Cryptography CS 555

Lecture 26: Introduction to Secure Multi-Party Computation

Department of Computer Sciences
Purdue University

Lecture Outline

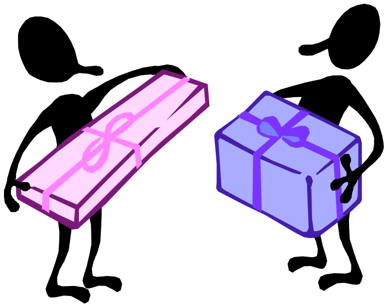
- Fair exchange protocols:
  - Definition
  - Classification
  - Applications
  - Certified email and contract signing examples
- Secure multi-party computation
  - Examples
  - 1-2 Oblivious Transfer
  - Two-Party Secure Computation

Fair Exchange Protocols

Definition

Given two parties A and B owing a and b. Then, a fair exchange protocol is an electronic protocol guaranteeing that **either both parties get what they want or none does** (either A gets b and B gets a, or none does).

At any point during the exchange, **no party gains any advantage** over the other one.



Examples of Fair Exchange Protocols

- **Certified email:** Assume A wants to send message M to B. A requires as proof of delivery B's signature on the message M.
- **Contract signing:** Given two parties A and B that negotiated the contract C, then A will sent its signature on C and B will sent its signature on C.
- **Fair purchase:** Items exchanged are merchandise and payment.

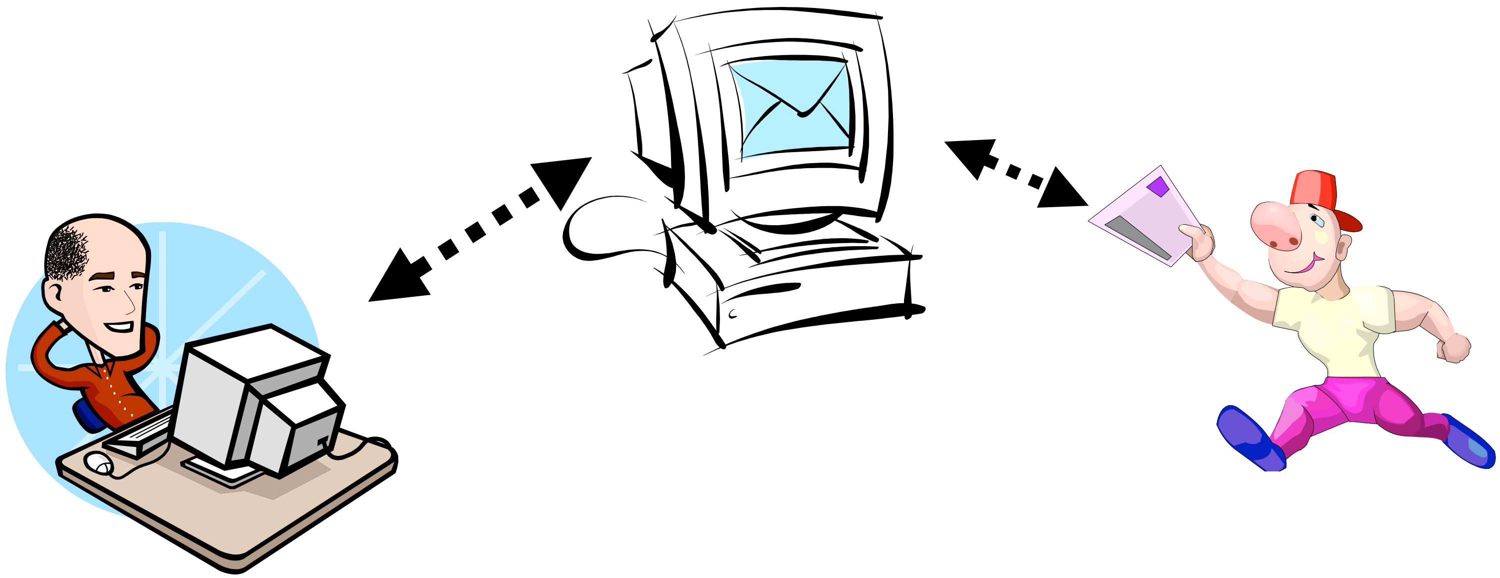
Simultaneity is very difficult to achieve in digital world!

Involvement of a Trusted Third Party (TTP)

- Protocols without TTP: the protocol does not rely on a TTP to facilitate the exchange. Works fine if parties are honest, but problematic if parties are cheating, requires parties to have the same computational power.
- Protocols using TTP: rely on a TTP
  - Visible TTP: the TTP has to be on line, it is involved in every transaction.
  - Invisible TTP: the TTP does not need to be permanently on-line; TTP involved only in case one of the parties cheats.

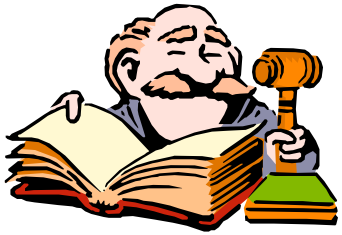
Visible TTP Approach

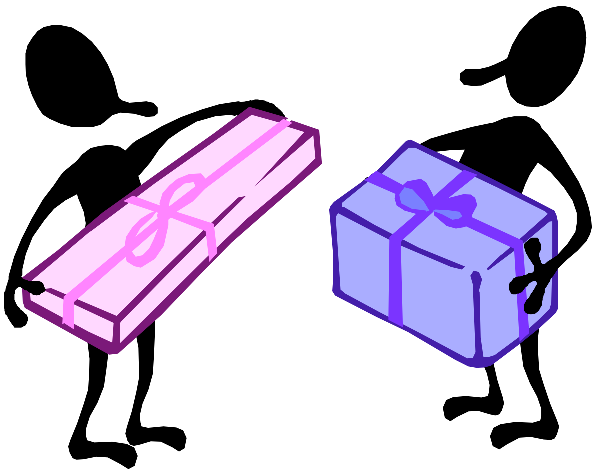
- Costly: TTP must be available any time; cost to maintain that.
- Congestion: all transaction going through one 'site'.
- Liability: the crash of the TTP can trigger losses.
- Single point of trust and failure.



Optimistic Protocols

- **TTP is invisible**: TTP will not participate in transactions in which parties are honest, but will solve disputes.
- **TTP acts on behalf of the cheated party**: if for example B cheats, TTP sends the information to A, as if B sent it.
- **Storage/privacy**: TTP does not need to store any secret of either party at any time.





Fair Exchange Certified Email without Privacy

A is not concerned about the privacy of the message sent to B.

1: A sends to B $Z = E^R_{TTP}(A, B, M)$
2: B sends to A $S = Sig_B(Z)$
3: After receiving S, A sends M and R to B
4: B verifies that $Z = E^R_{TTP}(A, B, M)$; if yes, then STOP. Otherwise sends Z to TTP.
5: If $Sig_B(Z)$ verifies, TTP decrypts Z; If the result is A, B, M, with randomness R, TTP sends message M to B and $Sig_B(Z)$ to A.

Fair Certified Email with Privacy

A is concerned about the privacy of the message to B.

1: A sends B $Z = E_{TTP}(A, B, E_B(M))$
2: B sends A $S = Sig_B(Z)$
3: After receiving S, if S verifies, A sends $E_B(M)$
4: If B receives X s.t. E(A,B,X) = Z, then decrypts X and gets M, then STOP. Otherwise B sends Z and $Sig_B(Z)$ to TTP.
5: If $Sig_B(Z)$ verifies, TTP decrypts Z; If the result is A, B, X, TTP sends X to B and $Sig_B(Z)$ to A.

Optimistic Fair Contract Signing

A and B aggreed on contract C and want to exchange $Sig_B(C, M)$ and $Sig_A(C,M)$, where M is a random message

1: A chooses a random message M, computes $Z = E_{TTP}(A,B,M)$ and sends to B: $Sig_A(C,Z)$

2: B sends to A, $Sig_B(C,Z)$ and $Sig_B(Z)$

3: If both $Sig_B(C,Z)$ and $Sig_B(Z)$ verify, A sends M to B

4: If B receives a string M s.t. $E_{TTP}(A, B, M) = Z$, then STOP. Otherwise sends Z, $Sig_B(C,Z)$ and $Sig_B(Z)$ to TTP.

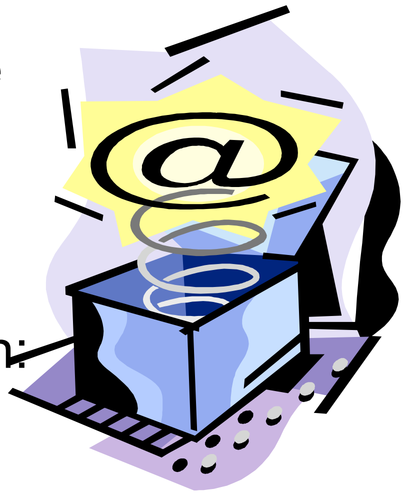
5: If $Sig_B(C,Z)$, $Sig_B(Z)$ verify, TTP decrypts Z; If the result is A, B, M, TTP sends message M to B and $Sig_B(C,Z)$ and $Sig_B(Z)$ to A.

Certified Email Based on Verifiable Encryption

- Passive role
  - No need for keeping state info
  - No dispute resolutions
  - No need for continuous connection
- Sender initiates the exchange

Verifiable Encryption of Digital Signatures

- It can be seen as a transparent box, Alice can see that the box contains Bob's digital signature, but she can not "extract" it from the box.
- TTP 'knows' a secret that allows him to open the transparent box.
- Example of application to verifiable encryption: certified email.

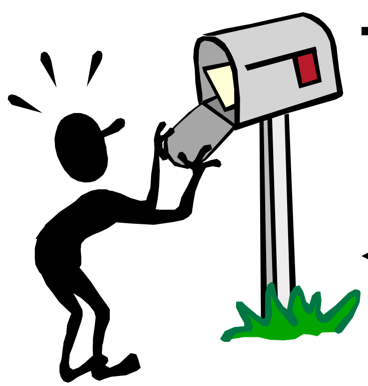


A Basic Protocol

$$Sign_A(H(m)), H(m) \longrightarrow$$

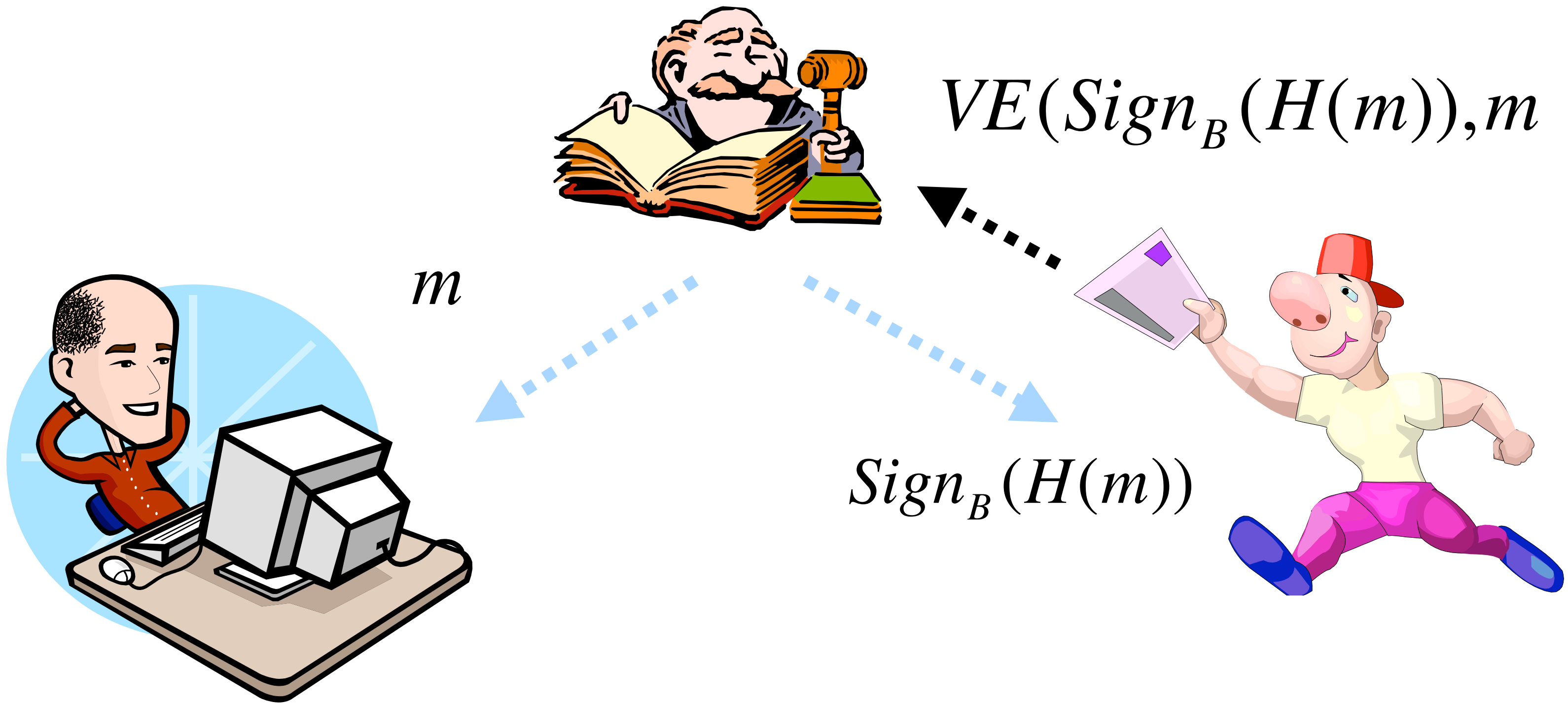
$$\longleftarrow VE(Sign_B(H(m)))$$

$$m \longrightarrow$$



$$\longleftarrow Sign_B(H(m))$$

Dispute Resolution



Lecture Outline

- Fair exchange protocols:
  - Definition
  - Classification
  - Applications
  - Certified email and contract signing examples
- Secure multi-party computation
  - Examples
  - 1-2 Oblivious Transfer
  - Two-Party Secure Computation

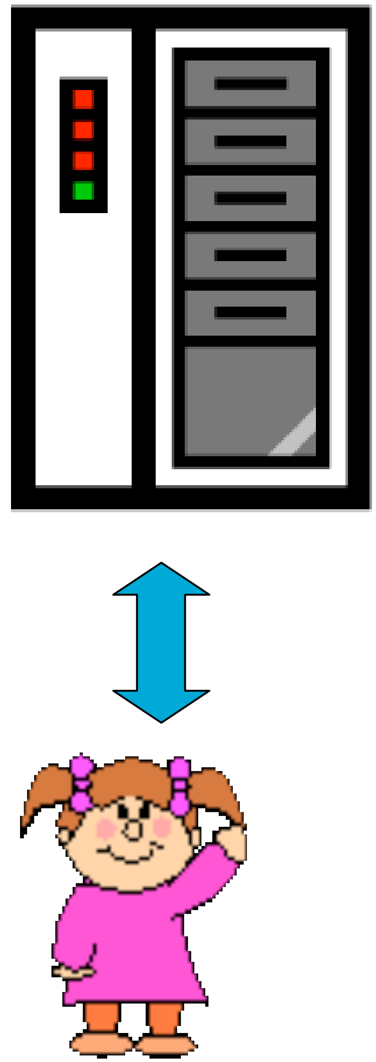
Example 1: The Millionare Problem (Yao 1982)

- Alice and Bob are two millionaires
- They want to know which one is richer
- None of them wants to reveal the exact amount of their wealth.

?

Example 2: Private Information Retrieval

- A client queries a database
- The server(s) should not learn the query
- The client should not learn more than the query he asked
- For example several servers hold copies of the data, client privately retrieve parts of the n bits of data stored in the database. Each queries give each individual database server no partial information



Example 3: Oblivious Transfer

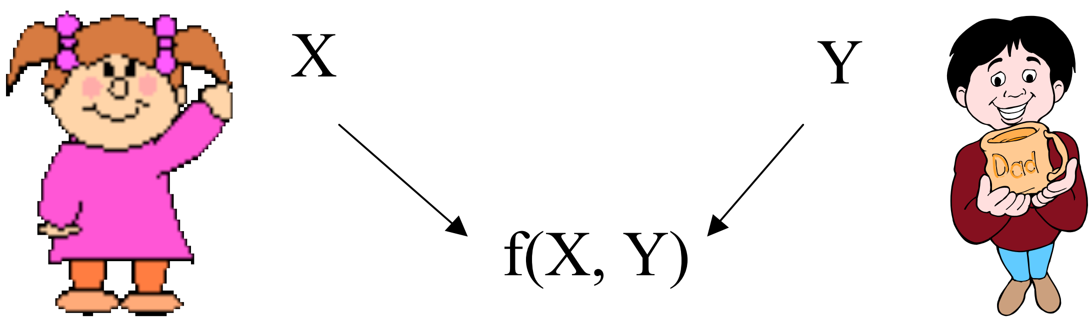
- Alice has two bits and she wants to send one of them to Bob
- Requirements are:
  - Alice does not know what bit did Bob learnt (received)
  - Bob does not learn anything else about the bits that Alice has, only that one bit

Example 4: Data Analysis on Sets while Preserving Privacy

- Companies need to analyze their data looking for patterns without compromising privacy
- Questions to answer:
  - What is the median of $A \cup B$
  - What is the intersection $A \cap B$
  - What matches an item x
  - What is the k-th ranked element

Secure Multi-Party Computation

- A set of parties with private inputs wish to compute a joint function ***f*** of their inputs.
- Parties wish to preserve security properties such as privacy and correctness.
- Security must be preserved in the face of adversarial behavior by some of the participants, or by an external party.



Adversary Models

- **Semi-honest adversary:** follows the protocol, but he is curious, he saves all intermediate computations; can later be used to compromise security.
- **Malicious adversary:** does not follow the protocol properly - could refuse to participate, replace inputs or partial computations, abort the protocol before it finishes.
- Assuming semi-honest model in general is enough.

1-out-of-2 Oblivious Transfer (OT) in Semi-Honest Model

- Inputs
  - Sender has two messages m_0 and m_1
  - Receiver has a single bit $\sigma \in \{0,1\}$
- Outputs
  - Sender receives nothing
  - Sender does not know what message the receiver learnt
  - Receiver obtain m_σ and learns nothing of $m_{1-\sigma}$

Details

- 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 selected without knowledge of its secret key:
  - Oblivious key generation: $pk \leftarrow OG$

Protocol for Oblivious Transfer

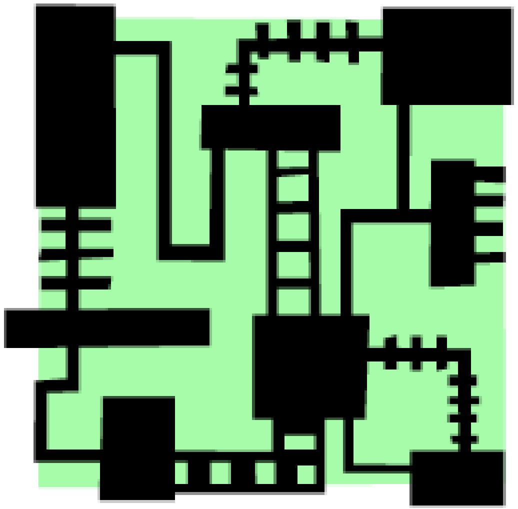
- Receiver (with input □):
  - Receiver chooses one key-pair (pk,sk) and one public-key pk' (obliviously of secret-key, does not know corresponding secret key).
  - Receiver sets $pk_{\square}$ = pk, $pk_{1-\square}$ = pk'
  - Note: receiver can decrypt for $pk_{\square}$ but not for $pk_{1-\square}$
  - 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 $c_{\square}$ using sk and obtains $m_{\square}$.

Can you think about how to generalize this protocol to k parties?

Can you think about attacks if the model is not semi-honest but malicious?

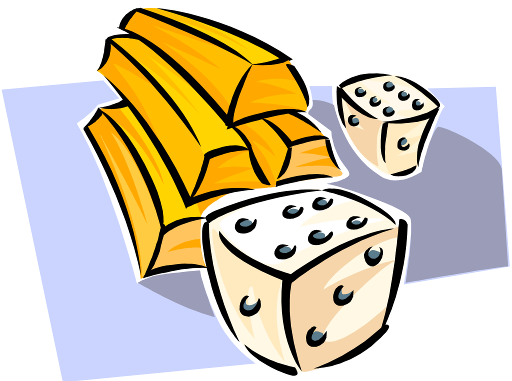
Secure Two-Party Computation

- Let f be the function that the parties wish to compute
- Represent f as an arithmetic circuit with addition and multiplication gates.
- The idea is to compute gate-by-gate, revealing only random shares each time



Definition of Random Shares

- a is a random shared; Then
  - Party 1 holds a random value a_1
  - Party 2 holds $a+a_1$
  - $a+a_1$ is just a random value revealing nothing of a.
  - We say that the parties hold random shares of a.
- The idea is to have computation such that all intermediate values are random shares (and so they reveal nothing).



Circuit Computation

- 1) Each party randomly shares its input with the other party
- 2) Compute gates of circuit: given random shares to the input wires, compute random shares of the output wires
- 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
  - $a_1+a_2=a$ and $b_1+b_2=b$
  - Wants to compute a+b
- To compute random shares of output $c=a+b$
  - Party 1 locally computes $c_1=a_1+b_1$
  - Party 2 locally computes $c_2=a_2+b_2$
  - $c_1+c_2=a_1+a_2+b_1+b_2=a+b=c$

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
  - Wants to compute $c = ab = (a_1+a_2)(b_1+b_2) = a_1b_1 + a_1b_2 + a_2b_1 + a_2b_2 = c_1 + c_2$
- Output:
  - Party 1: c1
  - Party 2: c2

Multiplication (cont)

- **Goal is that Party 1 computes c1 knowing its concrete share values and without knowing bits of Party 2**
- **Party 2's values are unknown to Party 1, but there are only 4 possibilities (depending on correspondence to 00,01,10,11)**
- Main idea: use oblivious transfer to transfer one of the 4 possible values, Party 1 is the sender and Party 2 is the receiver
- One party prepares the four possible outputs of f based on its own bits (which are known and fixed) and the (unknown) input bits of the other party.

1-out-of-4 Oblivious Transfer (OT) in Semi-Honest Model

- Inputs
  - Sender (Party 1) has four messages m_0, m_1 , m_2 , m_3
  - Receiver (Party 2) has 2 bit □□{00, 01, 10, 11}
- Outputs
  - Sender receives nothing
  - Receiver obtains one of the 4 messages and learns nothing of the other messages.

Multiplication (cont)

- Party 1 prepares a table as follows:
  - Row 1 corresponds to Party 2's input 00
  - Row 2 corresponds to Party 2's input 01
  - Row 3 corresponds to Party 2's input 10
  - Row 4 corresponds to Party 2's input 11
- 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 $a_2=1,b_2=1$

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

Summary

- By computing each gate this way, at the end the parties hold shares of the output wires.
- Function output generated by simply sending shares to each other.
- Secure multi-party computation has many applications.