Reminder: Course Feedback

Course Summary					
Course Code	Course Title	Survey Start Date	Survey End Date	Report Access Start	Response Rate
wl.202120.CS.55500. FNY.18101	Cryptography	4/19/2021 9:00 AM	5/2/2021 11:59 PM	5/12/2021 12:00 AM	50.00% (4/8)

- If you haven't already please complete your course evaluation before Sunday at 11:59PM.
- What did you like about the course? What could be improved? Let me know! Your feedback is valuable and I carefully read through any comments after the semester is over.
- Thanks to everyone who already completed there course evaluation!
- Your feedback is anonymous and will not impact your grade (I cannot view your feedback until after grades are entered).

Announcements

Quiz 6: Due tomorrow (4/28) at 11:59PM

Homework 5: Due Thursday (4/29) at 11:59PM

- Late submissions allowed up until Friday (3/30) at 11:59PM
- We plan to release the solutions on Saturday

Final Exam: Monday, May 3 at 10:30 AM

Location: FRNY B124 (right here!)

Time: 10:30AM – 12:30PM

(Practice Final Exam on Piazza)

Final Exam

- Cumulative, but will focus a bit more heavily on topics from the second half of the semester
- You are allowed to prepare one page (8.5x11) of handwritten notes. Double sided
- Practice Exam on Piazza
  - The real final will be shorter
  - The real exam will have more short answer questions

Cryptography
CS 555

Week 15:

- Zero-Knowledge Proofs
- Hot Topics in Cryptography
- Memory Hard Functions + Password Hashing

Recap: 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 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);

$\mathbf{x}$

$z = x^2 \text{ mod N}$

(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)

Bob (verifier);

Z

M$_1$,…M$_k$ where $M_i = y_i^2 z \bmod N$

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

Responses r$_1$,…,r$_k$ where $\boldsymbol{r_i} = \begin{cases} \boldsymbol{y_i} & \boldsymbol{if\ c_i = 0} \\ \boldsymbol{y_i}x & if\ \boldsymbol{c_i} = 1 \end{cases}$

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

Alice (prover);

$\mathbf{x}$

$z = x^2 \bmod \mathrm{N}$

$y_1, \ldots, y_k$
(random)

Simulator Power: Can program the random oracle

Non-Interactive Zero-Knowledge Proof (NIZK)

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

$\boldsymbol{challenges\ c = (c_1, \ldots, c_k) = H}(\mathrm{z}, \mathrm{M}_1, \ldots \mathrm{M}_k)$

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

Bob (verifier);

z

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

Alice (prover);

x

$z = x^2 \bmod \mathrm{N}$

$y_1, \ldots, y_k$
(random)

Completeness: If Alice is honest Bob will always accept

Non-Interactive Zero-Knowledge Proof (NIZK)

M$_1$,…M$_k$ where $M_i = y_i^2 z \; mod \; N$

$\boldsymbol{challenges\ c = (c_1, \ldots, c_k) = H}(\mathrm{z}, \mathrm{M}_1, \ldots \mathrm{M}_k)$

Responses r$_1$,…,r$_k$ where $\boldsymbol{r_i} = \begin{cases} \boldsymbol{y_i} & \boldsymbol{if\ c_i = 0} \\ \boldsymbol{y_i}x & if\ \boldsymbol{c_i} = 1 \end{cases}$

Bob (verifier);

z

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

Alice (prover);

x

$z = x^2 \bmod \mathrm{N}$

$y_1, \ldots, y_k$
(random)

Soundness: If the statement is false a malicious PPT prover should not be able to Produce a proof that Bob accepts.

Non-Interactive Zero-Knowledge Proof (NIZK)

Fact: If $z \neq x^2 \ mod\ N$ is not a square root then for each i< k either

1) M_i does not have a square root and Alice won't be able to respond when $\boldsymbol{c_i} = 1$, or

2) $z^{-1}M_i$ does not have a square root and Alice won't be able to respond when $\boldsymbol{c_i} = \boldsymbol{0}$

$M_i = y_i^2 z \ mod\ N$

$= (\boldsymbol{c_1}, \ldots, \boldsymbol{c_k}) = \boldsymbol{H}(\mathrm{z}, \mathrm{M}_1, \ldots \mathrm{M}_k)$

e $\boldsymbol{r_i} = \begin{cases} \boldsymbol{y_i} & \boldsymbol{if\ c_i = 0} \\ \boldsymbol{y_i}x & if\ \boldsymbol{c_i} = 1 \end{cases}$

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

Alice (prover);

X

$z = x^2 \bmod \mathrm{N}$

$y_1, \ldots, y_k$ (random)

Soundness: If the statement is false a malicious PPT prover should not be able to Produce a proof that Bob accepts.

Non-Interactive Zero-Knowledge Proof (NIZK)

Definition: Call a random oracle query $\boldsymbol{H}(\mathrm{M}_1,\ldots\mathrm{M}_k)$ lucky if Alice can respond to all challenges. Let L_j=1 if and only if query j is lucky.

Fact: $\Pr[L_j = 1 \mid L_1, \ldots, L_j \neq 1] \leq 2^{-k}$

Union Bound: $Pr[\exists j \leq q.\, L_j = 1\,] \leq q2^{-k}$

$M_i = y_i^2 z \bmod N$

$: = (c_1, \ldots, c_k) = \boldsymbol{H}(\mathrm{z},\mathrm{M}_1,\ldots\mathrm{M}_k)$

e $r_i = \begin{cases} y_i & \text{if } c_i = 0 \\ y_i x & \text{if } c_i = 1 \end{cases}$

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

Alice (prover);

x

$z = x^2 \bmod \mathrm{N}$

$y_1, \ldots, y_k$ (random)

Soundness: If the statement is false a malicious PPT prover should not be able to produce a proof that Bob accepts.

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'
  (H' must look like a random oracle)

- 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: $\mathrm{ADV_D} = |Pr[D^H(\text{z}, \pi_z) = 1] - Pr[D^{H'}(\text{z}, \pi'_z) = 1]|$
- **Zero-Knowledge:** Any PPT distinguisher D should have negligible advantage.
- NIZK proof π_z is transferrable (contrast with interactive ZK proof)

Non-Interactive Zero-Knowledge Proof (NIZK)

$M_1,\dots M_k$ where $M_i = y_i^2 z^{1-c_i} \bmod N$

$\boldsymbol{challenges\ c = (c_1, \dots, c_k)}$

Program $\boldsymbol{H}(M_1,\dots M_k) \coloneqq \boldsymbol{c}$

Responses $r_1,\dots,r_k$ **where** $\boldsymbol{r_i} = \begin{cases} \boldsymbol{y_i} & \boldsymbol{if\ c_i = 0} \\ \boldsymbol{y_i} & if\ \boldsymbol{c_i} = 1 \end{cases}$

Bob (verifier);

Simulator

z

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

$z = x^2 \bmod N$

$c = (c_1, \dots, c_k)$

$y_1, \dots, y_k$

(random)

Simulator Power: Can program the random oracle

Σ-Protocols

- Prover Input: instance/claim x and witness w
- Verifier Input: Instance x
- Σ-Protocols: three-message structure
  - Prover sends first message $m=P_1(x,w; r_1)$
  - Verifier responds with random challenge c
  - Prover sends response $R=P_2(x,w,r_1,c; r_2)$
  - Verifier outputs decision $V(x,m,c,R)$
  - **Completeness:** If w is a valid witness for instance x then $\Pr[V(x,c,R)=1]=1$
  - **Soundness:** If the claim x is false then $V(x,c,R)=0$ with probability at least ½
  - **Zero-Knowledge:** Simulator can produce computationally indistinguishable transcript

Σ-Protocols and Fiat-Shamir Transform

- Convert Σ-Protocols into Non-Interactive ZK Proof
- Prover Input: instance/claim x and witness w
- Verifier Input: Instance x
- **Step 1:** Prover generates first messages for n instances of the protocol
  - $m_i = P_1(x,w; r_i)$ for each i=1 to n
- **Step 2:** Prover uses random oracle to extract random coins $z_j=H(x,j, m_1,\ldots,m_n)$ for j=1 to n
  - Prover samples challenges $c_1,\ldots,c_n$ using random strings $z_1,\ldots,z_n$ i.e., c_i=SampleChallenge(z_i)
- **Step 3:** Prover computes responses $R_1,\ldots,R_n$
  - $R_i \leftarrow P_2(x,w,r_i,c_i)$
- **Step 4:** Prover outputs the proof $\{(m_i, c_i, z_i)\}_{i \le n}$

Σ-Protocols and Fiat-Shamir Transform

- **Step 1:** Prover generates first messages for n instances of the protocol
  - $m_i = P_1(x,w; r_i)$ for each i=1 to n
- **Step 2:** Prover uses random oracle to extract random coins $z_i=H(x,i, m_1,\ldots,m_n)$ for i=1 to n
  - Prover samples challenges $c_1,\ldots,c_n$ using random strings $z_1,\ldots,z_n$ i.e., c_i=SampleChallenge(z_i)
- **Step 3:** Prover computes responses $R_1,\ldots,R_n$
  - $R_i \leftarrow P_2(x,w,r_i,c_i)$
- **Step 4:** Prover outputs the proof $\pi = \{(m_i, c_i, R_i)\}_{i \leq n}$

Verifier: $V_{NI}(x,\pi)$ check that for all $i \leq n$

1. **V(x, (m_i, c_i, R_i))=1 and**
2. c_i=SampleChallenge(z_i) **where** $z_i=H(x,i, m_1,\ldots,m_n)$

Σ-Protocols and Fiat-Shamir Transform

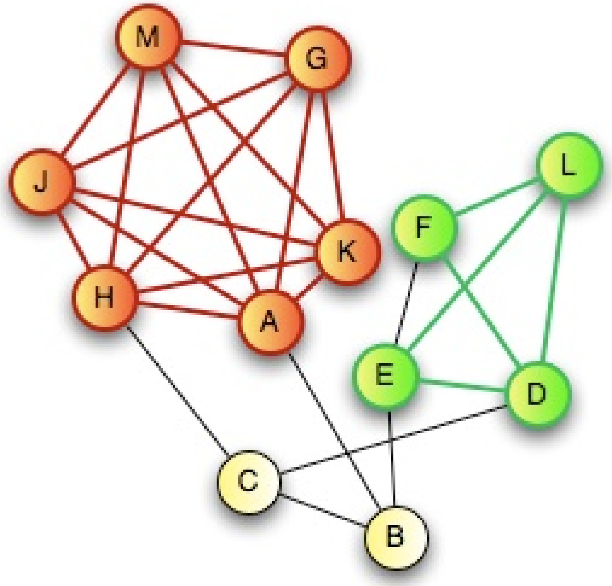
- **Step 1:** Prover generates first messages for n instances of the protocol
  - $m_i = P_1(x,w; r_i)$ for each i=1 to n
- **Step 2:** Prover uses random oracle to extract random coins $z_i=H(x,i, m_1,\ldots,m_n)$ for i=1 to n
  - Prover samples challenges $c_1,\ldots,c_n$ using random strings $z_1,\ldots,z_n$ i.e., $c_i=\mathrm{SampleChallenge}(z_i)$
- **Step 3:** Prover computes responses $R_1,\ldots,R_n$
  - $R_i \leftarrow P_2(x,w,r_i,c_i)$
- **Step 4:** Prover outputs the proof $\pi = \{(m_i, c_i, R_i)\}_{i \le n}$

Zero-Knowledge (Idea):

Step 1: Run simulator for Σ n-times to obtain n transcripts (m_i, c_i, R_i) **for each** $i \le n$.

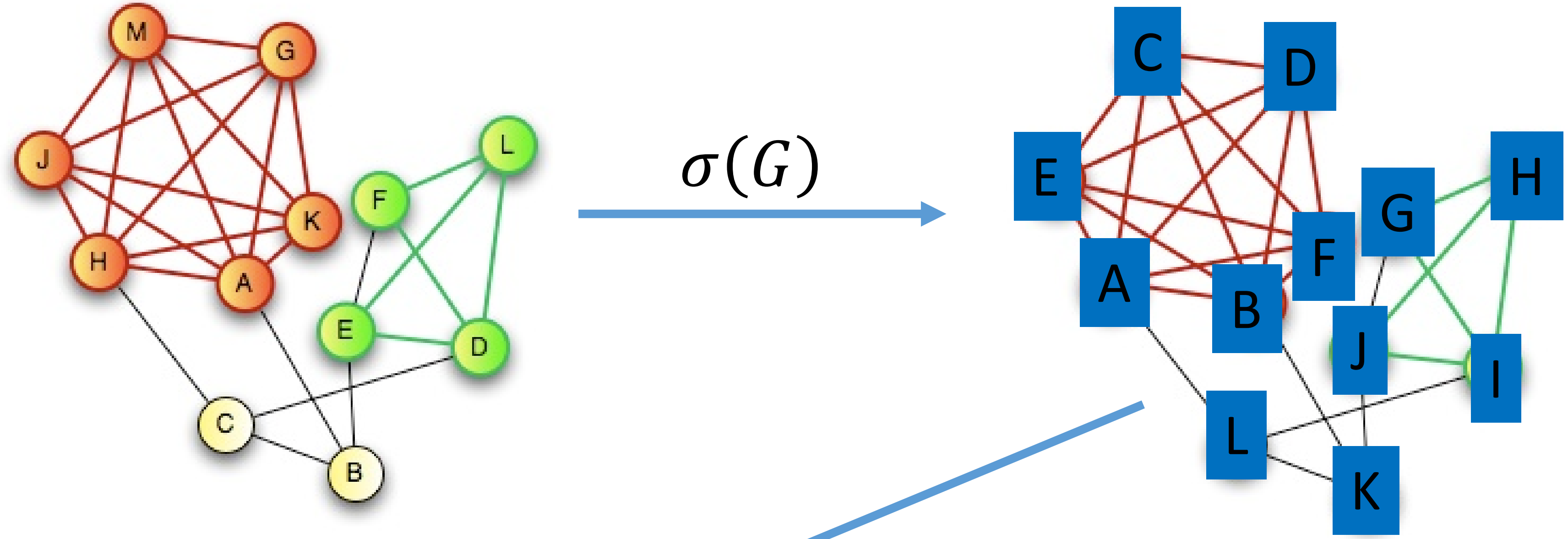
Step 2: Program the random oracle so that $H(x,i, m_1,\ldots,m_n)=z_i$ where $c_i=\mathrm{SampleChallenge}(z_i)$

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



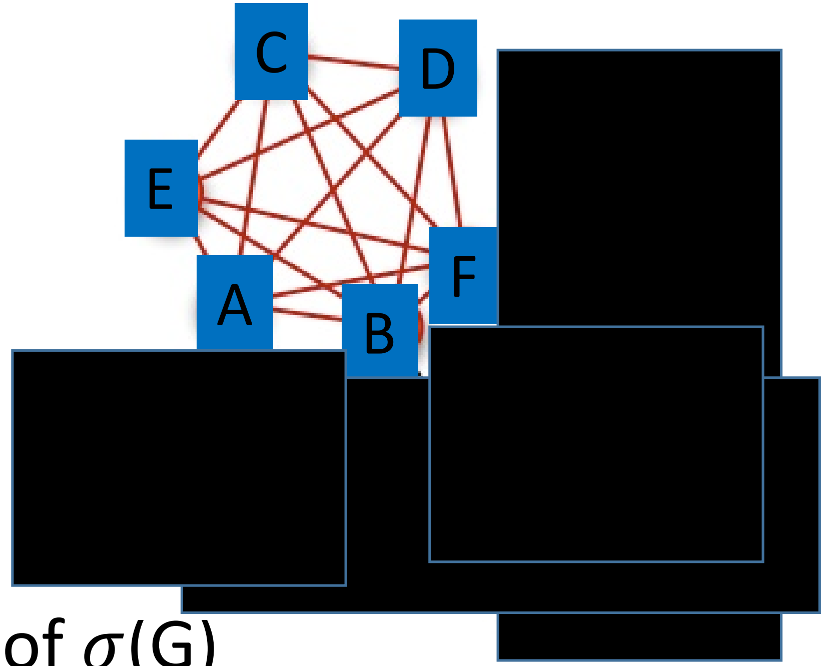
Adjacency matrix $A_{\sigma(G)}$

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

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

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

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

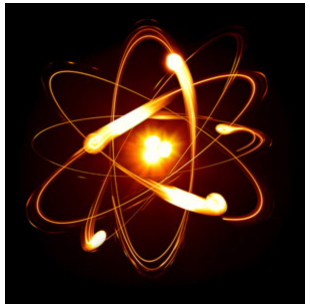
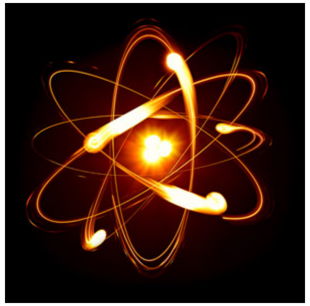
- **Completeness**: Honest prover can always make honest verifier accept
- **Soundness**: If prover commits to adjacency matrix $A_{\sigma(G)}$ of σ(G) and can reveal a clique in submatrix of $A_{\sigma(G)}$ then G itself contains a k-clique. Proof invokes binding property of commitment scheme.
- **Zero-Knowledge:** 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

Lattices

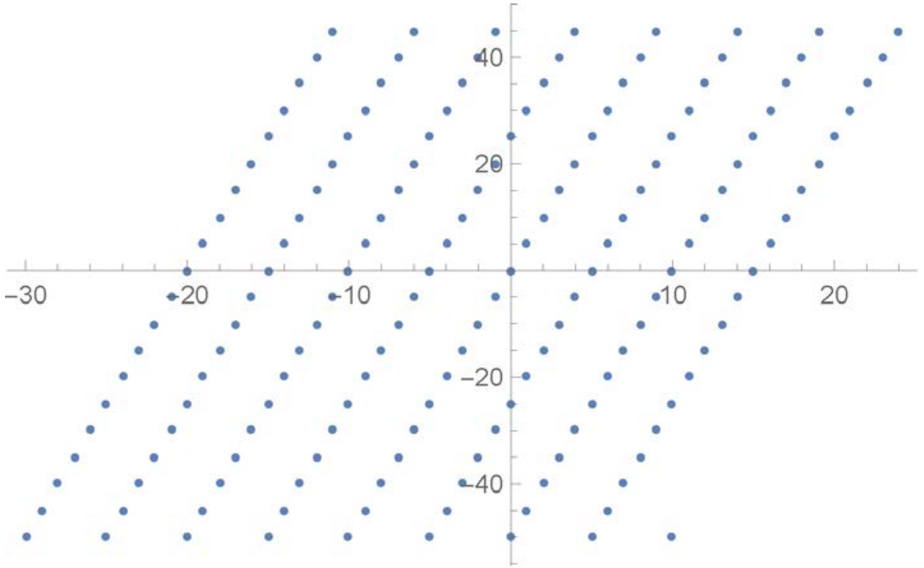
Each $\boldsymbol{b}_i$ is a vector $\boldsymbol{b}_i \in \mathbb{R}^n$, $\mathrm{span}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n) = \mathbb{R}^n$

- Basis: $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$
- Lattice:

integers

$$\mathrm{L}(\mathrm{B}) := \{\sum_{i \le n} a_i\, \boldsymbol{b}_i | a_1, \dots, a_n \in \mathbb{Z}\}$$

- Example: $\boldsymbol{b}_1 = (1,5)$ and $\boldsymbol{b}_2 = (5,0)$
  - $(0,25) = -5\boldsymbol{b}_1 + \boldsymbol{b}_2$ is in the lattice $\mathrm{L}(\mathrm{B})$
  - $(0, 2)$ is not in the lattice $\mathrm{L}(\mathrm{B})$



- Shortest Vector Problem: Find shortest (non-zero) vector in $\mathrm{L}(\mathrm{B})$

Usually defined using Euclidean Norm

Lattices

Each $\boldsymbol{b}_i$ is a vector $\boldsymbol{b}_i \in \mathbb{R}^n$, $\mathrm{span}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n) = \mathbb{R}^n$

- Basis: $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$
- Lattice:

integers

$$\mathrm{L(B)} \coloneqq \{\textstyle\sum_{i \le n} a_i\, \boldsymbol{b}_i \,|\, a_1, \dots, a_n \in \mathbb{Z}\}$$

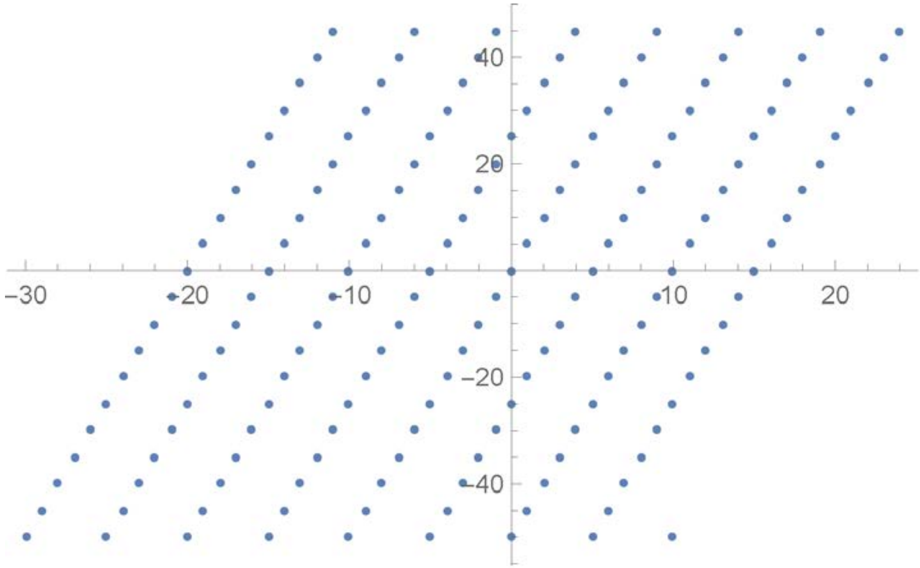
- Example: $\boldsymbol{b}_1 = (1,5)$ and $\boldsymbol{b}_2 = (5,0)$
    - $(0,25) = -5\boldsymbol{b}_1 + \boldsymbol{b}_2$ is in the lattice $\mathrm{L(B)}$
    - $(0, 2)$ is not in the lattice $\mathrm{L(B)}$

- Closest Vector Problem: Given $\boldsymbol{v} \in \mathbb{R}^n$

find vector in $\mathrm{L(B)}$ closest to $\boldsymbol{v}$

Example: $(0,25)$ is lattice point closest to

$\boldsymbol{v} = (1,24)$



Hard Lattice Problems

- Shortest Vector Problem: Find shortest (non-zero) vector in $\mathrm{L}(\mathrm{B})$

- Closest Vector Problem: Given $\boldsymbol{v} \in \mathbb{R}^n$ find vector in $\mathrm{L}(\mathrm{B})$ closest to $\boldsymbol{v}$

- Approximation versions
  - Relax requirement to find shortest/closest vector
- No known (quantum) algorithm to solve above problems
  - Even approximation is hard
  - Conjectured to be ``average case" hard (needed for crypto)

GGH Encryption Scheme

- Goldreich-Goldwasser-Halevi (GGH)
- Security Assumption (Lattices): Closest Vector Problem is Hard
  - No known quantum algorithm breaks the assumption

- Private Key: Matrix B and a unimodular matrix U
  - $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ is the basis of a lattice L with ``nice properties”
    - All vectors $\boldsymbol{b}_i$ are short and nearly orthogonal e.g., inner product $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ is small
- Public Key: $B' = UB$
  - B' is a second basis for the lattice L

GGH Encryption Scheme

- Private Key: Matrix B and a unimodular matrix U
  - $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ is the basis of a lattice L with ``nice properties"
    - All vectors $\boldsymbol{b}_i$ are short and nearly orthogonal e.g., inner product $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ is small
- Public Key: $B' = UB$
  - $B' = (\boldsymbol{b}_1{}', \dots, \boldsymbol{b}_n{}')$ is a second basis for the lattice L

Encryption: message $\mathrm{m} = (m_1, \dots, m_n)$ with each $-\mathrm{M} < m_i < M$

1. Compute $v = m \cdot B' = \sum_{i \le n} m_i \, \boldsymbol{b}_i{}'$
2. Pick a random error vector e with small norm
3. Return $c = v + e$

GGH Encryption Scheme

- Private Key: Matrix B and a unimodular matrix U
  - $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ is the basis of a lattice L with ``nice properties”
    - All vectors $\boldsymbol{b}_i$ are short and nearly orthogonal e.g., inner product $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ is small
- Public Key: $B' = UB$
  - $B' = (\boldsymbol{b}_1{}', \dots, \boldsymbol{b}_n{}')$ is a second basis for the lattice L

Decryption: ciphertext $c = v + e = e + m \cdot B' = e + m \cdot UB$

Compute $\mathrm{y} = c \cdot B^{-1} = m \cdot U + e \cdot B^{-1}$

Babai Rounding Technique Removes small error term $e \cdot B^{-1}$ from y

Return $m = (y - e \cdot B^{-1}) \cdot U^{-1} = (m \cdot U) \cdot U^{-1}$

GGH Encryption Scheme

- Private Key: Matrix B and a unimodular matrix U
  - $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ is the basis of a lattice L with ``nice properties"
    - All vectors $\boldsymbol{b}_i$ are short and nearly orthogonal e.g., inner product $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ is small
- Public Key: $B' = UB$
  - $B' = (\boldsymbol{b}_1', \dots, \boldsymbol{b}_n')$ is a second basis for the lattice L

Design Key Encapsulation Mechanism from GGH or other schemes like NTRU

- CPA/CCA-security

Fully Homomorphic Encryption (FHE)

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

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

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

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

- 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	…	???

[DMNS06, DKMMN06]

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

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

D

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?

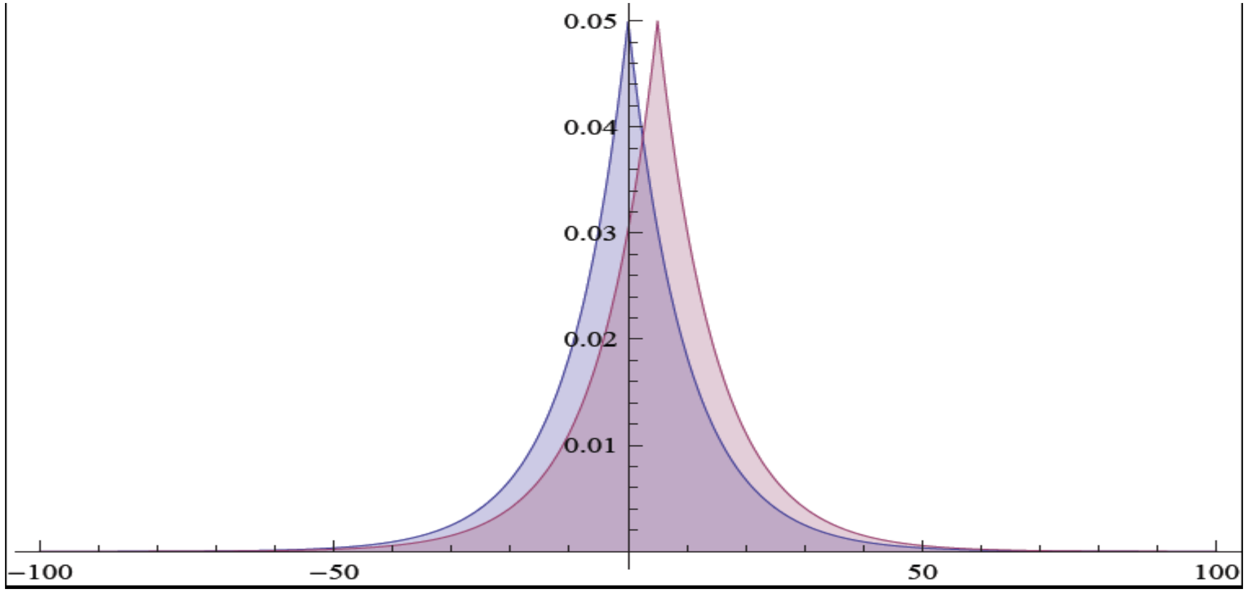
- Cryptography
  - Insiders/Outsiders
  - Only those with decryption key(s) can reveal secret
  - Multiparty Computation: Alice and Bob learn nothing other than f(x,y)

Traditional Differential Privacy Mechanism

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

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

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



Google

differential privacy

Scholar

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**

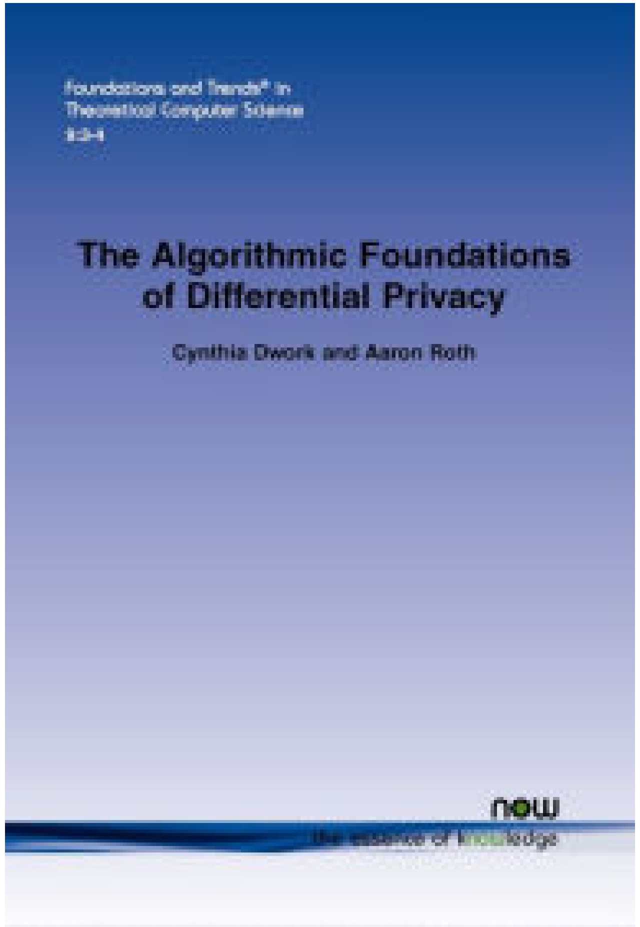
F McSherry, K Talwar - … of Computer Science, 2007. FOCS'07. …, 2007 - ieeexplore.ieee.org

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

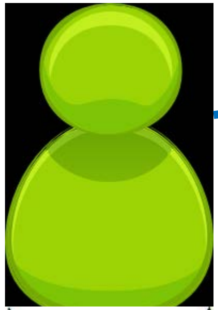


BARNES & NOBLE

- $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**	85e23cfe0021f58 4e3db87aa72630 a9a2345c062

SHA1(12345689d978034a3f6)=85e23cfe 0021f584e3db87aa72630a9a2345c062

+

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

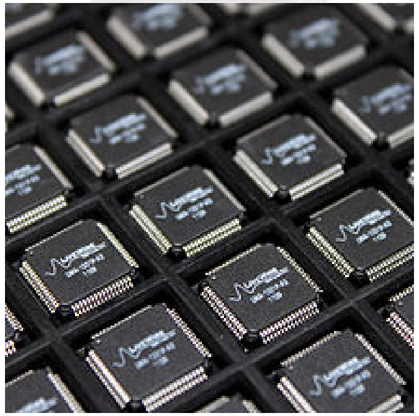
By Robert McMillan and Ryan Knutson

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

ASHLEY MADISON® Life is short. Have an affair.® YAHOO! Dropbox

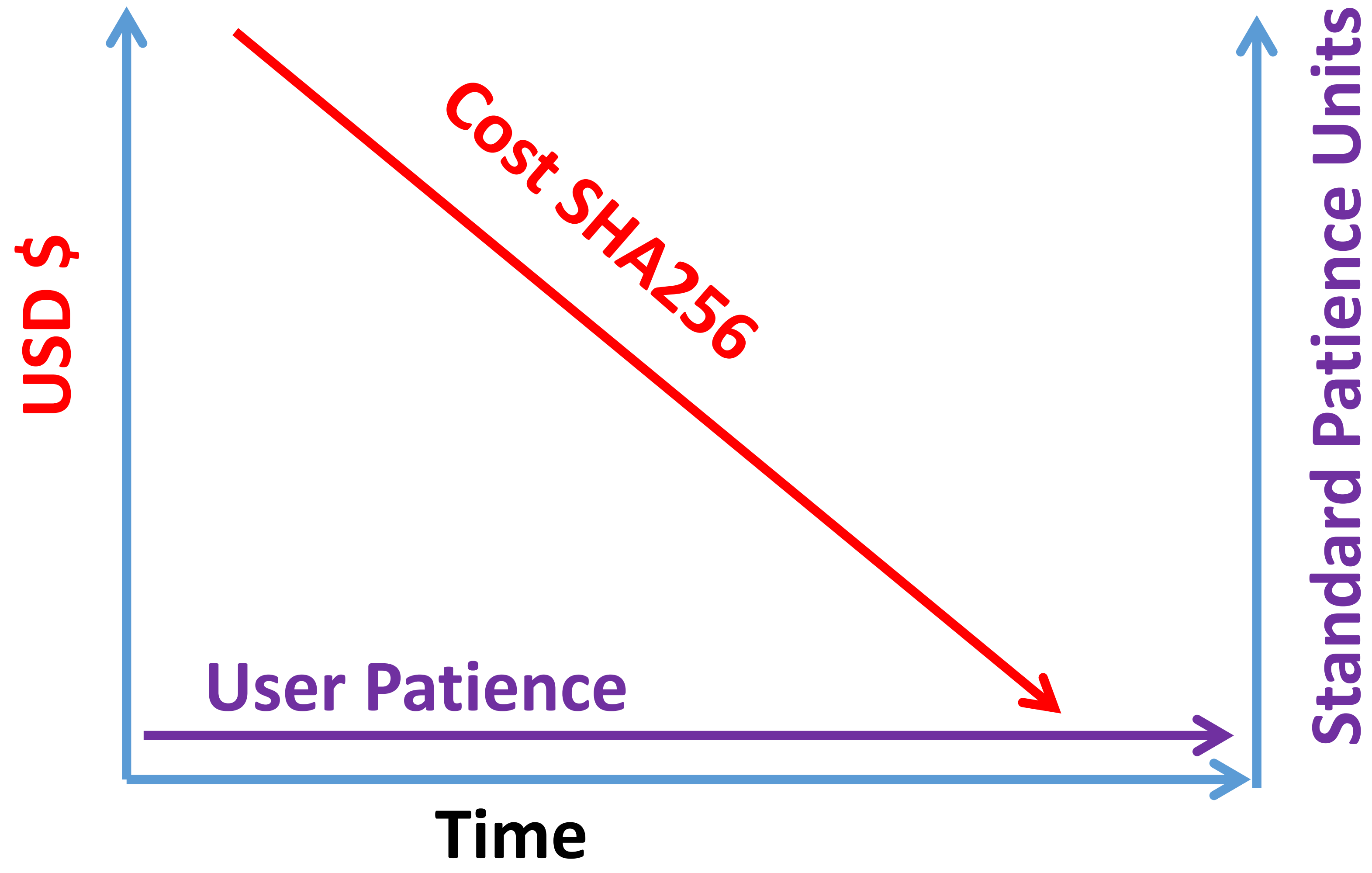
- PBKDF2

LastPass ****

100,000 SHA256 computations (iterative)

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

The Challenge

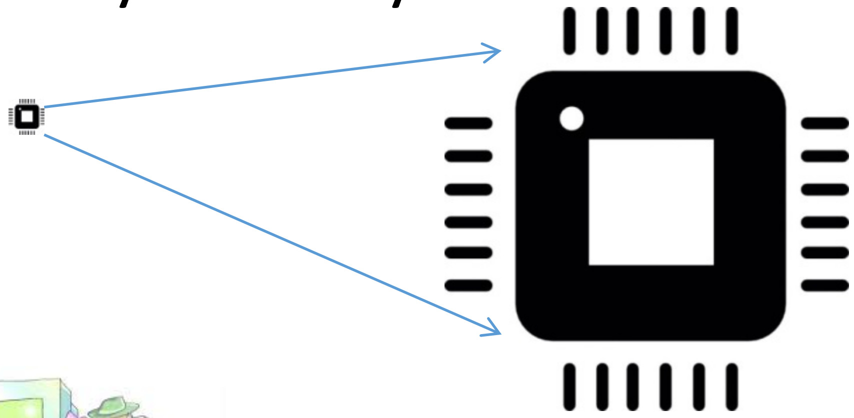


Disclaimer: This slide is entirely for humorous effect.

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

We recommend that you use Argon2…

(2013-2015)

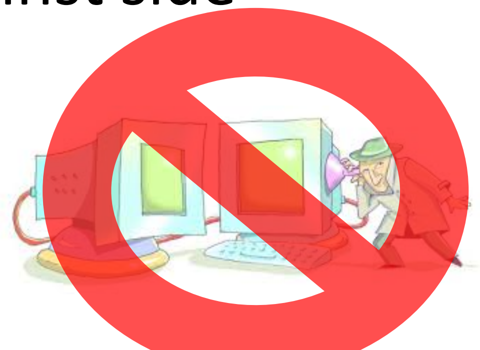
https://password-hashing.net/

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



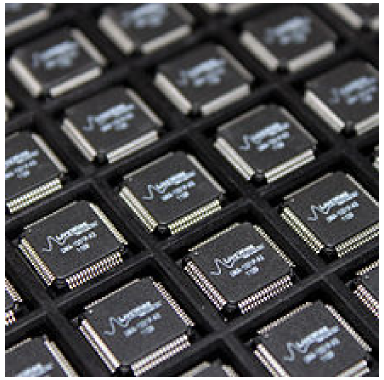
https://password-hashing.net/

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

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.

…n2i and 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).

…ected 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 …lidates in the literature. In particular, we obtain the following …meters σ and τ (and thread-count) such that $n = \sigma * \tau$.

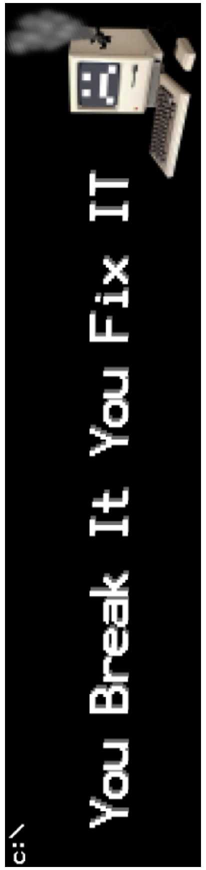
…[FLW13] has AT and energy complexities $O(n^{1.67})$.

…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

1 INTRODUCTION

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