UP TO NOW
UP TO NOW

- Assumptions, reductions
UP TO NOW

- Assumptions, reductions
- DL, CDH, DDH
UP TO NOW

- Assumptions, reductions
  - DL, CDH, DDH
- Key exchange
UP TO NOW

✧ Assumptions, reductions
   ✧ DL, CDH, DDH

✧ Key exchange

✧ Idea of secure computation
UP TO NOW

- Assumptions, reductions
  - DL, CDH, DDH
- Key exchange
- Idea of secure computation
- Modularization, Elgamal, malleability
THIS TIME
THIS TIME

- This time: secure computation
This time:

- **This time:** secure computation
- Concrete two-message protocols
This time: secure computation

Concrete two-message protocols

Some of them are toy, some are useful
This time: secure computation

Concrete two-message protocols

Some of them are toy, some are useful

Note: here Enc will denote lifted Elgamal
FUNCTIONALITY

\[ a \in S_a \]

\[ b \in S_b \]
FUNCTIONALITY

\[ a \in S_a \]

\[ b \in S_b \]
FUNCTIONALITY

\[ a \in S_a \]

\[ b \in S_b \]

\[ f_a(a, b) \]

Goal
IDEAL MODEL

\[ a \in S_a \]

\[ b \in S_b \]
IDEAL MODEL

\[ a \in S_a \quad \text{TTP} \quad b \in S_b \]
IDEAL MODEL

\[ a \in S_a \overset{a}{\rightarrow} TTP \overset{b}{\rightarrow} b \in S_b \]
IDEAL MODEL

\[ a \in S_a \] \rightarrow \text{TTP} \rightarrow \text{b} \in S_b
IDEAL MODEL

\[ a \in S_a \]

\[ b \in S_b \]

\[ f_a(a, b) \]

Goal
REAL MODEL

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

\[ f_a(a, b) \]

Protocol

Goal
REAL MODEL

\[ a \in S_a \]

\[ b \in S_b \]

\[ f_a(a, b) \]

Protocol

Goal

Tool
A protocol is a protocol ... usually correct output only guaranteed when inputs come from correct sets (semihonest model: we assume parties follow the protocol).
IDEAL MODEL: VETO

\[ a \in \{0,1\} \quad \text{and} \quad b \in \{0,1\} \]

\[ \text{TTP} \]

\[ a \quad \text{and} \quad b \]
REAL MODEL: VETO

$a \in \{0,1\}$

$b \in \{0,1\}$

$a \land b$

Protocol
REAL MODEL: VETO

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

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

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

✧ \((a, a \& b)\) leaks something about \(b\)
FUNCTIONALITY VS PROTOCOL

✧ \((a, a \& b)\) leaks something about \(b\)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>
FUNCTIONALITY VS PROTOCOL

\[ (a, a \& b) \text{ leaks something about } b \]
FUNCTIONALITY VS PROTOCOL

- \((a, a \& b)\) leaks something about \(b\)
  - If \(a = 1\) then \(a \& b = b\)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(a &amp; b)</td>
</tr>
<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>
FUNCTIONALITY VS PROTOCOL

- \((a, a \& b)\) leaks something about \(b\)
- If \(a = 1\) then \(a \& b = b\)
- Problem of **functionality**, not of protocol

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>O</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>O</td>
<td>I</td>
<td>O</td>
</tr>
<tr>
<td>b</td>
<td>I</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>a&amp;b</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>
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!

<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>
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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>
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
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**
IDEA: LINEARIZATION

Recall: lifted Elgamal enables to compute affine functions
IDEA: LINEARIZATION

- Recall: lifted Elgamal enables to compute affine functions
  - **Affine** in the inputs of **one** party
IDEA: LINEARIZATION

- Recall: lifted Elgamal enables to compute affine functions
  - **Affine** in the inputs of one party
- **Linearization:**
IDEA: LINEARIZATION

- Recall: lifted Elgamal enables to compute affine functions
  - **Affine** in the inputs of **one** party
- **Linearization:**
  - transform given **non-linear** function $f$ into a **linear/affine** (in inputs of one party) function $f$
  - that agrees with $f$ on the restricted input set
IDEA: LINEARIZATION

- Recall: lifted Elgamal enables to compute affine functions
  - **Affine** in the inputs of **one** party

- **Linearization:**
  - transform given **non-linear** function \( f \) into a **linear/affine** (in inputs of one party) 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

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

\[ pk \]
\[ b \in \{0, 1\} \]
VETO PROTOCOL

pk, sk
a ∈ \{0, 1\}

pk
b ∈ \{0, 1\}

r ← \mathbb{Z}_q

c ← Enc(a; r)
VETO PROTOCOL

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

pk b \in \{0, 1\}

r \leftarrow \mathbb{Z}_q

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

\ c
The VETO PROTOCOL is illustrated in the diagram. It involves the following steps:

1. Choose a random value \( r \leftarrow \mathbb{Z}_q \).
2. Compute \( c \leftarrow \text{Enc}(a; r) \).
3. Compute \( d \leftarrow c^b \).

The protocol is initiated with the public key \( pk \) and secret key \( sk \), where \( a \in \{0, 1\} \). Both parties compute and exchange \( c \) and \( d \) as part of the protocol's execution.
VETO PROTOCOL

\[ \begin{align*}
    &a \in \{0,1\} \\
    &pk, sk \\
    &r \leftarrow \mathbb{Z}_q \\
    &c \leftarrow \text{Enc}(a; r) \\
    &c \leftarrow \text{Enc}(a; r) \\
    &d \leftarrow c^b \\
    &d \leftarrow c^b \\
\end{align*} \]
VETO PROTOCOL

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

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

\[ d \leftarrow c^b \]

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

\[ pk, b \in \{0, 1\} \]
VETO: CORRECTNESS

\[ \text{pk, sk} \quad a \in \{0, 1\} \]

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

\[ c \]

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

\[ d \leftarrow c^b \]

\[ \text{pk} \quad b \in \{0, 1\} \]
VETO: CORRECTNESS

\[ \begin{align*}
\text{pk, sk} & \quad a \in \{0, 1\} \\
\text{pk} & \quad b \in \{0, 1\} \\
\end{align*} \]

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

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

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

\[ d \leftarrow c^b \]
**VETO: CORRECTNESS**

\[ \text{pk, sk} \]
\[ a \in \{0, 1\} \]
\[ M \leftarrow \text{Dec}(d) \]
\[ c \leftarrow \text{Enc}(a; r) \]

\[ r \leftarrow \mathbb{Z}_q \]

\[ d \leftarrow c^b \]

Correctness: \( \text{ok} \)

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

Decryption succeeds, since \( ab \) is small when \( a, b \in \{0, 1\} \)
VETO: ALICE'S PRIVACY

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

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

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

Alice's privacy: ok, Bob sees only random ciphertext
QUIZ: BOB’S PRIVACY

Bob's privacy: Ok?

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

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

\( r \leftarrow \mathbb{Z}_q \)

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

\( d \leftarrow c^b \)

\( c \)

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

\( M = ab \)

\( M \leftarrow \text{Dec}(d) \)
QUIZ: BOB'S PRIVACY

\[
\begin{align*}
pk, sk & \quad a \in \{0, 1\} \\
pk & \quad b \in \{0, 1\} \\
\text{Bob's privacy: } & \quad \text{Ok?}
\end{align*}
\]

\[
\begin{align*}
r & \leftarrow \mathbb{Z}_q \\
c & \leftarrow \text{Enc}(a; r) \\
d & \leftarrow c^b \\
c & \leftarrow \text{Enc}(a; r) \\
M & \leftarrow \text{Dec}(d) \\
d & = \text{Enc}(ab; rb) \\
M & = ab
\end{align*}
\]
QUIZ: BOB’S PRIVACY

pk, sk
a ∈ {0, 1}

pk
b ∈ {0, 1}

Bob's privacy: Ok?

Leaks information (consider b = 0)

pk, sk
a ∈ {0, 1}

c ← Enc(a; r)

M ← Dec(d)

c

d ← c^b

d = Enc(ab; rb)

M = ab
BLINDING PROPERTY
BLINDING PROPERTY

\[ \text{Enc}(m; r) \cdot \text{Enc}(0; r') = \text{Enc}(m; r + r') \]
BLINDING PROPERTY

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

follows from the definition of (lifted) Elgamal:
BLINDING PROPERTY

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

follows from the definition of (lifted) Elgamal:

\[ (g^{mh}, g^r) \cdot (g^0, g^{r'}) = (g^{mh+r}, g^{r+r'}) \]
BLINDING PROPERTY

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

  follows from the definition of (lifted) Elgamal:

  - $(g^{mb^r}, g^r) \cdot (g^{0b^{r'}, g^{r'}}) = (g^{mb^{r+r'}}, g^{r+r'})$

  - $r'$ random $\Rightarrow r + r'$ random for any $r$
**BLINDING PROPERTY**

- $\text{Enc}(m; r) \cdot \text{Enc}(0; r') = \text{Enc}(m; r + r')$
  - follows from the definition of (lifted) Elgamal:
    - $(g^{mb}r, g^r) \cdot (g^{0b}r', g^{r'}) = (g^{mb}r + r', g^{r + r'})$
  - $r'$ random $\Rightarrow$ $r + r'$ random for any $r$

- **Blinding property:**
BLINDING PROPERTY

- \( \text{Enc}(m; r) \cdot \text{Enc}(0; r') = \text{Enc}(m; r + r') \)
  - follows from the definition of (lifted) Elgamal:
    - \( (g^{mbr}, gr) \cdot (g^{0br'}, gr') = (g^{mbr+r'}, gr+r') \)
  - \( r' \) random \( \Rightarrow r + r' \) random for any \( r \)

- **Blinding property:**
  - given any encryption of (unknown) \( m \) \( \Rightarrow \) can compute random encryption of \( m \)
VETO PROTOCOL

$pk, sk \quad a \quad Protocol \quad b$
VETO PROTOCOL

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

\[ pk, sk \]
\[ a \]

\[ pk \]
\[ b \]
Protocol

VETO PROTOCOL

\[ \begin{align*}
    r & \leftarrow \mathbb{Z}_q \\
    c & \leftarrow \text{Enc}(a; r)
\end{align*} \]
VETO PROTOCOL

- $r \leftarrow \mathbb{Z}_q$
- $c \leftarrow \text{Enc}(a; r)$
- $r' \leftarrow \mathbb{Z}_q$
- $d \leftarrow c^b \cdot \text{Enc}(0; r')$

Protocol

$pk, sk$

$a$

$pk$

$b$
VETO PROTOCOL

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

\[ r' \leftarrow \mathbb{Z}_q \]
\[ d \leftarrow c^b \cdot \text{Enc}(0; r') \]
**Protocol**

\[ r \leftarrow \mathbb{Z}_q \]

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

\[ d \leftarrow c^b \cdot \text{Enc}(0; r') \]

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

$r \leftarrow \mathbb{Z}_q$
c$\leftarrow$ Enc($a$; $r$)

$c$ $\leftarrow$ Enc($a$; $r$)

$d \leftarrow c^b \cdot$ Enc($0$; $r'$)

d = Enc($ab$; ... + $r'$)

pk, sk

pk

M$\leftarrow$ Dec($d$)

a

b
**VETO PROTOCOL**

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

\[ r' \leftarrow \mathbb{Z}_q \]
\[ d \leftarrow c^b \cdot \text{Enc}(0; r') \]

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

\[ d = \text{Enc}(ab; \ldots + r') \]
\[ M = ab \]

**Correctness:** ok
**VETO PROTOCOL**

- **pk, sk**
  - $a$

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

- $r' \leftarrow \mathbb{Z}_q$
  - $d \leftarrow c^b \cdot \text{Enc}(0; r')$

- $M \leftarrow \text{Dec}(d)$
  - $d = \text{Enc}(ab; ... + r')$
  - $M = ab$

**Correctness:** ok

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

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

**Correctness:** ok

1. \( r \leftarrow \mathbb{Z}_q \)
2. \( c \leftarrow \text{Enc}(a; r) \)
3. \( d \leftarrow c^b \cdot \text{Enc}(0; r') \)
4. \( M \leftarrow \text{Dec}(d) \)

**Proof:**

\( M = ab \)
VETO PROTOCOL

```
\begin{align*}
    r & \leftarrow \mathbb{Z}_q \\
    c & \leftarrow \text{Enc}(a; r) \\
    d & \leftarrow c^b \cdot \text{Enc}(0; r') \\
    M & \leftarrow \text{Dec}(d)
\end{align*}
```

- **Correctness:** ok
- **Alice's privacy:** ok, Bob sees only random ciphertext
- **Random since $r'$ is random**
Bob's privacy:
Ok, Alice sees random encryption of intended output

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

\[ \begin{align*}
    r & \leftarrow \mathbb{Z}_q \\
    c & \leftarrow \text{Enc}(a; r) \\
    d & \leftarrow c^b \cdot \text{Enc}(0; r') \\
    M & \leftarrow \text{Dec}(d)
\end{align*} \]

Correctness:
ok

Random since \(r'\) is random
FUNCTIONALITY: SCALAR PRODUCT

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

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

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

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

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

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

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

TTP

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

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

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

\[ <a, b> = \Sigma a_i b_i \]
FUNCTIONALITY: SCALAR PRODUCT

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

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

\[ \langle a, b \rangle = \Sigma a_i b_i \]
SCALAR PRODUCT PROTOCOL

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

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

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

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

pk
b_i ∈ \{0, 1\}^L
b = \{b_i; i ∈ [n]\}
SCALAR PRODUCT PROTOCOL

for $i = 1$ to $n$

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

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

\{c_i\}
SCALAR PRODUCT PROTOCOL

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

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

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

\[ r' \leftarrow \mathbb{Z}_q \]
\[ d \leftarrow \prod_{i=1}^n c_i^{b_i} \cdot \text{Enc}(0; r') \]
SCALAR PRODUCT PROTOCOL

\[ \text{for } i = 1 \text{ to } n \]
\[ r_i \leftarrow \mathbb{Z}_q \]
\[ c_i \leftarrow \text{Enc}(a_i; r_i) \]
\[ r' \leftarrow \mathbb{Z}_q \]
\[ d \leftarrow \prod_{i=1}^{n} c_i^{b_i} \cdot \text{Enc}(0; r') \]

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

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

for $i = 1$ to $n$

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

$c = \{c_i\}$

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

$d \leftarrow \prod_{i=1}^{n} c_i^{b_i} \cdot \text{Enc}(0; r')$
SCALAR PRODUCT PROTOCOL

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

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

\[ d = \text{Enc}(\sum a_i b_i; ... + r') \]

\[ \text{for } i = 1 \text{ to } n \]

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

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

\[ \{c_i\} \]

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

\[ d \leftarrow \prod_{i=1}^{n} c_i^{b_i} \cdot \text{Enc}(0; r') \]

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

\[ \text{Protocol} \]

\[ \begin{align*}
\text{pk, sk} & \quad a \in \{0, 1\}^L \\
& \quad a = \{a_i : i \in [n]\}
\end{align*} \]

\[ \text{pk, sk} \]

\[ \begin{align*}
& \quad r_i \leftarrow \mathbb{Z}_q \\
& \quad c_i \leftarrow \text{Enc}(a_i; r_i)
\end{align*} \]

\[ \{c_i\} \]

\[ \begin{align*}
& \quad r' \leftarrow \mathbb{Z}_q \\
& \quad d \leftarrow \prod_{i=1}^{n} c_i^{b_i} \cdot \text{Enc}(0; r')
\end{align*} \]

\[ \begin{align*}
& \quad M \leftarrow \text{Dec}(d)
\end{align*} \]

\[ \begin{align*}
& \quad d = \text{Enc}(\Sigma a_i b_i, ... + r') \\
& \quad M = \Sigma a_i b_i
\end{align*} \]

Correctness: ok

for \( i = 1 \) to \( n \)

\[ r_i \]

\[ c_i \]

\[ \{c_i\} \]

\[ r' \]

\[ d \]

\[ \prod_{i=1}^{n} c_i^{b_i} \cdot \text{Enc}(0; r') \]

\[ M \]

\[ \Sigma a_i b_i \]
SCALAR PRODUCT PROTOCOL

\[
\begin{align*}
pk, sk & \quad a \in \{0, 1\}^L \\
\{a_i : i \in [n]\} & \quad b \in \{b_i : i \in [n]\}
\end{align*}
\]

for \(i = 1\) to \(n\)

\[
\begin{align*}
r_i & \leftarrow \mathbb{Z}_q \\
c_i & \leftarrow \text{Enc}(a_i; r_i)
\end{align*}
\]

\[
\begin{align*}
r' & \leftarrow \mathbb{Z}_q \\
d & \leftarrow \prod_{i=1}^{n} c_i^{b_i} \cdot \text{Enc}(0; r')
\end{align*}
\]

\[
\begin{align*}
d & = \text{Enc}(\Sigma a_i b_i; \ldots + r') \\
M & = \Sigma a_i b_i
\end{align*}
\]

Correctness: \(\text{ok}\)

Decryption succeeds when \(\Sigma a_i b_i \leq n 2^L\) is small, e.g., \(n 2^L < 2^{40}\)
SCALAR PRODUCT PROTOCOL

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

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

\( \{c_i\} \)

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

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

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

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

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

\( M = \Sigma a_i b_i \)

Alic's privacy: ok, Bob sees only ciphertexts

Correctness: ok

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

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

\( M \leftarrow Dec(d) \)
\( d = Enc(\sum a_i b_i, \ldots + r') \)
\( M = \sum a_i b_i \)

Correctness:
ok

for \( i = 1 \) to \( n \)

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

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

Alice's privacy:
ok, Bob sees only ciphertexts

\( r' \leftarrow \mathbb{Z}_q \)
\( d = \prod_{i=1}^{n} c_i^{b_i} \cdot Enc(0; r') \)

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

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

- Veto and scalar product are "linear" functions
MORE FUN...

- Veto and scalar product are "linear" functions
- ... thus straightforward to implement by using lifted Elgamal
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

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

\[ b \in \{0, 1\}^n \]
FUNCTIONALITY: HAMMING DISTANCE

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

TTP

\[ b \in \{0,1\}^n \]
FUNCTIONALITY: HAMMING DISTANCE

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

\[ b \in \{0,1\}^n \]
FUNCTIONALITY: HAMMING DISTANCE

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

\[ b \in \{0,1\}^n \]
FUNCTIONALITY: HAMMING DISTANCE

\[ a \in \{0,1\}^n \quad \text{and} \quad b \in \{0,1\}^n \]

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

$\delta(a, b) := | \{ i : a_i \neq b_i \} |$
FUNCTIONALITY: HAMMING DISTANCE

Does not seem to be "linear" at all???

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

Note $a_i$ and $b_i$ are Boolean!
QUIZ: LINEARIZATION OF HD

- Note $a_i$ and $b_i$ are Boolean!
- $a_i \not= b_i$ iff $a_i \text{ XOR } b_i = 1$
Quiz: Linearization of HD

- Note $a_i$ and $b_i$ are Boolean!
- $a_i \neq b_i$ iff $a_i \oplus b_i = 1$
- Moreover, Bob knows $b_i$
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 $b_i$

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

$$\delta(a, b) = \sum_{i=1}^{n} (b_i + (1 - 2b_i)a_i) = \sum_{i=1}^{n} (1 - 2b_i)a_i + \sum_{i=1}^{n} b_i$$
Note $a_i$ and $b_i$ are Boolean!

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

Moreover, Bob knows $b_i$

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

$\delta$ is “affine” for correct inputs and thus we can construct efficient 2-message protocol for Hamming distance.

$$\delta(a, b) = \sum_{i=1}^{n} (b_i + (1 - 2b_i)a_i) = \sum_{i=1}^{n} (1 - 2b_i)a_i + \sum_{i=1}^{n} b_i$$
HD PROTOCOL

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

- 
  - \( pk \)
  - \( b \in \{0, 1\}^n \)
HD PROTOCOL

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

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

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

pk
\( b \in \{0, 1\}^n \)

for \( i = 1 \) to \( n \)

\( r_i \leftarrow \mathbb{Z}_q \)

\( c_i \leftarrow \text{Enc}(a_i; r_i) \)
HD PROTOCOL

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

\[ r' \leftarrow \mathbb{Z}_q; \]
\[ d \leftarrow \prod_{i=1}^{n} c_i^{1-2b_i} \cdot \text{Enc} \left( \sum_{i=1}^{n} b_i; r' \right); \]
**HD PROTOCOL**

for $i = 1$ to $n$

$r_i \leftarrow \mathbb{Z}_q$

$c_i \leftarrow \text{Enc}(a_i; r_i)$

$r' \leftarrow \mathbb{Z}_q$

$d \leftarrow \prod_{i=1}^{n} c_i^{1-2b_i}$.

Enc $\left( \sum_{i=1}^{n} b_i; r' \right)$;
HD PROTOCOL

for $i = 1$ to $n$

$r_i \leftarrow \mathbb{Z}_q$

c_i \leftarrow \text{Enc}(a_i; r_i)$

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

$r' \leftarrow \mathbb{Z}_q$

d $\leftarrow \prod_{i=1}^{n} c_i^{1-2b_i}$

Enc $\left( \sum_{i=1}^{n} b_i; r' \right)$.
\[ pk, sk \]
\[ a \in \{0, 1\}^n \]
\[ M \leftarrow \text{Dec}(d) \]

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

\[ r' \leftarrow \mathbb{Z}_q; \]
\[ d \leftarrow \prod_{i=1}^{n} c_i^{1 - 2b_i}. \]
\[ \text{Enc} \left( \sum_{i=1}^{n} b_i; r' \right) ; \]

\[ d = \text{Enc}(\delta(a, b); ... + r') \]
HD PROTOCOL

For $i = 1$ to $n$

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

$d = \prod_{i=1}^{n} c_i^{1-2b_i} \cdot \text{Enc} \left( \sum_{i=1}^{n} b_i; r' \right)$

Correctness: $\text{ok}$

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

\[ pk, sk \]
\[ a \in \{0,1\}^n \]
\[ M \leftarrow \text{Dec}(d) \]

Correctness: ok

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

Decryption succeeds when \( \delta(a,b) \in \{0,...,n\} \) is small, e.g., \( n < 2^{40} \)

\[ d = \text{Enc}(\delta(a,b); \ldots + r') \]
\[ M = \delta(a,b) \]
HD PROTOCOL

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

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

\( \{c_i\} \)

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

Correctness: ok

\( d = \text{Enc}(\delta(a, b); ... + r') \)
\( M = \delta(a, b) \)

Alice's privacy: ok, Bob sees only ciphertexts

Decryption succeeds when \( \delta(a, b) \in \{0, ..., n\} \) is small, e.g., \( n < 2^{40} \)
Bob's privacy: ok, Alice sees random encryption of intended output

for $i = 1$ to $n$

$r_i \leftarrow \mathbb{Z}_q$

$c_i \leftarrow \text{Enc}(a_i; r_i)$

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

Correctness: ok

Alice's privacy: ok, Bob sees only ciphertexts

$r' \leftarrow \mathbb{Z}_q$

d $\leftarrow \prod_{i=1}^n c_i^{1-2b_i}$.

$\text{Enc} \left( \sum_{i=1}^n b_i; r' \right)$;

Decryption succeeds when $\delta(a,b) \in \{0,...,n\}$ is small, e.g., $n < 2^{40}$
REAL PROTOCOL: CPIR
REAL PROTOCOL: CPIR

- Assume any privacy-preserving database application
REAL PROTOCOL: CPIR

- Assume any privacy-preserving database application
- **Most trivial operation:** Alice queries one element from Bob's database
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
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?
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

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

\[ f = (f_0, f_1) \]

\[ f_i \in \{0,1\}^L \]
$(2,1)$-CPIR

- $x \in \{0,1\}$
- $f = (f_0, f_1)$
- $f_i \in \{0,1\}$
(2,1)-CPIR

\[ x \in \{0,1\} \quad \text{TTP} \quad f = (f_0, f_1) \quad f_i \in \{0,1\}^L \]
$x \in \{0,1\}$

$\mathbf{TTP}$

$f = (f_0, f_1)$

$f_i \in \{0,1\}^L$
\((2,1)\)-CPIR

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

\[ f = (f_0, f_1) \]

\[ f_i \in \{0,1\}^L \]

\[ f_x \]
$x \in \{0,1\}$

$\forall x \in \{0,1\}$

$f = (f_0, f_1)$

$f_i \in \{0,1\}^L$
(2,1)-CPIR

$x \in \{0,1\}$

$f = (f_0, f_1)$

$f_i \in \{0,1\}^L$

Does not seem to be "linear" at all???
QUIZ: CPIR PROTOCOL
QUIZ: CPIR PROTOCOL

❖ Note $x \in \{0, 1\}$
QUIZ: CPIR PROTOCOL

✧ Note $x \in \{0, 1\}$

✧ Moreover, Bob knows $f_0$ and $f_1$
Note $x \in \{0, 1\}$

Moreover, Bob knows $f_0$ and $f_1$

$$f_x = f_0 \cdot [x = 0] + f_1 \cdot [x = 1]$$
Note $x \in \{0, 1\}$

Moreover, Bob knows $f_0$ and $f_1$

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

$$f_x = f_0 \cdot (1 - x) + f_1 \cdot x = f_0 + (f_1 - f_0) \cdot x$$
QUIZ: CPIR PROTOCOL

- Note $x \in \{0, 1\}$
- Moreover, Bob knows $f_0$ and $f_1$
- $f_x = f_0 \cdot [x = 0] + f_1 \cdot [x = 1]$  
  \[ f_x = f_0 \cdot (1 - x) + f_1 \cdot x = f_0 + (f_1 - f_0) \cdot x \]

"affine" and thus we can construct 2-message protocol for (2, 1)-CPIR
(2, 1)-CPIR PROTOCOL

$pk, sk \ x \in \{0, 1\}$

$f = (f_0, f_1) \ f_i \in \{0, 1\}^L$
\((2, 1)\)-CPIR Protocol

Let \( pk, sk \) be the public and secret keys, respectively, and \( x \in \{0, 1\} \) be the input. The protocol proceeds as follows:

1. Choose \( r \leftarrow \mathbb{Z}_q \).
2. Compute \( c \leftarrow Enc(x; r) \).

The protocol allows for the computation of \( f = (f_0, f_1) \), where \( f_i \in \{0, 1\}^L \) for both \( i = 0, 1 \).
(2, 1)-CPIR PROTOCOL

\[ pk, sk \]
\[ x \in \{0, 1\} \]

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

\[ pk \]
\[ f = (f_0, f_1) \]
\[ f_i \in \{0, 1\}^L \]
(2, 1)-CPIR PROTOCOL

\[ \begin{align*}
    \text{pk}, \text{sk} & \quad x \in \{0, 1\} \\
    r & \leftarrow \mathbb{Z}_q \\
    c & \leftarrow \text{Enc}(x; r) \\
    r' & \leftarrow \mathbb{Z}_q \\
    d & \leftarrow c^{f_i - f_0} \cdot \text{Enc}(f_0; r') \\
\end{align*} \]
(2, 1)-CPIR PROTOCOL

\[ pk, sk \]
\[ x \in \{0, 1\} \]

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

\[ r' \leftarrow \mathbb{Z}_q \]
\[ d \leftarrow c^{f_i-f_0} \cdot \text{Enc}(f_0; r') \]

\[ pk \]
\[ f = (f_0, f_1) \]
\[ f_i \in \{0, 1\}_L \]
(2, 1)-CPIR PROTOCOL

Let $x \in \{0, 1\}$.

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

Let $f = (f_0, f_1)$ with $f_i \in \{0, 1\}^L$.

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

- $M \leftarrow \text{Dec}(d)$
CPIR: SECURITY

\( \text{pk}, \text{sk} \)
\( x \in \{0, 1\} \)

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

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

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

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

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

\[ pk, sk \]
\[ x \in \{0, 1\} \]

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

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

\[ d = \text{Enc}(f_x; \ldots + r') \]
\[ M = f_x \]

Correctness: ok

\[ pk \]
\[ f = (f_0, f_1) \]
\[ f_i \in \{0, 1\}^L \]
CPIR: SECURITY

\[ pk, sk \]
\[ x \in \{0, 1\} \]

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

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

\[ d = \text{Enc}(f_x; \ldots + r') \]
\[ M = f_x \]

Correctness:
\[ \text{ok} \]

Decryption succeeds when \( f_x \leq 2^L \) is small, e.g., \( 2^L < 2^{40} \)
CPIR: SECURITY

$pk, sk \ x \in \{0, 1\}$

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

Correctness: ok

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

$d = \text{Enc}(f_x; \ldots + r')$
$M = f_x$

Alice's privacy: ok, Bob sees only ciphertext

$p_k$

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

Decryption succeeds when $f_x \leq 2^L$ is small, e.g., $2^L < 2^{40}$
CPIR: SECURITY

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

Alice's privacy: ok, Bob sees only ciphertext

Correctness: ok

\[ \begin{align*}
    &pk, sk \\
    &x \in \{0, 1\} \\
    &M \leftarrow \text{Dec}(d) \\
    &d = \text{Enc}(f_x; \ldots + r') \\
    &M = f_x \\
    &d = \text{Enc}(f_x; \ldots + r') \\
    &M = f_x \\
\end{align*} \]

Decryption succeeds when \( f_x \leq 2^L \) is small, e.g., \( 2^L < 2^{40} \)
**EFFICIENCY: 2-ROUND PROTOCOLS**

**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 $n'$

$M_j \leftarrow \text{Dec}(d_j)$

$M \leftarrow \text{Answer}(a, \{M_j\}, ...)$
EFFICIENCY: 2-ROUND PROTOCOLS

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 $r'$
$\{d_j\} \leftarrow \text{Reply}(b, \{c_i\}; r')$

for $j = 1$ to $n'$

$M_j \leftarrow \text{Dec}(d_j)$
$M \leftarrow \text{Answer}(a, \{M_j\},...)$
ON EFFICIENCY
ON EFFICIENCY

Alice:
ON EFFICIENCY

Alice:

$n$ encryptions + $n'$ decryptions
ON EFFICIENCY

Alice:

- $n$ encryptions + $n'$ decryptions
- that is, $(3n + \Theta(n'))$ exponentiations + $n'$ DL-s
ON EFFICIENCY

Alice:

- $n$ encryptions + $n'$ decryptions
- that is, $(3n + \Theta(n'))$ exponentiations + $n'$ DL-s

Bob:
Alice:

- $n$ encryptions + $n'$ decryptions
- that is, $(3n + \Theta(n'))$ exponentiations + $n'$ DL-s

Bob:

- depends very much on protocol
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
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
"BITWISE" TRICKS

❍ DL timing: $\Theta(n' 2^{L/2})$ bit-ops /* linear in $n'$ / exp. in $L$ */
"BITWISE" TRICKS

❖ DL timing: $\Theta(n' 2^{L/2})$ bit-ops /* linear in $n'$ / exp. in $L$ */
❖ Common trick:
"BITWISE" TRICKS

- DL timing: $\Theta(n' 2^{L/2})$ bit-ops /* linear in $n'$ / exp. in $L$ */
- Common trick:
  - let Alice encrypt every bit separately, and then construct the protocol so that every ciphertext output by Bob also encrypts a bit
"BITWISE" TRICKS

- **DL timing**: $\Theta(n' 2^{L/2})$ bit-ops /* linear in $n'$ / exp. in $L$ */
- **Common trick:**
  - let Alice encrypt every bit separately, and then construct the protocol so that every ciphertext output by Bob also encrypts a bit
  - $\Theta(L)$ times more DL-s, but each DL gets "1-bit" input
"BITWISE" TRICKS

- **DL timing:** $\Theta(n' 2^{L/2})$ bit-ops /* linear in $n'$ / exp. in $L$ */
- **Common trick:**
  - let Alice encrypt every bit separately, and then construct the protocol so that every ciphertext output by Bob 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
"BITWISE" TRICKS

- **DL timing:** $\Theta(n' 2^{L/2})$ bit-ops /* linear in $n'$ / exp. in $L$ */
- **Common trick:**
  - let Alice encrypt every bit separately, and then construct the protocol so that every ciphertext output by Bob 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 $\Theta(L)$
"BITWISE" TRICKS

- **DL timing**: $\Theta(n' 2^{L/2})$ bit-ops /* linear in $n'$ / exp. in $L$ */
- **Common trick**: 
  - let Alice encrypt every bit separately, and then construct the protocol so that every ciphertext output by Bob 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

\[
\text{pk, sk} \quad x \in \{0, 1\}
\]

\[
\text{pk} \quad f = (f_0, f_1) \quad f_i \in \{0, 1\}^L
\]
BITWISE (2, 1)-CPIR

pk, sk
x ∈ \{0, 1\}

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

\( f_i \in \{0, 1\}^L \)
BITWISE (2, 1)-CPIR

pk, sk
x ∈ \{0, 1\}

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

\[ f = (f_0, f_i) \]
\[ f_i \in \{0, 1\}^L \]
BITWISE (2, 1)-CPIR

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

- $pk, sk$
  - $x \in \{0, 1\}$

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

- $for i = 1 to L$
  - $r_i' \leftarrow \mathbb{Z}_q$
  - $y_i \leftarrow f_1i - f_0i$
  - $d_i \leftarrow c^{y_i} \cdot \text{Enc}(f_0i; r_i')$
**BITWISE (2, 1)-CPIR**

- **pk, sk**
  - $x \in \{0, 1\}$

- **c**
  - for $i = 1$ to $L$
  - $r'_i \leftarrow \mathbb{Z}_q$
  - $y_i \leftarrow f_{1i} - f_{0i}$
  - $d_i \leftarrow c^{y_i} \cdot \text{Enc}(f_{0i}; r'_i)$

- $r \leftarrow \mathbb{Z}_q$
- $c \leftarrow \text{Enc}(x; r)$
**BITWISE (2, 1)-CPIR**

**Protocol**

\[
\begin{align*}
&\text{pk, sk} \\
&x \in \{0, 1\} \\
&\text{for } i = 1 \text{ to } L \\
&r_i' \leftarrow \mathbb{Z}_q \\
&y_i \leftarrow f_{1i} - f_{0i} \\
&d_i \leftarrow c^{y_i} \cdot \text{Enc}(f_{0i}; r_i') \\
&c \leftarrow \text{Enc}(x; r) \\
&\text{pk} \\
&f = (f_0, f_1) \\
&f_i \in \{0, 1\}^L \\
&\text{for } i = 1 \text{ to } L: \\
&M_i \leftarrow \text{Dec}(d_i) \\
&M \leftarrow (M_1, \ldots, M_L)
\end{align*}
\]
CPIR COMPLEXITY

pk, sk \ x \in \{0,1\}

\begin{align*}
    & r \leftarrow \mathbb{Z}_q \\
    & c \leftarrow (g^x h^r, g^r) \\
    & M_i \leftarrow \text{dlog} \left( d_i / d_i^{sk} \right) \\
    & M \leftarrow (M_1, \ldots, M_L)
\end{align*}

for i = 1 to L

\begin{align*}
    & r'_i \leftarrow \mathbb{Z}_q \\
    & y_i \leftarrow f_{1i} - f_{0i} \in \{-1,0,1\} \\
    & d_i \leftarrow (c_1, c_2)^{y_i} \cdot (g_{f_{0i}} h_i^{r'_i}, g_i^{r'_i})
\end{align*}
**CPIR COMPLEXITY**

<table>
<thead>
<tr>
<th></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>$3 \exp + 1 \cdot DL (L \text{ bits})$</td>
<td>5</td>
</tr>
<tr>
<td>Bitwise protocol</td>
<td>$2L + 2$</td>
<td>$(L + 2) \exp + L \cdot DL(1 \text{ bits})$</td>
<td>$2L$ Can be precomputed!</td>
</tr>
</tbody>
</table>

- Different trade-offs possible
- Computing $g^b$ is for free, for bit $b$

**Protocol**

\[
egin{align*}
\text{pk, sk} & \quad x \in \{0,1\} \\
\text{pk} & \quad f = (f_0, f_1) \\
\text{M}_i & \quad \text{dlog} \left( \frac{f_i}{d_{12} \text{sk}} \right) \\
\text{M} & \quad (\text{M}_1, \ldots, \text{M}_L)
\end{align*}
\]
CPIR: ALICE'S COMP.

Numbers are approximate, remember the slope.
Remarks

- Bitwise execution:
REMARKS

- **Bitwise execution:**
  - simple (more) efficient solution for **CPIR**
REMARKS

- Bitwise execution:
  - simple (more) efficient solution for CPIR
  - In other protocols, not so easy
Remarks

- **Bitwise execution:**
  - simple (more) efficient solution for **CPIR**
  - In other protocols, not so easy
- **Millionaire:** output bit depends on all input bits
REMARKS

- **Bitwise execution:**
  - simple (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
REMARKS

- **Bitwise execution:**
  - simple (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
ON ART OF PROTOCOL DESIGN

- Figure out how much resources you have
- Communication, Alice's and Bob's computation
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
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
ABOUT SECURITY

- Currently: **semihonest model**
  - Alice and Bob follow protocol but "eavesdrop"
ABOUT SECURITY

- Currently: **semihonest model**
  - Alice and Bob follow protocol but "eavesdrop"
- **Malicious model**
ABOUT SECURITY

- Currently: **semihonest model**
  - Alice and Bob follow protocol but "eavesdrop"
- **Malicious model**
  - Later lectures
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
ABOUT SECURITY
ABOUT SECURITY

揥 The protocols are "designed" to be secure
ABOUT SECURITY

- The protocols are "designed" to be secure
- Correctness (semihonest model)
ABOUT SECURITY

- The protocols are "designed" to be secure
- **Correctness (semihonest model)**
  - Follows from protocol design
ABOUT SECURITY

- The protocols are "designed" to be secure
- Correctness (semihonest model)
  - Follows from protocol design
- Alice's privacy (semihonest model)
ABOUT SECURITY

- 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
ABOUT SECURITY

- 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)**
ABOUT SECURITY

- 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)*
DEFINING SECURITY
DEFINING SECURITY

Alice's privacy:
DEFINING SECURITY

❖ Alice's privacy:
  ❖ Defined like IND-CPA for public-key encryption
DEFINING 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 $a_1$
DEFINING 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'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:
- Defined via simulation: Alice obtains no information about Bob's input \( b \), except what is obvious from her input \( a \) and correct output \( f_a(a, b) \)
DEFINING 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:**
  - Defined via simulation: Alice obtains no information about Bob's input $b$, except what is obvious from her input $a$ and correct output $f_a(a, b)$
  - Stronger notion than IND-CPA!
DEFINING 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:
  ❖ Defined via simulation: Alice obtains no information about Bob's input $b$, except what is obvious from her input $a$ and correct output $f_a(a, b)$
  ❖ Stronger notion than IND-CPA!

some computational assumption
DEFINING 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:**
  ❖ Defined via *simulation*: Alice obtains *no* information about Bob's input $b$, except what is obvious from her input $a$ and correct output $f_a(a, b)$
  ❖ Stronger notion than IND-CPA!
IND-CPA SECURITY OF PROTOCOLS
IND-CPA SECURITY OF PROTOCOLS

\[ \Pi = (\text{Setup, Query, Reply, Answer}) \]
**IND-CPA SECURITY OF PROTOCOLS**

- $\Pi = (\text{Setup}, \text{Query}, \text{Reply}, \text{Answer})$

**Game $\text{IND}_{\Pi, \mathcal{A}}(\kappa)$**

\[
gk \leftarrow \text{Setup}(\kappa) \\
(\text{sk}, \text{pk}) \leftarrow \text{Keygen}(gk) \\
(a_0, a_1) \leftarrow \mathcal{A}(gk, \text{pk}) \\
\beta \leftarrow \mathcal{R}_{\{0, 1\}} \\
\mathcal{R}_A \leftarrow \mathcal{R}_{\{0, 1\}} \\
c \leftarrow \text{Query}_{pk}(a_\beta, r) \\
\beta^* \leftarrow \mathcal{A}(gk, \text{pk}, c) \\
\text{Return } \beta = \beta^* \ ? 1 : 0
\]

Here $a_i$ are two possible $\mathcal{A}$'s inputs
IND-CPA SECURITY OF PROTOCOLS

\[ \Pi = (\text{Setup}, \text{Query}, \text{Reply}, \text{Answer}) \]

\[ \text{Adv}_{\Pi, \mathcal{A}}^{\text{IND}}(\kappa) := 2 \cdot \left| \Pr[\text{IND}_{\Pi, \mathcal{A}}(\kappa) = 1] - 1/2 \right| \]

\begin{align*}
gk & \leftarrow \text{Setup}(\kappa) \\
(\text{sk}, \text{pk}) & \leftarrow \text{Keygen}(gk) \\
(a_0, a_1) & \leftarrow \mathcal{A}(gk, pk) \\
\beta & \leftarrow \{0, 1\} \\
r & \leftarrow \mathcal{R}_{\mathcal{A}} \\
c & \leftarrow \text{Query}_{\text{pk}}(a\beta, r) \\
\beta^* & \leftarrow \mathcal{A}(gk, pk, c) \\
\text{Return} \beta = \beta^* \oplus 1 : 0
\end{align*}

Here \( a_i \) are two possible \( \mathcal{A}'s inputs \)
IND-CPA SECURITY OF PROTOCOLS

- $\Pi = (\text{Setup, Query, Reply, Answer})$
- $\text{Adv}^{\text{IND}}_{\Pi, A}(\kappa) := 2 \cdot \left| \Pr[\text{IND}^L_{\Pi, A}(\kappa) = 1] - \frac{1}{2} \right|$
- $A \in \text{breaks IND-CPA security of } \Pi \text{ iff } \text{Adv}^{\text{IND}}_{\Pi, A}(\kappa) \geq \epsilon$

Game $\text{IND}^L_{\Pi, A}(\kappa)$

- $gk \leftarrow \text{Setup } (1^\kappa)$
- $(sk, pk) \leftarrow \text{Keygen } (gk)$
- $(a_0, a_1) \leftarrow A(gk, pk)$
- $\beta \leftarrow \{0, 1\}$
- $r \leftarrow \mathcal{R}_A$
- $c \leftarrow \text{Query}_{pk}(a_\beta, r)$
- $\beta^* \leftarrow A(gk, pk, c)$
- Return $\beta = \beta^* \oplus 1 : 0$

Here $a_i$ are two possible $A$'s inputs
IND-CPA SECURITY OF PROTOCOLS

- \( \Pi = (\text{Setup}, \text{Query}, \text{Reply}, \text{Answer}) \)
- \( \text{Adv}_{\Pi, \mathcal{A}}^{\text{IND}}(\kappa) := 2 \cdot \left| \Pr[\text{IND}_{\Pi, \mathcal{A}}(\kappa) = 1] - 1/2 \right| \)
- \( \mathcal{A} \) \( \epsilon \)-breaks IND-CPA security of \( \Pi \) iff \( \text{Adv}_{\Pi, \mathcal{A}}^{\text{IND}}(\kappa) \geq \epsilon \)
- \( \Pi \) is \((\tau, \epsilon)\)-IND-CPA secure iff no PPT adversary \( \epsilon \)-breaks IND-CPA security of \( \Pi \) in time \( \leq \tau \)

Game \( \text{IND}_{\Pi, \mathcal{A}}(\kappa) \)

- \( \text{gk} \leftarrow \text{Setup}(\kappa) \)
- \( (\text{sk}, \text{pk}) \leftarrow \text{Keygen}(\text{gk}) \)
- \( (a_0, a_1) \leftarrow \mathcal{A}(\text{gk}, \text{pk}) \)
- \( \beta \leftarrow \mathcal{R}_{\{0, 1\}} \)
- \( r \leftarrow \mathcal{R}_{\mathcal{A}} \)
- \( c \leftarrow \text{Query}_{\text{pk}}(a_\beta, r) \)
- \( \beta^* \leftarrow \mathcal{A}(\text{gk}, \text{pk}, c) \)
- Return \( \beta = \beta^* ? 1 : 0 \)

Here \( a_i \) are two possible \( \mathcal{A} \)'s inputs
IND-CPA SECURITY OF PROTOCOLS

- $\Pi = (\text{Setup, Query, Reply, Answer})$
  
  $\text{Adv}_{\Pi,A}(\kappa) := 2 \cdot \left| \Pr[\text{IND}_{\Pi,A}(\kappa) = 1] - 1/2 \right|$

- A $\epsilon$-breaks IND-CPA security of $\Pi$ iff $\text{Adv}_{\Pi,A}(\kappa) \geq \epsilon$

- $\Pi$ is $(\tau,\epsilon)$-IND-CPA secure iff no PPT adversary $\epsilon$-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 $\text{IND}_{\Pi,A}(\kappa)$**

- $gk \leftarrow \text{Setup} (\kappa)$
- $(sk, pk) \leftarrow \text{Keygen} (gk)$
- $(a_0, a_1) \leftarrow A(gk, pk)$
- $\beta \leftarrow \{0, 1\}$
- $r \leftarrow \mathcal{R}_A$
- $c \leftarrow \text{Query}_{pk}(a_\beta, r)$
- $\beta^* \leftarrow A(gk, pk, c)$
- Return $\beta = \beta^* \oplus 1 : 0$

Here $a_i$ are two possible $A$'s inputs
IND-CPA SECURITY PROOFS
IND-CPA security proofs of all “correctly formed” 2-round protocols are tautologies
IND-CPA SECURITY PROOFS

- IND-CPA security proofs of all “correctly formed” 2-round 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
BOB'S PRIVACY PROOFS

- Also tautology
BOB'S PRIVACY PROOFS

- Also tautology
  - if the protocol is well constructed
BOB'S PRIVACY PROOFS

- Also tautology
- **if** the protocol is well constructed
- Alice only sees random encryption of the intended output
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
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
- Formally proven by using simulation
SIMULATION PARADIGM
SIMULATION PARADIGM

A protocol between honest Bob and adversary $\mathcal{A}$ is simulatable if there exists an efficient simulator $\text{Sim}$ such that for any (efficient?) $\mathcal{A}$, the following two distributions look indistinguishable:
SIMULATION PARADIGM

- A protocol between honest Bob and adversary $A$ is **simulatable** if there exists an efficient simulator $Sim$ such that for any (efficient?) $A$, the following two distributions look indistinguishable:

- **Distribution 1:**
  - $A$’s view when talking to Bob who knows $(x, w)$
A protocol between honest Bob and adversary $A$ is **simulatable** if there exists an **efficient** simulator $\text{Sim}$ such that for any (efficient?) $A$, the following two distributions look indistinguishable:

<table>
<thead>
<tr>
<th>Distribution 1:</th>
<th>Distribution 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$’s view when talking to Bob who knows $(x, w)$</td>
<td>$A$’s view when talking to $\text{Sim}$ who knows $x$ and the intended output of $A$ but nothing else about $w$</td>
</tr>
</tbody>
</table>
A protocol between honest Bob and adversary $A$ is simulatable if there exists an efficient simulator $\text{Sim}$ such that for any (efficient?) $A$, the following two distributions look indistinguishable:

- **Distribution 1:** $A$’s view when talking to Bob who knows $(x, w)$
- **Distribution 2:** $A$’s view when talking to $\text{Sim}$ who knows $x$ and the intended output of $A$ but nothing else about $w$

Since $\text{Sim}$ does not know $w$, the output of $\text{Sim}$ does not reveal anything about $w$
**Simulation Paradigm**

- A protocol between honest Bob and adversary $A$ is **simulatable** if there exists an efficient simulator $Sim$ such that for any (efficient?) $A$, the following two distributions look indistinguishable:

<table>
<thead>
<tr>
<th>Distribution 1:</th>
<th>Distribution 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$’s view when talking to Bob who knows $(x, w)$</td>
<td>$A$’s view when talking to $Sim$ who knows $x$ and the intended output of $A$ but nothing else about $w$</td>
</tr>
</tbody>
</table>

Since $Sim$ does not know $w$, the output of $Sim$ does not reveal anything about $w$.

**Simulation:** one of the most important paradigms in crypto.
A protocol between honest Bob and adversary $\mathcal{A}$ is **simulatable** if there exists an efficient simulator $\text{Sim}$ such that for any (efficient?) $\mathcal{A}$, the following two distributions look indistinguishable:

- **Distribution 1:** $\mathcal{A}$’s view when talking to Bob who knows $(x, w)$
- **Distribution 2:** $\mathcal{A}$’s view when talking to $\text{Sim}$ who knows $x$ and the intended output of $\mathcal{A}$ but nothing else about $w$

Since $\text{Sim}$ does not know $w$, the output of $\text{Sim}$ does not reveal anything about $w$

**Simulation:** one of the most important paradigms in crypto

The precise definition depends
EXAMPLE: SIMULATING CPIR

- Simulator $\text{Sim}(gk, pk, f_x)$ does:
EXAMPLE: SIMULATING CPIR

 Simulator Sim (gk, pk, fx) does:

 1. Create $r \leftarrow \mathbb{Z}_q$
EXAMPLE: SIMULATING CPIR

- Simulator $\text{Sim} \left( gk, pk, f_x \right)$ does:

1. Create $r \leftarrow \mathbb{Z}_q$

2. Compute $d_{\text{sim}} \leftarrow \text{Enc} \left( f_x; r \right)$
EXAMPLE: SIMULATING CPIR

Simulator Sim (gk, pk, fx) does:

1. Create $r \leftarrow \mathbb{Z}_q$

2. Compute $d_{\text{sim}} \leftarrow \text{Enc}(fx; r)$

3. Output $d_{\text{sim}}$ as simulated Reply
EXAMPLE: SIMULATING CPIR

- Simulator Sim \( (gk, pk, f_x) \) does:
  
  1. Create \( r \leftarrow \mathbb{Z}_q \)
  
  2. Compute \( d_{\text{sim}} \leftarrow \text{Enc}(f_x; r) \)
  
  3. Output \( d_{\text{sim}} \) as simulated Reply

- Clearly, \( d \) and \( d_{\text{sim}} \) have the same distribution given \( f_x \)
STUDY OUTCOMES
STUDY OUTCOMES

- Functionality vs protocol
STUDY OUTCOMES

- Functionality vs protocol
- 2-round protocols, examples
STUDY OUTCOMES

- Functionality vs protocol
- 2-round protocols, examples
- Different efficiency aspects
STUDY OUTCOMES

- Functionality vs protocol
- 2-round protocols, examples
- Different efficiency aspects
  - DL dominates when outputs get longer...
STUDY OUTCOMES

- Functionality vs protocol
- 2-round protocols, examples
- Different efficiency aspects
  - DL dominates when outputs get longer...
  - Tricks (some of them!): linearization, bitwise
STUDY OUTCOMES

- Functionality vs protocol
- 2-round protocols, examples
- Different efficiency aspects
  - DL dominates when outputs get longer...
  - Tricks (some of them!): linearization, bitwise
- Security of 2-round protocols
NEXT LECTURE

- How to construct protocols with large outputs
NEXT LECTURE

- How to construct protocols with large outputs

- without “bit-wising"
How to construct protocols with large outputs

- without “bit-wising"

- Trapdoor discrete logarithm
NEXT LECTURE

- How to construct protocols with large outputs
  - without “bit-wising"
- Trapdoor discrete logarithm
- Paillier cryptosystem