SMC Continued: Garbled Circuits and Zero Knowledge Proofs

Giulia Fanti
Fall 2019
Based on slides by Vitaly Shmatikov, Piotr Mardziel, and Andrej Bogdanov
Administrative

• HW4 due on Friday, 11:59 pm

• Additional OH on Friday (Sruti)
  • Regular location and time

• Final project
  • Presentations last week of class: Mon. Dec. 2 and Wed. Dec. 4
  • Sign up here: https://docs.google.com/spreadsheets/d/1ylz1MWLtlAJvxUkpTAT0fVtqKabFQXanGh3wqo1g-tc/
  • PLEASE ADD YOUR CANVAS GROUP NUMBER
  • Final writeup due on Dec. 11, 11:59 pm EDT
The Apple Card Didn't 'See' Gender—and That's the Problem

The way its algorithm determines credit lines makes the risk of bias more acute.

While Goldman Sachs, the issuing bank for the Apple Card, insisted right away that there isn't any gender bias in the algorithm, it failed to offer any proof. Then, finally, Goldman landed on what sounded like an ironclad defense: The algorithm, it said, has been vetted for potential bias by a third party; moreover, it doesn't even use gender as an input.

But the very fact that customers' gender is not collected would make such an audit less effective. According to Thomas, companies must, in fact, “actively measure protected attributes like gender and race” to be sure their algorithms are not biased on them.

The Brookings report also recommends hiring legal as well as technical experts to monitor algorithms for unintended bias after they’ve been deployed.
In-class Quiz

• On Canvas
Example: Naor-Pinkas Oblivious Transfer

Setting: order-\(q\) subgroup of \(\mathbb{Z}_p\), \(p\) is prime, \(q\) divides \(p - 1\)
\(g\) is a generator group for which CDH assumption holds

Server

- Chooses random \(k\)
- Sets \(PK_s = g^k\), \(PK_{1-s} = C/PK_s\)

Chooser

- Chooses random \(r\)
- Computes \(PK_1\) by taking \(C/PK_0\)

Choices: bit \(\sigma\)

Messages \(m_0\) and \(m_1\)

- Computes \((g^r)^k = (PK_\sigma)^r\)
- Decrypts \(m_\sigma\)

Chooser does not know the discrete log of \(PK_{1-\sigma}\), thus cannot distinguish between a random value \(g_Z\) and \((PK_{1-\sigma})^r\)
A. Yao

Protocols for Secure Computations

(FOCS 1982)
Yao’s Protocol

• Compute any function securely
  ... in the semi-honest model; can be extended to malicious

• First, convert the function into a boolean circuit

Truth tables:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</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>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
1: Pick Random Keys For Each Wire

- Evaluate **one gate** securely
  - Later generalize to the entire circuit
- Alice picks two **random keys** for each wire
  - One key corresponds to “0”, the other to “1”
  - 6 keys in total for a gate with 2 input wires
2: Encrypt Truth Table

- Alice encrypts each row of the truth table by encrypting the output-wire key with the corresponding pair of input-wire keys.

Original truth table:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</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>

Encrypted truth table:

- $E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$
- $E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
- $E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
- $E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
3: Send Garbled Truth Table

- Alice randomly permutes ("garbles") encrypted truth table and sends it to Bob

Garbled truth table:

\[ E_{k_{0x}}(E_{k_{0y}}(k_{0z})) \]
\[ E_{k_{0x}}(E_{k_{1y}}(k_{0z})) \]
\[ E_{k_{1x}}(E_{k_{0y}}(k_{0z})) \]
\[ E_{k_{1x}}(E_{k_{1y}}(k_{1z})) \]

Alice does not know which row of garbled table corresponds to which row of original table.
4: Send Keys For Alice’s Inputs

- Alice sends the key corresponding to her input bit
  - Keys are random, so Bob does not learn what this bit is

Garbled truth table:

- $E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
- $E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
- $E_{k_{0x}}(E_{k_{0y}}(k_{1z}))$
- $E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
- $E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

If Alice’s bit is 1, she simply sends $k_{1x}$ to Bob; if 0, she sends $k_{0x}$

Learns $K_{b'x}$ where $b'$ is Alice’s input bit, but not $b'$ (why?)
5: Use OT on Keys for Bob’s Input

- Alice and Bob run oblivious transfer protocol
  - Alice’s input is the two keys corresponding to Bob’s wire
  - Bob’s input into OT is simply his 1-bit input on that wire

Alice

Bob

k₀x, k₁x
k₀y, k₁y
k₀z, k₁z

AND

z

Knows $K_{b'x}$ where $b'$ is Alice’s input bit and $K_{by}$ where $b$ is his own input bit

Run oblivious transfer
Alice’s input: $k₀y, k₁y$
Bob’s input: his bit $b$
Bob learns $k_{by}$

What does Alice learn?

Garbled truth table:

- $E_{k₁x}(E_{k₀y}(k₀z))$
- $E_{k₀x}(E_{k₁y}(k₀z))$
- $E_{k₁x}(E_{k₁y}(k₁z))$
- $E_{k₀x}(E_{k₀y}(k₀z))$

slide 12
6: Evaluate One Garbled Gate

- Using the two keys that he learned, Bob decrypts exactly one of the output-wire keys
  - Bob does not learn if this key corresponds to 0 or 1
    - Why is this important?

Garbled truth table:

- Suppose $b'=0$, $b=1$
  - This is the only row Bob can decrypt. He learns $K_{0z}$

Knows $K_{b'x}$ where $b'$ is Alice’s input bit and $K_{by}$ where $b$ is his own input bit
7: Evaluate Entire Circuit

- In this way, Bob evaluates entire garbled circuit
  - For each wire in the circuit, Bob learns only one key
  - It corresponds to 0 or 1 (Bob does not know which)
    - Therefore, Bob does not learn intermediate values (why?)

- Bob tells Alice the key for the final output wire so she can determine if it corresponds to 0 or 1
  - Bob does not tell her intermediate wire keys (why?)
Example application:
How good is my password?
Brief Discussion of Yao’s Protocol

• Function must be converted into a circuit
  • For many functions, circuit will be huge (can use BDD)
• If m gates in the circuit and n inputs, then need 4m encryptions and n oblivious transfers
  • Oblivious transfers for all inputs can be done in parallel
• Yao’s construction gives a constant-round protocol for secure computation of any function in the semi-honest model
  • Number of rounds does not depend on the number of inputs or the size of the circuit!
Zero-Knowledge Proofs

Part II
Authentication

What happens when you type in your password?
Naïve authentication

- The server knows your password
- So they can impersonate you at other web sites where you use the same password
“Zero-knowledge” authentication

Can you convince the server that you know your password, without revealing it (or any other information)?
What is knowledge?

What is ignorance?
(lack of knowledge)

• Example 1: Tomorrow’s lottery numbers

2  31  12  7  28  11

We are ignorant of them because they are random
What is ignorance?

• Example 2: A difficult math problem

We are ignorant because it takes a lot of work to figure out the answer

• Questions of this type include
  • Finding satisfying assignments to Boolean formulas
  • Finding cliques in graphs
  • All NP-hard problems

Prove that \( P \neq NP \)
Using ignorance to our advantage

I know the password  
Can you prove it?  
acme.com

We want to convince the server that we know the password, while keeping it ignorant of the password itself.

The server is convinced, but gains zero-knowledge!
Goals

• Prover convinces verifier of statement.
  • “I know password for account gfanti”.
• Verifier cannot use what they learned to convince anyone else of the statement.
  • Prevent website from proving they know password for account gfanti.
Interactive Protocols

Example 1: Non-Color-blindness protocol
Example 2: Cave password protocol
Example 3: Graph coloring
Prove that you know blue from red

I pull at random either two balls from same box or one ball from box 1 and one from box 2

You say “same color” or “different color”

We repeat 10 times

If you got all the answers right, I am convinced you know red from blue

But I did not gain any other knowledge!
Properties

• Soundness
  • If Verifier accepts then the property (Prover is not color blind) holds with high probability

• Completeness
  • If property (Prover is not color blind) holds then the Verifier always accepts
Cave password protocol

• Alice proves to Bob that she knows password.
• Without revealing password to anyone.
Zero-knowledge

The verifier’s view of the interaction with the prover can be efficiently simulated \textit{without} interacting with the prover

\[ S(V) \cong P \leftrightarrow V \]

Probability distributions on transcripts are indistinguishable
ZK Proof Outline for Non-Color Blindness

• Verifier V*’s view in real world interaction with Prover P
  1. \(wp \ p\): draw two balls from same box; Prover says “same color”
  2. \(wp \ (1-p)\): draw one ball from box 1 and one ball from box 2; Prover says “different color”

• Simulator S uses V* as a black box; does not interact with P
  1. When V* says “draw two balls from same box” it does so and says “same color” to simulate P (\(wp \ p\))
  2. When V* says “draw one ball from box 1 and one ball from box 2” it does so and says “different color” to simulate P (\(wp \ 1-p\))

\((P,V^*)\) interaction transcript \(\approx S(V^*)\) transcript
Comments

• Verifier is polynomial time

• Prover has unbounded computation power

• ZK property has to hold for all verifiers $V^*$ (not just the honest verifier $V$)
Zero-knowledge password authentication

acme.com

Oded Goldreich  Silvio Micali  Avi Wigderson
Important Notion: Commitments

• Locked box analogy
  • Hiding – hard to tell which message is committed to
  • Binding – there is a unique message corresponding to each commitment
Graph coloring

**Task:** Assign one of 3 colors to the vertices so that no edge has both endpoints of same color

\[ 3\text{COL} = \{ G: G \text{ has a valid 3-coloring} \} \]

- Theorem

3\text{COL} is \text{NP-complete}

- However, it is easy to create 3-colored graphs.
Password authentication via 3-coloring

- Step 0: When you register for the web service, choose your password to be a valid 3-coloring of some (suitable) graph.
Password authentication via 3-coloring

- When the server asks for your password

Step 1: do not send the password, but send the graph $G$ instead (without the colors)

password:  

\[ \text{password: } \text{color1} \text{ color2} \text{ color3} \text{ color4} \text{ color5} \]

\[ G \]

\[ \text{acme.com} \]
Intuition about registration phase

• Because 3-coloring is hard, the server will not be able to figure out your password (coloring) from $G$.

• Later, when you try to log in, you will convince the server that you know how to color $G$, without revealing the coloring itself.

• The server will be convinced you know your password but remain ignorant about what it is.
The login phase, Step 2

password:

You randomly permute the colors

You lock each of the colors in a box and send the boxes to the server

The server chooses an edge at random and asks for the keys to the boxes at the endpoints

You send the requested keys

The server unlocks the two boxes and checks the colors are different

Repeat this 1000 times. Login succeeds if colors always different
Analysis in the login phase

Completeness

If you know the coloring then you will always successfully convince the server
Analysis in the login phase

• Soundness

  • If you are an impostor, you won’t know how to color the graph, so you will reveal two nodes of the same color for at least one edge.

  • If the verifier tries this \( n \) times on a graph with \(|E|\) edges, it will fail to notice with probability at most

    \[
    \left(1 - \frac{1}{|E|}\right)^n
    \]

  • So let’s run the protocol \( n = |E|^2 \) times (polynomial)

    \[
    \lim_{{|E| \to \infty}} \left(1 - \frac{1}{|E|}\right)^{|E|^2} = \frac{1}{e^{|E|}}
    \]
Analysis in the login phase

Zero Knowledge

If you are honest, the server remains ignorant about your password because all he sees are two random different colors
ZK Proof Outline for 3-COL

• Simulator S
  • Internally select random edge \((i, j)\) and random permutation

1. \(P \rightarrow V^*:\) Generate coloring s.t. \(\text{color}(i)\) not equal \(\text{color}(j)\); send permuted colors in locked boxes

2. \(V^* \rightarrow P:\) Open colors for edge \((i, j)\)

3. \(P \rightarrow V^*:\) Reveal \(\text{color}(i)\) and \(\text{color}(j)\) (note by step 1 they are not equal)

Note: If \(V^*\) is not honest, use \(V^*\) as a blackbox to output edge \(e\) in step 2; rewind if \(e\) not equal to \((i, j)\)
Seminal Results

• IP and ZK defined [GMR’85]

• ZK for all NP languages [GMW’86]
  • Assuming one way functions exist

• ZK for all of IP [BGGHKMR’88]
  • Everything that can be proven can be proven in ZK assuming one way functions exist
Next time

• How are these tools be used in blockchains?

• Summary of the course