CRYPTOGRAPHIC PROTOCOLS
2016, LECTURE 11

MALICIOUS MODEL. IDEA OF ZERO KNOWLEDGE

HELGER LIPMAA
UNIVERSITY OF TARTU, ESTONIA
UP TO NOW

- Introduction to the field
- **Secure computation protocols**
  - Can do "everything" but not necessarily efficiently
- **Big assumption**: parties follow protocol
THIS TIME

- Why semihonest model is not sufficient
- Malicious model: desiderata
- Example protocols and problems
- Short introduction to zero knowledge
- "Conceptual" lecture - few really technical details
REMINDER: GENERAL MODEL

Protocol

\[ f_1(x_1, \ldots, x_n) \]

\[ f_2(x_1, \ldots, x_n) \]

\[ f_3(x_1, \ldots, x_n) \]

\[ f_4(x_1, \ldots, x_n) \]
Adversary sees protocol messages and inputs/outputs of some parties. Goal: deduce information about inputs/outputs of anybody else.
SECURITY: MALICIOUS MODEL

Adversary sees and modifies protocol messages and inputs/outputs of some parties. Goal: deduce information about inputs/outputs of anybody else and/or change their outputs (undetected)
SECURITY GOALS AND MODELS

- **Privacy:**
  - no more information is revealed than necessary

- **Correctness:**
  - parties obtain correct outputs

- **Semihonest model/passive security:**
  - assuming everybody is semi-honest

- **Malicious model/active security:**
  - even if some parties are malicious

*Note: correctness in semihonest model is "easy"*
MALICIOUS VS SEMIHONEST

- Malicious model is much more complicated
  - Parties can behave arbitrarily, still need security

- Attacks can be surprising/unpredicted
  - even if getting passive security is trivial

- Guaranteeing active security: often costly
CONCRETE PROTOCOLS

- We will now see a few concrete protocols and possible attacks
- Needed to get some intuition
- We use this intuition to somewhat-formalize malicious model
Reminder: (2, 1)-CPIR Protocol

\[ r \sim \mathbb{Z}_n^* \]
\[ c \leftarrow \text{Enc}(x; r) \]

\[ r' \sim \mathbb{Z}_n^* \]
\[ d \leftarrow c f_i - f_o \cdot \text{Enc}(f; r') \]

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

\[ M = (f_1 - f_o) x + f_o = f_x \]
QUIZ: WHAT CAN BAD ALICE DO?

1. not send anything
2. send an non-ciphertext
3. encrypt $x$ that does not come from a valid domain
4. create a valid ciphertext without even knowing what she encrypted
5. ...
WHAT CAN ALICE DO?

pk, sk ∈ \{0, 1\}

1. **not send anything**
   - cannot protect against it --- but then Alice will just not get her output

2. **send a non-ciphertext**
   - with most cryptosystems, Bob can efficiently detect it
QUIZ: WHAT CAN BAD ALICE DO?

- $x \in \{0, 1\}$
- $r \leftarrow \mathbb{Z}_n^*$
- $c \leftarrow \text{Enc}(x; r)$
- $r' \leftarrow \mathbb{Z}_n^*$
- $d \leftarrow c f_i - f_0 \cdot \text{Enc}(f_0; r')$
- $M \leftarrow \text{Dec}(d)$
- $M = (f_1 - f_0) x + f_0 = f_x$

- **encrypt $x$ that does not come from a valid domain** // $x$ is not Boolean
- if Alice encrypts $x = 2^L$:
  - $M = (f_1 - f_0) 2^L + f_0$
  - since $f_i \in \{0, 1\}^L$ then Alice can recover from $M$ first $f_0$ and then $f_i$
- **Corollary**. Alice can completely break Bob's privacy by encrypting wrong input

Somewhat non-intuitive that Alice can break privacy
QUIZ: WHAT CAN BAD ALICE DO?

- **Create a valid ciphertext without even knowing what she encrypted**
- This does not matter with Paillier: she knows secret key and thus can decrypt her message
- With Elgamal: she will not learn more than by "just encrypting $2^L$"
- However, this attack is relevant with some other protocols
QUIZ: WHAT CAN BAD BOB DO?

**pk, sk**

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

1. not send anything
2. send a non-ciphertext
3. use incorrect database but correct linear function
4. send encryption of wrong linear function
5. ...

**M \leftarrow Dec(d)\right.**

\[ M = (f_1 - f_0) \cdot x + f_0 = f_x \]
QUIZ: WHAT CAN BAD BOB DO?

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

\[ r \leftarrow \mathbb{Z}_n^* \]
\[ c \leftarrow \text{Enc}(x; r) \]
\[ \text{M} \leftarrow \text{Dec}(d) \]

\[ \text{not send anything} \]
- Alice can detect it but she can do nothing about it
- It is one of the attacks we cannot protect against even theoretically
**QUIZ: WHAT CAN BAD BOB DO?**

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

- **send a non-ciphertext**
- Easy to detect by Alice with practically all cryptosystems

\[
M = (f_1 - f_0) x + f_0 = f_x
\]

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

- \( \text{pk, sk} \)
- \( f = (f_0, f_1) \)
- \( f_i \in \{0, 1\} \)
- \( L \)
- \( c \leftarrow \mathbb{Z}_n^* \)
- \( c \leftarrow \text{Enc}(x; r) \)
- \( r \leftarrow \mathbb{Z}_n^* \)
- \( c \leftarrow \text{Enc}(x; r') \)
- \( r' \leftarrow \mathbb{Z}_n^* \)
- \( d \leftarrow c f_i - f_0 \cdot \text{Enc}(f_0; r') \)
- \( M \leftarrow \text{Dec}(d) \)
- \( M = (f_1 - f_0) x + f_0 = f_x \)
**QUIZ: WHAT CAN BAD BOB DO?**

- Alice obtains $F_x$ where $F$ is a wrong database
- Second attack against which we cannot protect even theoretically: Bob enters wrong inputs

**Use incorrect database but correct linear function**

$M = (f_1 - f_0) x + f_0 = f_x$
**QUIZ: WHAT CAN BAD BOB DO?**

- **pk, sk**
- **x ∈ {0, 1}**

- \( r \leftarrow \mathbb{Z}_n^* \)
- \( c \leftarrow \text{Enc}(x; r) \)

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

- \( M = (f_1 - f_0) x + f_0 = f_x \)

- **send encryption of wrong linear function**
- For example, \( M = 0 \) (no dependence on \( x \))
- Here, this attack is equal to the previous attack in severity
- \( M = ax + b = \text{CPIR with database} (b, a + b) \)
- But there exist applications where this attack is more severe
IDEAL MODEL

Trusted third party

$a \in S_1$

$f_a(a, b)$

$b \in S_2$
IDEAL MODEL

Possible attacks in ideal model:
1. either party aborts after seeing their output
2. either party sends wrong inputs
   - however, the same wrong inputs have to be used consistently throughout the protocol

Abort or no abort

$f_a(a^*, b^*)$ if no abort: $f_b(a^*, b^*)$ else send "abort"
REAL MODEL: SECURITY

Protocol

Goal: (under a cryptographic assumption) the only possible attacks should be the two attacks that are also possible in the ideal model.

Abort or no abort

$\forall a \in S_1 \exists b \in S_2 \forall a^* \in S_1 \exists b^* \in S_2 \implies f_a(a^*, b^*)$

If no abort: $f_b(a^*, b^*)$ else send "abort"
REMARKS

- Only need to protect security of an honest user, so we say consider honest Alice and her security against malicious Bob

- Everything we said generalises to the case of multiple parties

- **Def.** A cryptographic protocol implements a functionality **securely** if only the following two attacks are possible:
  - any of the parties aborts protocol in the middle
  - some parties run the protocol but on wrong inputs

  Even if inputs are wrong they should be consistently wrong
RECALL: HOMOMORPHIC E-VOTING

Vote collector: sees who sent which ciphertext, cannot decrypt

Tallier: sees anonymous ciphertext, can decrypt

Enc(f(c_i)), Enc(Σf(c_i))

pk

pk

sk

complete tally can be computed from this efficiently

c_i ∈ {0,...,C - 1}
**QUIZ: MALICIOUS VOTER**

- **Enc**($f(c_i)$)
- **Enc(∑f(c_i))**

- **pk**
- **sk**

- $c_i \in \{0, \ldots, C - 1\}$

- **Voter** can:
  - not vote or vote for a "wrong candidate" (= attack in ideal model)
  - Voter computer can also cheat (won't have a separate slide)

- Can encrypt invalid encoding:
  - Enc (10 $f(c_i)$) - 10 votes for his favorite candidate
  - Enc (($V + 1$) $f(c_i)$) - makes tally invalid
  - Enc ($f(c_j)$) (possible due malleability) - to be different from voter $v_j$
  - ...

- Vote collector: sees who sent which ciphertext, cannot decrypt
- Tallier: sees anonymous ciphertext, can decrypt
MALLEABILITY: AUCTIONS

- Auctions: encrypt your bid, highest bidder wins

- Alice encrypts $c \leftarrow \text{Enc} (\text{bid}_{Alice})$

- Bob computes $d \leftarrow c \cdot \text{Enc} (1; r) = \text{Enc} (\text{bid}_{Alice} + 1)$

- Bob wins, with minimal possible bid
MALLEABILITY: RANDOM COIN

**Goal:** // toss a random coin to decide on something
- encrypt random $x, y \in \{0, 1\}$
- output is $z = x \oplus y = x + y - 2xy$

**Alice:** encrypts $c \leftarrow \text{Enc} \,(x)$

**Honest Bob:** encrypts $d \leftarrow \text{Enc} \,(y)$

**Attack:**
- Bob chooses any $z \in \{0, 1\}$, sets $d \leftarrow c^{(1 - 2z)} \cdot \text{Enc} \,(z; r)$

$$y = (1 - 2z) x + z = x + z - 2xz = x \oplus z,$$  thus really

$$z = x \oplus y$$
QUIZ: MALICIOUS VOTE COLLECTOR

Vote collector: sees who sent which ciphertext, cannot decrypt.
Tallier: sees anonymous ciphertext, can decrypt.

$\Sigma f(c_i)$

- Can abort protocol (= attack in ideal model)
- Can forward anything to tallier:
  - drop some votes, insert some votes
  - just encrypt "Putin won 100%"
  - encrypt invalid value (makes tally incorrect)
  - ...

$c_i \in \{0, \ldots, C - 1\}$
QUIZ: MALICIOUS TALLIER

- Can abort protocol (= attack in ideal model)
- Can output wrong plaintext:
  - Putin won 100%
  - invalid tally
  - something plainly unbelievable...

\[ c_i \in \{0, \ldots, C - 1 \} \]

\[ \text{Enc}(f(c_i)) \]

Vote collector: sees who sent which ciphertext, cannot decrypt

Tallier: sees anonymous ciphertext, can decrypt

\[ \text{Enc}(\Sigma f(c_i)) \]

\[ \Sigma f(c_i) \]
**Quiz: Main Problem**

- **Question:** what is the main problem with this voting protocol?

- **Answer:**
  - unaccountability
  - Not possible to make sure
  - there was cheating
  - who cheated
IDEAS FOR IMPROVEMENTS

- Do not use encodings $f(c_i)$ but just encrypt $c_i$
  - Voter sends $\text{Enc}(c_i)$. If its incorrect, then her vote does not count

- Additional benefit:
  - With encodings, need to use Paillier since encoding is sparse. Without encoding can use Elgamal

- How? Obviously then we need to use other tricks
  - Recall that we need to separate the guy who can connect voters to ciphertexts from the guy who can decrypt
MIXNET BASED VOTING: IDEA

- Encrypted votes are "shuffled" by a mixserver
  - More precisely, shuffled and blinded
- There is usually more than one mixserver
- Mixservers cannot decrypt, but the first mixserver can connect voters to ciphertexts
- After final mixing, the resulting ciphertext vector is threshold-decrypted and tally is published
MIXNET BASED E-VOTING

\[ C_i = \operatorname{Enc}(c_i) \]

\[ C'_i = C_{\pi(i)} \cdot \operatorname{Enc}(0; r_i) \]

\[ C''_i = C'_{\pi'(i)} \cdot \operatorname{Enc}(0; r'_{i'}) \]

\( \pi \): random permutation

\( r_i \) - random randomizers

\( \pi' \): random permutation

\( r'_{i'} \) - random randomizers
MIXNET: SECURITY PROPERTIES

- $C_i = \text{Enc}(c_i)$
- $C'_i = C_{\pi(i)} \cdot \text{Enc}(0; r_i)$
- $C''_i = C'_{\pi(i)} \cdot \text{Enc}(0; r'_i)$

- Voter can: not vote or vote for a "wrong candidate", ...
  (=attack in ideal model)
  - If voter computer does it, then it is a different thing
  - Can encrypt invalid encoding:
    - but then his vote is not valid / not counted at the end, so this is ok

pk: threshold

\{(c_i)\} in some order
\[ C_i = \text{Enc}(c_i) \]

\[ C_i' = C_{\pi(i)} \cdot \text{Enc}(0; r_i) \]

\[ C_i'' = C'_{\pi(i)} \cdot \text{Enc}(0; r'_i) \]

- Can abort the protocol (= attack in the ideal model)
- Can output invalid ciphertexts:
  - too few, too many, not all are ciphertexts - easy to detect
  - output is not a shuffle: for no permutation \( \pi \) and for no randomizer \( r_i \) it holds that \( C_i' = C_{\pi(i)} \cdot \text{Enc}(0; r_i) \)
MIXNET: SECURITY PROPERTIES

- $C_i = \text{Enc}(c_i)$
- $C_i' = C_{\pi(i)} \cdot \text{Enc}(0; r_i)$
- $C_i'' = C_{\pi'(i)} \cdot \text{Enc}(0; r_i')$

- Can abort the protocol (=attack in ideal model)
- Can output invalid plaintexts:
  - too few, too many - easy to detect
  - $c_i$ is not the decryption of $C_i''$, for at least one $i$
E-voting as other complex real-life protocols have many entities of different functionalities and capabilities/goals.

We already omitted some entities here for simplicity.

Voter computer, PKI (provision of public keys), the Internet (service provider?)
HOW TO GET SECURITY?

✧ First idea:

✧ let the mixserver publish $\pi$ and all $r_i$. Then verify that the output is correct.

✧ let the voter to publish his vote and ciphertext randomizer. Check they are also correct.

✧ Bad for privacy

such secret value, revealing of which proves correct action, is called a witness
SECOND IDEA

✧ Do not reveal the witness

✧ Instead let the party prove that such a witness exists

✧ so that the proof does not reveal any side information apart from that

Zero-knowledge proof
If this idea sounds crazy, think about authentication.
ZK PROOF: SHORT DEFINITION

- **Syntax**: ZK proof is a protocol between a prover $P$ and a verifier $V$, at the end of which $V$ either accepts or rejects.

- ZK proof satisfies the following security requirements:
  - **Completeness**: honest $V$ accepts honest $P$.
  - **Soundness**: honest $V$ does not accept malicious $P^*$.
  - **Zero-knowledge**: malicious $V^*$ learns from the proof with a honest $P$ that $P$ is honest and nothing else.

Formal definitions are much more complicated, see the next lecture.
RECALL: HOMOMORPHIC E-VOTING

pk

Vote collector: sees who sent which ciphertext, cannot decrypt

Enc(f(c_i))

c_i \in \{0, ..., C - 1\}

pk

Enc(\Sigma f(c_i))

sk

Tallier: sees anonymous ciphertext, can decrypt

\Sigma f(c_i)
RECALL: HOMOMORPHIC E-VOTING

\[ \text{Enc}(f(c_i)) \]

Vote collector: sees who sent which ciphertext, cannot decrypt

\[ \text{Enc}(\Sigma f(c_i)) \]

Tallier: sees anonymous ciphertext, can decrypt

no need for ZK proof (product of public ciphertexts)

\[ c_i \in \{0, \ldots, C - 1\} \]

\[ \Sigma f(c_i) \]

+ ZK proof that decryption was correct

\[ \text{pk} \]

\[ \text{pk} \]

\[ \text{sk} \]

\[ \text{pk} \]

\[ \text{sk} \]
RECALL: MIXNET BASED E-VOTING

\[ C_i = \text{Enc}(c_i) \]

\[ C_i' = C_{\pi(i)} \cdot \text{Enc}(0; r_i) \]

\[ C_i'' = C'_{\pi'(i)} \cdot \text{Enc}(0; r_i') \]

\[ \pi: \text{random permutation} \]

\[ r_i: \text{random randomizers} \]

\[ \pi': \text{random permutation} \]

\[ r_i': \text{random randomizers} \]

\[ \text{pk, sk: threshold} \]

+ ZK proof that the shuffle is correct

+ ZK proof that decryption was correct

\( \{ c \text{ in some order} \} \)
Some ZK proofs are obviously much more complex than others.

Proof of correct decryption:

- with Paillier, tallier can not only compute \( m \) but also \( r \)

- proof = \((m, r)\)

Proof of correct shuffle: ???
GENERAL PROTOCOL DESIGN

- Design a passively secure protocol
  - I.e., a protocol that achieves correctness/privacy given participants follow it
  - ... take any protocol we have seen up to now

- Make it secure in the malicious model by adding ZK proofs to all messages

of course this needs "some" care
STUDY OUTCOMES

- Semihonest vs malicious model
- Ideal model and our goal in the real model
- Some protocols and how they can be subverted by malicious parties
- Idea: ZK proofs
NEXT LECTURE

- More information about ZK proofs

- Initial study:
  - special type of ZK proofs, called Σ-protocols