Course Business

- **Homework 5 Extended**
  - **Due Saturday @11PM on Gradescope**
- Practice Final Released Next Week

Homework 4 Statistics

Minimum Value	**72.00**
Maximum Value	**110.00**
Range	**38.00**
Average	**94.93**
Median	**96.00**
Standard Deviation	**12.02**

Cryptography
CS 555

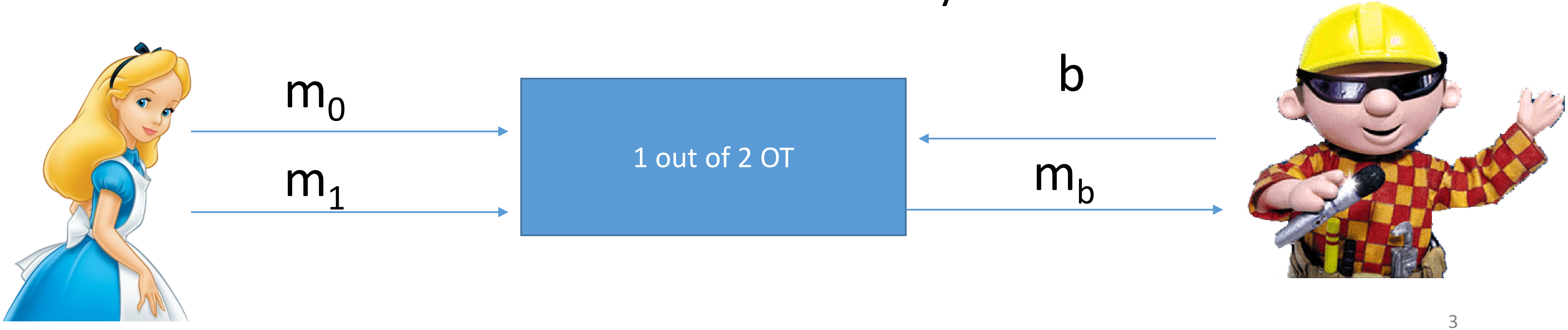
Week 15:

- Oblivious Transfer
- Yao's Garbled Circuits
- Zero-Knowledge Proofs

Readings: Katz and Lindell Chapter 10 & Chapter 11.1-11.2, 11.4

Oblivious Transfer (OT)

- 1 out of 2 OT
  - Alice has two messages m_0 and m_1
  - At the end of the protocol
    - Bob gets exactly one of m_0 and m_1
    - Alice does not know which one
- Oblivious Transfer with a Trusted Third Party

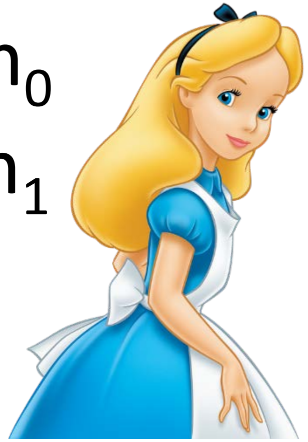


Bellare-Micali 1-out-of-2-OT protocol

- Oblivious Transfer without a Trusted Third Party
  - g is a generator for a prime order group G_q in which CDH problem is hard

m_0

m_1



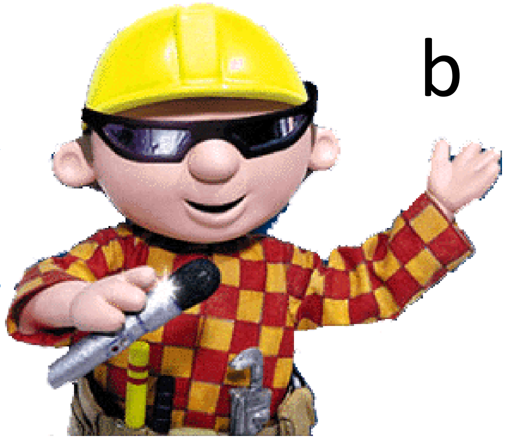
$c \leftarrow_R G_q$

c

z_0, z_1

$$C_0 = \left[g^{r_0}, H\left(z_0^{r_0}\right) \oplus m_0\right]$$
$$C_1 = \left[g^{r_1}, H\left(z_1^{r_1}\right) \oplus m_1\right]$$

b



$k \leftarrow_R Z_q$

$$z_b = g^k, z_{1-b} = cg^{-k}$$

Bob can decrypt C_b

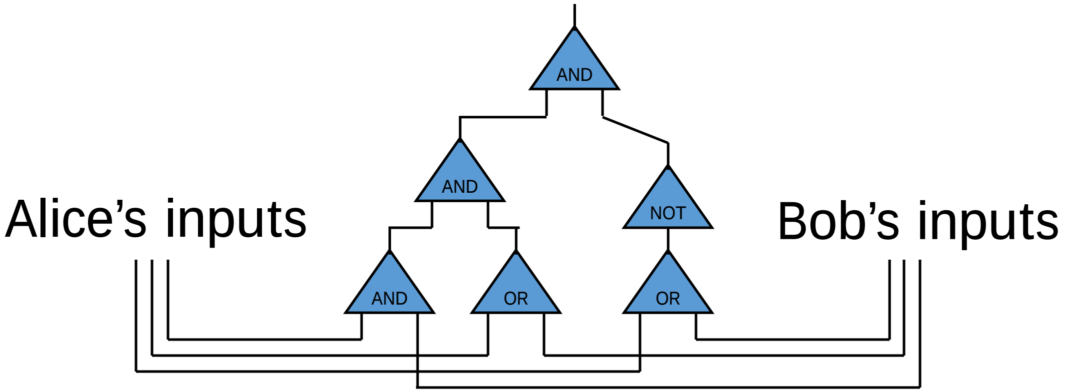
$$z_b^{r_b} = g^{kr_b}$$

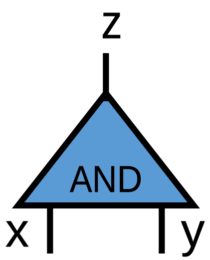
Yao's Protocol

Vitaly Shmatikov

Yao’s Protocol

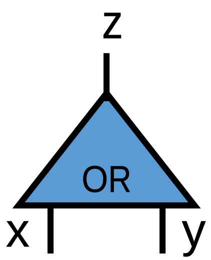
- Compute any function securely
  - … in the semi-honest model
- First, convert the function into a boolean circuit





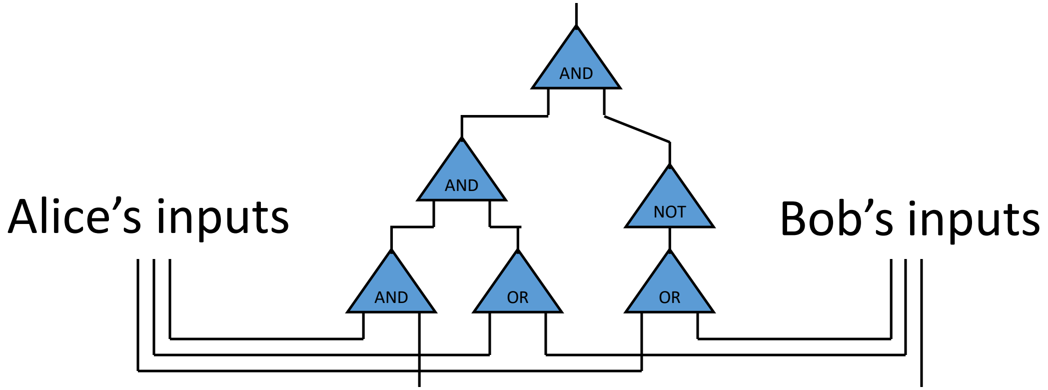
Truth table:

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



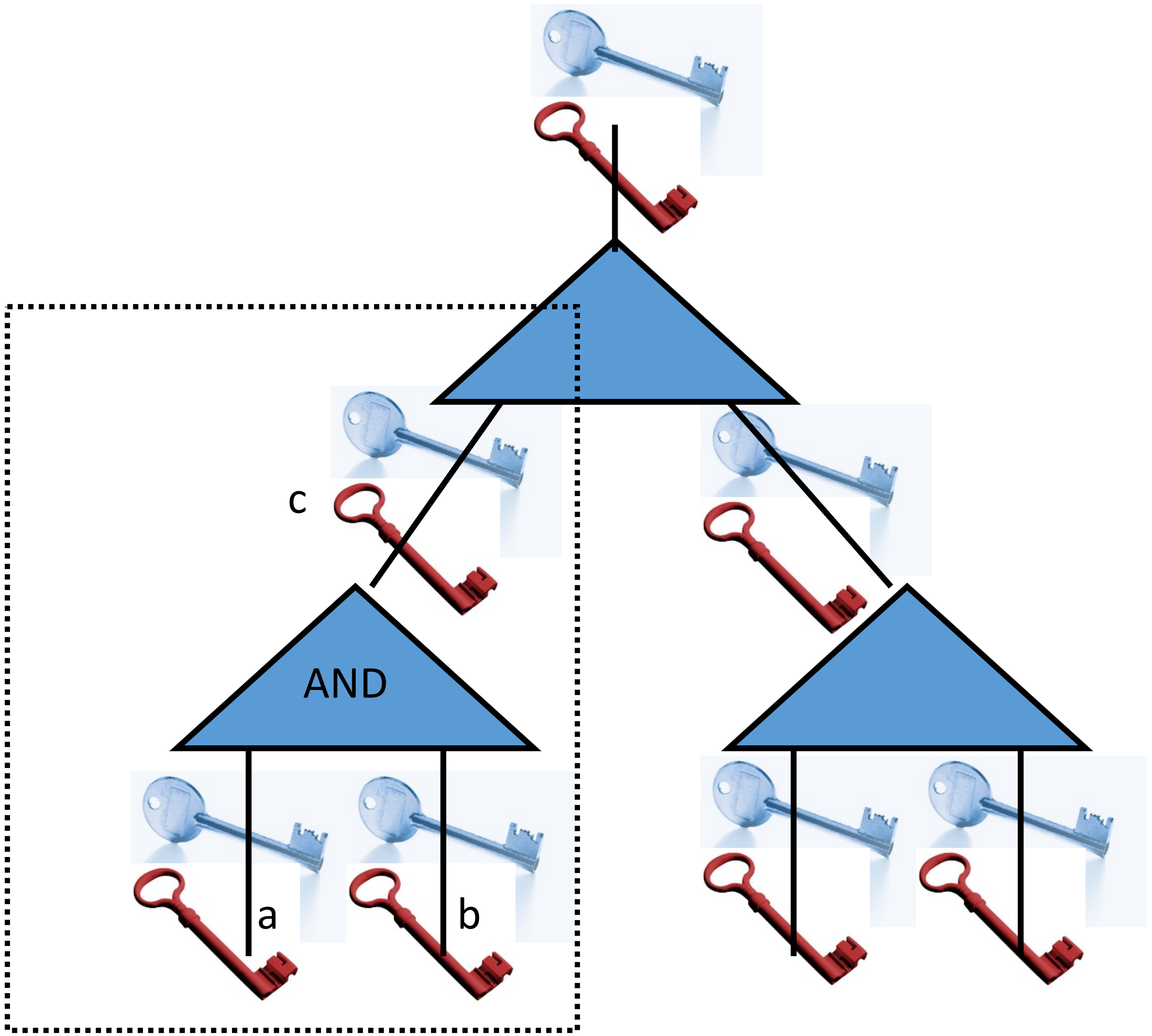
Overview:

1. Alice prepares "garbled" version **C'** of C
2. Sends "encrypted" form **x'** of her input x
3. Allows Bob to obtain "encrypted" form **y'** of his input y via OT
4. Bob can compute from **C'**,**x'**,**y'** the "encryption" **z'** of z=C(x,y)
5. Bob sends **z'** to Alice and she decrypts and reveals to him z

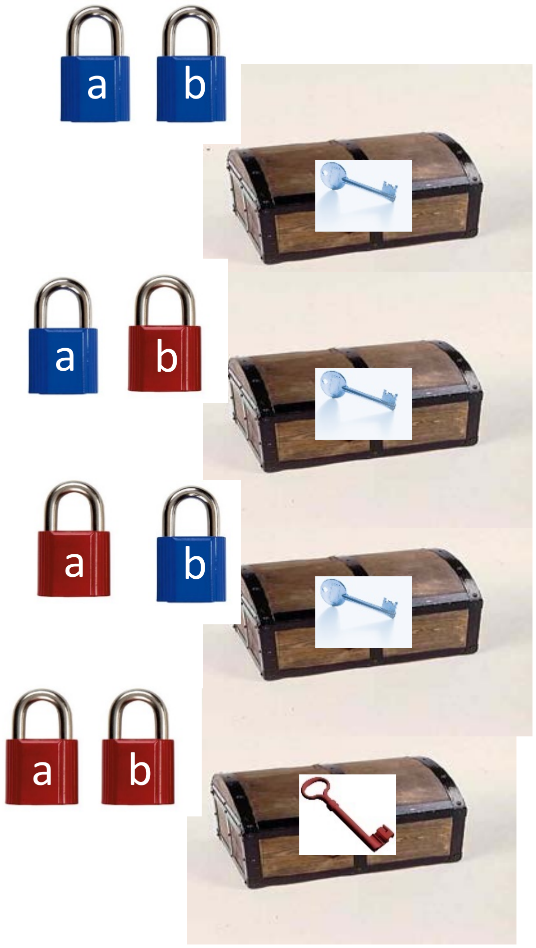
Crucial properties:

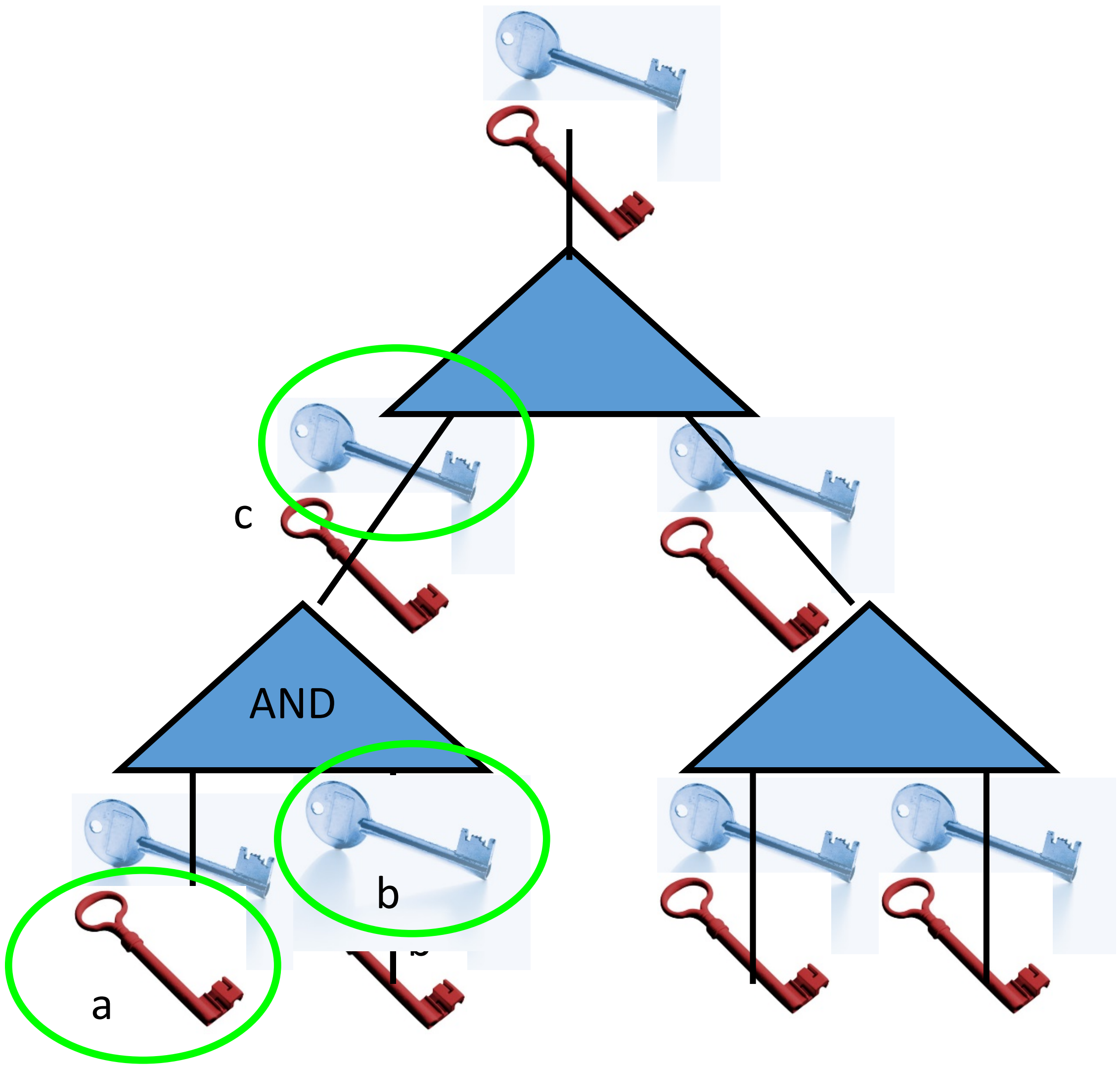
1. Bob never sees Alice's input x in unencrypted form.
2. Bob can obtain encryption of y without Alice learning y.
3. Neither party learns intermediate values.
4. Remains secure even if parties try to cheat.

Intuition



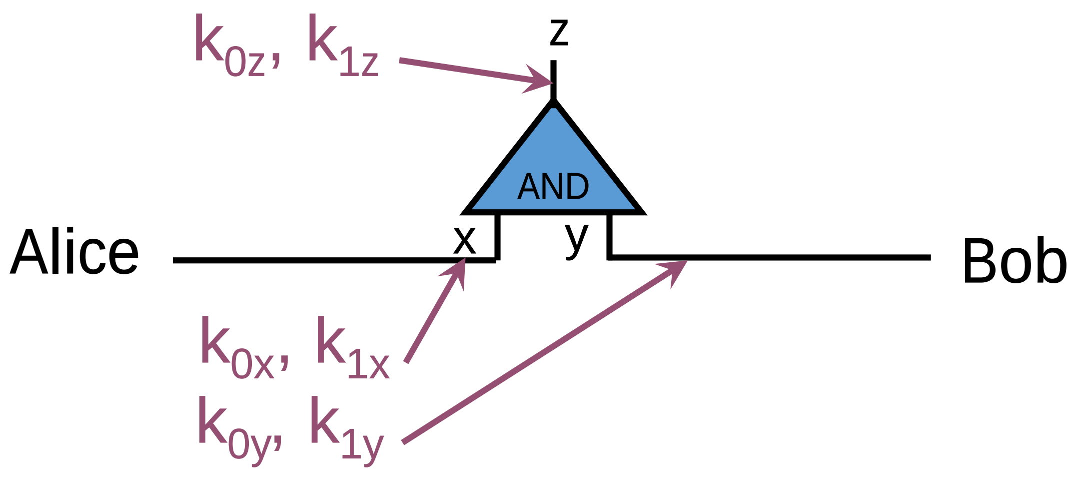
Intuition





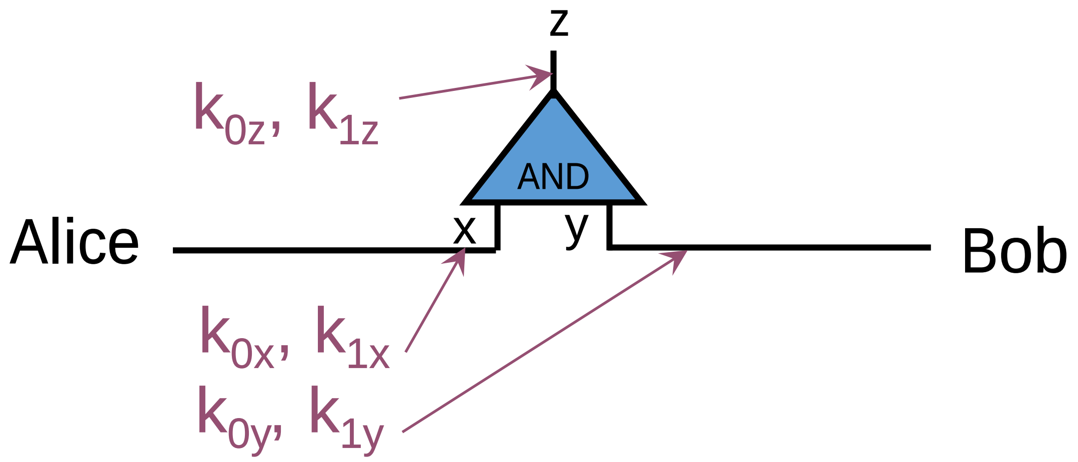
1: Pick Random Keys For Each Wire

- Next, evaluate <u>one gate</u> 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:

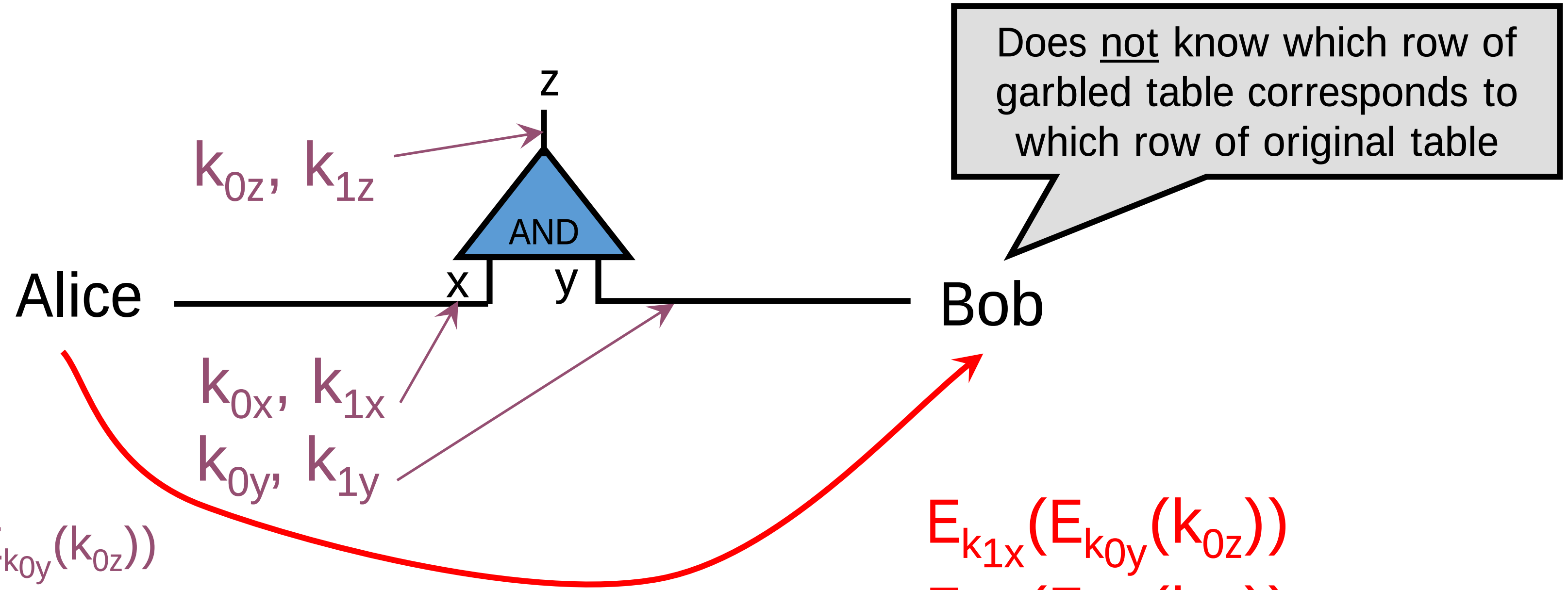
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

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



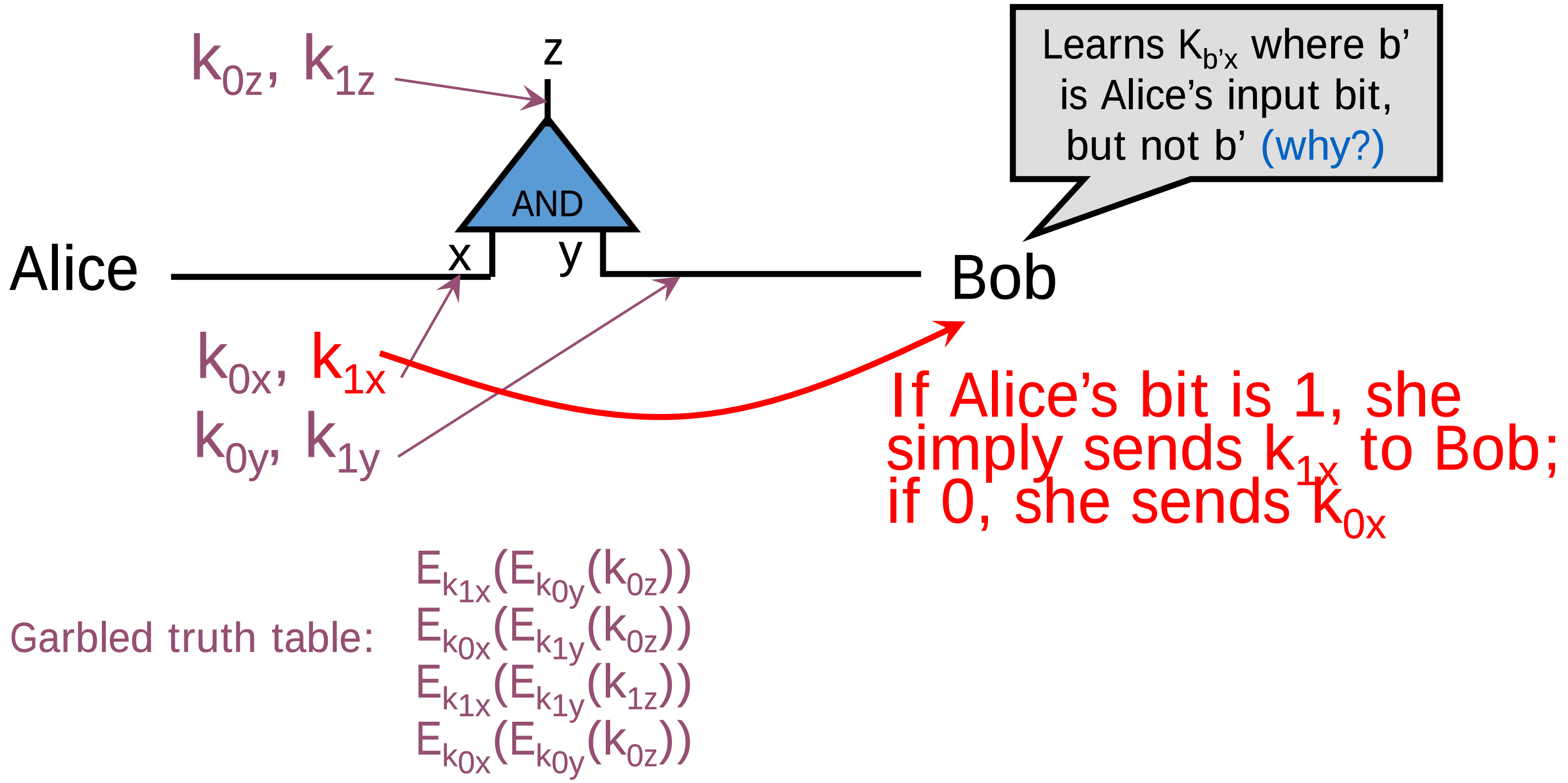
$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$

Garbled truth table:

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

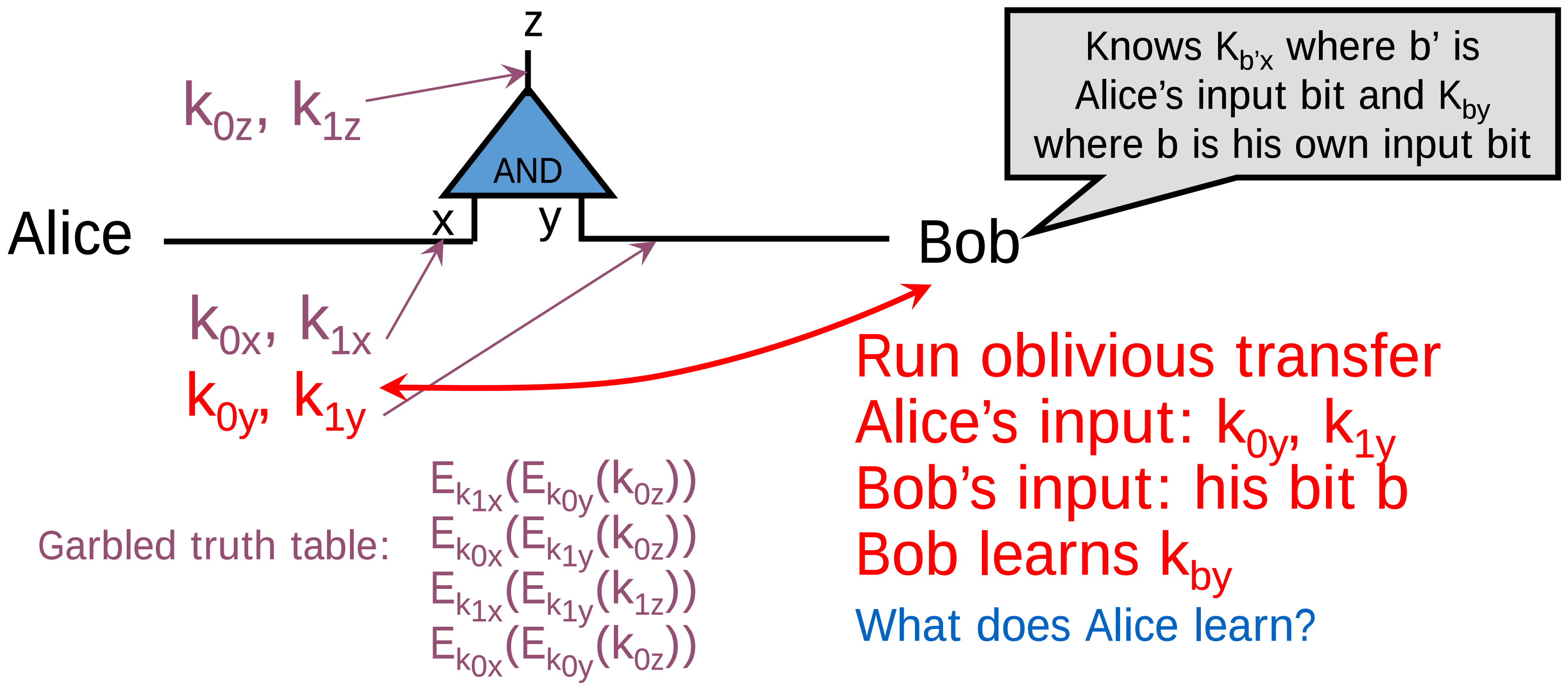
4: Send Keys For Alice's Inputs

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



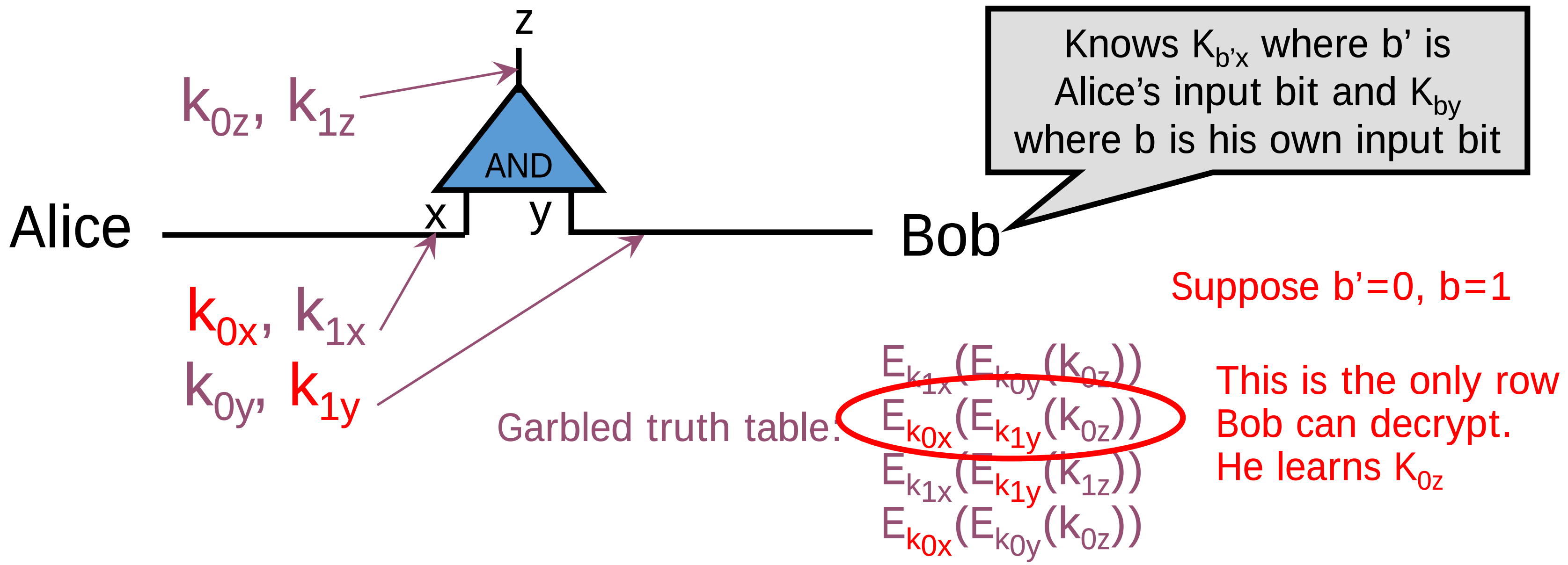
5: Use OT on Keys for Bob's Input

- 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



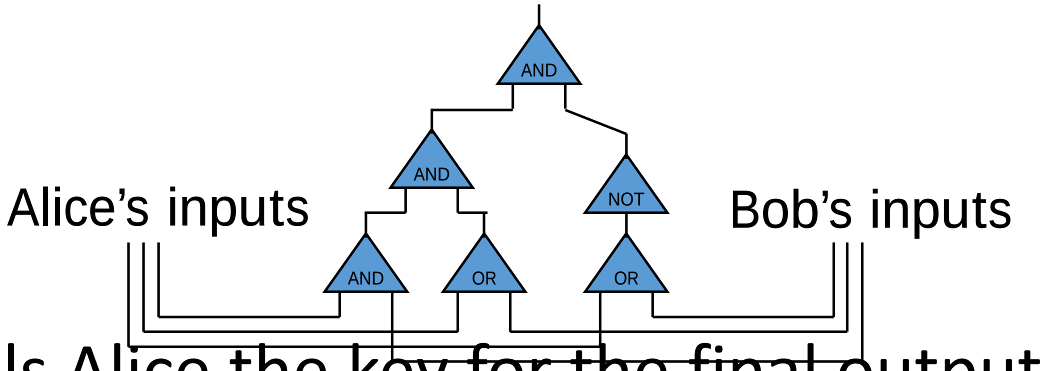
6: Evaluate Garbled Gate

- 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?



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 and she tells him if it corresponds to 0 or 1
  - Bob does <u>not</u> tell her intermediate wire keys (why?)

Different Circuits $f_A(x, y)$ and $f_B(x, y)$?

- (Regular Protocol for f_A): Alice Garbles circuit C_A computing f_A and Bob evaluates garbled circuit $C_A{}'$ and sends Alice garbled output $z_A{}'$.
  - Alice can ungarble the output z_A' to obtain $z_A \ = f_A(x, y)$ but does not send this value to Bob.

- (Swap Roles) Bob garbles circuit C_B computing f_B. Alice evaluates garbled circuit $C_B{}'$ and sends Bob the garbled output $z_B{}'$.
  - Bob can ungarble the output z_B' to obtain $z_B \ = f_B(x, y)$ but does not send this value to Alice.

Security (Semi-Honest Model)

- **Security**: Assuming that Alice and Bob are both semi-honest (follow the protocol) then there exist PPT simulators S_A and S_B s.t.

$$\{A_n\}_{n\in\mathbb{N}} \equiv_C \{S_A(n, x, f_A(x, y)\ )\}_{n\in\mathbb{N}}$$
$$\{B_n\}_{n\in\mathbb{N}} \equiv_C \left\{S_B\left(n, y, f_B(x, y)\right)\right\}_{n\in\mathbb{N}}$$

- **Remark**: Simulator S_A is only shown Alice's output $f_A(x, y)$ (similarly, S_B is only shown Bob's output $f_B(x, y)$)

Theorem (informal): If the oblivious transfer protocol is secure, and the underlying encryption scheme is CPA-secure then Yao's protocol is secure in the semi-honest adversary model.

Security (Semi-Honest Model)

- **Security**: Assuming that Alice and Bob are both semi-honest (follow the protocol) then there exist PPT simulators S_A and S_B s.t.

$$\{A_n\}_{n\in\mathbb{N}} \equiv_C \{S_A(n, x, f_A(x, y)\ )\}_{n\in\mathbb{N}}$$
$$\{B_n\}_{n\in\mathbb{N}} \equiv_C \left\{S_B\left(n, y, f_B(x, y)\right)\right\}_{n\in\mathbb{N}}$$

- **Simulating Bob’s View (Intuition):**
  - Garble the circuit following the honest algorithm Alice would use
  - Pick a random input x’ for Alice
    - Send Bob garbled circuits, plus garbled keys for x’
    - Allow Bob to obtain garbled keys for his input y via OT
  - Bob obtains garbled output $z_A{}'$ of $f_A(x', y)$ but cannot distinguish from garbled key for $f_A(x, y)$

Brief Discussion of Yao's Protocol

- Function must be converted into a circuit
  - For many functions, circuit will be huge
- If m gates in the circuit and n inputs from Bob, 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!

Fully Malicious Security?

1. Alice could initially garble the wrong circuit $C(x,y)=y$.
2. Given output of $C(x,y)$ Alice can still send Bob the output $f(x,y)$.
3. Can Bob detect/prevent this?

Fix: Assume Alice and Bob have both committed to their input: $c_A=\mathrm{com}(x|r_A)$ and $c_B=\mathrm{com}(y|r_B)$.

- Alice and Bob can use zero-knowledge proofs to convince other party that they are behaving honestly.
- **Example**: After sending a message A Alice proves that the message she just sent is the same message an honest party would have sent with input x s.t. $c_A=\mathrm{com}(x|r_A)$
- Here we assume that Alice and Bob have both committed to correct inputs (Bob might use y which does not represent his real vote etc… but this is not a problem we can address with cryptography)

Fully Malicious Security (Sketch)

- Assume Alice and Bob have both committed to their input: c_A=com(x|r_A) and c_B=com(y|r_B).
  - Here we assume that Alice and Bob have both committed to correct inputs (Bob might use y which does not represent his real vote etc… but this is not a problem we can address with cryptography)
  - Alice has c_B and can unlock c_A
  - Bob has c_A and can unlock c_B

1. Alice sets C_f = GarbleCircuit(f,r).
   1. Alice sends to Bob.
   2. Alice convinces Bob that C_f = GarbleCircuit(f,r) for some r (using a zero-knowledge proof)
2. For each original oblivious transfer if Alice's inputs were originally x_0,x_1
   1. Alice and Bob run OT with y_0,y_1 where y_i=$Enc_K(x_i)$
   2. Bob uses a zero-knowledge proof to convince Alice that he received the correct y_i (e.g. matching his previous commitment c_B)
   3. Alice sends K to Bob who decrypts y_i to obtain x_i

Course Feedback

- What did you like? What could be improved?
- Your feedback is valuable!

This statistic is not differentially private ☺

Dear Jeremiah Blocki,

Below is the current response rate in course(s) that you supervise or for which you are evaluated.

Course Num - Sec	Course Name	Number Expected	Number Received	Survey Open	Survey Close
CS55500 - LE1	Cryptography	27	0	Nov 26 1:40 AM	Dec 9 11:59 PM

Student access to complete evaluations will be closed at 11:59pm on December 9.

Visit http://www.purdue.edu/idp/courseevaluations/CE_Faculty.html at any time to view response rates.

Please reply to this message if you have any questions.

Sincerely,
Chantal Levesque-Bristol
Director, Center for Instructional Excellence

CS 555:Week 15: Zero-Knowledge Proofs

Computational Indistinguishability

- Consider two distributions X_ℓ and Y_ℓ (e.g., over strings of length ℓ).
- Let D be a distinguisher that attempts to guess whether a string s came from distribution X_ℓ or Y_ℓ.

The advantage of a distinguisher D is

$$Adv_{D,\ell} = \left|Pr_{s \leftarrow X_\ell}[D(s) = 1] - Pr_{s \leftarrow Y_\ell}[D(s) = 1]\right|$$

Definition: We say that an ensemble of distributions $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are <u>computationally indistinguishable</u> if for all PPT distinguishers D, there is a negligible function negl(n), such that we have

$$Adv_{D,n} \leq negl(n)$$

Computational Indistinguishability

Notation: $\{X_n\}_{n\in\mathbb{N}} \equiv_C \{Y_n\}_{n\in\mathbb{N}}$ means that the ensembles are computationally indistinguishable.

- Consider two d ℓ).
- Let D be a disti came from distribution X_ℓ

The advantage of a distinguisher D is

$$Adv_{D,\ell} = \left|Pr_{s\leftarrow X_\ell}[D(s)=1] - Pr_{s\leftarrow Y_\ell}[D(s)=1]\right|$$

Definition: We say that an ensemble of distributions $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are <u>computationally indistinguishable</u> if for all PPT distinguishers D, there is a negligible function negl(n), such that we have

$$Adv_{D,n} \le negl(n)$$

P vs NP

- **P** problems that can be solved in polynomial time

- **NP** --- problems whose solutions can be **verified** in polynomial time
  - Examples: SHORT-PATH, COMPOSITE, 3SAT, CIRCUIT-SAT, 3COLOR,
  - DDH
    - **Input:** $A = g^{x_1}, \mathrm{B} = g^{x_2}$ and Z
    - **Goal:** Decide if $\mathrm{Z} = g^{x_1 x_2}$ or $\mathrm{Z} \neq g^{x_1 x_2}$.
  - **NP-Complete** --- hardest problems in NP (e.g., all problems can be reduced to 3SAT)
- **Witness**
  - A short (polynomial size) string which allows a verify to check for membership
  - DDH Witness: x_1,x_2.

Zero-Knowledge Proof

Two parties: Prover P (PPT) and Verifier V (PPT)

(P is given witness for claim e.g.,)

- **Completeness:** If claim is true honest prover can always convince honest verifier to accept.
- **Soundness:** If claim is false then Verifier should reject with probability at least ½. (Even if the prover tries to cheat)
- **Zero-Knowledge:** Verifier doesn't learn anything about prover's input from the protocol (other than that the claim is true).
- Formalizing this last statement is tricky
- **Zero-Knowledge:** should hold even if the attacker is dishonest!

Zero-Knowledge Proof

Trans(1^n,V',P,x,w,r_p,r_v) transcript produced when V' and P interact

- V' is given input X (the problem instance e.g., $X = g^x$)
- P is given input X and w (a witness for the claim e.g., w=x)
- V' and P use randomness r_p and r_v respectively
- Security parameter is n e.g., for encryption schemes, commitment schemes etc…

$\boldsymbol{X_n} =$ **Trans(1^n,V',P,x,w)** is a distribution over transcripts (over the randomness r_p,r_v)

(Blackbox Zero-Knowledge): There is a PPT simulator S such that for every V' (possibly cheating) S, with oracle access to V', can simulate X_n without a witness w. Formally,

$$\{X_n\}_{n\in\mathbb{N}} \equiv_C \left\{S^{V'(.)}(x, 1^n)\right\}_{n\in\mathbb{N}}$$

Zero-Knowledge Proof

Trans(1^n,V',P,x,w,r_p,r_v) transcript produced when V' and P interact

- V' is given input x (the problem instance e.g., $A = g^{x_1}$, $\mathrm{B} = g^{x_2}$ and z_b)
- P … r the claim e.g., …
- V' … espectively
- Se … yption schemes, …

X_n … n over transcript

Simulator S is not given witness w

Oracle V'(x,trans) will output the next message V' would output given current transcript trans

(Blackbox Zero-Knowledge): There is a PPT simulator S such that for every V' (possibly cheating) S, with oracle access to V', can simulate X_n without a witness w. Formally,

$$\{X_n\}_{n\in\mathbb{N}} \equiv_C \{S^{V'(.)}(x, 1^n)\}_{n\in\mathbb{N}}$$

Zero-Knowledge Proof for Discrete Log Solution

Bob (verifier);
A

$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c} \in \{\mathbf{0}, \mathbf{1}\}$$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y} + x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ B = g^r\ \ and\ AB = C} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ C = g^r}\ \ and\ \boldsymbol{AB = C} \\ 0 & otherwise \end{cases}$$

Alice (prover);
x s.t.
$A = g^x$,
$B = g^y$,
(random y)

Claim: There is some integer x such that $A = g^x$

Zero-Knowledge Proof for Discrete Log Solution

$$\boldsymbol{B = g^{y}, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c} \in \{\boldsymbol{0}, \boldsymbol{1}\}$$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y} + x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} \boldsymbol{1} & \boldsymbol{if\ c = 0\ and\ B = g^{r}\ \ and\ AB = C} \\ \boldsymbol{1} & \boldsymbol{if\ c = 1\ and\ C = g^{r}}\ \ and\ \boldsymbol{AB = C} \\ 0 & otherwise \end{cases}$$

Bob (verifier);
$A = g^{x},$

Alice (prover);
x
$A = g^{x},$
$B = g^{y},$
(random y)

Correctness: If Alice and Bob are honest then Bob will always accept

Zero-Knowledge Proof for Discrete Log Solution

$\boxed{B = g^y}, C = g^{x+y}$

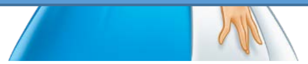
Case 1: Challenge (c=0)

$challenge\ c \in \{0, 1\}$

Response $r = \begin{cases} \boxed{y} & if\ c = 0 \\ y + x & if\ c = 1 \end{cases}$

Bob (verifier);
$A = g^x,$

$Decision\ d = \begin{cases} 1 & if\ c = 0\ and\ \boxed{B = g^r\ and\ AB = C} \\ 1 & if\ c = 1\ and\ C = g^r\ and\ AB = C \\ 0 & otherwise \end{cases}$



Alice (prover);
x
$A = g^x,$
$B = g^y,$
(random y)

Correctness: If Alice and Bob are honest then Bob will always accept

Zero-Knowledge Proof for Discrete Log Solution

Case 2: Challenge (c=1)

$$\boldsymbol{B = g^y}, \boxed{\boldsymbol{C = g^{x+y}}}$$

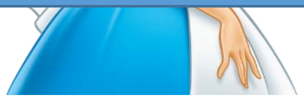
$$\boldsymbol{challenge\ c} \in \{\mathbf{0}, \mathbf{1}\}$$

Response $r = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boxed{\boldsymbol{y} + x} & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ B = g^r\ \ and\ AB = C} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and} \boxed{\boldsymbol{C = g^r}\ \ and\ \boldsymbol{AB = C}} \\ 0 & otherwise \end{cases}$$

Bob (verifier);

$A = g^x$,



Alice (prover);

x

$A = g^x$,

$B = g^y$,

(random y)

Correctness: If Alice and Bob are honest then Bob will always accept

Zero-Knowledge Proof for Discrete Log Solution

$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c \in \{0,1\}}$$

Response $r = \begin{cases} y & if\ c = 0 \\ y + x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} 1 & if\ c = 0\ and\ B = g^r\ and\ AB = C \\ 1 & if\ c = 1\ and\ C = g^r\ and\ AB = C \\ 0 & otherwise \end{cases}$$

Bob (verifier);
$A = g^x,$

Alice (prover);
x
$A = g^x,$
$B = g^y,$
(random y)

Soundness: If $A \neq g^x$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Zero-Knowledge Proof fo

Assume that AB=C, now
If $B = g^y$ and $C = g^{x+y}$ for some x,y then $A = g^x$

Bob (verifier);
$A = g^x$,

$\boldsymbol{B = g^y, C = g^{x+y}}$

$\boldsymbol{challenge\ c \in \{0,1\}}$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ y + x & if\ c = 1 \end{cases}$

$\boldsymbol{Decision\ d} = \begin{cases} \boldsymbol{1} & \boldsymbol{if\ c = 0\ and\ B = g^r\ and\ AB = C} \\ \boldsymbol{1} & \boldsymbol{if\ c = 1\ and\ C = g^r\ and\ AB = C} \\ 0 & otherwise \end{cases}$

Alice (prover);
x
$A = g^x$,
$B = g^y$,
(random y)

Soundness: If $A \neq g^x$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Case 1: for all r $B \neq g^r$

$\rightarrow Pr[reject] \geq Pr[c=0] = \frac{1}{2}$

Assume that AB=C, now If $B = g^y$ and $C = g^{x+y}$ for some x,y then $A = g^x$

$\boldsymbol{B = g^y, C = g^{x+y}}$

$\boldsymbol{challenge\ c \in \{0,1\}}$

$\boxed{\text{Response } r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c=0} \\ \boldsymbol{y}+x & if\ c=1 \end{cases}$

$\boldsymbol{Decision\ d} = \begin{cases} \boldsymbol{1} & \boldsymbol{if\ \boxed{c=0\ and\ B=g^r}\ and\ AB=C} \\ \boldsymbol{1} & \boldsymbol{if\ c=1\ and\ C=g^r\ and\ AB=C} \\ 0 & otherwise \end{cases}$

Bob (verifier);
$A = g^x$,

Alice (prover);
x
$A = g^x$,
$B = g^y$,
(random y)

Soundness: If $A \neq g^x$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Case 2: for all r $C \neq g^r$

$\rightarrow Pr[reject] \geq Pr[c = 1] = \frac{1}{2}$

Assume that AB=C, now

If $B = g^y$ and $C = g^{x+y}$ for some x,y then $A = g^x$

$\boldsymbol{B = g^y, C = g^{x+y}}$

$\boldsymbol{challenge\ c \in \{0, 1\}}$

Response r $= \begin{cases} y & if\ c = 0 \\ y + x & if\ c = 1 \end{cases}$

Bob (verifier);

$A = g^x,$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ B = g^r\ and\ AB = C} \\ \mathbf{1} & \boldsymbol{if\ \boxed{c = 1\ and\ C = g^r}\ and\ AB = C} \\ 0 & otherwise \end{cases}$$

Alice (prover);

x

$A = g^x,$

$B = g^y,$

(random y)

Soundness: If $A \neq g^x$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Zero-Knowledge Proof for Discrete Log Solution

$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$\boldsymbol{challenge\ c = V'(A, (B, C)) \in \{0, 1\}}$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ y + x & if\ c = 1 \end{cases}$

$\boldsymbol{Decision\ d = V'(A, (B, C), c, r)}$

Dishonest (verifier);
$A = g^x,$

Alice (honest);
x
$A = g^x,$
$B = g^y,$
(random y)

Transcript: $\boldsymbol{View_{V'}} = (A, (B, C), c, r, d)$

Zero-Knowledge Proof for Discrete Log Solution

Dishonest (verifier);
$A = g^x$,

$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c = V'(A,(B,C)) \in \{0,1\}}$$

Response $r = \begin{cases} y & if\ c = 0 \\ y + x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d = V'(A,(B,C),c,r)}$$

Alice (honest);
x
$A = g^x$,
$B = g^y$,
(random y)

Zero-Knowledge: For all PPT V' exists PPT Sim s.t $\boldsymbol{View_{V'}} \equiv_C \mathrm{Sim}^{V'(.)}(A)$

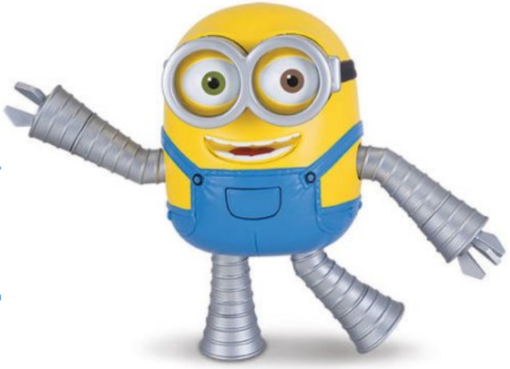
Zero-Knowledge Proof for Discrete Log Solution

$$\begin{cases} B = g^y, C = AB & \text{if b=0} \\ B = \frac{C}{A}, C = g^y & otherwise \end{cases}$$

$challenge\ c = V'(A, (B, C)) \in \{0, 1\}$

Response $r = \begin{cases} y & if\ c{=}b \\ \perp & otherwise \end{cases}$

$Decision\ d = V'(A, (B, C), c, r)$



Dishonest (verifier);
$A = g^x,$

Simulator
Cheat bit b,
$A = g^x,$
$B = g^y,$
(random y)

Zero-Knowledge: For all PPT V' exists PPT Sim s.t $\boldsymbol{View}_{V'} \equiv_C \mathrm{Sim}^{V'(.)}(A)$

Zero-Knowledge Proof for Discrete Log Solution

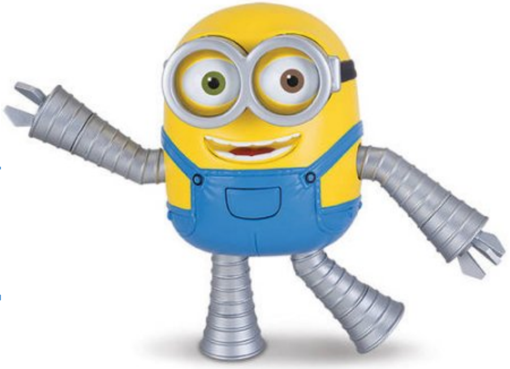
$$\begin{cases} B = g^y, C = AB & \text{if b=0} \\ B = \frac{C}{A}, C = g^y \ otherwise & \end{cases}$$

$challenge\ c = V'(A, (B, C)) \in \{0, 1\}$

Response $r = \begin{cases} y & if\ c{=}b \\ \perp & otherwise \end{cases}$

$Decision\ d = V'(A, (B, C), c, r)$

Dishonest (verifier);
$A = g^x,$



Simulator
Cheat bit b,
$A = g^x,$
$B = g^y,$
(random y)

Zero-Knowledge: Simulator can produce identical transcripts (Repeat until $r \neq \perp$)

Zero-Knowledge Proof for Discrete Log Solution

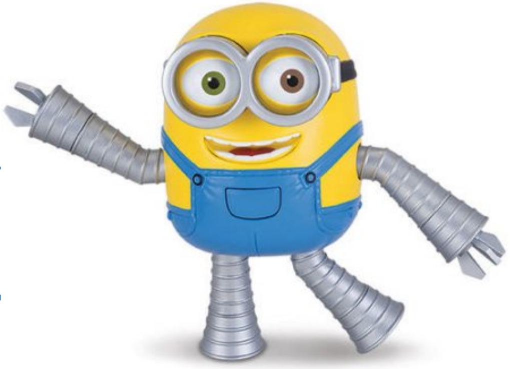
Dishonest (verifier);
$A = g^x$,

$$\begin{cases} \boldsymbol{B = g^y, C = AB} & \textbf{if b=0} \\ \boldsymbol{B = \frac{C}{A}, C = g^y} & \boldsymbol{otherwise} \end{cases}$$

$\boldsymbol{challenge\ c = V'(A, (B, C)) \in \{0, 1\}}$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c{=}b} \\ \bot & \boldsymbol{otherwise} \end{cases}$

$\boldsymbol{Decision\ d = V'(A, (B, C), c, r)}$



Simulator
Cheat bit b,
$A = g^x$,
$B = g^y$,
(random y)

Zero-Knowledge: If $A = g^x$ for some x then $\boldsymbol{View_{V'}} \equiv_C \mathrm{Sim}^{V'(.)}(A)$

Zero-Knowledge Proof for Square Root mod N

Bob (verifier);

z

$$M = zy^2 \; mod \; N$$

$$\boldsymbol{challenge\ c} \in \{\mathbf{0}, \mathbf{1}\}$$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y}x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ M = zr^2} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ M = r^2}\ mod\ N \\ 0 & otherwise \end{cases}$$

Alice (prover);

X

$z = x^2$ mod N

(random y)

Completeness: If Alice knows x such $z = x^2$ mod N then Bob will always accept

Zero-Knowledge Proof for Square Root mod N

$$M = zy^2 \ mod\ N$$

$$\boldsymbol{challenge\ c} \in \{\mathbf{0}, \mathbf{1}\}$$

$$\textbf{Response } \boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y}x & if\ c = 1 \end{cases}$$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ M = zr^2} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ M = r^2}\ mod\ N \\ 0 & otherwise \end{cases}$$

Bob (verifier);

z

Alice (prover);

X

$z = x^2$ mod N

(random y)

Soundness: If $z \neq x^2$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Zero-Knowledge Proof for Square Root mod N

Bob (verifier);

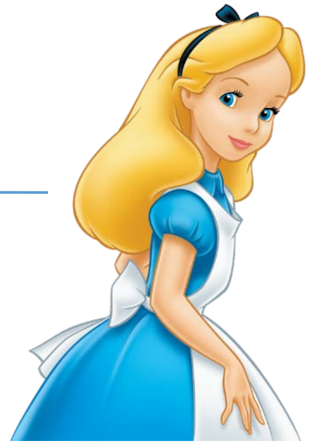
z

$$M = zy^2 \ mod \ N$$

$\boldsymbol{challenge\ c} \in \{\mathbf{0}, \mathbf{1}\}$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y}x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ M = zr^2} \text{ mod N} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ M = r^2} \text{ mod N} \\ 0 & otherwise \end{cases}$$



Alice (prover);

X

$z = x^2$ mod N

(random y)

Zero-Knowledge: How does the simulator work?

Zero-Knowledge Proof vs. Digital Signature

- Digital Signatures are transferrable
  - E.g., Alice signs a message m with her secret key and sends the signature σ to Bob. Bob can then send (m, σ) to Jane who is convinced that Alice signed the message m.

- Are Zero-Knowledge Proofs transferable?
  - Suppose Alice (prover) interacts with Bob (verifier) to prove a statement (e.g., z has a square root modulo N) in Zero-Knowledge.
  - Let $\boldsymbol{View_V}$ be Bob's view of the protocol.
  - Suppose Bob sends $\boldsymbol{View_V}$ to Jane.
  - Should Jane be convinced of the statement (e.g., z has a square root modulo N)>

Non-Interactive Zero-Knowledge Proof (NIZK)

$M_1, \ldots M_k$ where $M_i = y_i^2 z \bmod N$

$\boldsymbol{challenges\ c = (c_1, \ldots, c_k) = H}(M_1, \ldots M_k)$

Responses $r_1, \ldots, r_k$ where $r_i = \begin{cases} y_i & \text{if } c_i = 0 \\ y_i x & \text{if } c_i = 1 \end{cases}$

Bob (verifier);

z

$$\boldsymbol{Decision\ d} = \prod_i d_i \quad \boldsymbol{where\ d_i} = \begin{cases} 1 & \boldsymbol{if\ c_i = 0\ and\ M_i = r_i^2 z} \bmod N \\ 1 & \boldsymbol{if\ c_i = 1\ and\ M_i = r_i^2} \bmod N \\ 0 & otherwise \end{cases}$$

Alice (prover);

X

$z = x^2 \bmod N$

(random $y_1, \ldots, yk$)

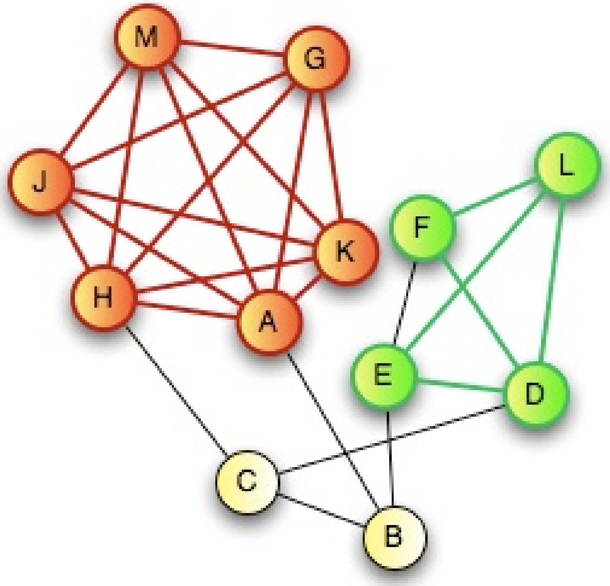
Simulator Power: Can program the random oracle

NIZK Security (Random Oracle Model)

- Simulator is given statement to proof (e.g., z has a square root modulo N)
- Simulator must output a proof π'_z and a random oracle H'

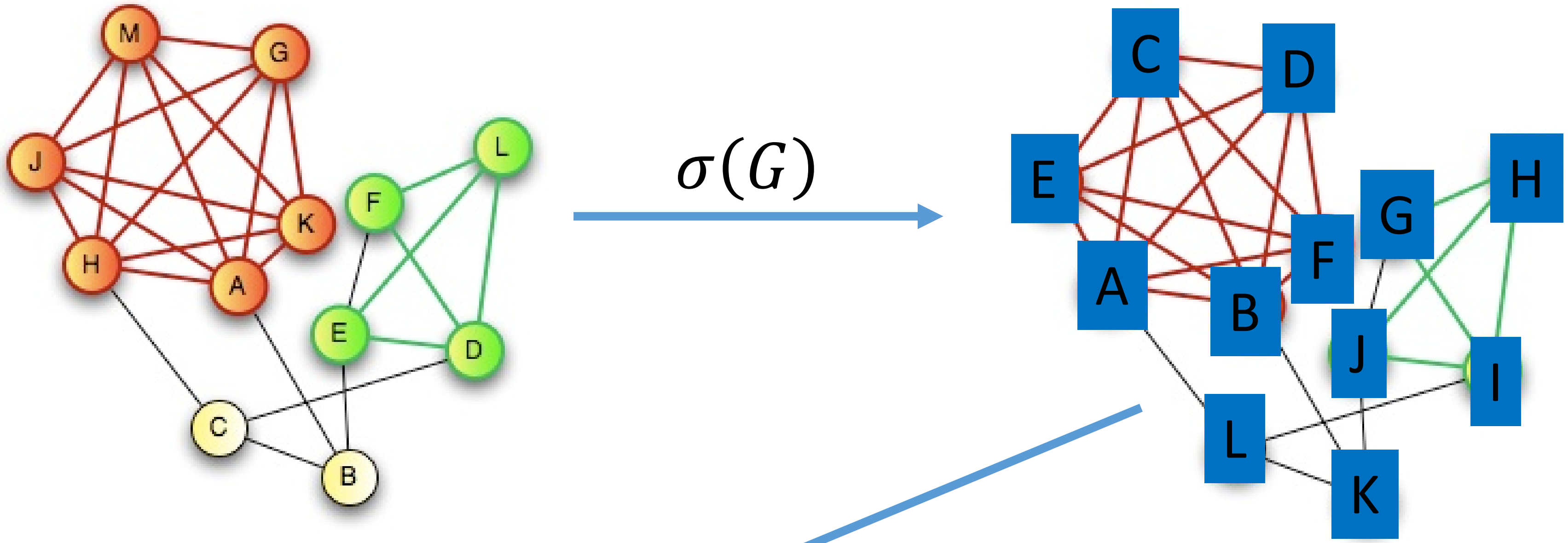
- Distinguisher D
  - World 1 (Simulated): Given z, π'_z and oracle access to H'
  - World 2 (Honest): Given z, π_z (honest proof) and oracle access to H
  - Advantage: $\mathrm{ADV_D} = |Pr[D^H(\mathrm{z}, \pi_z) = 1] - Pr[D^{H'}(\mathrm{z}, \pi'_z) = 1]|$
- **Zero-Knowledge:** Any PPT distinguisher D should have negligible advantage.
- NIZK proof π_z is transferrable (contrast with interactive ZK proof)

Zero-Knowledge Proof for all NP



- CLIQUE
  - Input: Graph G=(V,E) and integer k>0
  - Question: Does G have a clique of size k?
- CLIQUE is NP-Complete
  - Any problem in NP reduces to CLIQUE
  - A zero-knowledge proof for CLIQUE yields proof for all of NP via reduction
- Prover:
  - Knows k vertices $v_1,\ldots,v_k$ in G=(V,E) that form a clique

Zero-Knowledge Proof for all NP



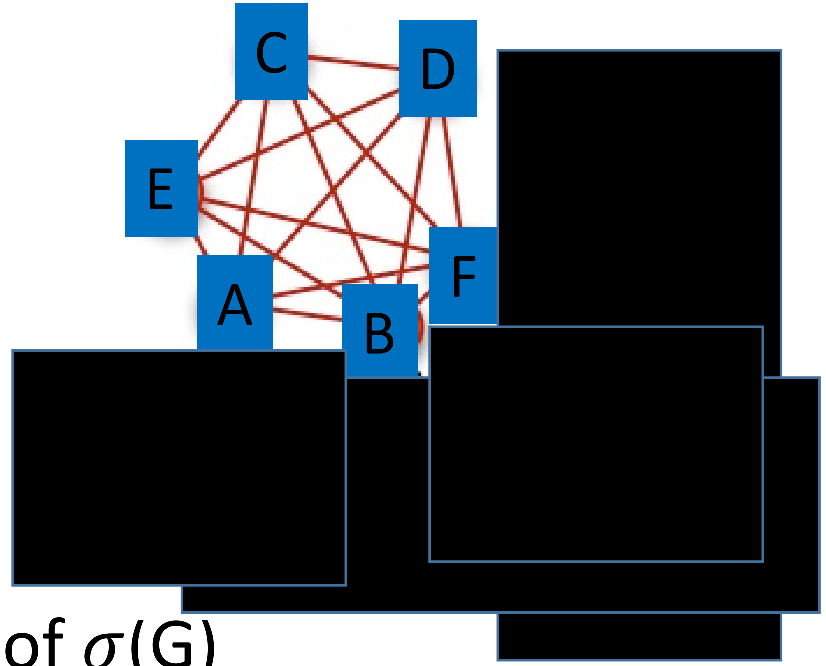
Adjacency matrix $A_{\sigma(G)}$

$$\begin{array}{cc} & \begin{array}{ccc} A & & L \end{array} \\ \begin{array}{c} A \\ \\ L \end{array} & \begin{pmatrix} 0 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{pmatrix} \end{array}$$

Commitment to $A_{\sigma(G)}$

$$\begin{array}{cc} & \begin{array}{ccc} A & & L \end{array} \\ \begin{array}{c} A \\ \\ L \end{array} & \begin{pmatrix} Com(0, r_{A,A}) & \cdots & Com(1, r_{A,L}) \\ \vdots & \ddots & \vdots \\ Com(1, r_{L,A}) & \cdots & Com(0, r_{L,L}) \end{pmatrix} \end{array}$$

Zero-Knowledge Proof for all NP



- Prover:
  - Knows k vertices $v_1,\ldots,v_k$ in G=(V,E) that for a clique

1. Prover commits to a permutation σ over V
2. Prover commits to the adjacency matrix $A_{\sigma(G)}$ of σ(G)
3. Verifier sends challenge c (either 1 or 0)
4. If c=0 then prover reveals σ and adjacency matrix $A_{\sigma(G)}$
   1. Verifier confirms that adjacency matrix is correct for σ(G)
5. If c=1 then prover reveals the submatrix formed by first rows/columns of $A_{\sigma(G)}$ corresponding to $\sigma(v_1),\ldots,\sigma(v_k)$
   1. Verifier confirms that the submatrix forms a clique.

Zero-Knowledge Proof for all NP

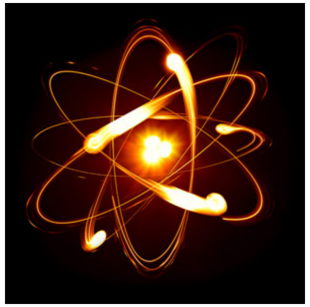
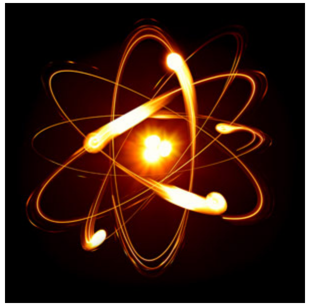
- **Completeness**: Honest prover can always make honest verifier accept
- **Soundness**: If prover commits to adjacency matrix $A_{\sigma(G)}$ of σ(G) and can reveal a clique in submatrix of $A_{\sigma(G)}$ then G itself contains a k-clique. Proof invokes binding property of commitment scheme.
- **Zero-Knowledge:** Simulator cheats and either commits to wrong adjacency matrix or cannot reveal clique. Repeat until we produce a successful transcript. Indistinguishability of transcripts follows from hiding property of commitment scheme.

Secure Multiparty Computation (Adversary Models)

- Semi-Honest (“honest, but curious”)
  - All parties follow protocol instructions, but…
  - dishonest parties may be curious to violate privacy of others when possible
- Fully Malicious Model
  - Adversarial Parties may deviate from the protocol arbitrarily
    - Quit unexpectedly
    - Send different messages
  - It is much harder to achieve security in the fully malicious model
- Convert Secure Semi-Honest Protocol into Secure Protocol in Fully Malicious Mode?
  - Tool: Zero-Knowledge Proofs
  - Prove: My behavior in the protocol is consistent with honest party

CS 555:Week 15: Hot Topics

Shor's Algorithm

- Quantum Algorithm to Factor Integers

- Running Time

$$O((\log N)^2(\log \log N)(\log \log \log N))$$

- Building Quantum Circuits is challenging, but...
- RSA is broken if we build a quantum computer
  - Current record: Factor 21=3x7 with Shor's Algorithm
  - **Source:** Experimental Realisation of Shor's Quatum Factoring Algorithm Using Quibit Recycling (https://arxiv.org/pdf/1111.4147.pdf)

https://en.wikipedia.org/wiki/Shor%27s_algorithm

Quantum Resistant Crypto

- Symmetric key primitives are believed to be safe
- …but Grover's Algorithm does speed up brute-force attacks significantly (2^n vs $\sqrt{2^n}$)
  - Solution: Double Key Lengths

- Integer Factoring, Discrete Log and Elliptic Curve Discrete Log are not safe
  - All public key encryption algorithms we have covered
  - RSA, RSA-OAEP, El-Gamal,….

https://en.wikipedia.org/wiki/Lattice-based_cryptography

Post Quantum Cryptography

- Symmetric key primitives are believed to be safe
- …but Grover's Algorithm does speed up brute-force attacks significantly (2^n vs $\sqrt{2^n}$)
  - Solution: Double Key Lengths
- Hashed Based Signatures
  - Lamport Signatures and extensions
- Lattice Based Cryptography is a promising approach for Quantum Resistant Public Key Crypto
  - Ring-LWE
  - NTRU

https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

Fully Homomorphic Encryption (FHE)

- Idea: Alice sends Bob $Enc_{PK_A}(x_1), \ldots, Enc_{PK_A}(x_n)$

$$Enc_{PK_A}(x_i) + Enc_{PK_A}(x_j) = Enc_{PK_A}(x_i + x_j)$$

and

$$Enc_{PK_A}(x_i) \times Enc_{PK_A}(x_j) = Enc_{PK_A}(x_i \times x_j)$$

- Bob cannot decrypt messages, but given a circuit C can compute

$$Enc_{PK_A}(C(x_1, \ldots, x_n))$$

- Proposed Application: Export confidential computation to cloud

https://simons.berkeley.edu/talks/shai-halevi-2015-05-18a (Lecture by Shai Halevi)

Fully Homomorphic Encryption (FHE)

- Idea: Alice sends Bob $Enc_{PK_A}(x_1), \ldots, Enc_{PK_A}(x_n)$
- Bob cannot decrypt messages, but given a circuit C can compute
$$Enc_{PK_A}\left(C(x_1, \ldots, x_n)\right)$$
- We now have candidate constructions!
  - Encryption/Decryption are polynomial time
  - …but expensive in practice.
  - Proved to be CPA-Secure under plausible assumptions

- Remark 1: Partially Homomorphic Encryption schemes cannot be CCA-Secure. Why not?

https://simons.berkeley.edu/talks/shai-halevi-2015-05-18a (Lecture by Shai Halevi)

Partially Homomorphic Encryption

- Plain RSA is multiplicatively homomorphic

$$Enc_{PK_A}(x_i) \times Enc_{PK_A}(x_j) = Enc_{PK_A}(x_i \times x_j)$$

- But not additively homomorphic

- Pallier Cryptosystem

$$Enc_{PK_A}(x_i) \times Enc_{PK_A}(x_j) = Enc_{PK_A}(x_i + x_j)$$

$$\left(Enc_{PK_A}(x_i)\right)^k = Enc_{PK_A}(k \times x_j)$$

- Not same as FHE, but still useful in multiparty computation

https://en.wikipedia.org/wiki/Paillier_cryptosystem

Program Obfuscation (Theoretical Cryptography)

- Program Obfuscation
  - Idea: Alice obfuscates a circuit C and sends C to Bob
  - Bob can run C, but cannot learn "anything else"
  - Lots of applications…
- Indistinguishability Obfuscation
  - Theoretically Possible
    - In the sense that $f(n) = 2^{100000000} n^{100000}$ is technically polynomial time
- Secure Hardware Module (e.g., SGX) can be viewed as a way to accomplish this in practice
  - Must trust third party (e.g., Intel)

https://simons.berkeley.edu/talks/amit-sahai-2015-05-19a (Lecture by Amit Sahai)

Differential Privacy

Release Aggregate Statistics?

- Question 1: How many people in this room have cancer?
- Question 2: How many students in this room have cancer?
- The difference (A1-A2) exposes my answer!

Differential Privacy: Definition

- n people
- Neighboring datasets:
  - Replace x with x’

Name	CS Prof?	…	STD?

Name	CS Prof?	…	STD?
Bjork	-1	…	???

D

[DMNS06, DKMMN06]

$$(\epsilon, \delta)\text{-differential privacy: } \forall (D, D'),\ \forall S$$

$$\Pr[\mathsf{ALG}(D) \in S] \leq e^{\epsilon} \Pr[\mathsf{ALG}(D') \in S] + \delta$$

Differential Privacy vs Cryptography

- ε is not negligibly small.
- We are not claiming that, when D and D' are neighboring datasets,
$$\boldsymbol{Alg(D) \equiv_C Alg(D')}$$
- Otherwise, we would have $\boldsymbol{Alg(X) \equiv_C Alg(Y')}$ for any two data-sets X and Y.
- Why?

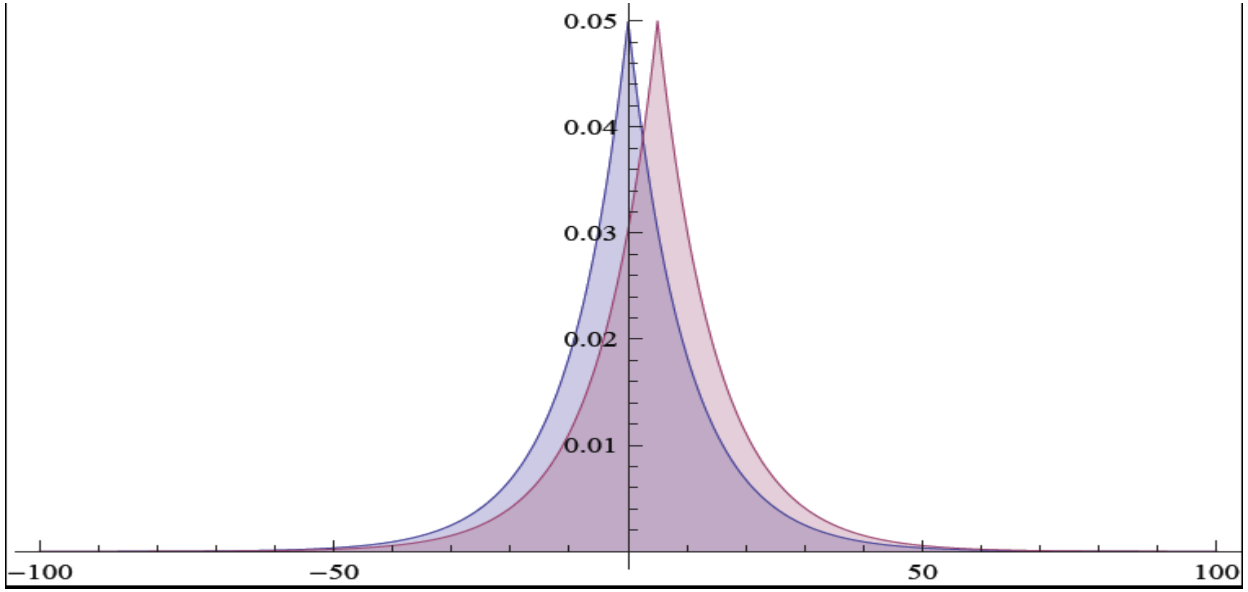
- Cryptography
  - Insiders/Outsiders
  - Only those with decryption key(s) can reveal secret
  - Multiparty Computation: Alice and Bob learn nothing other than f(x,y)

Traditional Differential Privacy Mechanism

Theorem: Let $\mathrm{D} = (x_1, \ldots, x_n) \in \{0,1\}^n$

$$\mathrm{A}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i + \mathrm{Lap}\left(\frac{1}{\varepsilon}\right),$$

satisfies $(\varepsilon, 0)$-differential privacy. (True Answer, Noise)



Google Scholar

differential privacy

About 3,000,000 results (**0.06** sec)

Articles

Case law

My library

Any time

Since 2016

Since 2015

Since 2012

Custom range...

Differential privacy: A survey of results

C Dwork - International Conference on Theory and Applications of …, 2008 - Springer

Abstract Over the past five years a new approach to **privacy**-preserving data analysis has born fruit [13, 18, 7, 19, 5, 37, 35, 8, 32]. This approach differs from much (but not all!) of the related literature in the statistics, databases, theory, and cryptography communities, in that **...**

Cited by 2557 Related articles All 32 versions Web of Science: 365 Cite Save More

Mechanism design via **differential privacy**

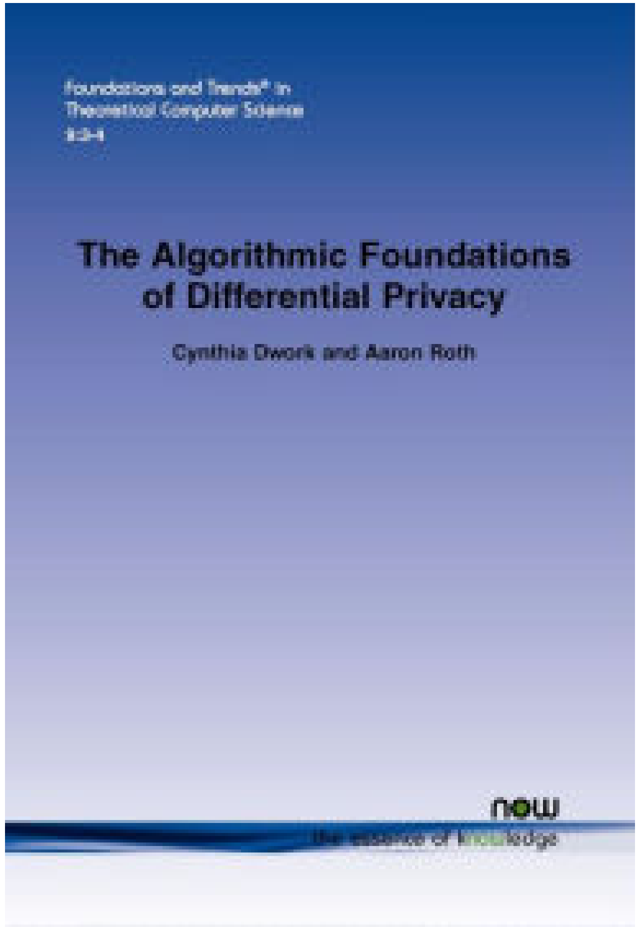
F McSherry, K Talwar - … of Computer Science, 2007. FOCS'07. …, 2007 - ieeexplore.ieee.org

Abstract We study the role that **privacy**-preserving algorithms, which prevent the leakage of specific information about participants, can play in the design of mechanisms for strategic agents, which must encourage players to honestly report information. Specifically, we **...**

Cited by 708 Related articles All 25 versions Cite Save

Microsoft® Research

Resources

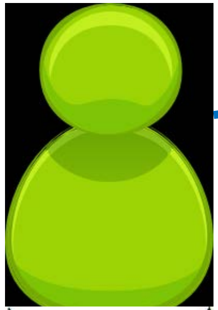


BARNES & NOBLE

- $99

Free PDF:
https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf

Password Storage and Key Derivation Functions



jblocki, 123456

Username	Salt	Hash
jblocki	89d978034a3f6	85e23cfe0021f584e3db87aa72630a9a2345c062

SHA1(12345689d978034a3f6)=85e23cfe0021f584e3db87aa72630a9a2345c062

+

Offline Attacks: A Common Problem

- Password breaches at major companies have affected ~~millions~~ **billions** of user accounts.

Offline Attacks: A Common Problem

- Password breaches at major companies have affected ~~millions~~ **billions**

TECH

Yahoo Triples Estimate of Breached Accounts to 3 Billion

Company disclosed late last year that 2013 hack exposed private information of over 1 billion users

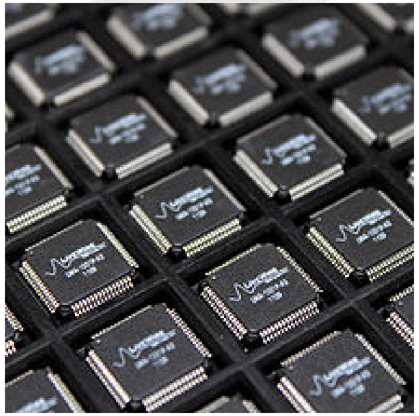
By Robert McMillan and Ryan Knutson

Updated Oct. 3, 2017 9:23 p.m. ET

A massive data breach at Yahoo in 2013 was far more extensive than previously disclosed, affecting all of its 3 billion user accounts, new parent company Verizon Communications Inc. said on Tuesday.

The figure, which Verizon said was based on new information, is three times the 1 billion accounts Yahoo said were affected when it first disclosed the breach in December 2016. The new disclosure, four months after Verizon completed its acquisition of Yahoo, shows that executives are still coming to grips with the extent of the...

Goal: Moderately Expensive Hash Function



Fast on PC and
Expensive on ASIC?

Attempt 1: Hash Iteration

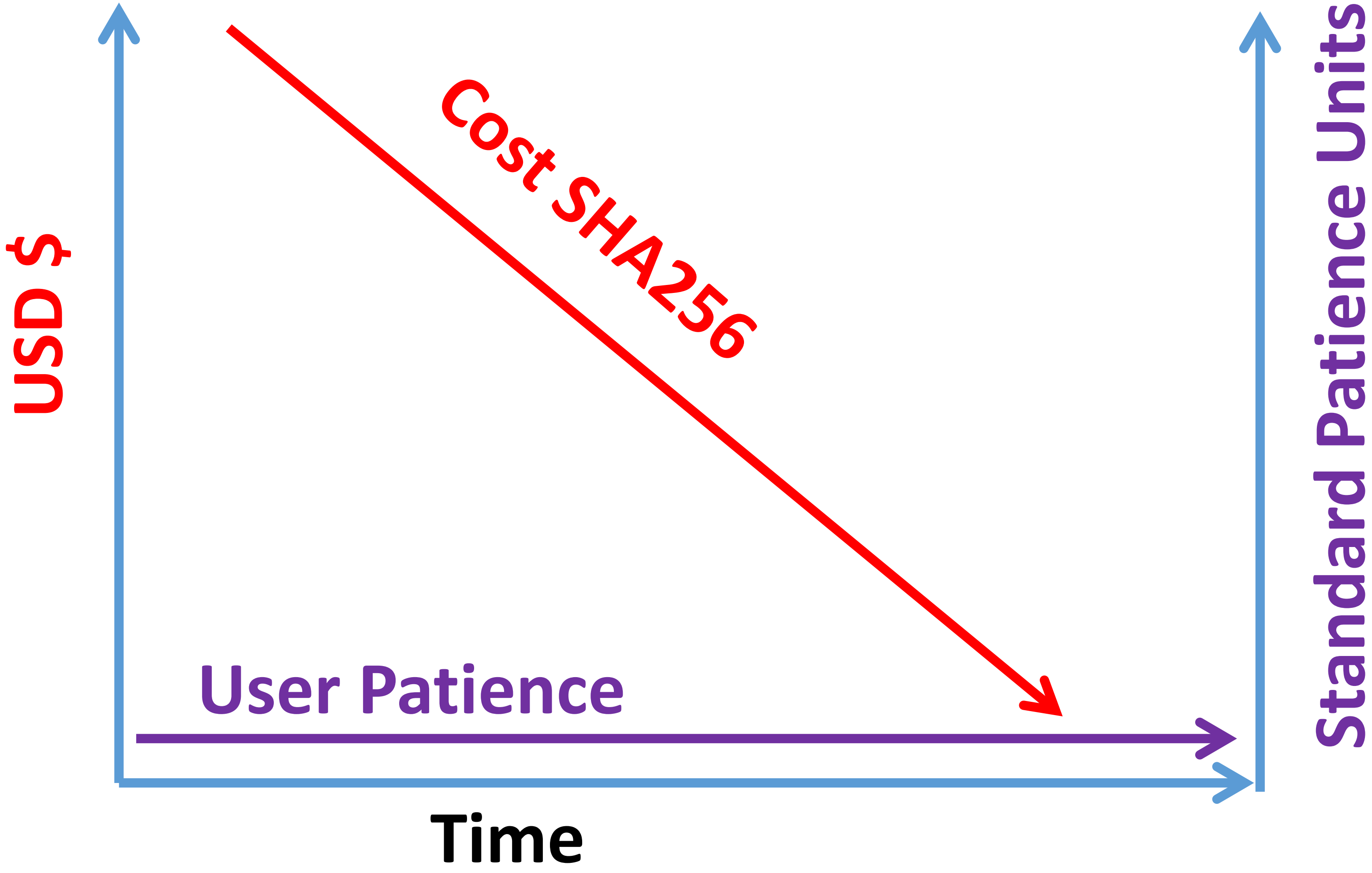
- BCRYPT

- PBKDF2

100,000 SHA256 computations (iterative)

Estimated Cost on ASIC: $1 per billion password guesses [BS14]

The Challenge

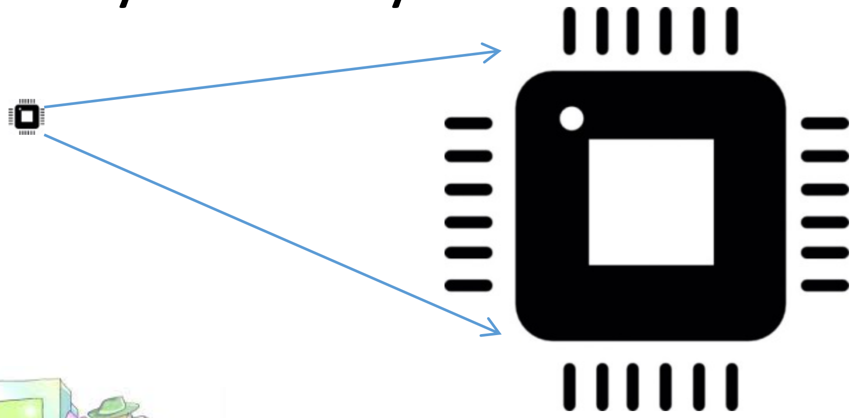


Disclaimer: This slide is entirely for humorous effect.

Memory Hard Function (MHF)

- Intuition: computation costs dominated by memory costs

vs.



sCrypt

- Data Independent Memory Hard Function (iMHF)
  - Memory access pattern should not depend on input

password hashing competition

(2013-2015)

https://password-hashing.net/

password
hashing
competition

(2013-2015)

We recommend that you use Argon2…

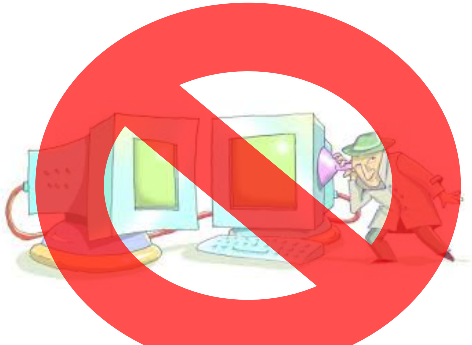
https://password-hashing.net/

password hashing competition

(2013-2015)

We recommend that you use Argon2…

There are two main versions of Argon2, **Argon2i** and Argon2d. **Argon2i** is the safest against side-channel attacks



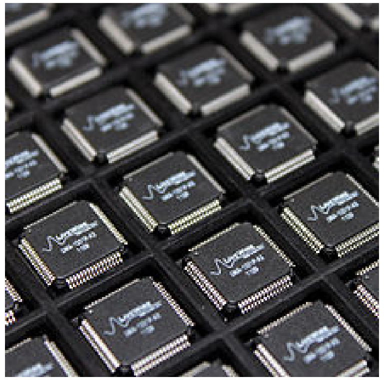
https://password-hashing.net/

Depth-Robustness: The Key Property

Necessary [AB16] and sufficient [ABP16] for secure iMHFs

Question

Are existing iMHF candidates based on depth-robust DAGs?



Answer: No

…n2i and Balloon Hashing

On the Depth-Robustness and Cumulative Pebbling Cost of Argon2i

Jeremiah Blocki* Samson Zhou†

August 4, 2017

Abstract

Argon2i is a data-independent memory hard function that won the password hashing competition. The password hashing algorithm has already been incorporated into several open source crypto libraries such as libsodium. In this paper we analyze the cumulative memory cost of computing Argon2i. On the positive side we provide a lower bound for Argon2i. On the negative side we exhibit an improved attack against Argon2i which demonstrates that our lower bound is nearly tight. In particular, we show that

(1) An Argon2i DAG is $\left(e, O\left(n^3/e^3\right)\right)$-reducible.

(2) The cumulative pebbling cost for Argon2i is at most $O\left(n^{1.768}\right)$. This improves upon the previous best upper bound of $O\left(n^{1.8}\right)$ [AB17].

(3) Argon2i DAG is $\left(e, \tilde{\Omega}\left(n^3/e^3\right)\right)$-depth robust. By contrast, analysis of [ABP17a] only established that Argon2i was $\left(e, \tilde{\Omega}\left(n^2/e^2\right)\right)$-depth robust.

(4) The cumulative pebbling complexity of Argon2i is at least $\tilde{\Omega}\left(n^{1.75}\right)$. This improves on the previous best bound of $\Omega\left(n^{1.66}\right)$ [ABP17a] and demonstrates that Argon2i has higher cumulative memory cost than competing proposals such as Catena or Balloon Hashing.

Jeremiah Blocki
Purdue University

For the Alwen-Blocki attack to fail against practical memory parameters, Argon2i-B must be instantiated with more than 10 passes on memory. The current IRTF proposal calls even just 6 passes as the recommended "paranoid" setting.
More generally, the parameter selection process in the proposal is flawed in that it tends towards producing parameters for which the attack is successful (even under realistic constraints on parallelism).

…cted acyclic graph (DAG) G on $n = \Theta(\sigma * \tau)$ nodes representing

…analyzing iMHFs. First we define and motivate a new complexity … (i.e. electricity) required to compute a function. We argue that, …portant as the more traditional AT-complexity. Next we describe … an iMHF based on an arbitrary DAG G. We upperbound both …nce evaluated in terms of a certain combinatorial property of G. …everal general classes of DAGs which include those underlying …didates in the literature. In particular, we obtain the following …meters σ and τ (and thread-count) such that $n = \sigma * \tau$.

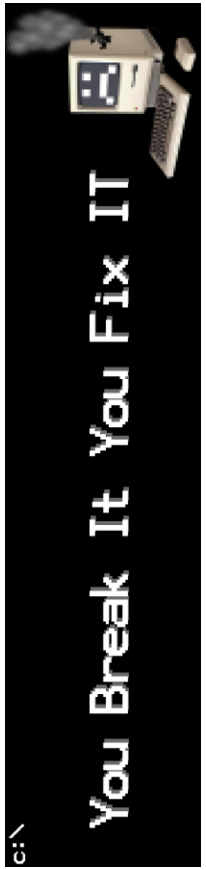
…[FLW13] has AT and energy complexities $O(n^{1.67})$.

…FLW13] has complexities is $O(n^{1.67})$.

…functions of [CGBS16] both have complexities in $O(n^{1.67})$.

- The Argon2i function of [BDK15] (winner of the Password Hashing Competition [PHC]) has complexities $O(n^{7/4}\log(n))$.

Can we build a secure iMHF?



Practical Graphs for Optimal Side-Channel Resistant Memory-Hard Functions

Joël Alwen*
IST Austria
jalwen@ist.ac.at

Jeremiah Blocki
Purdue University
jblocki@purdue.edu

Ben Harsha†
Purdue University
bharsha@purdue.edu

ABSTRACT

A memory-hard function (MHF) f_n with parameter n can be computed in sequential time and space n. Simultaneously, a high *amortized parallel* area-time complexity (aAT) is incurred per evaluation. In practice, MHFs are used to limit the rate at which an adversary (using a custom computational device) can evaluate a security sensitive function that still occasionally needs to be evaluated by honest users (using an off-the-shelf general purpose device). The most prevalent examples of such sensitive functions are Key Derivation Functions (KDFs) and password hashing algorithms where rate limits help mitigate off-line dictionary attacks. As the honest users' inputs to these functions are often (low-entropy) passwords special attention is given to a class of side-channel resistant MHFs called iMHFs.

Experimental benchmarks on a standard off-the-shelf CPU show that the new modifications do not adversely affect the impressive throughput of Argon2i (despite seemingly enjoying significantly higher aAT).

CCS CONCEPTS

• **Security and privacy** → **Hash functions and message authentication codes;**

KEYWORDS

hash functions; key stretching; depth-robust graphs; memory hard functions

Github: https://github.com/Practical-Graphs/Argon2-Practical-Graph