Advanced Tools from Modern Cryptography

Lecture 7
Secure 2-Party Computation: Yao’s Garbled Circuit
Plan (Still sticking with passive corruption):
- Two protocols, that are secure computationally
  - The “passive-GMW” protocol for any number of parties
  - A 2-party protocol using Yao’s Garbled Circuits
- Both rely on a computational primitive called Oblivious Transfer

Last time: OT and Passive-GMW
(Not exactly the version from the GMW’87 paper.)

Today: 2-Party protocol using Yao’s Garbled Circuits
2-Party SFE

Secure Function Evaluation (SFE) IDEAL:

- Trusted party takes \((X;Y)\). Outputs \(g(X;Y)\) to Alice, \(f(X;Y)\) to Bob
- Randomized Functions: \(g(X;Y;r)\) and \(f(X;Y;r)\) s.t. neither party knows \(r\) (beyond what is revealed by output)

OT is an instance of a (deterministic) 2-party SFE

\[ g(x_0,x_1;b) = \text{none}; f(x_0,x_1;b) = x_b \]

Single-Output SFE: only one party gets any output
2-Party SFE

- Can reduce general SFE (even randomized) to a single-output deterministic SFE

\[ f'(X, M, r_1; Y, r_2) = (g(X; Y; r_1 \oplus r_2) \oplus M, f(X; Y; r_1 \oplus r_2)). \]

- Compute \( f'(X, M, r_1; Y, r_2) \) with random \( M, r_1, r_2 \)
- Bob sends \( g(X, Y; r_1 \oplus r_2) \oplus M \) to Alice

- Passive secure

- For active security too: \( f' \) authenticates (one-time MAC) as well as encrypts \( g(X; Y; r_1 \oplus r_2) \) using keys input by Alice

- Generalizes to more than 2 parties too [Exercise]

- Yao: Reduces single-output deterministic 2-party SFE to OT

- Single round of interaction, but with only computational security (cf. GMW: information-theoretic, but many rounds)
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information “obliviously”

If we had a trusted third party

We Predict STOCKS!!

A: up, B: down

Sure

All 2 of them!

I need just one

But can't tell you which
Why is OT Useful?

Naïve 2PC from OT

- Say Alice’s input $x$, Bob’s input $y$, and only Bob should learn $f(x,y)$.
- Alice (who knows $x$, but not $y$) prepares a table for $f(x,\cdot)$ with $D = 2^{|y|}$ entries (one for each $y$).
- Bob uses $y$ to decide which entry in the table to pick up using 1-out-of-$D$ OT (without learning the other entries).
- Bob learns only $f(x,y)$ (in addition to $y$). Alice learns nothing beyond $x$.
- OT captures the essence of MPC: Secure computation of any function $f$ can be reduced to OT.

Secure protocol for $f$ using access to ideal OT

Problem: $D$ is exponentially large in $|y|$

Plan: somehow exploit efficient computation (e.g., circuit) of $f$
Functions as Circuits

- Directed acyclic graph
- Nodes: multiplication and addition gates, constant gates, inputs, output(s)
- Edges: wires carrying values from F
- Each wire comes out of a unique gate, but a wire might fan-out
- Can evaluate wires according to a topologically sorted order of gates they come out of
2-Party MPC for General Circuits

“General”: evaluate any arbitrary (boolean) circuit

One-sided output: both parties give inputs, only one party gets outputs

Either party maybe corrupted passively

Consider evaluating OR (single gate circuit)

Alice holds $x=a$, Bob has $y=b$; Bob should get $\text{OR}(x,y)$
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$

Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box).

She un-labels the four boxes and sends them in random order to Bob. Also sends the key $K_{x=a}$ (labeled only as $K_x$).

So far Bob gets no information.

Bob "obliviously picks up" $K_{y=b}$, and tries the two keys $K_x,K_y$ on the four boxes. For one box both locks open and he gets the output.
A Physical Protocol

Secure?

For curious Alice: only influence from Bob is when he picks up his key $K_y=b$

But this is done “obliviously”, so she learns nothing

For curious Bob: What he sees is predictable (i.e., can be simulated), given the final outcome

What Bob sees: His key opens $K_y$ in two boxes, Alice’s opens $K_x$ in two boxes; only one random box fully opens. It has the outcome.

Note when $y=1$, cases $x=0$ and $x=1$ appear same
Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire $w$ in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$.
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire $w$ in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$.

For each gate $G$ with input wires $(u,v)$ and output wire $w$, prepare 4 boxes $B_{uv}$ and place $K_{w=G(a,b)}$ inside box $B_{uv=ab}$. Lock $B_{uv=ab}$ with keys $K_{u=a}$ and $K_{v=b}$.

Give to Bob: Boxes for each gate, one key for each of Alice’s input wires.

Obliviously: one key for each of Bob’s input wires.

Boxes for output gates have values instead of keys.
Evaluation: Bob gets one key for each input wire of a gate, opens one box for the gate, gets one key for the output wire, and proceeds.

- Gets output from a box for the output gate.

Security similar to before.

Curious Alice sees nothing.

Bob can simulate his view given final output: Bob could prepare boxes and keys (stuffing unopenable boxes arbitrarily); for an output gate, place the output bit in the box that opens.
Garbled Circuit

That was too physical!

Yao’s Garbled circuit: boxes/keys replaced by Symmetric Key Encryption (specifically, using a Pseudorandom Function or PRF)

\[ \text{Enc}_K(m) = \text{PRF}_K(\text{index}) \oplus m, \text{ where index is a wire index} \]

(distinct for different wires fanning-out of the same gate)

Double lock: \( \text{Enc}_{K_x}(\text{Enc}_{K_y}(m)) \)

PRF in practice: a block-cipher, like AES

Uses Oblivious Transfer for strings: For passive security, can just repeat bit-OT several times to transfer longer keys

Security? Need to first define security when computational primitives are used! (Next time!)
Garbled Circuit

One issue when using encryption instead of locks

Given four doubly locked boxes (in random order) and two keys, we simply tried opening all locks until one box fully opened

With encryption, cannot quite tell if a box opened or not! Outcome of decryption looks random in either case.

Simple solution: encode the keys so that wrong decryption does not result in outputs that look like valid encoding of keys

Better solution: For each wire 0 & 1 keys have distinct “shape” labels, assigned at random. Each locked box marked with the shape of the two keys needed to unlock it.
Pseudorandomness

Basic notions in (symmetric-key) cryptography

A Pseudorandomness Generator (PRG) and a Pseudorandom Function (PRF)

PRG takes a short seed and (deterministically) outputs a longer string: $G_k: \{0,1\}^k \rightarrow \{0,1\}^{n(k)}$ where $n(k) > k$

A PRF is essentially a PRG with a “long” output, with an extra input (index) which specifies a block to be selected from this output

Security definitions based on computational indistinguishability
Indistinguishability

Distribution ensembles \{A_k\}, \{B_k\} \textit{computationally indistinguishable} if \(\forall\) Probabilistic Polynomial Time tests \(T\), \(\exists\) negligible \(v(k)\) s.t.
\[
\left| \Pr_{x \leftarrow A_k}[T(x)=1] - \Pr_{x \leftarrow B_k}[T(x)=1] \right| \leq v(k)
\]

Recall

\(A_k \approx B_k\)
Pseudorandomness Generator (PRG)

- Takes a short seed and (deterministically) outputs a long string
- \( G_k : \{0,1\}^k \rightarrow \{0,1\}^{n(k)} \) where \( n(k) > k \)

Security definition:
- \( \{G_k(x)\}_{x \leftarrow \{0,1\}^k} \approx U_{n(k)} \)
- \( \forall \text{ PPT} \), \( \text{REAL} \approx \text{IDEAL} \)

\( z \leftarrow \{0,1\}^n \) cannot be statistically indistinguishable from \( U_{n(k)} \) unless \( n(k) \leq k \) (Why?)
Pseudorandom Function (PRF)

- A PRF is essentially a PRG with a “long” output
  - A function $F(s;i)$ outputs the $i^{th}$ block of the pseudorandom string corresponding to seed $s$
  - When the number of blocks is small (polynomial in the security parameter), this is the same as a PRG with a longer output
  - This suffices for Garbled Circuits

- More generally a PRF supports exponentially many blocks (i.e., large domain for $i$)
  - Needs a new security definition as the adversary can’t be given the entire string
Pseudorandom Function (PRF)

- A compact representation of an exponentially long (pseudorandom) string
- Allows “random-access” (instead of just sequential access)
- A function $F(s; i)$ outputs the $i$th block of the pseudorandom string corresponding to seed $s$
- Exponentially many blocks (i.e., large domain for $i$)

Security definition

- Need to define pseudorandomness for a function (not a string)
- Idea: the view of an adversary arbitrarily interacting with the function is indistinguishable from its view when interacting with a random function

If the domain of $i$ is polynomial sized (as is sufficient for Garbled Circuits), can implement PRF using a PRG
A pseudorandom function (PRF) is a function $F(s, \cdot)$ that maps a string $s$ from $\{0,1\}^k$ to a function $F(s, \cdot)$, such that:

$$F: \{0,1\}^k \times \{0,1\}^m(k) \rightarrow \{0,1\}^n(k)$$

is a PRF if:

- For all probabilistic polynomial-time (PPT) adversaries, the distribution of $F(s, \cdot)$ is statistically indistinguishable from a random function $R(\cdot)$.

This can be formally stated as:

$$\forall \text{ PPT} \quad \text{REAL} \approx \text{IDEAL}$$

The diagram illustrates this concept with $s \leftarrow \{0,1\}^k$ and $F(s, \cdot)$ mapping to a random function $R(\cdot)$. The condition $\forall \text{ PPT}$ ensures that the distribution of $F(s, \cdot)$ is indistinguishable from a random function for all PPT adversaries.