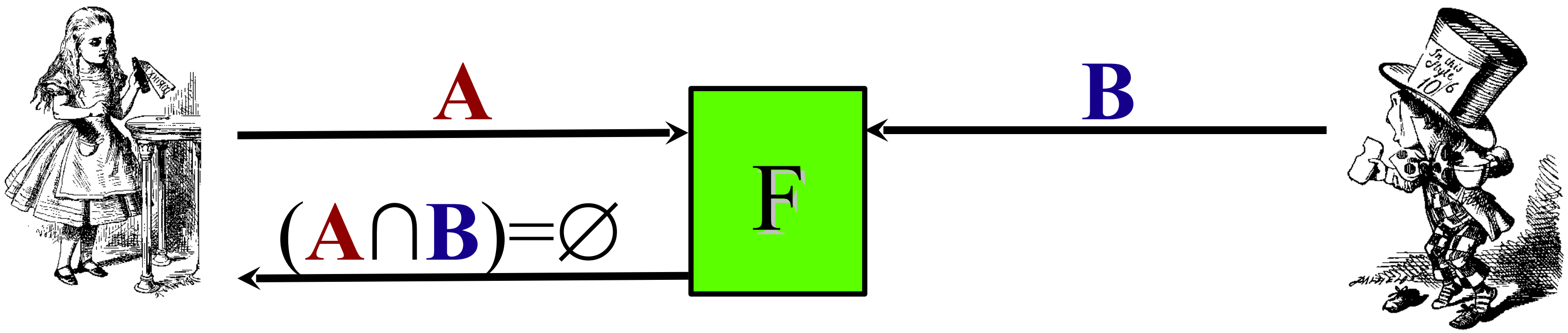
Honest-Verifier Private Disjointness Testing without Random Oracles

Susan Hohenberger
MIT CSAIL
IBM / John Hopkins

Stephen A. Weis
MIT CSAIL
Google

Workshop on Privacy Enhancing Technologies
June 28th, 2006

Private Disjointness Testing



- <u>PDT Problem:</u>
  - Two parties have a private set **A** and **B**.
  - One party learns the bit (**A**∩**B**)=∅.
  - Neither party learns any other information.

Organization

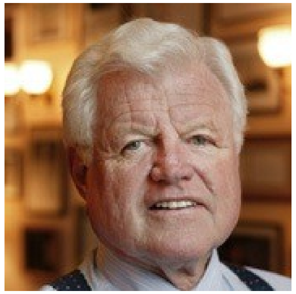
- Private disjointness testing applications
- Prior tools, solutions, and problems
- Our tools and solutions

Our Contributions

- *Testable and homomorphic commitments* (THCs) based on subgroup decision assumptions
- A private disjointness testing construction:
  - Efficient, based on standard tools
  - Secure against malicious provers
  - Does not use random oracles

Application: No Fly List

- Alice has a secret no-fly list **A**:


- Bob has a private passenger list **B**
- Alice should learn only whether $(\mathbf{A} \cap \mathbf{B}) = \emptyset$
- Neither party should learn anything else
- Bob can't cause false alarms

Application: Anonymous Logins

- Alice has a secret user list $\mathbf{A}$
- Bob wants to login anonymously with some identity in a set $\mathbf{B}$
- If $(\mathbf{A} \cap \mathbf{B}) = \varnothing$, Bob does not get access
- Otherwise, Alice knows he is *some* user

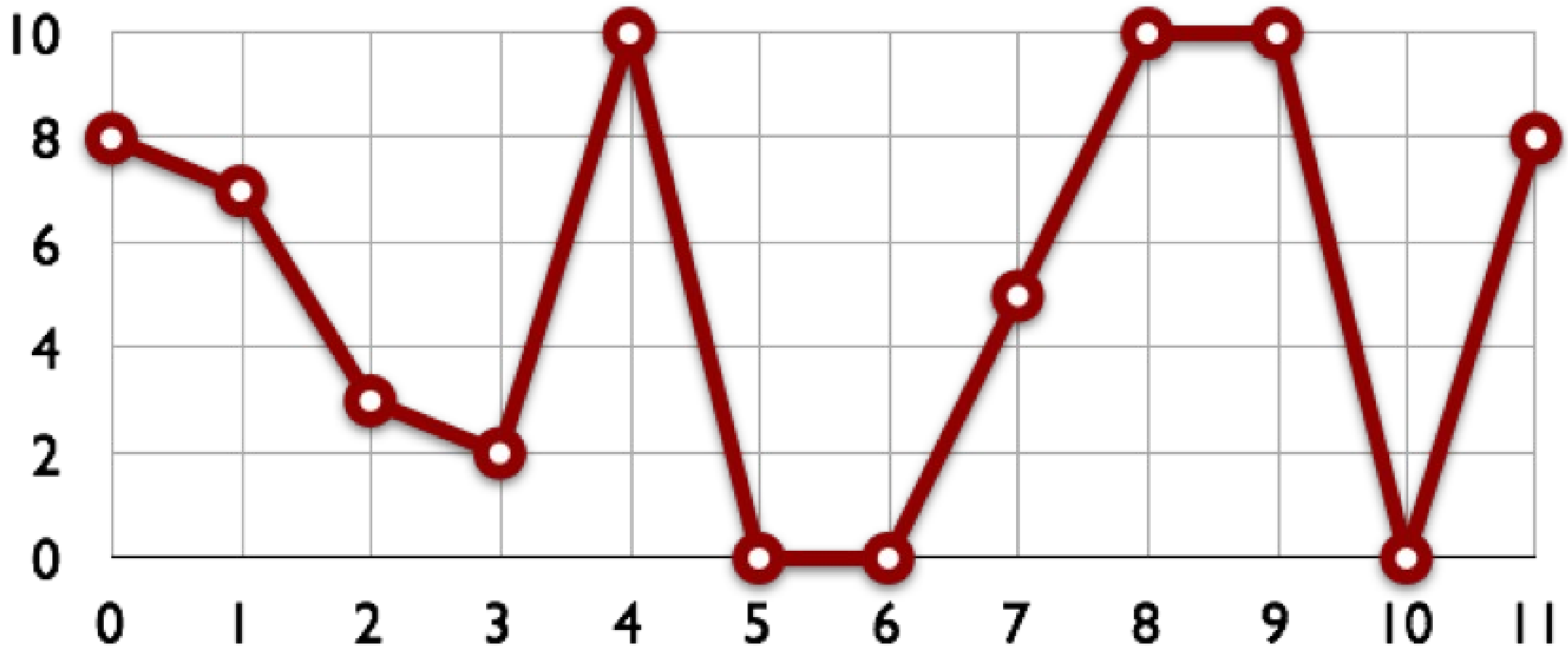
Prior Solution: Secure Function Evaluation

- Secure multi-party computation & secure function evaluation [GMW, BGW]
- Theses techniques are still too inefficient
- Specialized protocols are more practical

Prior Tool: Embedding Sets in Polynomials

Prime p, Input: Set $\mathbf{A} = \{a_1, a_2, \ldots, a_n\}$

$$f(x) = \Pi\ (x-a_j) = \Sigma\ \alpha_i x^i \pmod{p}$$



Example: $p = 11$, $\mathbf{A} = \{5, 6, 10\}$

$$f(x) = (x-5)(x-6)(x-10) = x^3+x^2+8x+8 \pmod{11}$$

Prior Tool: Homomorphic Encryption

- Allows operations between ciphertexts:
  - Addition:

    $$E_k(x)+E_k(y) = E_k(x+y)$$

  - Constant Multiplication:

    $$E_k(x)\cdot y = E_k(x\cdot y)$$

- Example: Paillier Cryptosystem

Prior Tool: Oblivious Polynomial Evaluation

- Given example $f(x) = x^3+x^2+8x+8 \pmod{11}$
- Publish encrypted coefficients:

  $\mathbf{c_3}=E_k(1)$, $\mathbf{c_2}=E_k(1)$, $\mathbf{c_1}=E_k(8)$, $\mathbf{c_0}=E_k(8)$
- Anyone can obliviously evaluate $f(\mathbf{b})$:

  $E_k(f(\mathbf{b})) = \mathbf{c_3 b^3+c_2 b^2+c_1 b+c_0}$
- If $f(\mathbf{b})=0$, then $\mathbf{b} \in \mathbf{A}$
- Not private: Semi-honest verifier learns $f(\mathbf{b})$

Prior Solution: FNP Private Set Membership

- Verifier (Alice) embeds set $\mathbf{A}$ in $f(x)$ and homomorphically encrypts coefficients $\mathbf{c_i}$
- Prover (Bob) chooses random $\mathbf{r}$ and obliviously evaluates $\mathbf{c}=E_k(f(\mathbf{b})\cdot\mathbf{r})$
- Verifier decrypts $\mathbf{c}$. If $D_k(\mathbf{c})=0$, *honest* prover evaluated f at some point in $\mathbf{A}$
- Otherwise, random $\mathbf{r}$ clobbers $f(\mathbf{b})$

Problem with Prior Work

- [FNP '04]: Paillier's homomorphic encryption
- [KM '05]: ElGamal "superposed encryption"
- Problem with both: Prover can just encrypt 0
- Prover sends $E_k(0)$ -- spurious intersection
- Can fix this with random oracles and expensive zero-knowledge sub-protocols

Our Tool: Testable and Homomorphic Commitments

- Private Operations:

Commit: $\mathrm{Com}(x,r)$

Equality Test: $\mathrm{Test}(\mathrm{Com}(x,r), y) = 1$ iff $(x = y)$

- Public Operations:

Add: $\mathrm{Com}(x_1,r_1)+\mathrm{Com}(x_2,r_2)=\mathrm{Com}(x_1+x_2, r_1+r_2)$

Constant Multiplication: $c\cdot\mathrm{Com}(x,r)=\mathrm{Com}(cx,cr)$

Prior Tool: Pedersen Commitments

- Perfectly hiding, homomorphic commitment
- Two bases, g and h, generate Z_p, for prime p
- $\mathrm{Com}(x,r) = g^x h^r$
- Open commitment by revealing x and r

Our THC Construction

- Pick group $\mathbf{G}$ of multiplicative order $n = p{\cdot}q$

  For example, $QR_{p'}$ *where* $p' = 2p{\cdot}q+1$
- Pick random group generator g of order n
- Pick random subgroup generator h of order q
- Publish $\mathbf{G}$ and n.
- Keep p, q, g, and h secret.

Our THC Construction

- Private Operations:

$$\mathrm{Com}(x,r) = g^x h^r$$

$$\mathrm{Test}(\mathrm{Com}(x,r),y) = (g^x h^r / g^y)^q = g^{q(x-y)}$$

- Public Operations:

$$\mathrm{Com}(x,r)\cdot\mathrm{Com}(y,s) = g^{x+y}\cdot h^{r+s} = \mathrm{Com}(x+y,\ r+s)$$

$$\mathrm{Com}(x,r)^c = (g^x\cdot h^r)^c = g^{cx}\cdot h^{cr} = \mathrm{Com}(cx,cr)$$

Our PDT Solution

- Verifier embeds set in polynomial $f(x)$
- Verifier encodes coefficients $\mathbf{c_i} = \mathrm{Com}(\alpha_i, r_i)$
- Prover evaluates $\mathrm{Com}(f(\mathbf{b}))=g^{f(\mathbf{b})}h^{r(\mathbf{b})}$ and multiplies it by a random R, $\mathbf{c} = g^{Rf(\mathbf{b})}h^{Rr(\mathbf{b})}$
- Verifier: If $\mathrm{Test}(\mathbf{c},0) = \mathbf{c}^q = 1$ then $\mathbf{b} \in \mathbf{A}$
- Otherwise, $\mathbf{c}$ is some random value in $\mathbf{G}$

Malicious Provers

Given $\mathbf{G}$, n, and the $\mathbf{c}_i$ coefficients...

- Can the prover extract info about $\mathbf{A}$?
- Can he cause spurious intersections?

Malicious Prover ZK

- Commitment perfectly hides coefficients
- Except values are in a group of order n with subgroups of order p and q
- If the prover can distinguish subgroup elements, it might learn info about $\mathbf{A}$
- *Subgroup Decision Assumption:* It is hard to distinguish subgroup elements

Soundness

- If malicious prover outputs an element of order q, verifier thinks there is an intersection
- Commitments reveal no information
- *Subgroup Computation Assumption:* Hard to compute subgroup elements

SDA and SCA

- Same assumptions made by:
  - [YS'01] - Private Information Retrieval
  - [BGN'05] - Encryption
  - [GOS'06] - Perfect Zero-Knowledge
- Factoring and discrete log must be hard for SDA/SCA to be hard
- Unknown relation to DDH/CDH

Malicious Verifiers?

- Malicious verifier might pick a group with many subgroups or a weird polynomial
- We can fix with random oracles, repeated invocations, ZK protocols, etc.
- But then our scheme is no better than existing techniques

Comparing Security Assumptions

Setting	FNP '04	KM '05	HW '06
Semi-Honest	Factoring	DDH	SDA & SCA
Malicious Prover	Random Oracles	NIZK Proofs Random Oracles	**NONE**
Malicious Verifier	Multiple Invocations	UC-Commits Random Oracles	ZK Proofs?

Further Work

- Security against malicious verifiers without random oracles
- Efficiently implementing other private set operations, e.g. union, intersection
- Relation of SDA/SCA to DDH/CDH
- Multi-party protocols based on SDA/SCA

Questions?

- E-mail: `sweis@mit.edu`
- Homepage:
  - `http://saweis.net`
  - Has link to HB+ page and bibliography
- This fall: Google