Secure Multiparty Computation

Li Xiong

CS573 Data Privacy and Security

Outline

- Secure multiparty computation
  - Problem and security definitions
  - Basic cryptographic tools and general constructions

Yao's Millionnare Problem

- Two millionaires, Alice and Bob, who are interested in knowing which of them is richer without revealing their actual wealth.
- This problem is analogous to a more general problem where there are two numbers a and b and the goal is to solve the inequality without revealing the actual values of a and b.

Secure Multiparty Computation

- A set of parties with private inputs
- Parties wish to jointly compute a function of their inputs so that certain security properties (like privacy and correctness) are preserved
- Properties must be ensured even if some of the parties maliciously attack the protocol
- Examples
  - Secure elections
  - Auctions
  - Privacy preserving data mining
  - …

Heuristic Approach to Security

1. Build a protocol
2. Try to break the protocol
3. Fix the break
4. Return to (2)

Another Heuristic Tactic

- Design a protocol
- Provide a list of attacks that (provably) cannot be carried out on the protocol
- Reason that the list is complete

A Rigorous Approach

- Provide an exact problem definition
  - Adversarial power
  - Network model
  - Meaning of security
- Prove that the protocol is secure

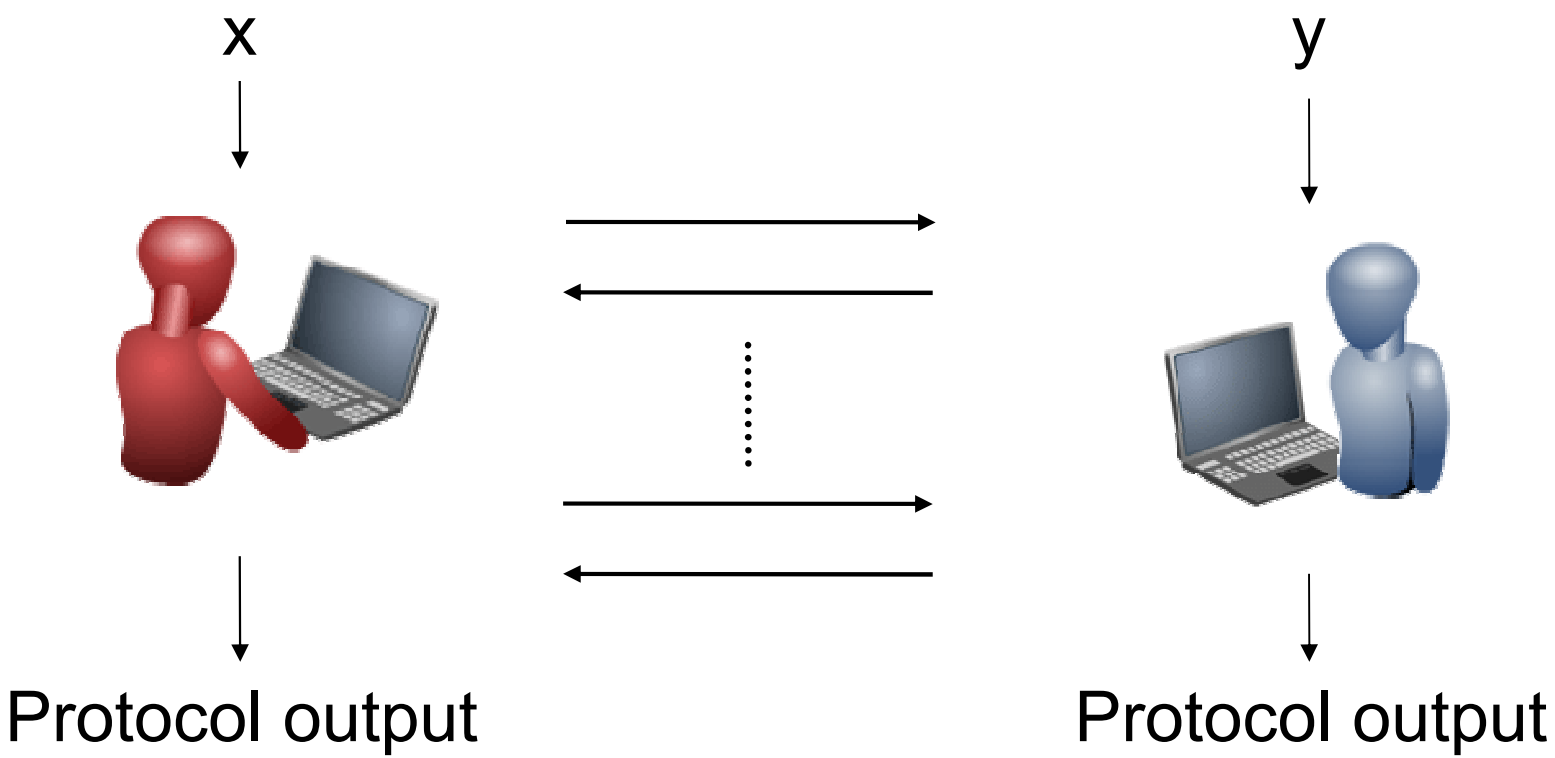
Secure Multiparty Computation

- A set of parties with private inputs wish to compute some joint function of their inputs.
- Parties wish to preserve some security properties. e.g., privacy and correctness.
  - Example: secure election protocol
- Security must be preserved in the face of adversarial behavior by some of the participants, or by an external party.

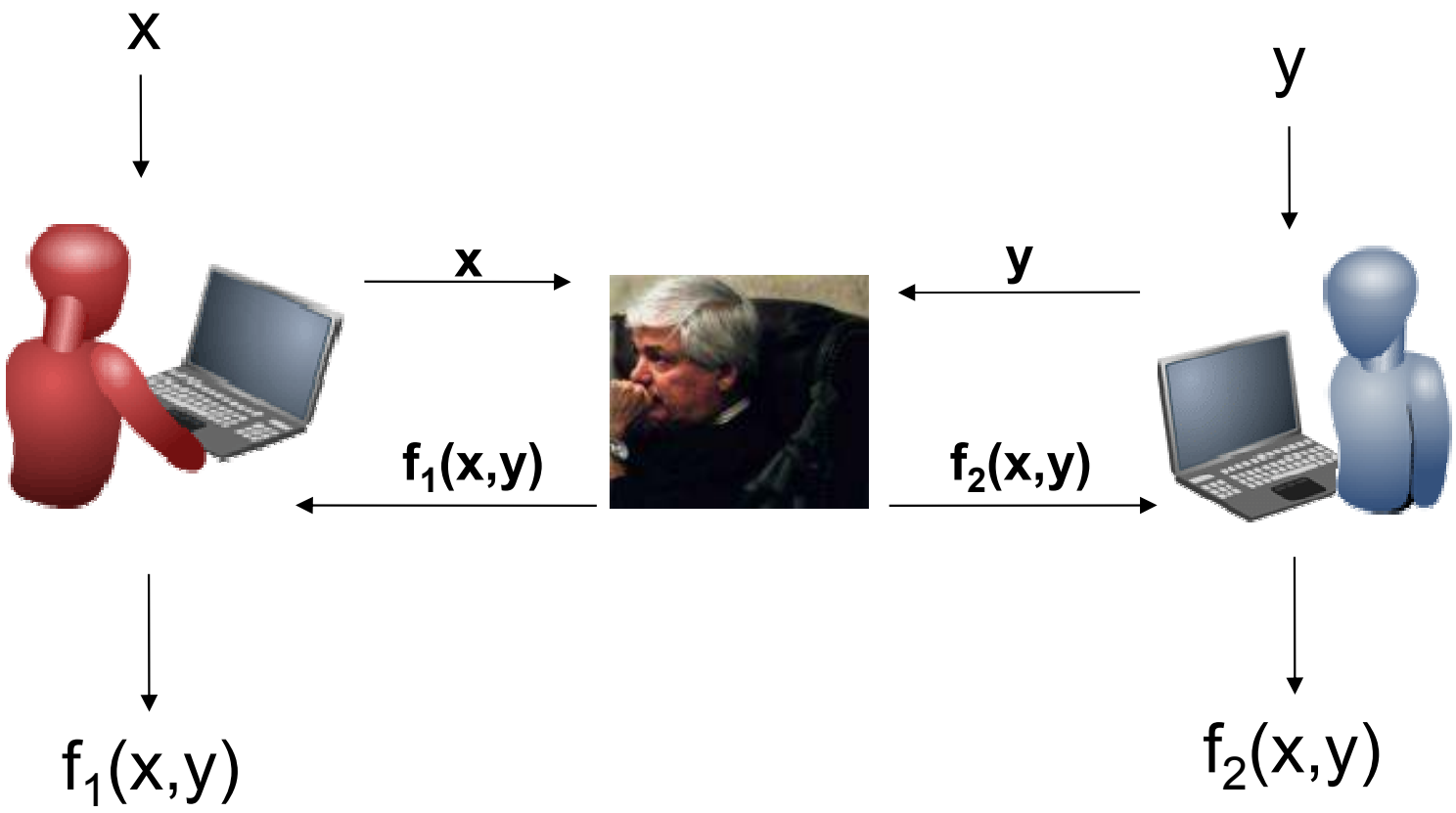
Defining Security

- The real/ideal model paradigm for defining security [GMW,GL,Be,MR,Ca]:
  - Ideal model: parties send inputs to a trusted party, who computes the function for them
  - Real model: parties run a real protocol with no trusted help
- A protocol is secure if any attack on a real protocol can be carried out in the ideal model

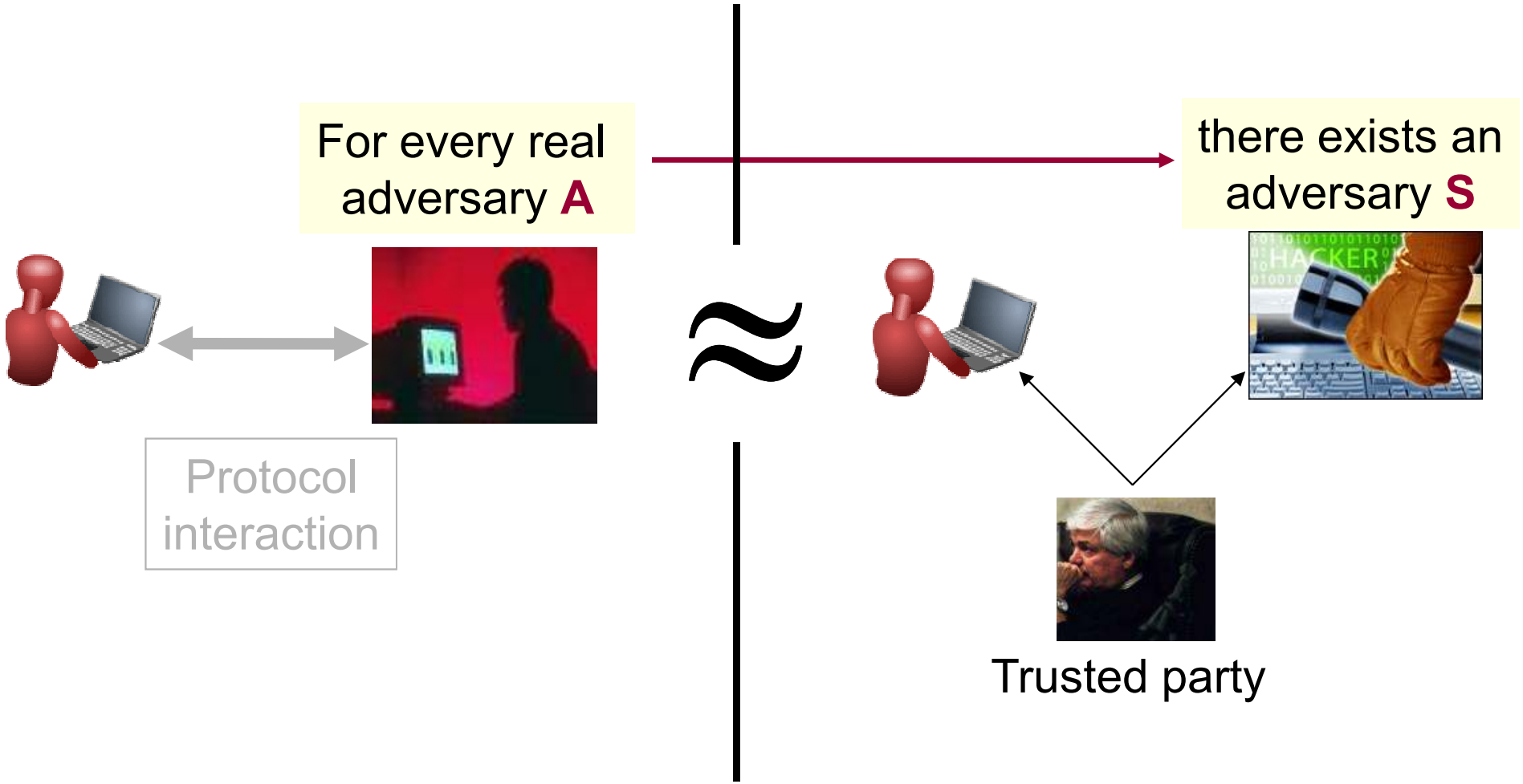
The Real Model



The Ideal Model



The Security Definition:



Properties of the Definition

- Privacy:
  - The ideal-model adversary cannot learn more about the honest party's input than what is revealed by the function output
  - Thus, the same is true of the real-model adversary
- Correctness:
  - In the ideal model, the function is always computed correctly
  - Thus, the same is true in the real-model
- Others:
  - For example, fairness, independence of inputs

Adversary Model

- **Computational power:**
  - polynomial-time versus all-powerful
- **Adversarial behaviour:**
  - Semi-honest: follows protocol instructions
  - Malicious: arbitrary actions
- **Corruption behaviour**
  - Static: set of corrupted parties fixed at onset
  - Adaptive: can choose to corrupt parties at any time during computation
- **Number of corruptions**
  - Honest majority versus unlimited corruptions

Security proof tools

- Real/ideal model: the real model can be simulated in the ideal model
  - Key idea – Show that whatever can be computed by a party participating in the protocol can be computed based on its input and output only
  - $\exists$ polynomial time S such that $\{S(x,f(x,y))\} \equiv \{View(x,y)\}$

Security proof tools

- Composition theorem
  - if a protocol is secure in the hybrid model *where the protocol uses* a trusted party that computes the (sub) functionalities, and we replace the calls to the trusted party by calls to secure protocols, *then the resulting protocol is secure*
  - Prove that component protocols are secure, then prove that the combined protocol is secure

Outline

- Secure multiparty computation
  - Defining security
  - Basic cryptographic tools and general constructions

Public-key encryption

- Let (G,E,D) be a public-key encryption scheme
  - G is a key-generation algorithm (pk,sk) ← G
    - Pk: public key
    - Sk: secret key
  - Terms
    - Plaintext: the original text, notated as m
    - Ciphertext: the encrypted text, notated as c
  - Encryption: $c = E_{pk}(m)$
  - Decryption: $m = D_{sk}(c)$
  - Concept of **one-way function**: knowing c, pk, and the function E_{pk}, it is still computationally intractable to find m.

*Different implementations available, e.g. RSA

Construction paradigms

- Passively-secure computation for two-parties
  - Use oblivious transfer to securely select a value
- Passively-secure computation with shares
  - Use secret sharing scheme such that data can be reconstructed from some shares
- From passively-secure protocols to actively-secure protocols
  - Use zero-knowledge proofs to force parties to behave in a way consistent with the passively-secure protocol

1-out-of-2 Oblivious Transfer (OT)

1-out-of-2 Oblivious Transfer (OT)

- Inputs
  - Sender has two messages m_0 and m_1
  - Receiver has a single bit $\sigma \in \{0,1\}$
- Outputs
  - Sender receives nothing
  - Receiver obtain m_σ and learns nothing of $m_{1-\sigma}$

Semi-Honest OT

- Let (G,E,D) be a public-key encryption scheme
  - G is a key-generation algorithm $(pk,sk) \leftarrow G$
  - Encryption: $c = E_{pk}(m)$
  - Decryption: $m = D_{sk}(c)$
- Assume that a public-key can be sampled without knowledge of its secret key:
  - Oblivious key generation: $pk \leftarrow OG$
  - El-Gamal encryption has this property

Semi-Honest OT

Protocol for Oblivious Transfer

- Receiver (with input σ):
  - Receiver chooses one key-pair (pk,sk) and one public-key pk' (oblivious of secret-key).
  - Receiver sets $pk_\sigma = pk$, $pk_{1-\sigma} = pk'$
  - Note: receiver can decrypt for pk_σ but not for $pk_{1-\sigma}$
  - Receiver sends pk_0, pk_1 to sender
- Sender (with input m_0, m_1):
  - Sends receiver $c_0 = E_{pk0}(m_0)$, $c_1 = E_{pk1}(m_1)$
- Receiver:
  - Decrypts c_σ using sk and obtains m_σ.

Security Proof

- Intuition:
  - Sender's view consists only of two public keys pk_0 and pk_1. Therefore, it doesn't learn anything about that value of σ.
  - The receiver only knows one secret-key and so can only learn one message
- Note: this assumes semi-honest behavior. A malicious receiver can choose two keys together with their secret keys.

Generalization

- Can define 1-out-of-k oblivious transfer
- Protocol remains the same:
  - Choose k-1 public keys for which the secret key is unknown
  - Choose 1 public-key and secret-key pair

Secrete Sharing Scheme

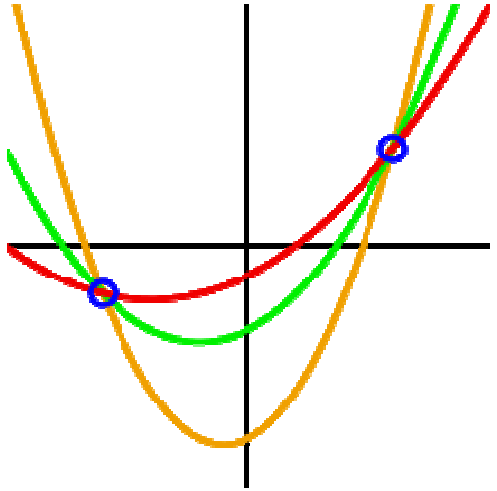
- Distributing a secret amongst n participants, each of whom is allocated a share of the secret
- The secret can be reconstructed only when a sufficient number (t) of shares are combined together
  - (t, n)-threshold scheme
- Secrete shares, random shares
  - individual shares are of no use on their own

Trivial Secrete Sharing Scheme

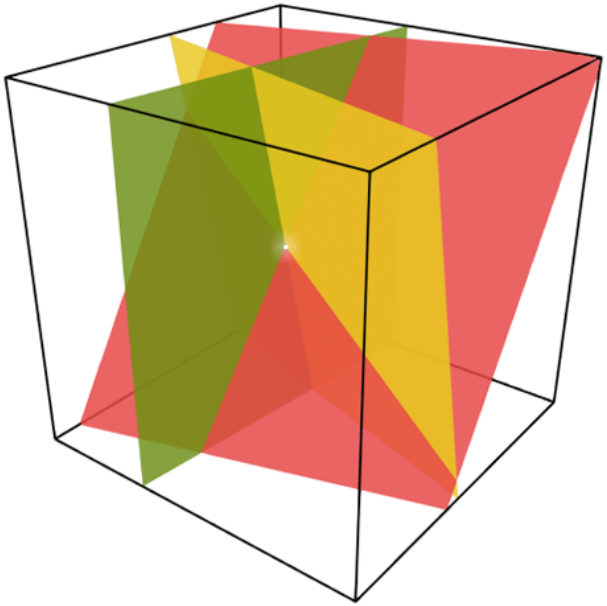
- Encode the secret as an integer s.
- Give to each player i (except one) a random integer r_i. Give to the last player the number $(s - r_1 - r_2 - ... - r_{n-1})$

Secrete sharing scheme

- Shamir's scheme
  - It takes t points to define a polynomial of degree t-1
  - Create a t-1 degree polynomial with secret as the first coefficient and the remaining coefficients picked at random. Find *n* points on the curve and give one to each of the players. Tt At least *t* points are required to fit the polynomial.



- Blakey's scheme
  - any n nonparallel n-dimensional hyperplanes intersect at a specific point
  - Secrete as the coordinate of the hyperplanes
  - Less space efficient



General GMW Construction

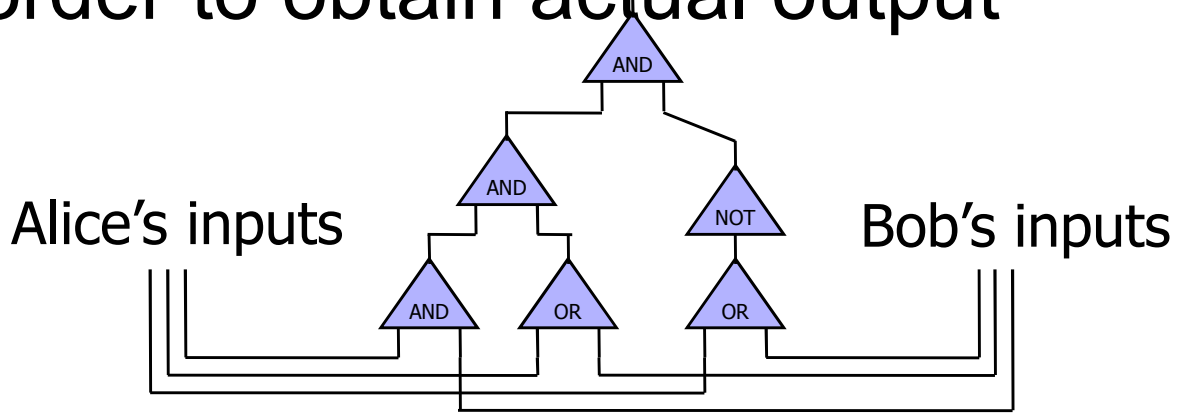
- For simplicity – consider two-party case
- Let f be the function that the parties wish to compute
- Represent f as an arithmetic circuit with addition and multiplication gates
- Aim – compute gate-by-gate, revealing only random shares each time

Random Shares Paradigm

- Let a be some value:
  - Party 1 holds a random value a_1
  - Party 2 holds $a+a_1$
  - Note that without knowing a_1, $a+a_1$ is just a random value revealing nothing of a.
  - We say that the parties hold random shares of a.
- The computation will be such that all intermediate values are random shares (and so they reveal nothing).

Circuit Computation

- Stage 1: each party randomly shares its input with the other party
- Stage 2: compute gates of circuit as follows
  - Given random shares to the input wires, compute random shares of the output wires
- Stage 3: combine shares of the output wires in order to obtain actual output



Addition Gates

- Input wires to gate have values a and b:
  - Party 1 has shares a_1 and b_1
  - Party 2 has shares a_2 and b_2
  - Note: $a_1+a_2=a$ and $b_1+b_2=b$
- To compute random shares of output $c=a+b$
  - Party 1 locally computes $c_1=a_1+b_1$
  - Party 2 locally computes $c_2=a_2+b_2$
  - Note: $c_1+c_2=a_1+a_2+b_1+b_2=a+b=c$

Multiplication Gates

- Input wires to gate have values a and b:
  - Party 1 has shares a_1 and b_1
  - Party 2 has shares a_2 and b_2
  - Wish to compute $c = ab = (a_1+a_2)(b_1+b_2)$
- Party 1 knows its concrete share values a_1 and b_1.
- Party 2's shares a_2 and b_2 are unknown to Party 1, but there are only 4 possibilities (00,01,10,11)

Multiplication (cont)

- Party 1 prepares a table as follows (Let r be a random bit chosen by Party 1):
  - Row 1 contains the value $a \cdot b + r$ when $a_2=0, b_2=0$
  - Row 2 contains the value $a \cdot b + r$ when $a_2=0, b_2=1$
  - Row 3 contains the value $a \cdot b + r$ when $a_2=1, b_2=0$
  - Row 4 contains the value $a \cdot b + r$ when $a_2=1, b_2=1$

Concrete Example

- Assume: $a_1=0$, $b_1=1$
- Assume: r=1

Row	Party 2's shares	Output value
1	$a_2=0,b_2=0$	$(0+0)\cdot(1+0)+1=1$
2	$a_2=0,b_2=1$	$(0+0)\cdot(1+1)+1=1$
3	$a_2=1,b_2=0$	$(0+1)\cdot(1+0)+1=0$
4	$a_2=1,b_2=1$	$(0+1)\cdot(1+1)+1=1$

The Gate Protocol

- The parties run a 1-out-of-4 oblivious transfer protocol
- Party 1 plays the sender: message i is row i of the table.
- Party 2 plays the receiver: it inputs **1** if $a_2=0$ and $b_2=0$, **2** if $a_2=0$ and $b_2=1$, and so on…
- Output:
  - Party 2 receives $c_2=c+r$ – this is its output
  - Party 1 outputs $c_1=r$
  - Note: c_1 and c_2 are random shares of c, as required

Security

- Reduction to the oblivious transfer protocol
- Assuming security of the OT protocol, parties only see random values until the end. Therefore, simulation is straightforward.

- Note: correctness relies heavily on semi-honest behavior (otherwise can modify shares).

Outline

- Secure multiparty computation
  - Defining security
  - Basic cryptographic tools and general constructions
- Coming up
  - Applications in privacy preserving distributed data mining
  - Random response protocols

A real-world problem and some simple solutions

- Bob comes to Ron (a manager), with a complaint about a sensitive matter, asking Ron to keep his identity confidential
- A few months later, Moshe (another manager) tells Ron that someone has complained to him, also with a confidentiality request, about the same matter
- Ron and Moshe would like to determine whether the same person has complained to each of them without giving information to each other about their identities

Comparing information without leaking it. Fagin et al, 1996

Solutions

- Solution 1: Trusted third party
- Solution 7: message for Moshe
- Solution 8: Airline reservation
- Solution 9: Password

References

- Secure Multiparty Computation for Privacy-Preserving Data Mining, Pinkas, 2009
- Chapter 7: General Cryptographic Protocols (7.1 Overview), The Foundations of Cryptography, Volume 2, Oded Goldreich

  http://www.wisdom.weizmann.ac.il/~Eoded/foc-vol2.html
- Comparing information without leaking it. Fagin et al, 1996

Slides credits

- Tutorial on secure multi-party computation, Lindell

  www.cs.biu.ac.il/~lindell/research-statements/tutorial-secure-computation.ppt

- Introduction to secure multi-party computation, Vitaly Shmatikov, UT Austin

  www.cs.utexas.edu/~shmat/courses/cs380s_fall08/16smc.ppt