Privacy-preserving Distributed Information Sharing and Secure Function Evaluation

Dawn Song

dawnsong@cs.berkeley.edu

Thanks for Benny Pinkas for some of the slides

Project

- Milestone report
  - Do not affect grade
  - Just for status update
  - Feedback tomorrow
- Poster session:
  - Dec 5, 4-6pm, Woz
  - Report due by 4pm, Dec 5
    - Electronic submission to summary gmail account
    - Hardcopy submission to office mailbox
- Final report:
  - Single column, 11pt font, reasonable margin
  - 10 pg limit excluding bibliography & appendix
  - Similar to a conference paper format
    - Abstract
    - Introduction: problem motivation & introduction
    - Approach
    - Design & implementation
    - Evaluation: if something didn't work as expected, explain why
    - Related work
    - Conclusion
- Final submission
  - Tarball of all software (including make files, test scripts & environment), paper (including source files), poster slides

Samples of Cryptographic Constructions for Privacy-preserving Applications

- The following few lectures
- Show what can be done & give a flavor of how it is done
- It's OK if you get a little lost
  - Just focus on the high-level picture
- Later this semester
  - Privacy issues in applications
  - Guest lecture at end of semester
    - Real-world case studies on privacy
      - Court cases fought by EFF

Privacy-Preserving Distributed Information Sharing

- **Allow multiple data holders to collaborate in order to compute important information while protecting the privacy of other information.**
  - **Security-related information**
  - **Users' private information**
    - **Health information**
  - **Enterprises' proprietary information**

Example Scenario: Medical Research

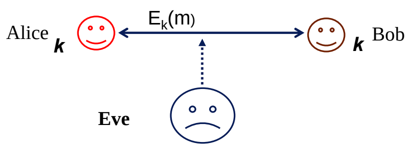
- **Medical research:**
  - **Trying to learn patterns in the data, in "aggregate" form.**
  - **Problem: how to enable learning aggregate data without revealing personal medical information?**
  - **Hiding names is not enough, since there are many ways to uniquely identify a person**
- **A single hospital/medical researcher might not have enough data**
- **How can different organizations share research data without revealing personal data?**

Issues and Tools

- **Best privacy can be achieved by not giving any data, but..**
- **Privacy tools: cryptography**
  - **Encryption:** data is hidden unless you have the decryption key. However, we also want to *use* the data.
  - **Secure function evaluation:** two or more parties with private inputs. Can compute any function they wish without revealing anything else.
  - **Strong theory. Starts to be relevant to real applications.**
- **Non-cryptographic tools**
  - **Query restriction:** prevent certain queries from being answered.
  - **Data/Input/output perturbation:** add errors to inputs – hide personal data while keeping aggregates accurate. (randomization, rounding, data swapping.)
  - **Can these be understood as well as we understand Crypto? Provide the same level of security as Crypto?**

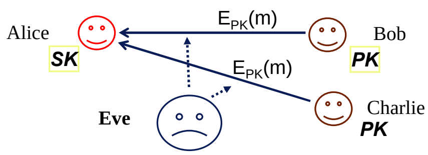
Crypto Primer: Symmetric Key Encryption

- **Alice wants to send a message $m \in \{0,1\}^n$ to Bob**
  - **Set-up phase is** secret
  - **Symmetric encryption: Alice and Bob share a secret key k**
- **They want to prevent** Eve **from *learning* anything about the message**



Crypto Primer: Public key encryption

- **Alice generates a private/public key pair *(SK,PK)***
- **Only Alice knows the secret key *SK***
- **Everyone (even Eve) knows the public key *PK,* and can encrypt messages to Alice**
- **Only Alice can decrypt (using *SK*)**



Problem: Secure Function Evaluation

- **A major topic of cryptographic research**
- **How to let n parties, $P_1,..,P_n$ compute a function $f(x_1,..,x_n)$**
  - **Where input x_i is known to party P_i**
  - **Parties learn the final input and nothing else**

The Millionaires Problem [Yao]

Whose value is greater?

Leak no other information!

Comparing Information without Leaking it

- **Output: Is x=y?**
- **The following solution is *insecure*:**
  - **Use a one-way hash function H()**
  - **Alice publishes H(x), Bob publishes H(y)**

Secure two-party computation – Security definition


Input: x y

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

As if…

Leak no other information

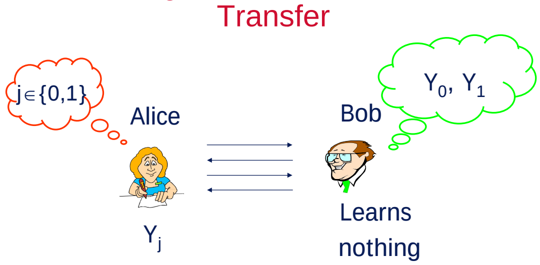
- A protocol is secure if it emulates the ideal solution
- Alice learns F(x,y), and therefore can compute everything that is implied by x, her prior knowledge of y, and F(x,y).
- Alice should not be able to compute anything else

- Simulation:
  - A protocol is considered secure if:
    For every adversary in the real world
    There exists a simulator in the ideal world, which outputs an indistinguishable ``transcript" , given access to the information that the adversary is allowed to learn

Secure Function Evaluation

- **Major Result [Yao]: "Any function that can be evaluated using polynomial resources can be securely evaluated using polynomial resources"**
  (under some cryptographic assumption)

SFE Building Block: 1-out-of 2 Oblivious Transfer



- 1-out-of-2 OT can be based on most public key systems
- There are implementations with two communication rounds

General Two party Computation

Two party protocol

- **Input:**
  - **Sender: Function F (some representation)**
    - **The sender's input Y is already embedded in F**
  - **Receiver: X $\in \{0,1\}^n$**
- **Output:**
  - **Receiver: F(x) and nothing else about F**
  - **Sender: nothing about x**

Representations of F

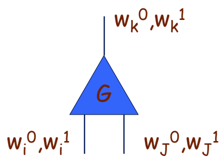
- **Boolean circuits [Yao,GMW,…]**
- **Algebraic circuits [BGW,…]**
- **Low deg polynomials [BFKR]**
- **Matrices product over a large field [FKN,IK]**
- **Randomizing polynomials [IK]**
- **Communication Complexity Protocol [NN]**

Secure two-party computation of general functions [Yao]

- **First, represent the function F as a Boolean circuit C**
  - **It's always possible**
  - **Sometimes it's easy (additions, comparisons)**
  - **Sometimes the result is inefficient (e.g. for indirect addressing, e.g. A[x])**
- **Then, "garble" the circuit**
- **Finally, evaluate the garbled circuit**

Garbling the circuit

- **Bob constructs the circuit, and then garbles it.**



W values will serve as cryptographic keys

$W_k^0 \equiv 0$ on wire k
$W_k^1 \equiv 1$ on wire k

(Alice will learn one string per wire, but not which bit it corresponds to.)

Gate tables

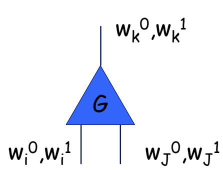
- **For every gate, every combination of input values is used as a key for encrypting the corresponding output**
- **Assume G=AND. Bob constructs a table:**
  - **Encryption of w_k^0 using keys w_i^0, w_J^0 (AND(0,0)=0)**
  - **Encryption of w_k^0 using keys w_i^0, w_J^1 (AND(0,1)=0)**
  - **Encryption of w_k^0 using keys w_i^1, w_J^0 (AND(1,0)=0)**
  - **Encryption of w_k^1 using keys w_i^1, w_J^1 (AND(1,1)=1)**
- **Result: given w_i^x, w_J^y, can compute $w_k^{G(x,y)}$**

Secure computation

- **Bob sends the table of gate G to Alice**
- **Given, e.g., w_i^0, w_J^1, Alice computes w_k^0 by decrypting the corresponding entry in the table, but she does not know the actual values of the wires.**

Encryption of w_k^0 using keys w_i^0, w_J^0
Encryption of w_k^0 using keys w_i^0, w_J^1
Encryption of w_k^1 using keys w_i^1, w_J^1
Encryption of w_k^0 using keys w_i^1, w_J^0

Permuted order



Secure computation

- **Bob sends to Alice**
  - **Tables encoding each circuit gate.**
  - **Garbled values (w's) of his input values.**
  - **Translation from garbled values of output wires to actual 0/1 values.**
- **If Alice gets garbled values (w's) of her input values, she can compute the output of the circuit, and nothing else.**

Alice's input

- **For every wire i of Alice's input:**
  - **The parties run an OT protocol**
  - **Alice's input is her input bit (s).**
  - **Bob's input is w_i^0, w_i^1**
  - **Alice learns w_i^s**
- **The OTs for all input wires can be run in parallel.**
- **Afterwards Alice can compute the circuit by herself.**

Secure computation – the big picture

- **Represent the function as a circuit C**
- **Bob sends to Alice 4|C| encryptions (e.g. 64|C| Bytes), 4 encryptions for every gate.**
- **Alice performs an OT for every input bit. (Can do, e.g. 100-1000 OTs per sec.)**
- **~One round of communication.**
- **Efficient for medium size circuits!**
- **Fairplay [MNPS]**
  - **a secure two-party computation system**
  - **implementing Yao's "garbled circuit" protocol**

Privacy-preserving Set Operations

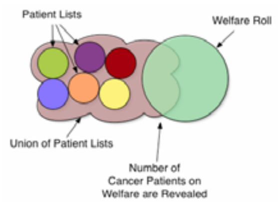
- Yao's Garbled Circuit is a generic construction
  - May be too expensive for complex functions
- For specific functions, we could design more efficient algorithms
  - E.g., privacy-preserving set operations [Kissner-Song]
- Data can often be represented as multisets
- Important operations often can be represented as set operations
- Thus, need methods for privacy-preserving set operations

Motivation (I): Do-Not-Fly List

- Do-not-fly list
  - Airlines must determine which passengers cannot fly
  - Government and airlines cannot disclose their lists

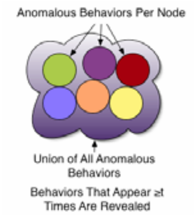
Motivation (II): Public Welfare Survey

- How many welfare recipients are being treated for cancer?
  - Cancer patients and welfare rolls are confidential
  - Compute private union and intersection operations



Motivation (III): Distributed Network Monitoring

- Each node keeps a list of anomalous events
- Identify anomalous events appearing at t or more nodes
- Compute private union and element reduction operations
- d-th Element reduction $Rd_d(S)$: If an element *a* appears *b* times in S, *a* appears *b-d* times in the d-th reduction of S



Private Set Operations

- Traditional approach: trusted third party (TTP)
- Private set operations:
  - No trusted third party
  - Provide the same privacy/security as in TTP case
- Results:
  - Efficient, composable, privacy-preserving operations on multisets: intersection, union, element reduction
  - $\gamma ::= s \mid Rd_d(\gamma) \mid \gamma \cap \gamma \mid s \cup \gamma \mid \gamma \cup s$
  - Can also compute multiset cardinality, subset relations
- Solution:
  - Polynomials as intermediate representation of sets
  - Use mathematical properties of polynomials for set operations
  - Homomorphic encryption to compute on encrypted polynomials

Computing Polynomial Representations of Set Operations

- Use polynomial f over Ring R to represent multiset S: roots are the set elements, f= $\prod_{a \in S}(x-a)$
- Given polynomials f and g representing multiset S and T, compute the polynomial representing:
  a) $S \cup T$;
  b) $S \cap T$;
  c) $Rd_d(S)$;
  with properties:
  1) Correctness: well-formed roots give correct result.
  2) Privacy: reveal no additional information about S & T.

Multiset Union

Multisets	Polynomial Rep.
S	f
T	g

$\boxed{S \cup T \longleftrightarrow f * g}$

- **Satisfies:**
  a) Correctness: polynomial multiplication preserves roots
  b) Privacy: trivial

Multiset Intersection: Strawman Approach

Multisets	Polynomial Rep.
S	f
T	g

$\boxed{S \cap T \longleftrightarrow f+g}$

- **Polynomial addition preserves shared roots**
- **However, reveals extra information about S and T**

Multiset Intersection

Multisets	Polynomial Rep.
S, T	f,g

$\boxed{S \cap T \longleftrightarrow f * r + g * s}$

- ***r,s:* uniformly distributed polynomials from $R^{Deg(f)}[x]$ (each coefficient chosen u.a.r. from R)**
- Lemma: *If **gcd(u,v)=1**, **Deg(u)=Deg(v)=p, r,s**$\leftarrow R^p[x]$, leading coefficients of u & v have multiplicative inverse, then **$u*r+v*s$ is uniformly distributed over $R^h[x]$, h=2p.***
- **Correctness & privacy from lemma**

Element Reduction

Multisets	Polynomial Rep.
S	f

$$\boxed{Rd_d(S) \longleftrightarrow \sum_{0\leq i\leq d} f^{(j)} * r_j * e_j}$$

- r_j: uniformly distributed polynomials from $R^{Deg(f)}[x]$ (each coefficient chosen u.a.r. from R)
- e_j: polynomial of degree j with certain properties
- Proof of correctness and privacy more complicated

Homomorphic Encryption (I)

- **Encrypt coefficients of polynomial using a *threshold additively homomorphic* cryptosystem**
- **We can perform the calculations needed for our techniques with encrypted polynomials (examples use Paillier cryptosystem)**
  - **Addition**

$$\begin{aligned} h &= f + g \\ h_i &= f_i + g_i \\ E(h_i) &= E(f_i) * E(g_i) \end{aligned}$$

Homomorphic Encryption (II)

- **Formal derivative**

$$\begin{aligned} h &= f' \\ h_i &= (i+1) f_{i+1} \\ E(h_i) &= E(f_i)^{i+1} \end{aligned}$$

- **Multiplication**

$$\begin{aligned} h &= f * g \\ h_i &= \sum_{j=0}^{k} f_j * g_{i-j} \\ E(h_i) &= \prod_{j=0}^{k} E(f_j)^{g_{i-j}} \end{aligned}$$

Multiset Intersection

- Let each player *i* (*1≤i≤n*) hold an input multiset S_i
- Each player calculates the polynomial f_i representing S_i and broadcasts $E(f_i)$
- For each *i*, each player *j* *(1≤j≤n)* chooses uniformly distributed polynomial $r_{i,j}$, and broadcasts $E(f_i * r_{i,j})$
- All players calculate and decrypt

$$E\left(\sum_{i=1}^{n} f_i * \left(\sum_{j=1}^{n} r_{i,j}\right)\right) = E(p)$$

- Players determine the intersection multiset:
  if $(x-a)^b \mid p$
  then *a* appears b times in the result

SFE: Other Side of the Story

- Provable security
  - Simulation to the ideal world
  - Learn nothing more than the final results
- However, the function needs to be well chosen first
  - Computing the median may leak sufficient info if the set is small

Summary

- Privacy-preserving distributed information sharing
- Secure function evaluation
  - Security definition
  - Possibility results & generic construction
  - More specialized construction
    - Private set operations
- Next class
  - Computation on encrypted data
  - Private operations on Untrusted Servers/Storage