Efficient Private Matching and Set Intersection

Mike Freedman, NYU
Kobbi Nissim, MSR
Benny Pinkas, HP Labs

(To appear in EUROCRYPT 2004)

A Story…

CVS/pharmacy

RITE AID It's not just a store, it's a solution

We think patients are misusing prescriptions to obtain drugs…

Here too..

We could share our lists of patients?

But, what about HIPAA? And we're competitors!

Have you heard of "secure function evaluation" ?

This is all "theory". It can't be efficient.

A Story…

CVS/pharmacy

RITE AID It's not just a store, it's a solution

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

We could share our lists of patients?

Have you heard of "secure function evaluation" ?

This is all "theory". It can't be efficient.

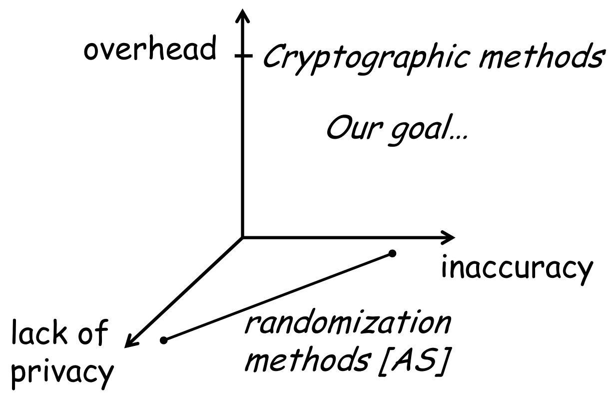
The Scenario

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

- Enterprises and government holding sensitive databases
- Peer-to-Peer networks
- Mobile wireless crowds (PDAs, cell phones)

Credit rating, CAPS II, shared interests (research, music), genetic compatibility, etc

Crypto vs. randomization methods



Related work

- Use a circuit for SFE [Yao,GMW,BGW]
- Use k^2 private equality tests
  - Single inputs x,y; return 1 iff x = y, 0 otherwise
  - (O(k) computation [NP])
- Diffie-Hellman based solutions [FHH99, EGS03]
  - Insecure against malicious adversaries
  - Depend on a "random oracle" assumption
- Our work: O(k ln ln k) overhead.
  - "Semi-honest" adversaries – no RO assumption
  - "Malicious" adversaries – with RO assumption

This talk…

- Overview
- Basic protocol in semi-honest model
- Efficient Improvements
- A little on…
  - Extending protocol to malicious model
  - Approximation bounds
  - Multi-party security
  - Fuzzy matching

Basic tool: Homomorphic Encryption

- Semantically-secure public-key encryption
- Given Enc(M1), Enc(M2), can compute
  - Enc(M1+M2) = Enc(M1) · Enc(M2)
  - Enc(c · M1) = $[\text{Enc}(M1)]^{c}$, for any constant c

  without knowing decryption key
- Examples: El Gamal, Paillier, DJ

The Protocol

- Client (C) defines a polynomial of degree k whose roots are her 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, $\forall y,\ r \leftarrow$ random

$$Enc(\ r \cdot P(y) + y\ )$$

if $y \in X \cap Y$ → Enc (y)

otherwise → Enc (random)

- S sends (permuted) results back to C

Variant protocols…cardinality

$$Enc(\ r \cdot P(y) + 1\ )$$

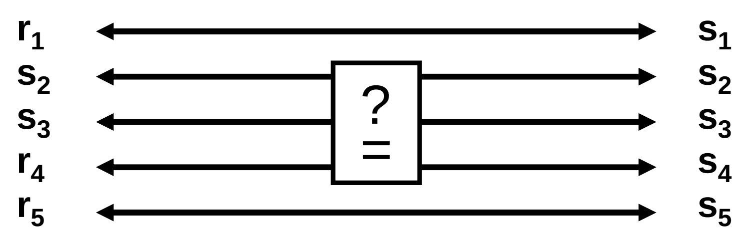
if $y \in X \cap Y$ → Enc (1)

otherwise → Enc (random)

- Computes size of intersection: # Enc (1)

Variant protocols…others

$$Enc(\ r \cdot P(y) + s\ )$$



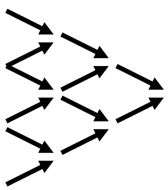
circuit

- $\forall y$, compute $r \cdot P(y) + s$, for $s \leftarrow$ random
- Perform Yao circuit on decrypted values

Variant protocols…others

Enc(r · P(y) + s)

$r_1 \neq$
$s_2 =$
$s_3 =$
$r_4 \neq$
$r_5 \neq$



circuit

e.g.,
| intersection | > threshold

s_1
s_2
s_3
s_4
s_5

- $\forall$y, compute r · P(y) + s, for s $\leftarrow$ random
- Perform Yao circuit on decrypted values

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 can simulate her view in the protocol, given the output of $X \cap Y$ alone: she can compute the enc's of items in $X \cap Y$ and of random items.

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·P(y)+y), using k exponentiations
    - Total $O(k^2)$ exponentiations

Improving Efficiency (1)

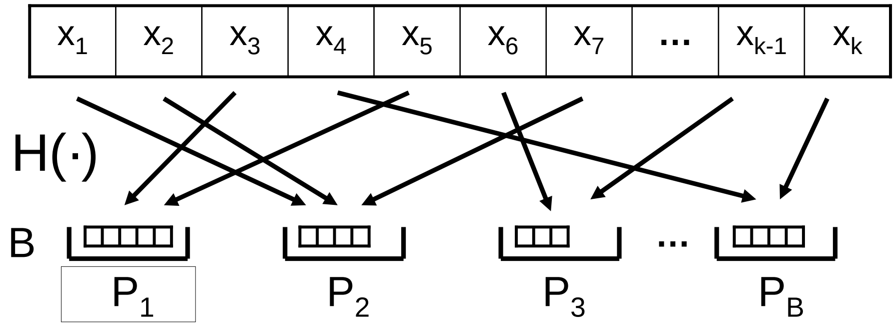
- Inputs typically from a "small" domain of D values. Represented by log D bits (…20)
- Use Horner's rule

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

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

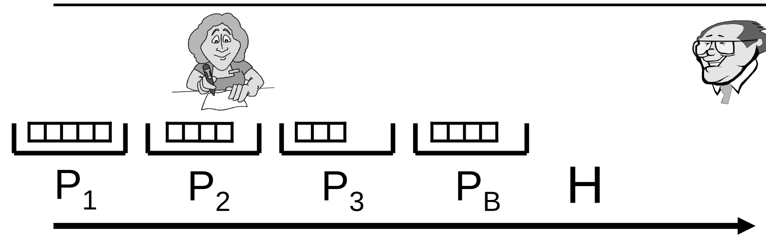
$\rightarrow$ Improve by factor of | modulus | / log D
e.g., 1024 / 20 ≈ 50

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

Improving Efficiency (2): Hashing



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

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

- 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 · M
- Server: k·M short exp's, k full exp's
  ($P_i(y)$) ($r \cdot P_i(y) + y$)
- How to make M small as possible?

  Balanced allocations [ABKU]:
  - H: Choose two bins, map to the 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
- A little on…
  - Extending protocol to malicious model
  - Approximation bounds
  - Multi-party security
  - Fuzzy matching

Malicious Adversaries

- Malicious clients
  - Without hashing: trivial. Parties use known a_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 computes $r \cdot P(y) + y'$
- Solution: Server uses RO to commit to seed, then uses resulting randomness to "prove" correctness of encryption

Is Approximation easier?

- Represent inputs sets as k-bit vectors

  0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1
- Approximate size of intersection (scalar product) with sublinear overhead? And securely?
- Lower bound:
  - Approximating $|X \cap Y|$ within $1 \pm \varepsilon$ factor requires $\Omega(k)$ communication
  - True even for randomized algorithms
  - Proof: Reduction from Razborov's lower bound for Disjointness
- We provide secure approximation protocol

Multi-party intersection

- N parties: (N-1) clients, 1 leader
- $\forall y$, leader prepares (N-1) shares that XOR to y
- Each client performs intersection protocol with leader, learns random share of y
- Clients XOR (N-1) decrypted values

  Recovers y iff $y \in |X_1 \cap X_2 \cap X_3 \cap \ldots \cap X_N|$
- Nice communication flow

Fuzzy matching

- Databases are not always accurate or full
  - Errors, omissions, inconsistent spellings, etc.
- How to report a match iff entries similar?
  - Match in t out of T “attributes”
- Adaption of earlier protocol, but requires T choose t overhead

Open problems

- More computationally-efficient protocol?
- Malicious parties
  - Protocol secure in standard model?
  - Secure, efficient set cardinality protocol?
- Fuzzy matching
  - Efficient protocol needed?
  - Security in malicious model?