Efficient Private Matching and Set Intersection

Mike Freedman, NYU

Kobbi Nissim, MSR

Benny Pinkas, HP Labs

EUROCRYPT 2004

A Story…

Is there any chance we might be compatible?

Maybe…

We could see if we have similar interests?

I don't really like to give personal information

Have you heard of "secure function evaluation" ?

I don't want to waste my entire night…

Making SFE more efficient…

1. Improvements to generic primitives (SFE, OT)
2. Improvements in specific protocol examples

We could see if we have similar interests?

Have you heard of "secure function evaluation" ?

I don't want to waste my entire night…

Secure Function Evaluation

Input: X Y

Output: F(x,y) *and nothing else*

As if… X Y

F(x,y) F(x,y)

What if such trustworthy barkeeps don't exist?

Proving SFE Protocols…

Real World

X → Y

X ← Y

≈ Views

X

Y

Ideal World

F(x,y)

F(x,y)

Can consider semi-honest and malicious models

Our Specific Scenario

		
	Client	Server
Input:	$X = x_1 \ldots x_k$	$Y = y_1 \ldots y_k$
Output:	$X \cap Y$ only	nothing

- Shared interests (research, music)
- Dating, genetic compatibility, etc.
- Credit card rating
- Terrorist watch list (CAPS II)

Related work

- Generic constructions [Yao,GMW,BGW,CCD]
  - Represent function as a circuit with combinatorial gates
  - Concern is size of circuit (as communication is O(|C|))
  - Simplest uses k^2 comparisons
- Diffie-Hellman based solutions [FHH99, EGS03]
  - Insecure against malicious adversaries
  - Considered in the "random oracle" model
- Our work: O(k ln ln k) overhead.
  - "Semi-honest" adversaries – in standard model
  - "Malicious" adversaries – in RO model

This talk…

- Overview
- Basic protocol in semi-honest model
- Efficient Improvements
- Extending protocol to malicious model
- Other results…

Basic tool: Homomorphic Encryption

- Semantically-secure public-key encryption
- Given Enc(M1), Enc(M2) can compute, without knowing decryption key,
  - $Enc(M1+M2) = Enc(M1) \cdot Enc(M2)$
  - $Enc(c \cdot M1) = [Enc(M1)]^{c}$, for any constant c
- Examples: El Gamal variant, Paillier, DJ

The Protocol

- Client (C) defines a polynomial of degree k whose roots are his inputs $x_1,\ldots,x_k$

$$P(y) = (x_1-y)(x_2-y)\ldots(x_k-y) = a_0 + a_1 y + \ldots + a_k y^k$$

- C sends to server (S) homomorphic encryptions of polynomial's coefficients

$$Enc(a_0),\ldots, Enc(a_k)$$

$$Enc(\ P(y)\ ) = Enc(\ a_0 + a_1 \cdot y^1 + \ldots + a_k \cdot y^k\ )$$

$$Enc(a_0) \cdot Enc\ (a_1)^{\ y1} \cdot \ldots \cdot Enc\ (a_k)^{\ yk}$$

The Protocol

- S uses homomorphic properties to compute,

  $\forall y,\ r_y \leftarrow \text{random}$

$$\text{Enc}(\ r_y \cdot P(y) + y\ )$$

if $y \in X \cap Y$ → Enc (y)

otherwise → Enc (random)

- S sends (permuted) results back to C
- C decrypts results, identifies y's

Variant protocols…cardinality

$\mathrm{Enc}(r_y \cdot P(y) + 1)$

if $y \in X \cap Y$ → Enc (1)

otherwise → Enc (random)

- Computes size of intersection: # Enc (1)
- Others… Output 1 iff $| X \cap Y | > t$

Security (semi-honest case)

- Client's privacy
  - S only sees semantically-secure enc's
  - Learning about C's input = breaking enc's
- Server's privacy (proof via simulation)
  - Client gets $X \cap Y$ in ideal (TTP) model
  - Given that, can compute E(y)'s and E(rand)'s and thus simulate real model

Efficiency

- Communication is O(k)
  - ✓ C sends k coefficients
  - ✓ S sends k evaluations on polynomial

- Computation
  - ✓ Client encrypts and decrypts k values
  - ✗ Server:
    - $\forall y \in Y$, computes $Enc(r_y \cdot P(y)+y)$, using k exponentiations
    - Total $O(k^2)$ exponentiations

Improving Efficiency (1)

- Inputs typically from a "small" domain of D values. Represented by log D bits (…20)

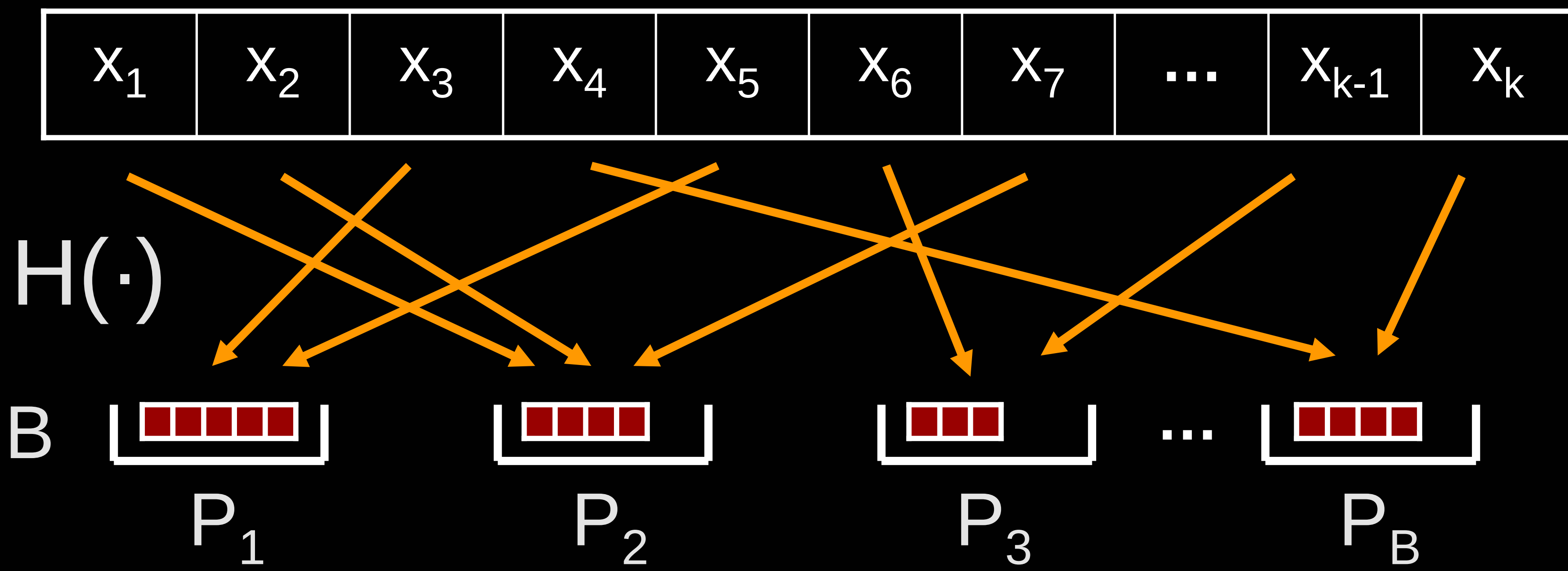
- Use Horner's rule

$$P(y) = a_0 + y\,(a_1 + \ldots y\,(a_{n-1} + y a_n)\ \ldots)$$

  - That is, exponents are only log D bits
  - Overhead of exponentiation is linear in | exponent |

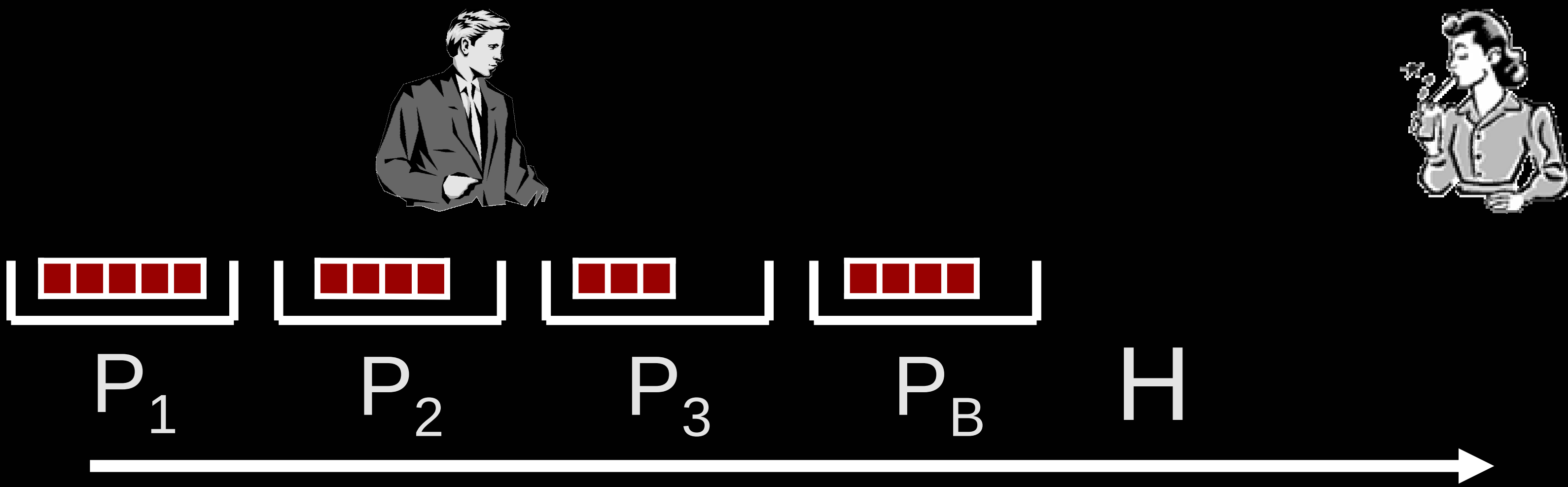
→ "Improve" by factor of | modulus | / log D
e.g., 1024 / 20 ≈ 50

Improving Efficiency (2): Hashing



- C uses PRF H(·) to hash inputs to B bins
- Let M bound max # of items in a bin
- Client defines B polynomials of deg M. Each poly encodes x’s mapped to its bin + enough “other” roots

Improving Efficiency (2): Hashing



$$\forall y,\ \ i \leftarrow H(y),\ r_y \leftarrow \text{rand}$$

$$\text{Enc}(\ r_y \cdot P_i(y) + y\ )$$

- Client sends B polynomials and H to server.
- For every y, S computes H(y) and evaluates the single corresponding poly of degree M

Overhead with Hashing

- Communication: $B \cdot M$ (# bins · # items per)
- Server: $k \cdot M$ short exp's, k full exp's

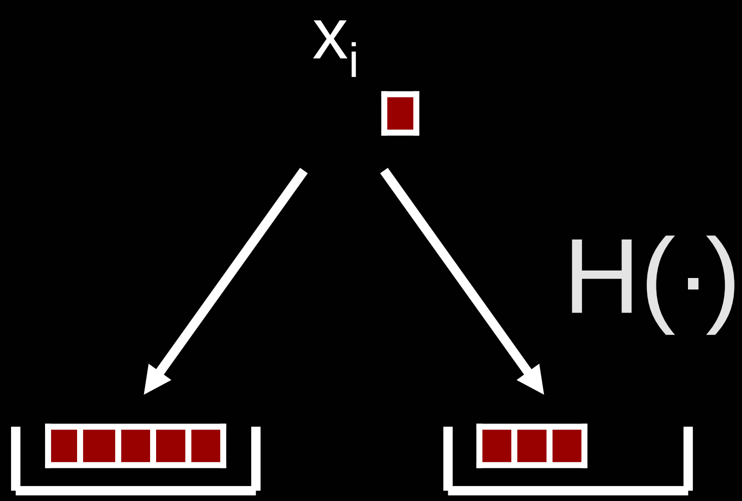
  ($P_i(y)$) ($r_y \cdot P_i(y) + y$)
- How to make M small as possible?
- Choose most balanced hash function

Balanced allocations [ABKU]

- H: Choose two bins, map to emptier bin
- $B = k / \ln \ln k$

  $\rightarrow M = O(\ln \ln k)$

  $M \leq 5$ [BM]



- Communication: $O(k)$
- Server: $k \ln \ln k$ short exp, k full exp in practice

This talk…

- Overview
- Basic protocol in semi-honest model
- Efficient Improvements
- Extending protocol to malicious model
- Other results…

Malicious Adversaries

- Malicious clients
  - Without hashing is trivial: Ensure $a_0 \neq 0$
  - With hashing
    - Verify that total # of roots (in all B poly's) is k
    - Solution using cut-and-choose
    - Exponentially small error probability
  - Still standard model
- Malicious servers
  - Privacy…easy:

    S receives semantically-secure encryptions

Security against Malicious Server

- Correctness: Ensure that there is an input of k items corresponding to S's actions
- Problem: Server can compute $r_y \cdot P(y) + y'$
- Solution: Server uses RO to commit to seed, then uses resulting randomness to "prove" correctness of encryption

Security against Malicious Server

$\forall y,\ s \leftarrow \text{rand},\ r \leftarrow H_1(s)$

$[e,h] \leftarrow [\ \text{Enc}\ (\ r_1 \cdot P(y) + s\ )\ ,\ \ H_2\ (r_2, y)\ ]$

Deterministic

$s^* \leftarrow \text{Dec}\ (e),\ r^* \leftarrow H_1(s^*)$

$?\ \exists\ x,\ \text{s.t.}$

$e = \text{Enc}\ (\ r^*_1 \cdot P(x) + s^*\ ) \quad \wedge \quad h = H_2\ (r^*_2,\ x)$

Other results and open problems

- Approximating size of intersection (scalar product)
  - Requires Ω(k) communication
  - Provide secure approximation protocol
- PM protocol extends efficiently to multiple parties
- Malicious-party protocol in standard model?
- Fuzzy Matching?
  - Databases are not always accurate or full
  - Report iff entries match in t out of V “attributes”

Questions?