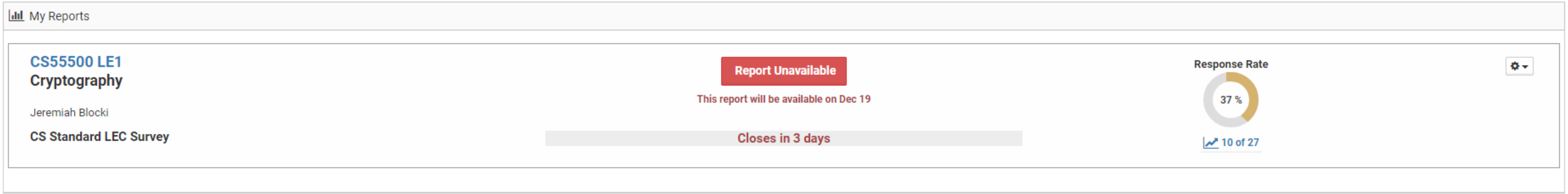
Homework 5 Statistics

Minimum Value	59.00
Maximum Value	100.00
Range	41.00
Average	82.73
Median	83.50
Standard Deviation	12.74

Course Evaluation

- Please Complete Your Course Evaluations
- Your feedback is valuable!



My Reports

CS55500 LE1

Cryptography

Jeremiah Blocki

CS Standard LEC Survey

This report will be available on Dec 19

Closes in 3 days

Response Rate

37 %

10 of 27

- Homework 5 Solutions and Practice Final Available on Piazza

Final Exam

- Time: Tuesday, December 11th at 8AM
- Location: LWSN B151

- Comprehensive
  - …but heavier coverage of material covered in second half of semester
- Format
  - Multiple choice
  - Fill in the blank (expect more of these questions)
  - true/false/more information
- Practice Exam on Piazza
- Solutions to practice exam distributed on Thursday (Do not distribute!)

Review: Attacker Models

- Passive Eavesdropping Attacker (Eve)

- Active Attacker
  - **Chosen Plaintext Attack:** Attacker can control/influence messages that are encrypted

  - **Chosen Ciphertext Attack:** Attacker can convince honest party to (partially) decrypt ciphertexts of his/her choosing.
- MPC: Semi-Honest vs Malicious
- Man-In-The-Middle Attacker

Review: Key Concepts for Symmetric Key Crypto

- **Building Blocks:** OWFs, OWPs, PRGs, PRFs, CRHFs, PRPs (Block Cipher)
  - **Constructions:** PRFs from PRGs, PRPs via Feistel Network etc…
- Should understand syntax (e.g., PRF uses a key, but a PRG doesn't) and security definitions (e.g., PRG vs PRF)

- MAC vs. Encryption
  - Confidentiality vs Integrity
  - Syntax
  - Security Definition(s): Authenticated Encryption, CCA-Security, CPA-Security Perfect Secrecy, MAC-forgery game

Review: Collision Resistant Hash Functions (CRHF)

- CRHFs are a unique object in cryptography
  - No secret key (public seed) --- security definition (e.g., seeded) vs practice (e.g., SHA3)
- Davies-Meyer construction in Ideal Cipher Model

- Handling long inputs
  - Merkle Tree
  - Merkle-Damgård

- Collision/Inversion Attacks
  - Birthday Attack
  - Small Space Birthday Attack
  - Pre-Computation Attacks (Time/Space Tradeoffs)

- Random Oracle Methodology

Review: Key Principles

- Sufficient Key Space Principle
    - Resist brute-force attacks

- Penguin Principle
    - Issues with stateless/deterministic encryption schemes
    - Importance of nonces

- Independent Key Principle

Review: Asymmetric Key Crypto

- Key Assumptions:
    - FACTORING
    - RSA-Inversion Problem
    - Discrete Logarithm Problem
    - DDH vs CDH
    - OWFs (for Certain Signature Schemes)
- Public Key Encryption
    - Syntax
    - Security Definition(s): CPA vs CCA-security
    - Constructions: Plain RSA, El Gamal, RSA-OAEP
- Key Encapsulation Mechanism (and how to use them)

Review: Signatures

- Goal: Message Integrity
- Signature Properties:
  - Public Verification
  - Transferrable: Bob receives signature from Alice and can forward to Joe
  - Can identify sender
  - Cannot identify intended recipient
    - **Example:** Alice signs message “I promise to pay you $50” and sends to Bob
    - Eve can copy signature and forward to Joe who believes that Alice will pay him $50.
    - **Solution:** Can bind signature to recipient, by indicating recipient inside the message
      - E.g., “I promise to pay you (Bob) $50”
- Contrast with MAC
  - Need secret key for verification
  - Cannot identify sender (anyone who has secret key)

Review: Signatures and MACs

- What are some secure constructions of signatures?
    - RSA-FDH
    - Schnorr-Signatures (Fiat-Shamir)
    - DSA/ECDSA
- How to build a MAC?
    - HMAC
    - PRF: $t=F_k(m)$
- Handling Long Messages: Hash and sign/mac
- How to build an (in)secure signature/MAC scheme?

Review: Multi-Party Computation

- Malicious vs Semi-Honest Security Models

- Security Definition (Simulator)
  - Captures intuition that Alice learns “nothing else” about Bob’s input

- Yao’s Protocol (Garbled Circuits)
  - What is security model?
  - Building Blocks: Oblivious Transfer, CPA-Secure Encryption
- Use of Zero-Knowledge Proofs in MPC

Review: Zero-Knowledge

- Decision Problem (e.g., DDH, SAT, CLIQUE)

- Properties
  - Completeness
    - Honest prover can always get verifier to accept a true statement
  - Soundness
    - A cheating prover can't consistently get honest verifier to accept
  - Zero-Knowledge
    - How to build a simulator?
- Interactive vs Non-Interactive Zero-Knowledge

Practice Problem 1: NIZK

- Build a NIZK for the group membership problem
- Verifier: Knows h, wants to be sure that h is in <g>
- Prover: Knows x such that $h=g^x$
  - Prover picks r and sets $\mathrm{z} = g^{x+r}$
  - Prover selects the challenge b= LSB(H(z)), and sets the response R=r+bx.
  - Prover outputs the proof (z,R)
- Verifier computes b= LSB(H(z)) and checks that $h^{1-b}z = g^R$

- Problem?

Practice Problem 1: NIZK (FIX)

- Build a NIZK for the group membership problem
- Verifier: Knows h, wants to be sure that h is in <g>
- Prover: Knows x such that $h=g^x$
  - Prover picks $r_1,\ldots,r_k$ and sets $\mathrm{z_i} = g^{x+r_i}$ for each i.
  - Prover selects the challenge $b_1,\ldots,b_k$= $H(z_1,\ldots,z_k)$ and sets the responses $R_i=r_i+b_ix$.
  - Prover outputs the proof $(\mathrm{z}_1,R_1),\ldots,(\mathrm{z}_\mathrm{k},R_k)$
- Verifier computes $b_1,\ldots,b_k$= H(z) and checks that $h^{1-b_i}z_i = g^{R_i}$ for each i.

- How to build the simulator?

Practice Problem 2: Better Soundness

- Build an (interactive) Zero-Knowledge Proof for the group membership problem with soundness 2^{-k} instead of k.
- Verifier: Knows h, wants to be sure that h is in <g>
- Prover: Knows x such that $h=g^x$

Protocol:

1. Prover picks $r_1,\ldots,r_k$ and sets $z_i = g^{x+r_i}$ for each i.
2. Verifier selects the challenge $b_1,\ldots,b_k$
3. Prover computes the responses $R_i=r_i+b_ix$.
4. Verifier checks that $h^{1-b_i}z_i = g^{R_i}$ for each i.

- How to build the simulator?

Practice Problem 2: Better Soundness in ZK

Protocol:

1. Prover picks $r_1,\ldots,r_k$ and sets $\mathrm{z_i} = g^{x+r_i}$ for each i.
2. Verifier selects the challenge $b_1,\ldots,b_k$
3. Prover computes the responses $R_i=r_i+b_ix$.
4. Verifier checks that $h^{1-b_i}z_i = g^{R_i}$ for each i.

- Trick Question!
- Simulator should not be able to output NIZK for claim (without tampering with random oracle)
- Dishonest verifier can set $b_1,\ldots,b_k = H(z_1,\ldots,z_k)$ to obtain NIZK proof π!
  - $\pi = (\mathrm{z_1},R_1), \ldots, (\mathrm{z_k},R_k)$

Practice Problem 2: Better Soundness in ZK

Protocol 2:

1. Verifier selects nonce b and sends y=H(b) to the prover.
2. Prover picks r_1,…,r_k and sets $\mathrm{z_i} = g^{x+ri}$ for each i.
3. Verifier reveals b and sets challenges b_1,…,b_k=b
4. Prover computes the responses R_i=r_i+b_ix.
5. Verifier checks that $h^{1-b_i}z_i = g^{R_i}$ for each i.

Practice Problem 3: Garbled Circuit Reuse

- Let f(a1,a2,b1,b2)=(a1 AND b1) OR (a2 AND b2)
- Alice sends Bob a Garbled Circuit with keys
    - Keys $K_{W,0}$ and $K_{W,1}$ for each input/output wire W.
    - Suppose Alice first runs the protocol with input (0,1) and Bob's input (1,1)
    - Which keys can Bob recover during the protocol?
        - $K_{a1,0}$, $K_{a2,1}$, $K_{b1,1}$, $K_{b2,1}$ (initial inputs),
        - $K_{AND_1,0}$, $K_{AND_2,1}$ (AND gates),
        - $K_{OR,1}$ (output)
- Later suppose Alice runs the protocol with new input (1,0) but does not re-garble the circuit (Bob's input is the same)
    - What keys can Bob recover after second iteration?
        - **Answer:** Every key except for $K_{b1,0}$, $K_{b2,0}$

Practice Problem 4: RSA Authentication

- RSA Based Authentication
  - Verifier sends random nonce r mod N to Prover
  - Prover authenticates with $R = r^d \bmod N$
  - Verifier checks that $R^e = r \bmod N$
- What would security definition look like for generic authentication protocol?
  - Define the game
- Is this protocol secure?
  - Yes (assuming RSA-Inversion assumption)

Practice Problem 5: RSA Overuse

- RSA Based Authentication
  - Verifier sends random nonce r mod N to Prover
  - Prover authenticates with $R= r^d \bmod N$
  - Verifier checks that $R^e=r \bmod N$
- Suppose we use the same secret key e for Key Encapsulation and for RSA Authentication?
  - KEM: outputs $(y,K=H(x))$ where $y=x^e \bmod N$

- What could go wrong?

Cryptography
CS 555

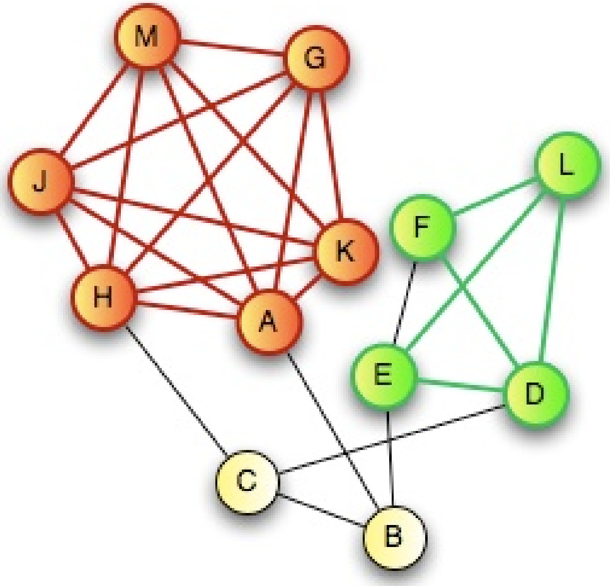
Week 16:

- Zero-Knowledge Proofs,
- Hot Topics in Cryptography
- Review for Final Exam

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

CS 555:Week 15: Zero-Knowledge Proofs

Zero-Knowledge Proof for all NP

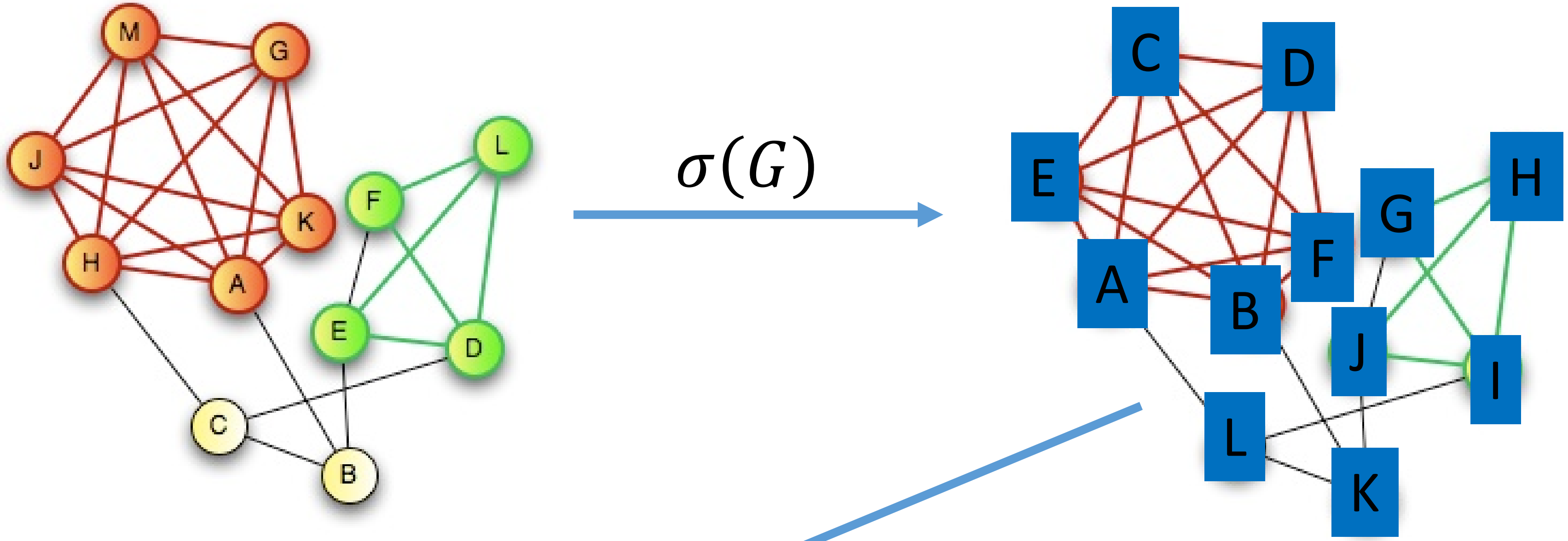


- CLIQUE
  - Input: Graph G=(V,E) and integer k>0
  - Question: Does G have a clique of size k?

- 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



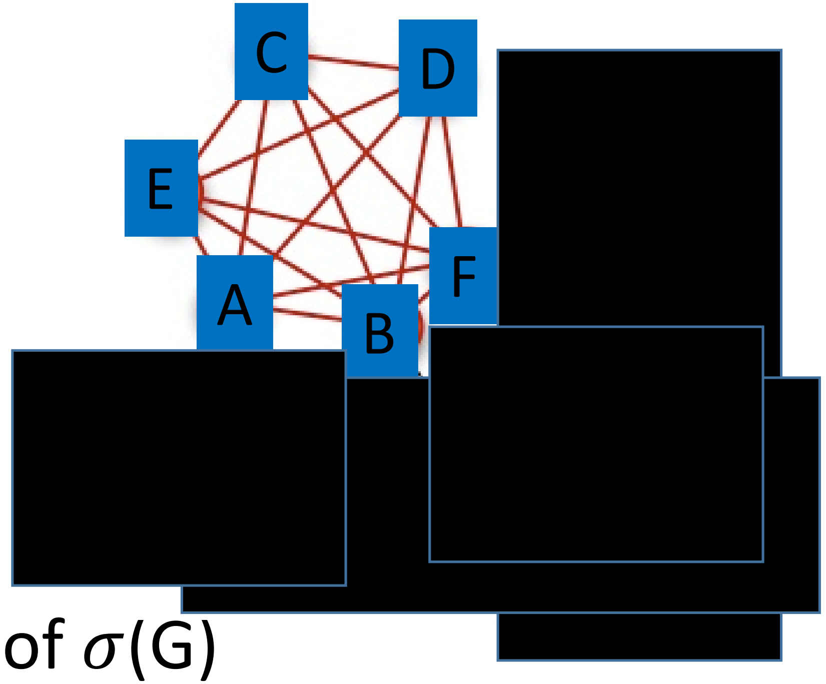
Adjacency matrix $A_{\sigma(G)}$

$$\begin{array}{cccc} & A & & L \\ A & 0 & \cdots & 1 \\ & \vdots & \ddots & \vdots \\ L & 1 & \cdots & 0 \end{array}$$

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

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

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 selects 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.

Soundness and Completeness

- **Completeness:** If the prover knows a clique he can always respond to the challenge.

- **Soundness:** If no clique exists then either
  1. The prover commits to (permutation of) the original graph
     →Cannot respond to challenge (c=1) to reveal *submatrix* containing clique
  2. The prover commits to a different (not-isomorphic) graph
     → Cannot respond to challenge to reveal permutation σ

Zero-Knowledge Proof Simulator

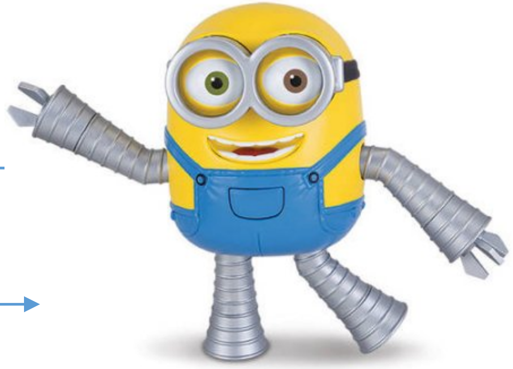
Dishonest (verifier);
$G = (V, E)$,

$$Com(A) = \begin{pmatrix} \mathrm{H}(A_{1,1}, r_{1,1}) & \cdots & \mathrm{H}(A_{1,n}, r_{1,n}) \\ \vdots & \ddots & \vdots \\ \mathrm{H}(A_{n,1}, r_{n,1}) & \cdots & \mathrm{H}(A_{n,n}, r_{n,n}) \end{pmatrix} \text{ if b=0}$$

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

$$\text{Response } \boldsymbol{r} = \begin{cases} \begin{pmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{n,1} & \cdots & r_{n,n} \end{pmatrix} \boldsymbol{and\ \sigma} & \boldsymbol{if\ c{=}b} \\ \bot & otherwise \end{cases}$$

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



Simulator
Cheat bit b,
$G = (V, E)$,
A= $\sigma(G)$
(random σ)

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

Zero-Knowledge Proof Simulator

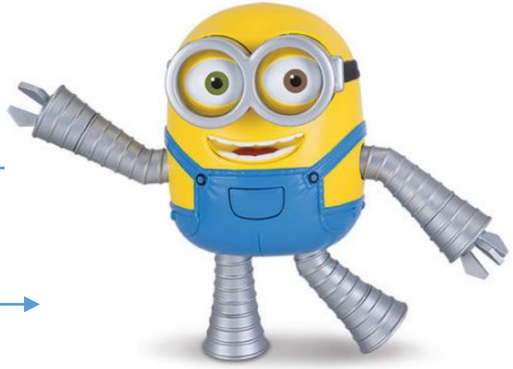
Dishonest (verifier);
$G = (V, E)$,

$$Com(K_n) = \begin{pmatrix} \mathrm{H}(0, r_{1,1}) & \cdots & \mathrm{H}(1, r_{1,n}) \\ \vdots & \ddots & \vdots \\ \mathrm{H}(1, r_{n,1}) & \cdots & \mathrm{H}(0, r_{n,n}) \end{pmatrix}$$ if b=0

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

$$\boldsymbol{r} = \begin{cases} \begin{pmatrix} r_{\sigma(1),\sigma(1)} & \cdots & r_{\sigma(1),\sigma(k)} \\ \vdots & \ddots & \vdots \\ r_{\sigma(1),\sigma(k)} & \cdots & r_{\sigma(k),\sigma(k)} \end{pmatrix} & \textit{if } c=b \\ \perp & \textit{otherwise} \end{cases}$$

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



Simulator
Cheat bit b,
$G = (V, E)$,
A= $\sigma(G)$
(random σ)

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

Zero-Knowledge Proof for all NP

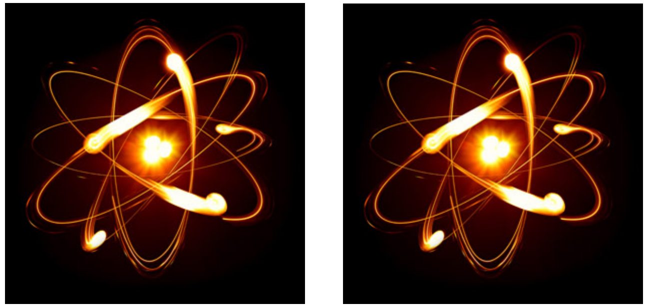
- **Completeness**: Honest prover can always make honest verifier accept
- **Soundness**: If prover commits to adjacency matrix $A_{\sigma(G)}$ of σ(G) and can reveal a clique in submatrix of $A_{\sigma(G)}$ then G itself contains a k-clique. Proof invokes binding property of commitment scheme.
- **Zero-Knowledge:** 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 are unsafe ☹
  - 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 are believed to be safe
  - Lamport One-Time Signatures and extensions to many-time signatures
- 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

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

- Bob has PK_A and can also include his own encrypted inputs $Enc_{PK_A}(y_i)$
- **Many Applications:**
  - Export confidential computation to cloud
  - Secure Multiparty Computation,…

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

Partially Homomorphic Encryption

- **Secret Key:** Large (prime) number p.
- **Public Key:** $\mathrm{N = pq \ and} \ x_i = pq_i + 2r_i + 1$ for each $i \le t$ where $r_i \ll p$

- Encrypting a Bit b:
    - Select Random Subset: $S \subset [t]$ and random $r \ll p$
    - Return $c = b + 2r + \sum_{i \in S} x_i \ mod \ N = p \sum_{i \in S} q_i + 2(r + \sum_{i \in S} r_i) + b \ mod \ N$
- Decrypting a ciphertext:
    - *As long as* $2(r + \sum_{i \in S} r_i) < p$
    - $(c \ mod \ p) mod \ 2 = (2(r + \sum_{i \in S} r_i) + b\ ) \ mod \ 2 = b$

Partially Homomorphic Encryption

- Encrypting a Bit b:
  - Select Random Subset: $S \subset [t]$ and random $r \ll p$
  - Return $c = b + 2r + \sum_{i \in S} x_i \ mod\ N = p \sum_{i \in S} q_i + 2(r + \sum_{i \in S} r_i) + b$
- **Adding two ciphertexts**

$$c + c' = p\left(\sum_{i \in S} q_i + \sum_{i \in S'} q_i\right) + 2\underbrace{\left(r + r' + \sum_{i \in S} r_i + \sum_{i \in S'} r_i\right)}_{\textbf{Noise increases a bit}} + b + b'$$

Partially Homomorphic Encryption

- Encrypting a Bit b:
  - Select Random Subset: $S \subset [t]$ and random $r \ll p$
  - Return $c = b + 2r + \sum_{i \in S} x_i \ mod\ N = p \sum_{i \in S} q_i + 2(r + \sum_{i \in S} r_i) + b$
- **Multiply two ciphertexts**

$$cc' = p\left(\sum_{i \in S} q_i \sum_{i \in S'} q_i + \sum_{i \in S} q_i \sum_{i \in S'} r_i + \cdots\right) + \underbrace{4\left(\left(\boldsymbol{r} + \sum_{i \in S} \boldsymbol{r}_i\right)\left(\boldsymbol{r}' + \sum_{i \in S'} \boldsymbol{r}_i\right)\right) + 2b\left(r + \sum_{i \in S'} r_i\right) + 2b'\left(r + \sum_{i \in S} r_i\right)}_{\textbf{Noise increases a bit more (multiplicative)}} + bb'$$

Bootstrapping (Gentry 2009)

- Transform Partially Homomorphic Encryption Scheme into Fully Homomorphic Encryption Scheme
- **Key Idea:**
  - Maintain two public keys pk_1 and pk_2 for partially homomorphic encryption
    - Also, encrypt sk_1 using pk_2 and encrypt sk_2 under pk_1
    - The ciphertexts are included in the public key
  - Run homomorphic evaluation using pk_1 until the noise gets to be too large
  - Let $c_1,…,c_k$ be intermediate ciphertext(s) (under key pk_1)
  - Encrypt $c_1,…,c_k$ bit by bit under (under key pk_2)
  - Then evaluate the decryption circuit homorphically (under key pk_2)
  - **Challenge:** Need to make sure that decryption circuit is shallow enough to evaluate…
- Expensive, but there are tricks to reduce the running time

Fully Homomorphic Encryption Resources

- Implementation: https://github.com/shaih/HElib
- Tutorial: https://www.youtube.com/watch?v=jIWOR2bGC7c

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
  - “Best Possible Obfuscation” cannot distinguish O(C) from O(C’) when |C| = |C’| compute the same function
  - 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

United States™ Census Bureau

YAHOO!®

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]

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

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

Differential Privacy vs Cryptography

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

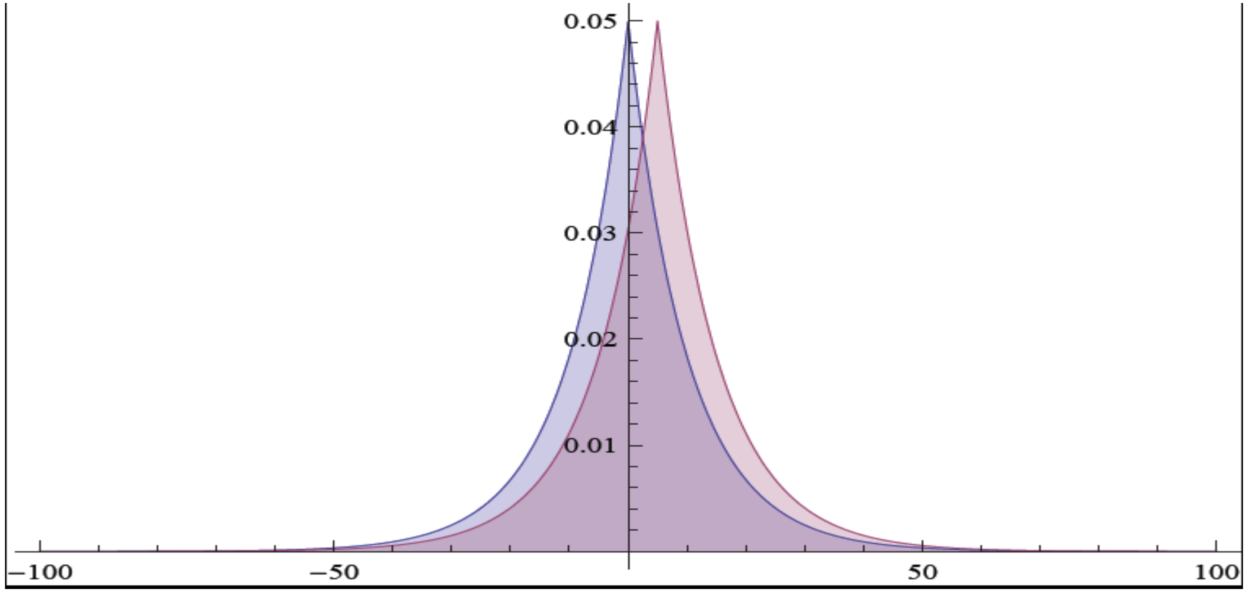
- Cryptography
  - Insiders/Outsiders
  - Only those with decryption key(s) can reveal secret
  - Multiparty Computation: Alice and Bob learn nothing other than f(x,y)

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)



Traditional Differential Privacy Mechanism

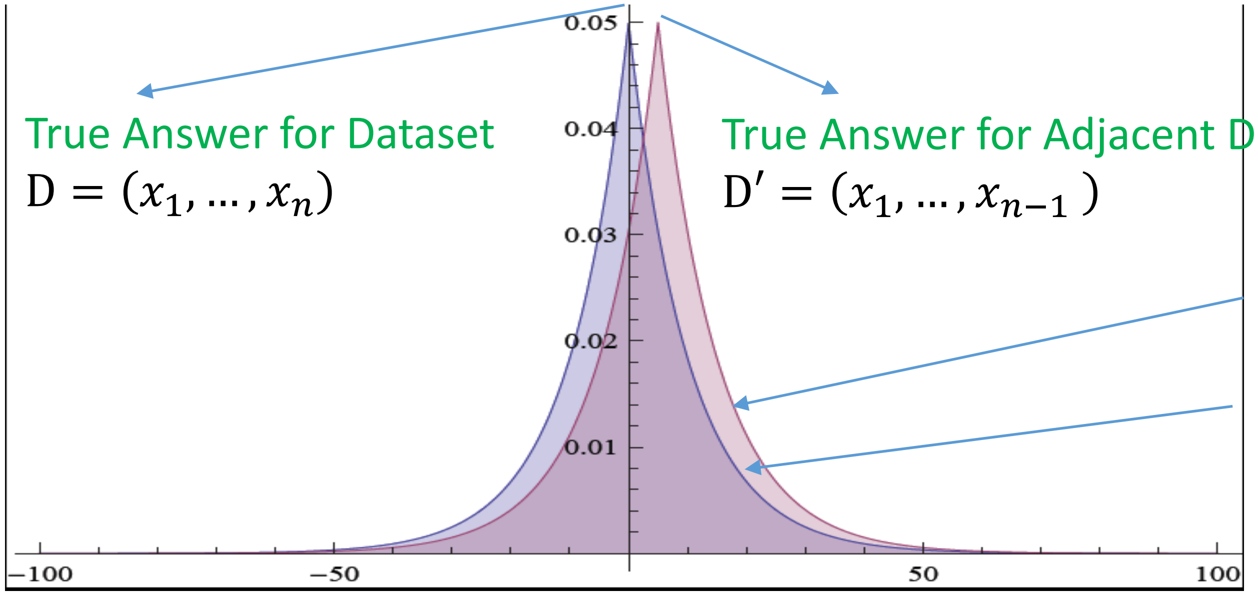
Theorem: Let $\mathrm{D} = (x_1, \dots, x_n) \in \{0,1\}^n$

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

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

Observe:

$$\frac{\Pr[\mathrm{A}(\mathrm{D}') = 20]}{\Pr[\mathrm{A}(\mathrm{D}) = 20]} = \frac{e^{-|19-0|\varepsilon}}{e^{-|20-0|\varepsilon}} = e^{\varepsilon}$$

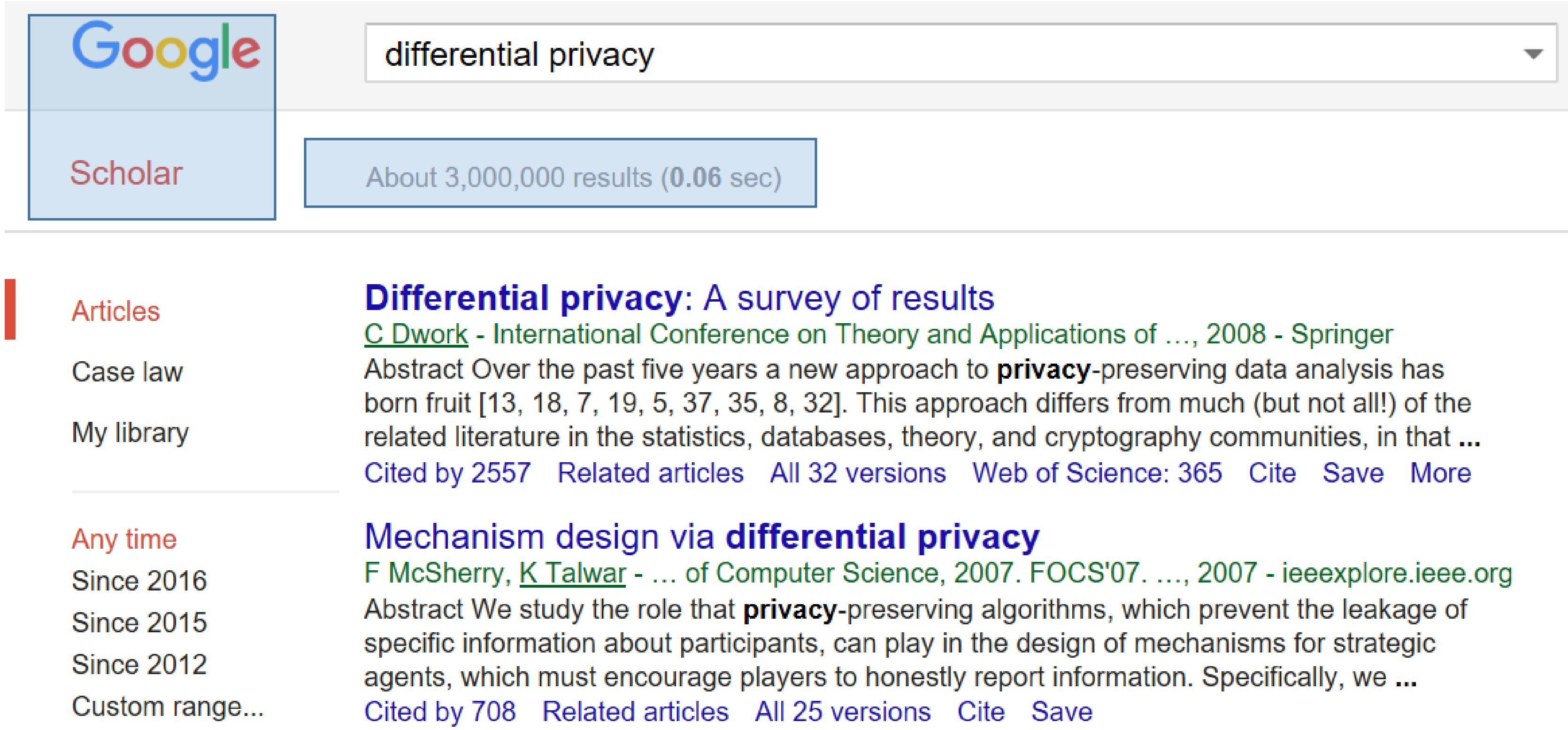


True Answer for Dataset $\mathrm{D} = (x_1, \dots, x_n)$

True Answer for Adjacent Dataset $\mathrm{D}' = (x_1, \dots, x_{n-1})$

$\Pr[\mathrm{A}(\mathrm{D}') = 20] \propto e^{-|19-0|\varepsilon}$

$\Pr[\mathrm{A}(\mathrm{D}) = 20] \propto e^{-|20-0|\varepsilon}$

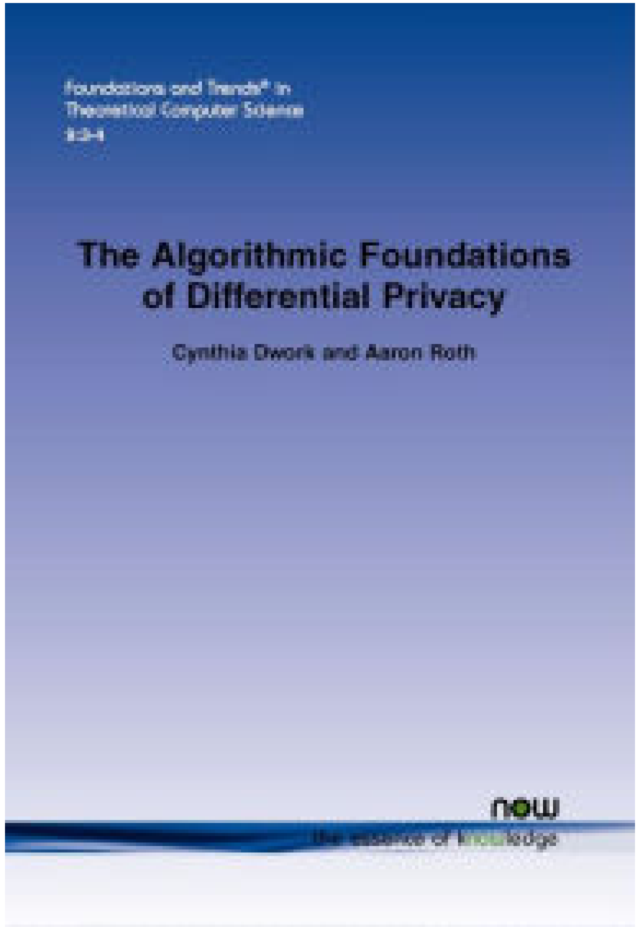


Differential privacy: A survey of results
C Dwork - International Conference on Theory and Applications of …, 2008 - Springer
Abstract Over the past five years a new approach to **privacy**-preserving data analysis has born fruit [13, 18, 7, 19, 5, 37, 35, 8, 32]. This approach differs from much (but 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



Foundations and Trends® in
Theoretical Computer Science

The Algorithmic Foundations
of Differential Privacy

Cynthia Dwork and Aaron Roth

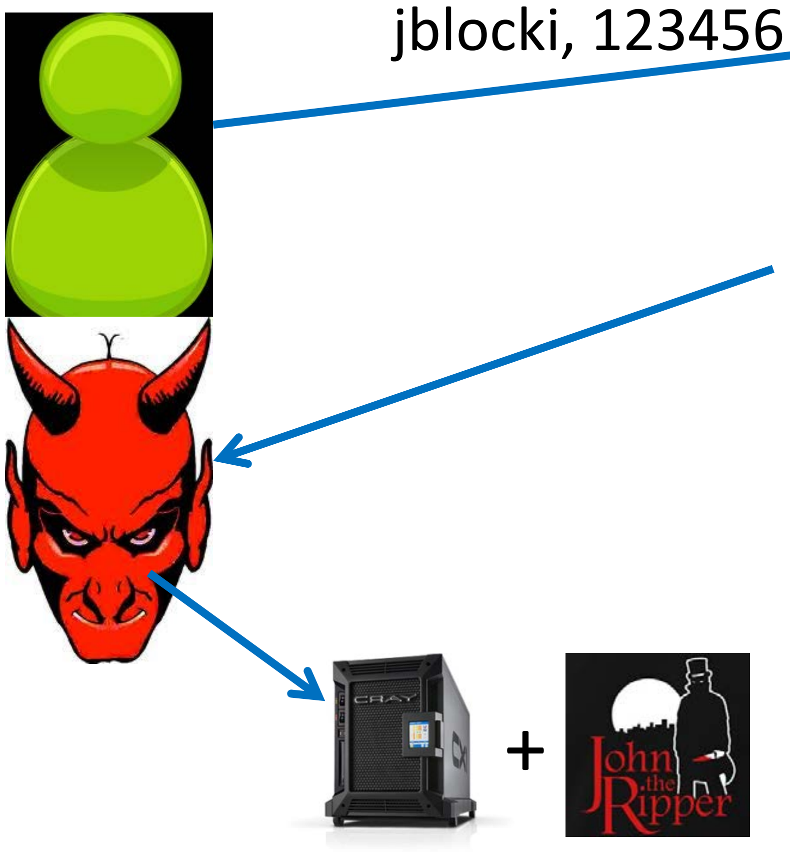
now
the essence of knowledge

BARNES & NOBLE

- $99

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

Password Storage and Key Derivation Functions



Username	Salt	Hash
jblocki	89d978034a3f6	85e23cfe0021f58 4e3db87aa72630 a9a2345c062

SHA1(12345689d978034a3f6)=85e23cfe0021f584e3db87aa72630a9a2345c062

Offline Attacks: A Common Problem

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

LastPass ****

SONY

ebay

AdultFriendFinder®

ASHLEY MADISON®
Life is short. Have an affair.®

LinkedIn®

Dropbox

Zappos.com
the web's most popular shoe store!

rockyou

YAHOO!

Adobe

GAWKER

livingsocial®

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

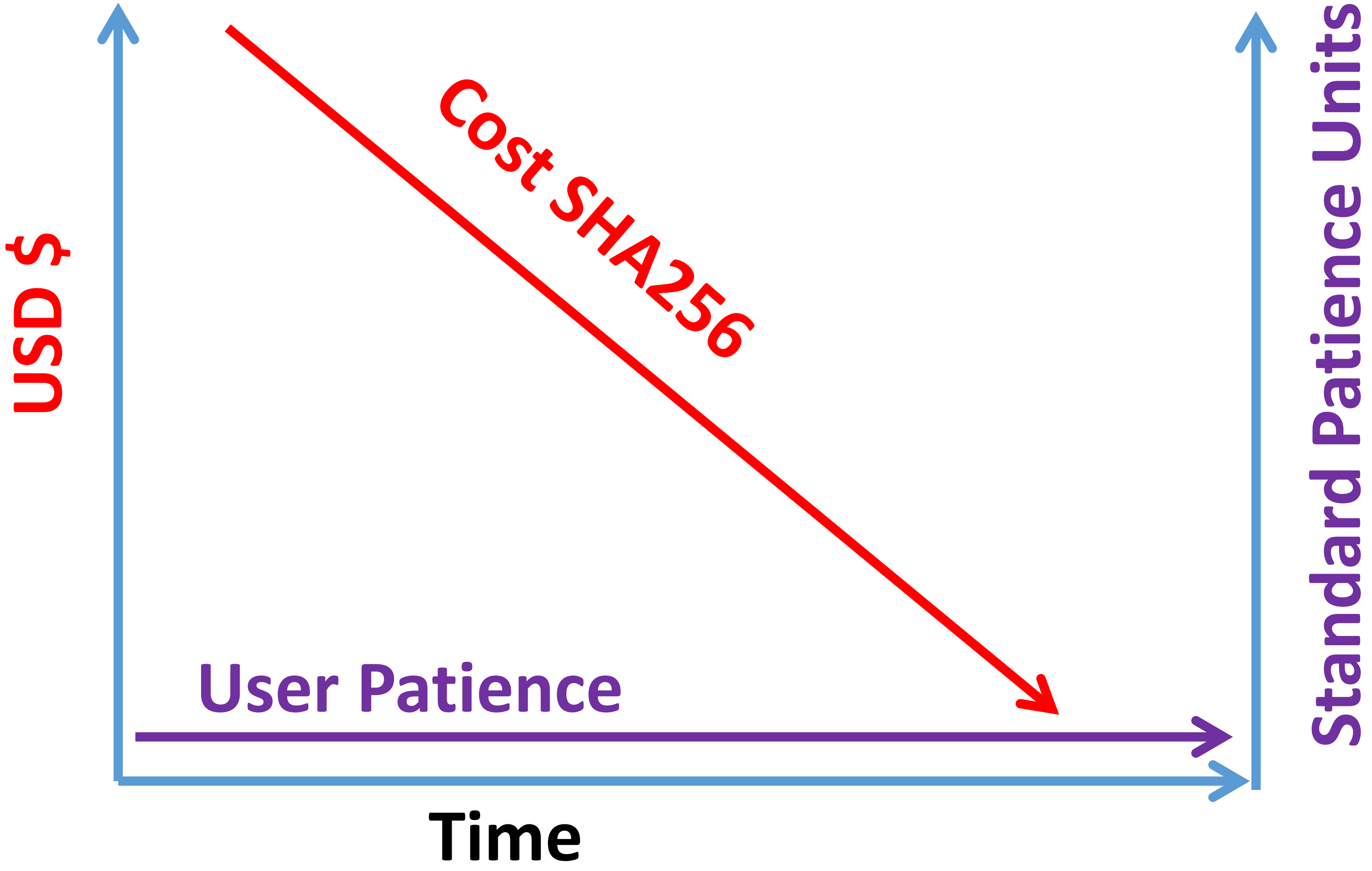
- PBKDF2

LastPass ****

100,000 SHA256 computations (iterative)

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

The Challenge

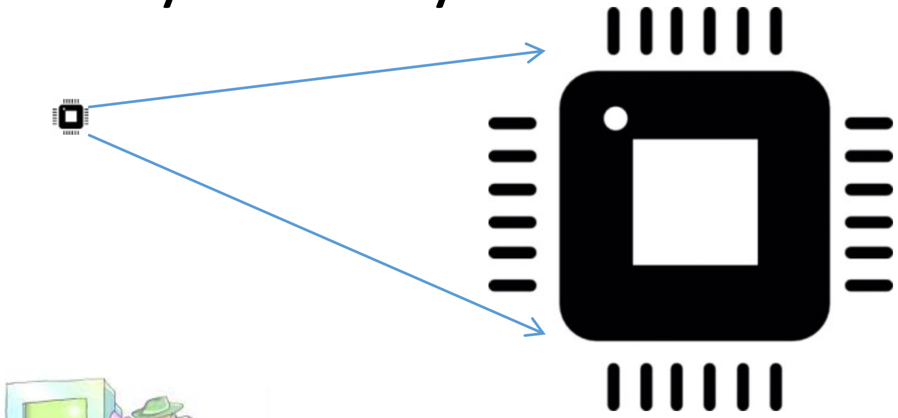


Disclaimer: This slide is entirely for humorous effect.

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

(2013-2015)

We recommend that you use Argon2…

https://password-hashing.net/

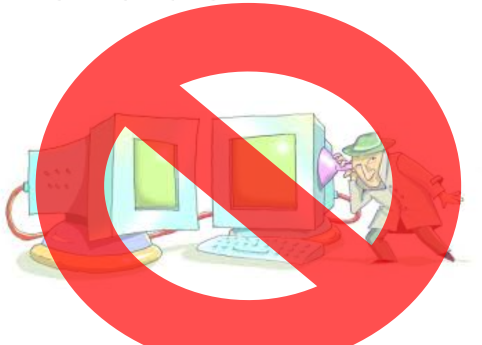
password
hashing
competition

(2013-2015)

https://password-hashing.net/

We recommend that you use Argon2…

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

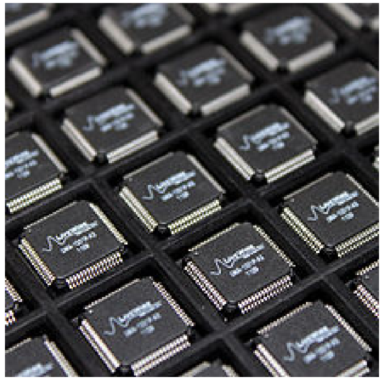


Depth-Robustness: The Key Property

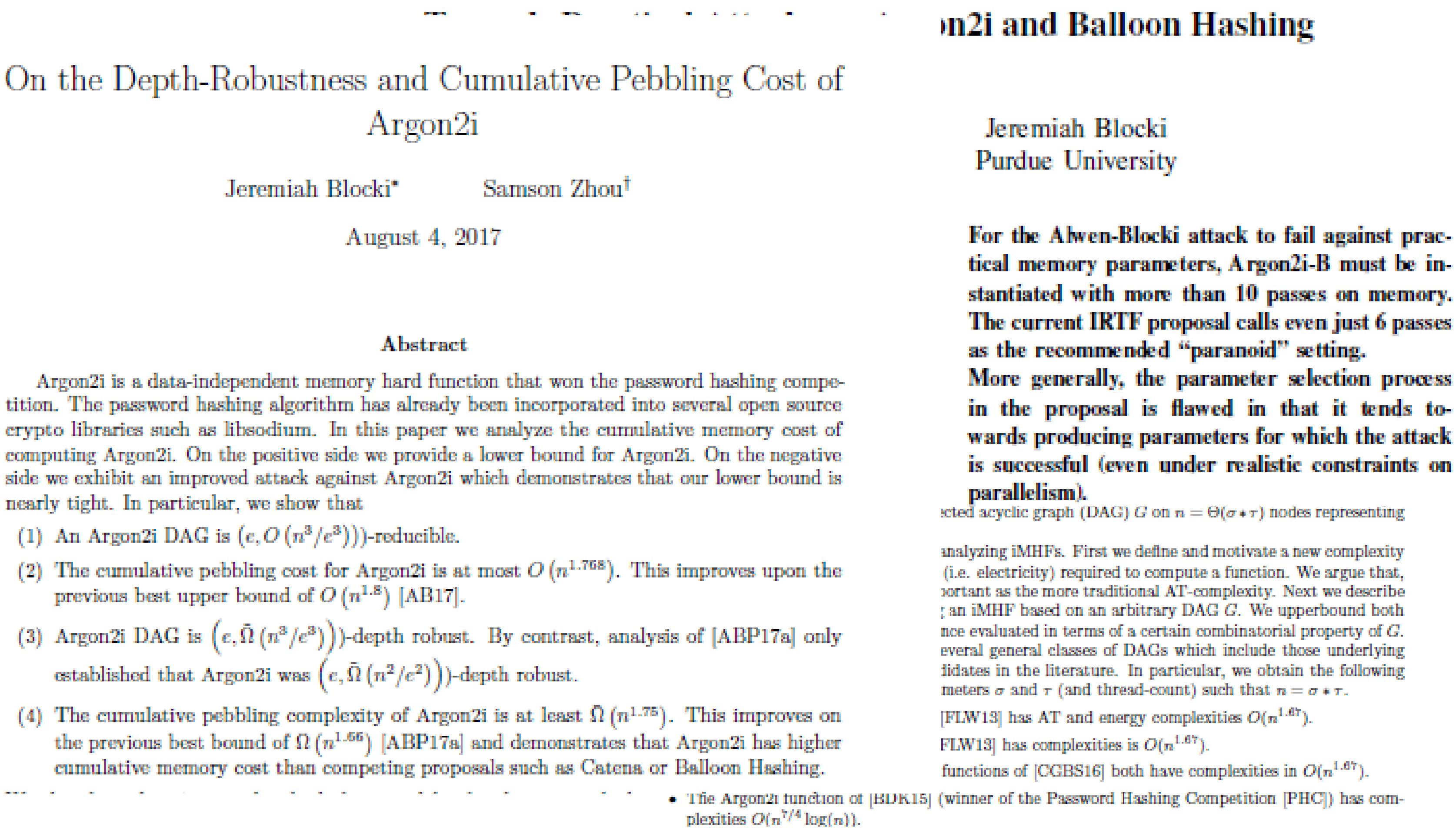
<u>Necessary</u> [AB16] and <u>sufficient</u> [ABP16] for secure iMHFs

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).

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

nalyzing iMHFs. First we define and motivate a new complexity
(i.e. electricity) required to compute a function. We argue that,
ortant 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 $n = \sigma * \tau$.

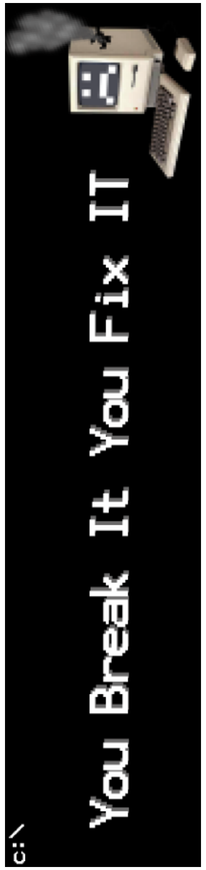
[FLW13] has AT and energy complexities $O(n^{1.67})$.

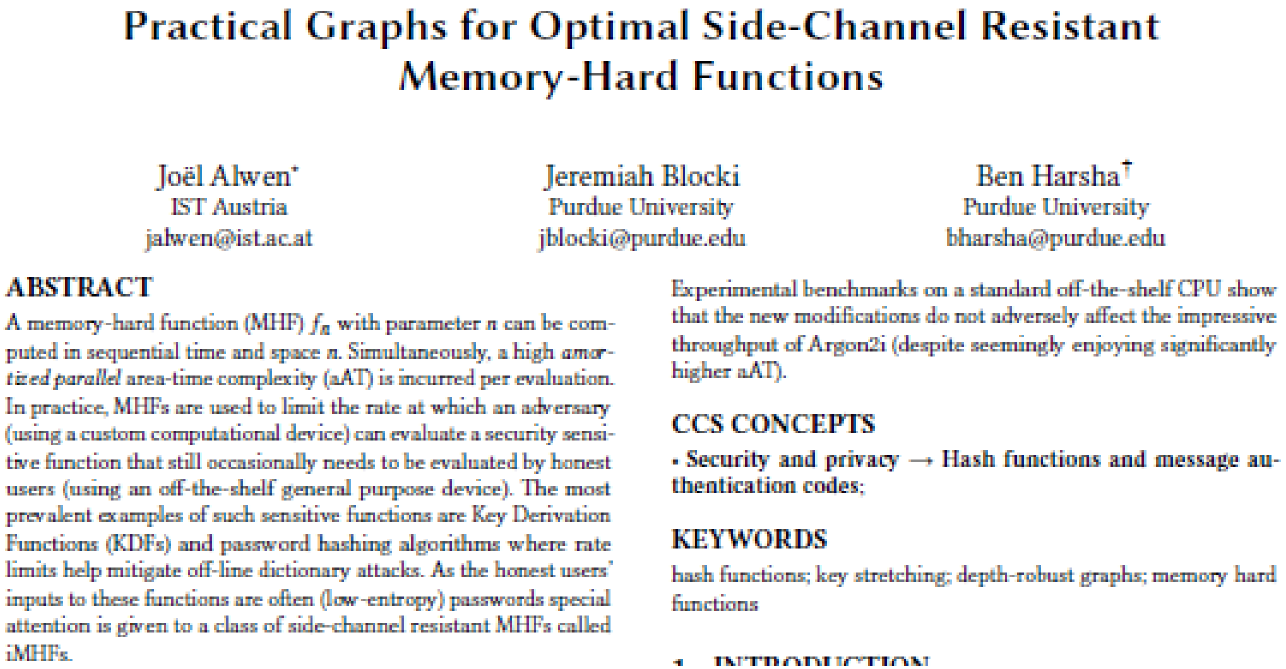
FLW13] has complexities is $O(n^{1.67})$.

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