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

WILL KNIGHT BUSINESS 11.19.2019 09:15 AM

WIRED

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.

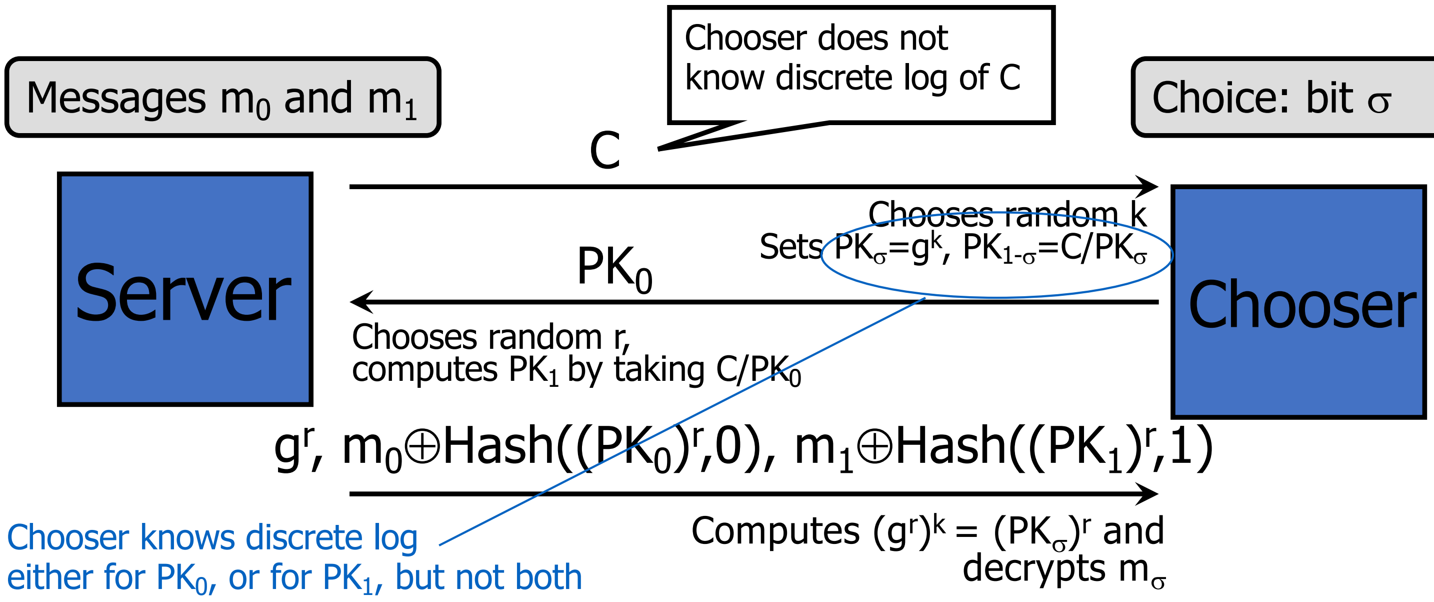
The Brookings report also recommends hiring legal as well as technical experts to monitor algorithms for unintended bias after they've been deployed.

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



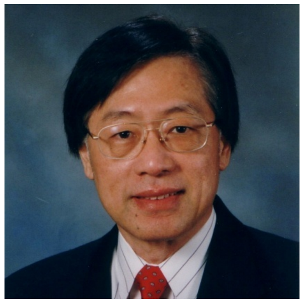
Messages m_0 and m_1
Chooser does not know discrete log of C
Choice: bit σ
C
Server
Chooser
Chooses random k
Sets $PK_\sigma=g^k$, $PK_{1-\sigma}=C/PK_\sigma$
PK_0
Chooses random r, computes PK_1 by taking C/PK_0
g^r, $m_0 \oplus Hash((PK_0)^r,0)$, $m_1 \oplus Hash((PK_1)^r,1)$
Computes $(g^r)^k = (PK_\sigma)^r$ and decrypts m_σ
Chooser knows discrete log either for PK_0, or for PK_1, but not both

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)

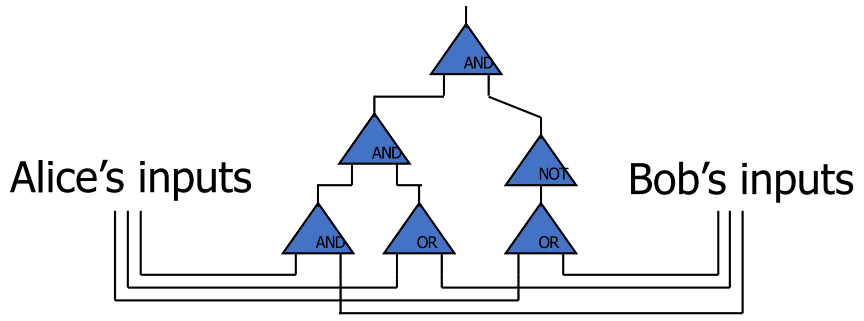


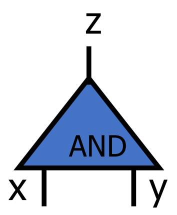
Yao’s Protocol

- Compute any function securely

  … in the semi-honest model; can be extended to malicious

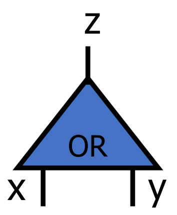
- First, convert the function into a boolean circuit





Truth table:

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

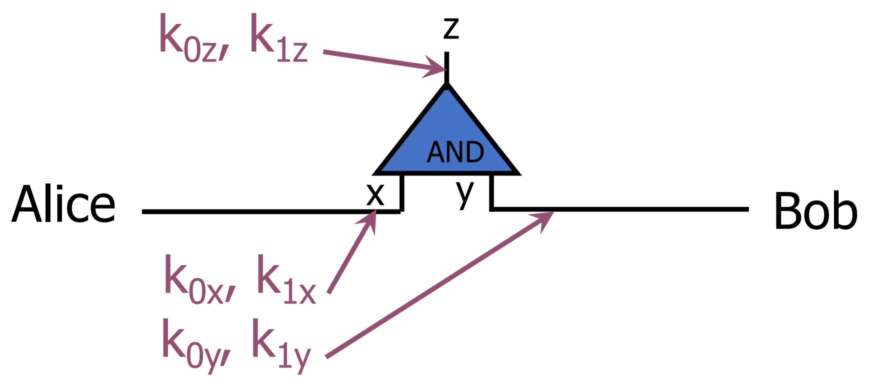


Truth table:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

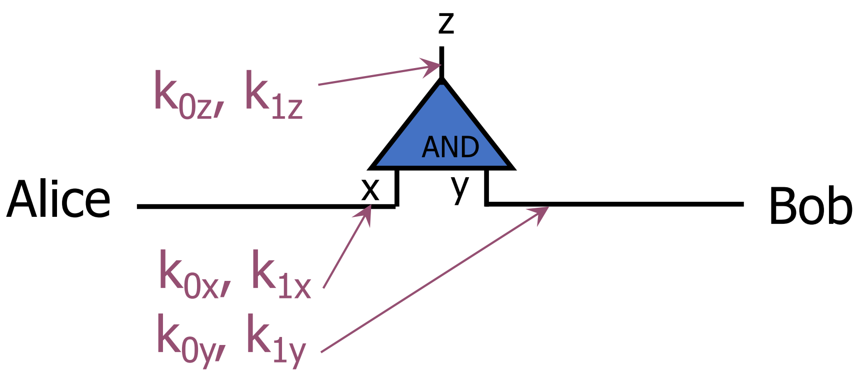
1: Pick Random Keys For Each Wire

- 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:

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Encrypted truth table:

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

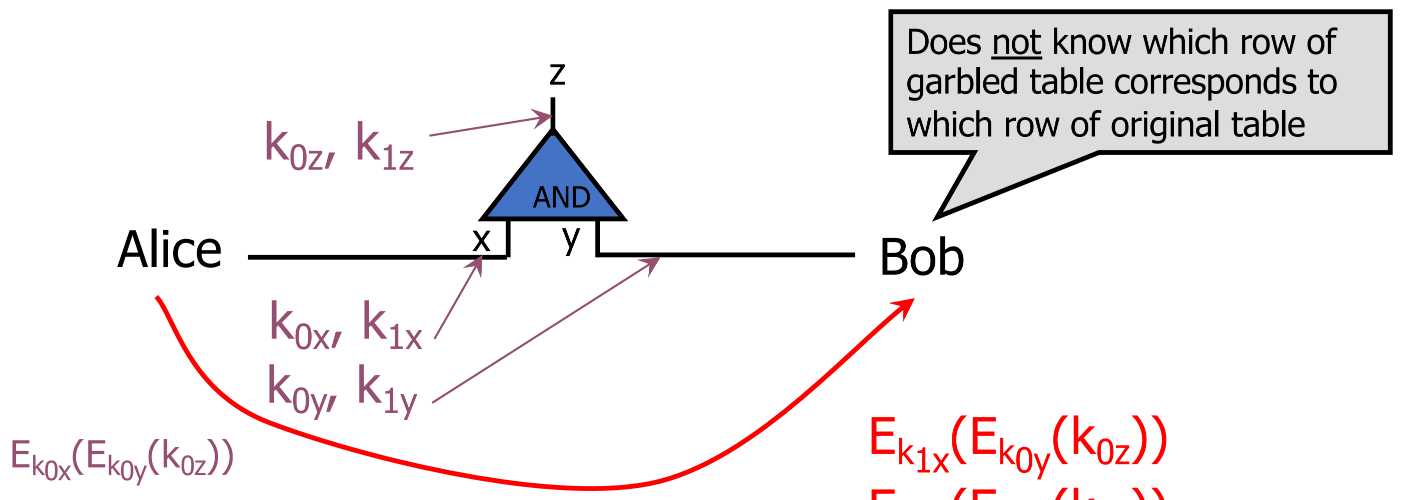
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$

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

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

3: Send Garbled Truth Table

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


Does not know which row of garbled table corresponds to which row of original table
z
k_{0z}, k_{1z}
AND
Alice
x
y
Bob
k_{0x}, k_{1x}
k_{0y}, k_{1y}

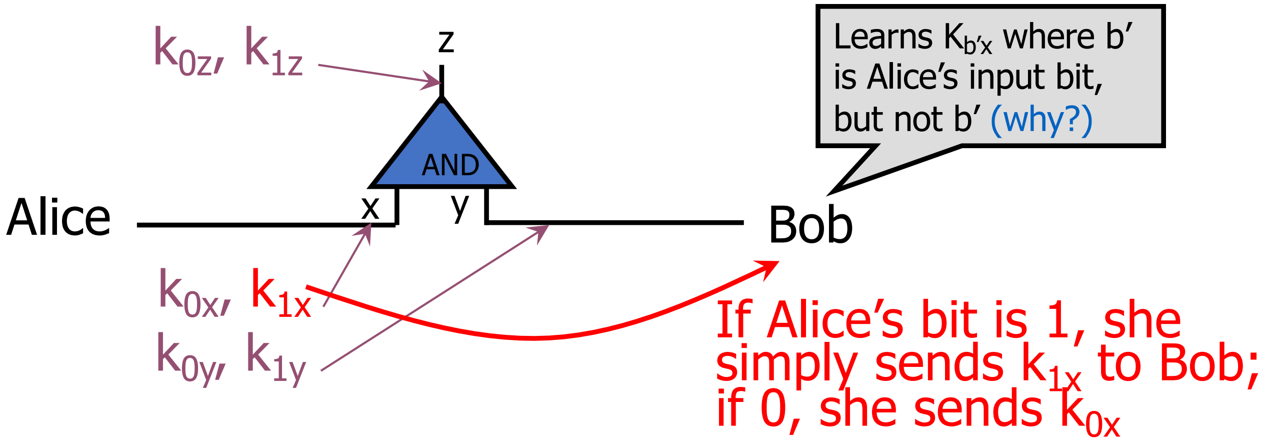
$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$

Garbled truth table:

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

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

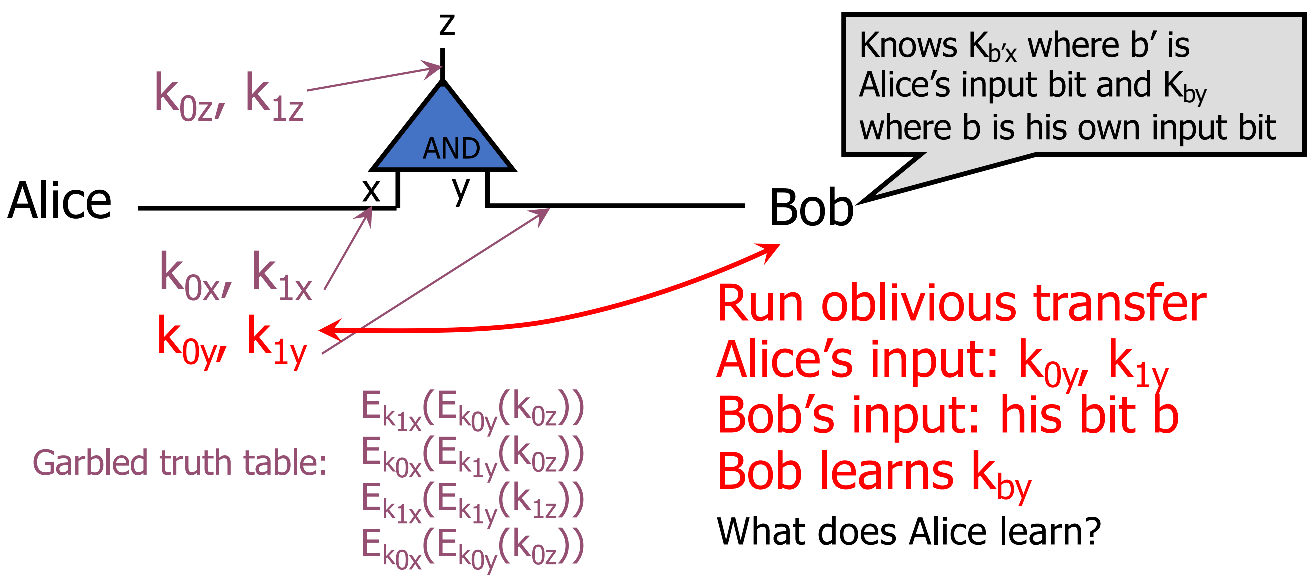

k_{0z}, k_{1z}
z
AND
x
y
Alice
Bob
k_{0x}, k_{1x}
k_{0y}, k_{1y}
Learns $K_{b'x}$ where b' is Alice's input bit, but not b' (why?)
If Alice's bit is 1, she simply sends k_{1x} to Bob; if 0, she sends k_{0x}

Garbled truth table:

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

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

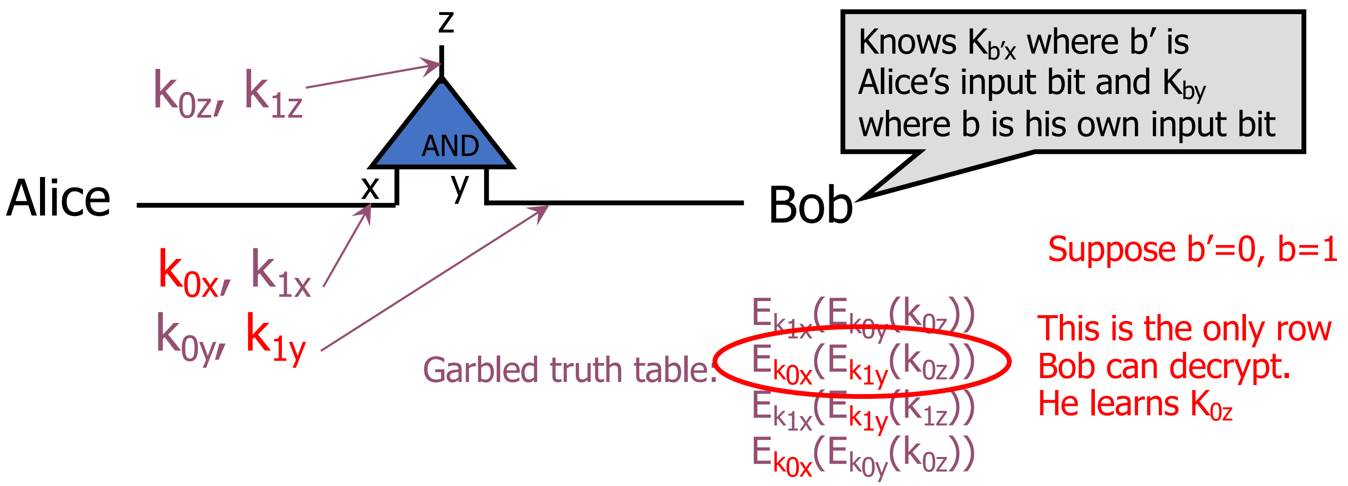


z
k_{0z}, k_{1z}
AND
Alice
x
y
Bob
Knows $K_{b'x}$ where b' is Alice's input bit and K_{by} where b is his own input bit
k_{0x}, k_{1x}
k_{0y}, k_{1y}
Garbled truth table:
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

Run oblivious transfer
Alice's input: k_{0y}, k_{1y}
Bob's input: his bit b
Bob learns k_{by}
What does Alice learn?

6: Evaluate One Garbled Gate

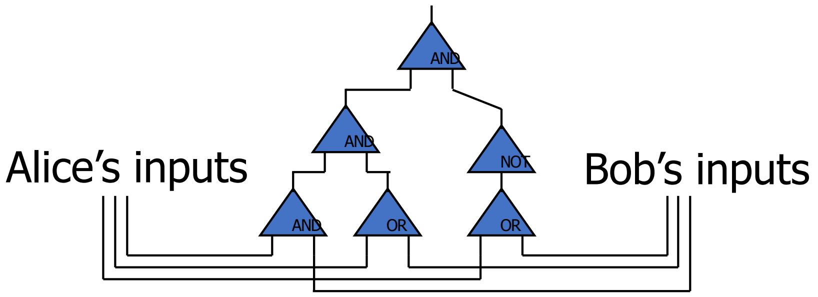
- Using the two keys that he learned, Bob decrypts exactly one of the output-wire keys
  - Bob does not learn if this key corresponds to 0 or 1
    - Why is this important?



z
k_{0z}, k_{1z}
AND
x
y
Alice
Bob
k_{0x}, k_{1x}
k_{0y}, k_{1y}
Knows $K_{b'x}$ where b' is Alice's input bit and K_{by} where b is his own input bit
Suppose b'=0, b=1
Garbled truth table:
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$
This is the only row Bob can decrypt. He learns K_{0z}

7: Evaluate Entire Circuit

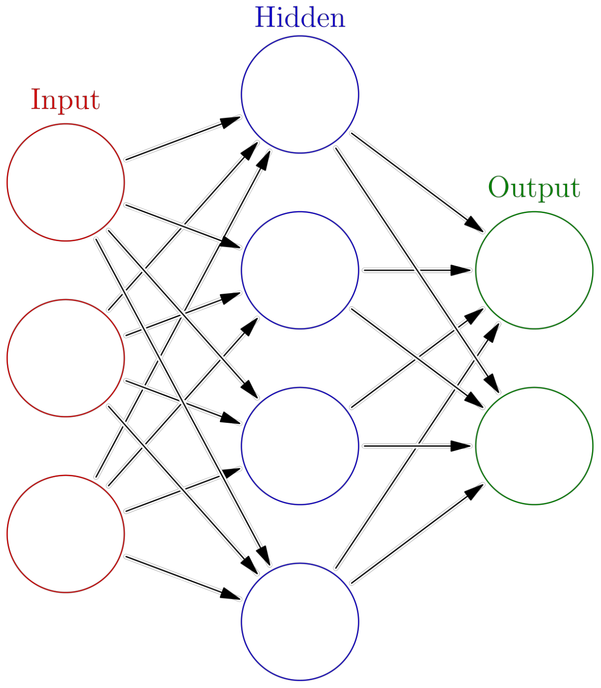
- In this way, Bob evaluates entire garbled circuit
  - For each wire in the circuit, Bob learns only one key
  - It corresponds to 0 or 1 (Bob does not know which)
    - Therefore, Bob does not learn intermediate values (why?)



- 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?

Network Parameters θ



$f(x,\theta)$

Input password x

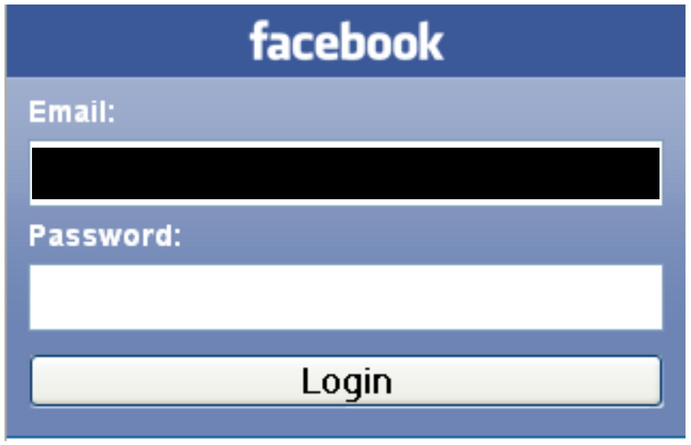
Brief Discussion of Yao's Protocol

- 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

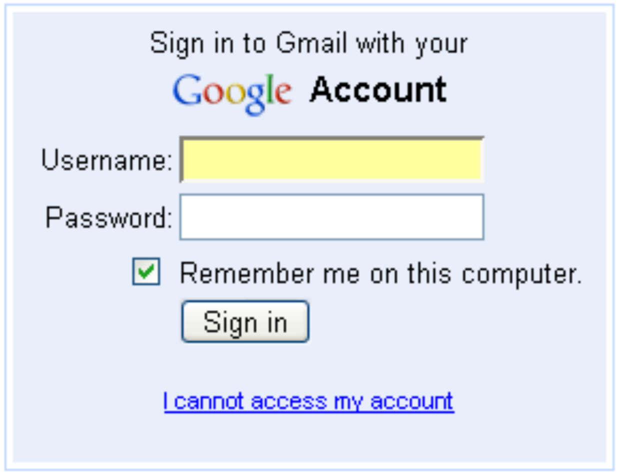

facebook

Email:

[redacted]

Password:

Login


Sign in to Gmail with your

Google Account

Username:

Password:

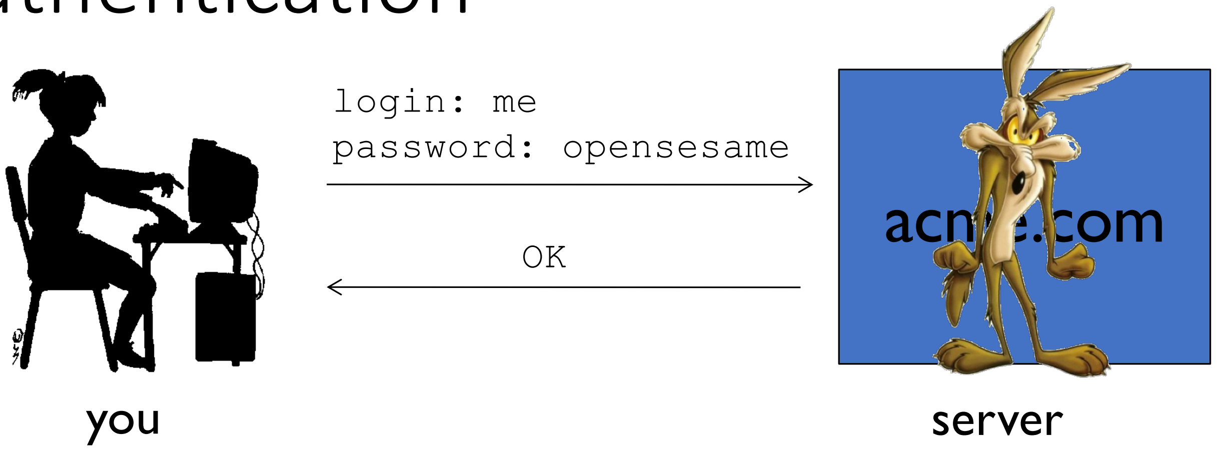
✓ Remember me on this computer.

Sign in

I cannot access my account

What happens when you type in your password?

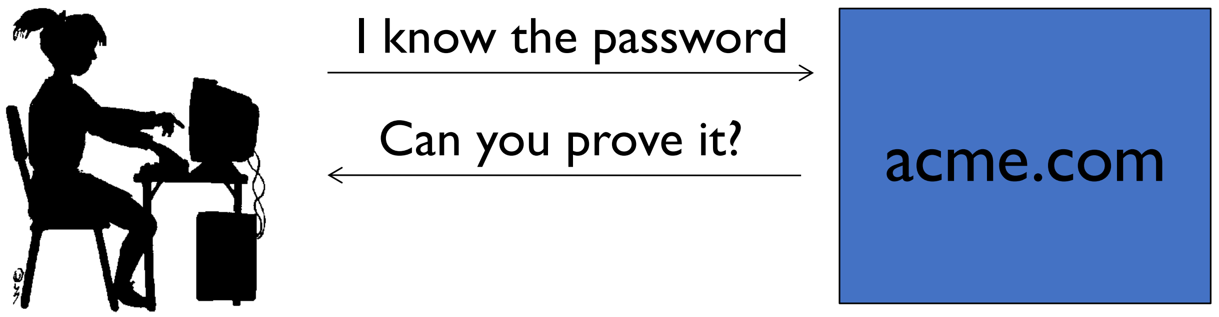
Naïve authentication



- The server knows your password

- So they can impersonate you at other web sites where you use the same password

“Zero-knowledge” authentication



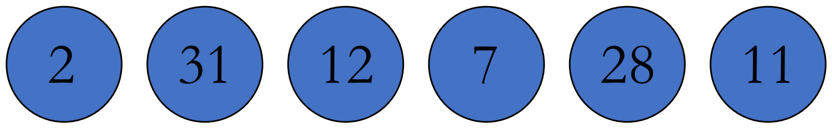
Can you convince the server that you know your password, without revealing it (or any other information)?

What is knowledge?

What is ignorance?

(lack of knowledge)

- Example 1: Tomorrow's lottery numbers



We are ignorant of them because they are random

What is ignorance?

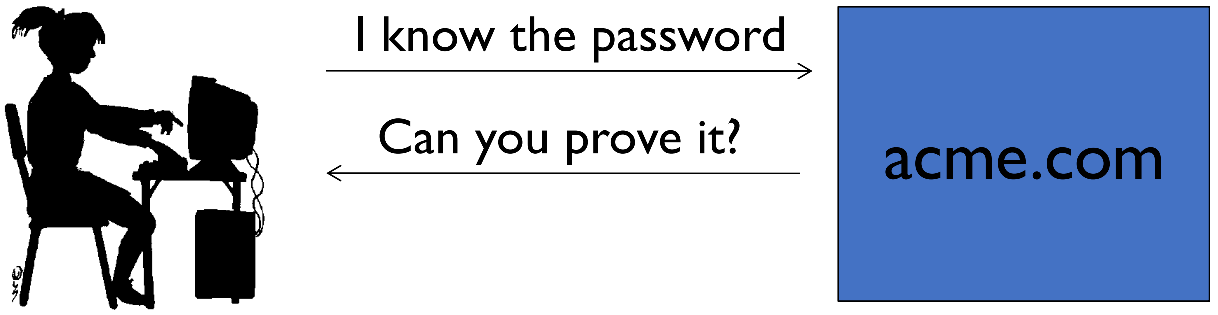
- Example 2: A difficult math problem

Prove that $\mathrm{P} \neq \mathrm{NP}$

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

Using ignorance to our advantage



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!

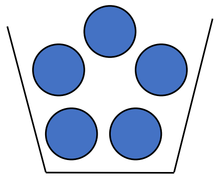
Goals

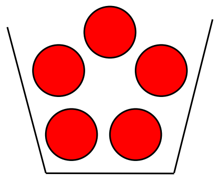
- Prover convinces verifier of statement.
    - “I know password for account gfanti”.
- Verifier cannot use what they learned to convince anyone else of the statement.
    - Prevent website from proving they know password for account gfanti.

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

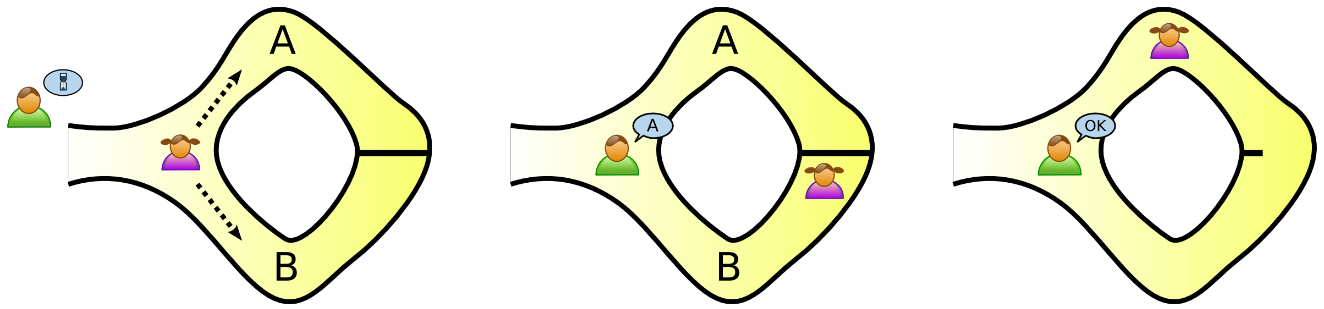
But I did not gain any other knowledge!

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

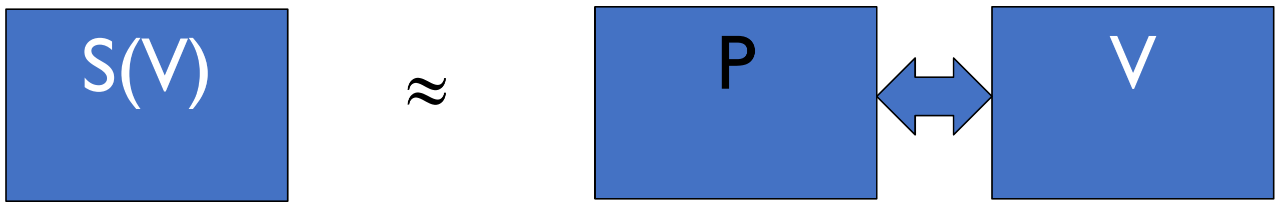
Cave password protocol

- 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 without interacting with the prover



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"

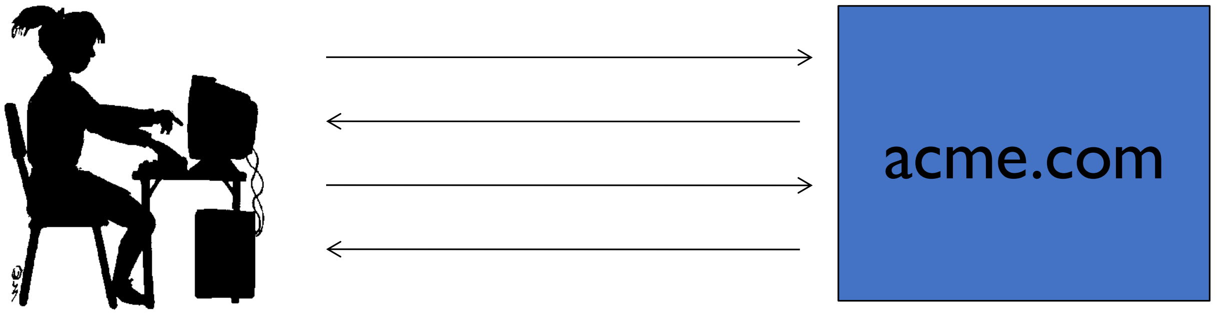
(P,V*) interaction transcript ≈ S(V*) transcript

- Simula
  1. W s "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*)

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



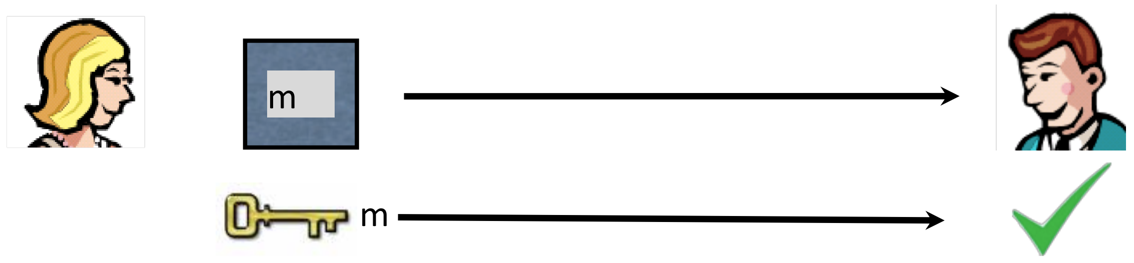
Oded Goldreich

Silvio Micali

Avi Wigderson

Important Notion: Commitments

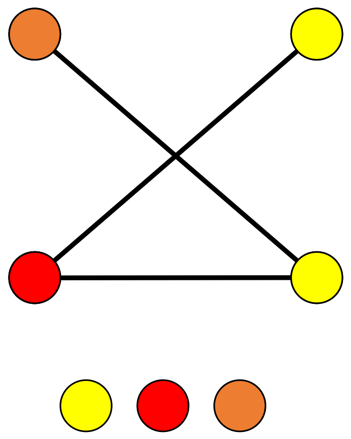
- Locked box analogy
  - Hiding – hard to tell which message is committed to
  - Binding – there is a unique message corresponding to each commitment



Graph coloring

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

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



- Theorem

3COL is NP-complete

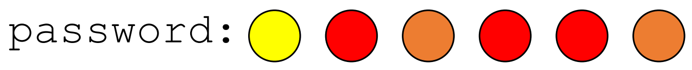
- However, it is easy to create 3-colored graphs.

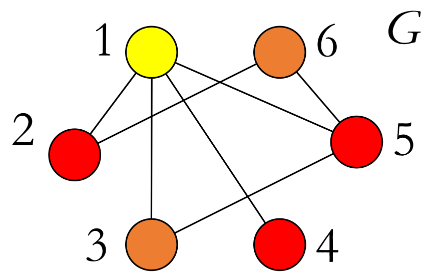
Password authentication via 3-coloring

- Step 0: When you register for the web service,

registration
acme.com

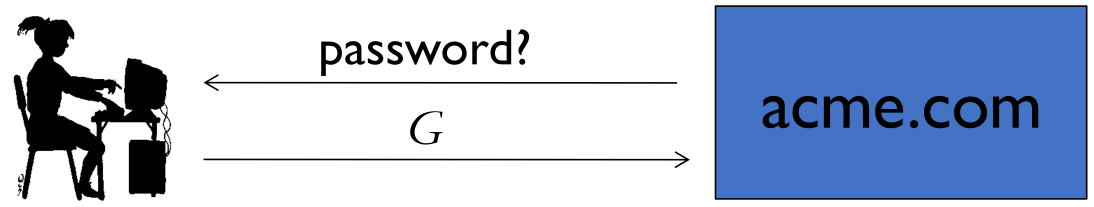
choose your password to be a valid 3-coloring of some (suitable) graph



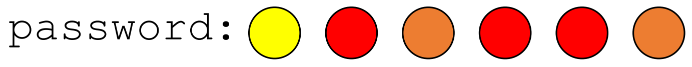


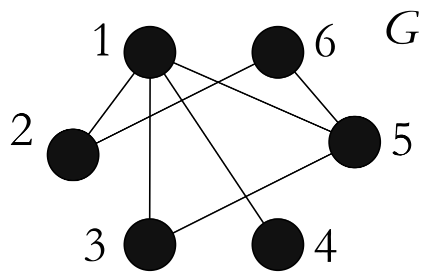
Password authentication via 3-coloring

- When the server asks for your password



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





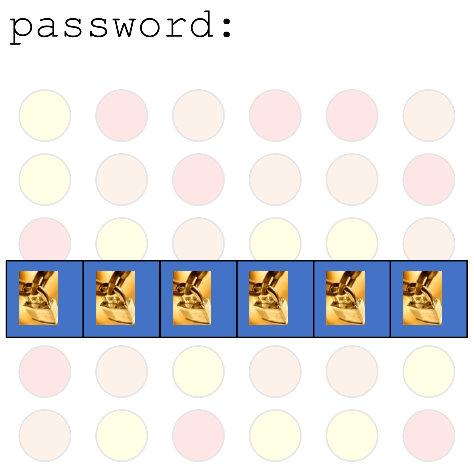
Intuition about registration phase

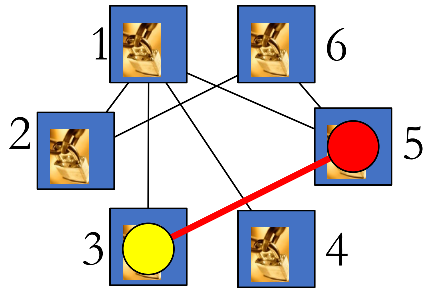
- Because 3-coloring is hard, the server will not be able to figure out your password (coloring) from G

- 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





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→ uted
   col
2. V*
3. P→ al)

(P,V*) interaction transcript ≈ S(V*) transcript

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