L</td>
</tr>
</tbody>
</table>

- Different trade-offs possible
- Computing \(g^b\) is for free, for bit \(b\)
CPIR: ALICE'S COMP.

Numbers are approximate, remember the slope.
Bitwise execution:
- logical and (more) efficient solution for CPIR
- In other protocols, not so easy
  - Millionaire: output bit depends on all input bits
  - Lifted Elgamal-based millionaire protocol with >40-dimensional vectors is way too slow
- Need other solutions
ON ART OF PROTOCOL DESIGN

- Figure out how much resources you have
  - Communication, Alice's and Bob's computation
- Design a protocol that achieves a good tradeoff
- Also: efficiency vs assumption
ABOUT SECURITY

- Currently: **semihonest model**
  - Alice and Bob follow protocol but "eavesdrop"
- **Malicious model**
  - Later lectures
  - Given protocol secure in semihonest model, add zero knowledge proofs
The protocols are "designed" to be secure

Correctness (semihonest model)
- Follows from protocol design

Alice's privacy (semihonest model)
- Bob only sees some random ciphertexts, thus follows from the IND-CPA security of the cryptosystem

Bob's privacy (semihonest model)
- Alice only sees random encryptions of intended output(s)
MORE ON SECURITY

- **Alice's privacy:**
  - Defined like IND-CPA for public-key encryption
  - Bob cannot distinguish between Alice's messages corresponding to Alice's any two inputs $a_0$ and $a_1$

- **Bob's privacy:**
  - Alice should obtain *no information* about Bob's input $b$, except what is obvious from her input $a$ and correct output $f_a(a, b)$
**IND-CPA Security of Protocols**

- \( \Pi = (\text{Setup}, \text{Query}, \text{Reply}, \text{Answer}) \)
- \( \text{Adv[IND]} := | \text{Pr[IND} = 1] - 1/2 | \)
- An \( \varepsilon \)-breaks IND-CPA security of \( \Pi \) iff \( \text{Adv[IND]} \geq \varepsilon \)
- \( \Pi \) is \((\tau, \varepsilon)\)-IND-CPA secure iff no PPT adversary \( \varepsilon \)-breaks IND-CPA security of \( \Pi \) in time \( \leq \tau \)
- \( \Pi \) is IND-CPA secure iff it is \((\text{poly}(\kappa), \text{negl}(\kappa))\)-IND-CPA secure

**Game IND(1^\kappa, \Pi, A)**

- \( gk \leftarrow \text{Setup}(1^\kappa) \)
- \( (sk, pk) \leftarrow \text{Keygen}(gk) \)
- \( (m_0, m_1) \leftarrow A(gk, pk) \)
- \( \beta \leftarrow \{0, 1\} \)
- \( r \leftarrow R_\Pi \)
- \( c \leftarrow \text{Query}_{pk}(m_\beta, r) \)
- \( \beta^* \leftarrow A(gk, pk, c) \)
- Return \( \beta = \beta^* \oplus 1 \)

Here \( m_i \) are two possible Alice’s inputs.
IND-CPA SECURITY PROOFS

- IND-CPA security proofs of all 2MAH protocols are tautologies

- Since Elgamal is IND-CPA secure, and Bob only sees random ciphertexts, the protocol is IND-CPA secure
BOB'S PRIVACY PROOFS

- Also tautology
  
  - if the protocol is well constructed

- Alice only sees random encryption of the intended output

- Thus even if Alice is omnipotent, Alice can only recover intended output
STUDY OUTCOMES

- Functionality vs protocol
- 2MAH protocols, examples
- Different efficiency aspects
  - DL dominates when outputs get longer...
  - Tricks (some of them!): linearization, bitwise
- Security of 2MAH protocols
How to construct protocols with large outputs

- without "bit-wising"

- Trapdoor discrete logarithm

- Paillier cryptosystem