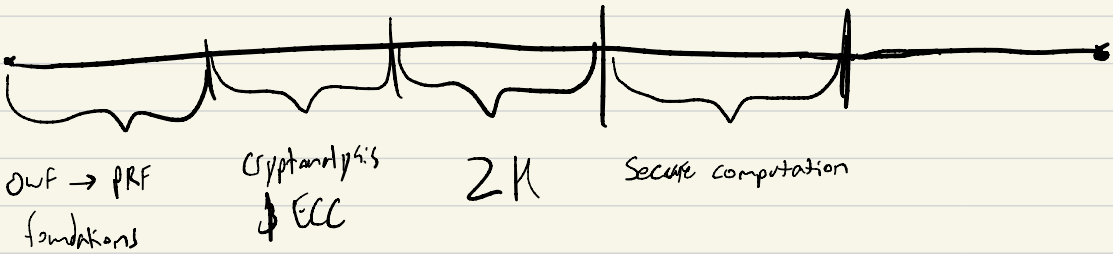
Lecture 12: OT / 2PC

5/14/20

Plan

- What is secure computation / 2PC
- Oblivious Transfer (OT)
- Garbled Circuits

Course Outline



Next week we'll discuss multiparty computation (MPC), of which 2PC is a special case.

Two-party Computation (2PC)

In this and the next few lectures, we will discuss secure computation (A.K.A. multiparty computation (MPC)), where multiple parties, each holding secret inputs, can compute a function on all their inputs without revealing their inputs to each other!

Today we will focus on the special case of Two-party Computation (2PC)

Setting:

Alice knows $x \in \{0,1\}^*$

Bob knows $y \in \{0,1\}^*$

Both Alice & Bob know a function $f: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$

Sometimes called the "functionality"

2PC protocol: Alice and Bob both learn $f(x,y)$, and nothing else

Note: more generally, we could even have two different functions f_A and f_B where Alice learns $f_A(x,y)$ and Bob learns $f_B(x,y)$.

Examples of 2PC problems

2PC is a very broad abstraction and many privacy problems can be thought of as 2PC problems:

Yao's millionaire problem: 2 millionaires want to find out who is richer without revealing how much money they each have.

$f(x,y)$: output 1 if $x>y$, 0 otherwise.

Online advertising: Google & Macy's want to find out how successful an ad campaign is without revealing their own business information.

Google input: who saw ad

Macy's input: who bought stuff

2PC output: Number of people who saw ad and bought

Private contact discovery: which of my contacts use Signal?

My input: Contact list

Signal input: list of Signal users

My 2PC output: intersection of the lists

Signal's 2PC output: nothing

↑ Private Set Intersection (PSI)

Even Zero-Knowledge can be thought of as a 2PC:

Alice input: (x, w)

Bob input: x

Alice's output: nothing

Bob's output: $x \stackrel{?}{\in} L$

Defining Security for 2PC

Two main kinds of security we can consider:

Semi-honest: Alice & Bob follow protocol exactly (like HV2K)

Malicious: Alice & Bob can deviate from protocol (like 2K)

We'll focus on semi-honest security

Def: An interactive protocol $\langle A, B \rangle$ for functionality f,

$f: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ must satisfy

Correctness: for all inputs $x, y \in \{0,1\}^*$

$$\Pr[\langle A(x), B(y) \rangle = f(x,y)] = 1$$

Privacy: There exist efficient simulators Sim_A, Sim_B s.t. $\forall x, y \in \{0,1\}^*$

$$Sim_A(x, f(x,y)) \approx_c view_A(\langle A(x), B(y) \rangle)$$

$$Sim_B(y, f(x,y)) \approx_c view_B(\langle A(x), B(y) \rangle)$$

Oblivious Transfer (OT)

A surprising (and surprisingly useful) 2PC. We'll see an awesome application later this lecture.

We'll call the two parties S and R for sender & receiver

S input: 2 messages m_0, m_1, S output: nothing

R input: a selection bit b, R output: m_b

S sends m_b to R without knowing what it sent!

R learns m_b and nothing else! (so S can't just send both)

We'll right down security for OT explicitly even though it's a special case of the general 2PC security definition above.

Def: protocol $\langle S, R \rangle$ is an oblivious transfer protocol if it satisfies these properties:

Correctness: $\forall m_0, m_1 \in \{0,1\}^*$, $\Pr[\text{output}_R(\langle S(m_0,m_1), R(b)\rangle) = m_b] = 1$

Sender Privacy: There exists Sim_R s.t. $\forall m_0, m_1$ and $b \in \{0,1\}$

$$\text{View}_R(\langle S(m_0,m_1), R(b)\rangle) \approx_c \text{Sim}_R(b, m_b)$$

Receiver Privacy: There exists Sim_S s.t. $\forall m_0, m_1$ and $b \in \{0,1\}$

$$\text{View}_S(\langle S(m_0,m_1), R(b)\rangle) \approx_c \text{Sim}_S(m_0, m_1)$$

Building OT

(Bellare-Micali OT construction)

Let G be a group of prime order q with generator $g \in G$.

Let $H: G \to \{0,1\}^{\ell}$ be hash function modeled as Random Oracle.

(ℓ: length of msgs)

Construction is based on El-Gamal encryption

$S(m_0, m_1 \in \{0,1\}^{\ell})$		$R(b)$
$C \xleftarrow{R} G$	$\xrightarrow{C}$	$K \xleftarrow{R} \mathbb{Z}_q$] El-Gamal secret key
		$Y_b \leftarrow g^K$] 2 different El-Gamal public keys (but R only knows secret key for Y_b)
		$Y_{1-b} \leftarrow C / g^K$]
	$\xleftarrow{Y_0, Y_1}$	
check $Y_0 \cdot Y_1 = C$ (abort if not)		
$r_0, r_1 \xleftarrow{R} \mathbb{Z}_q$		
$C_0 \leftarrow (g^{r_0}, H(Y_0^{r_0}) \oplus m_0)$] El-Gamal encryption. We use the hash and $\oplus$ instead of just multiplying because $m_0, m_1 \in \{0,1\}^{\ell}$ not G.		
$C_1 \leftarrow (g^{r_1}, H(Y_1^{r_1}) \oplus m_1)$		
	$\xrightarrow{C_0, C_1}$	parse $C_b = (U, V)$
		decrypt $m_b \leftarrow H(U^K) \oplus V$
		El-Gamal decryption

Proof idea:

Correctness — R has key to decrypt m_b

Sender privacy — follows from CDH

R only has secret key for one of y_0, y_1

So only one of c_0, c_1 can be decrypted by

the security of the encryption scheme.

Receiver privacy — is information-theoretic

Note that the only message S gets is y_0, y_1

But y_b is uniformly random in G since $k \xleftarrow{R} \mathbb{Z}_q$

And y_{1-b} is uniformly random in G since c and y_b are

and $y_{1-b} \leftarrow c/y_b$

To complete these proofs, we would need to write down Sim_R, Sim_S and show they output the right distributions using the ideas above.

Yao's Garbled Circuits

Generic 2PC protocol for ANY functionality f

f must be represented as a boolean circuit:

AND/XOR is enough to compute anything → - AND/XOR gates
- No loops
- input wire for each bit of input, output wire for each bit of output

Circuit garbling only requires a symmetric encryption scheme & OT!

High-level plan: 1. Alice takes circuit for f, "garbles" it, and sends the circuit to Bob.

2. Bob uses OT to learn secret information needed to "ungarble" the circuit only on Bob's inputs and finishes the computation.

Warm up: Garbling a single AND gate

Idea: Associate each row of truth table with a ciphertext and each pair of inputs with distinct keys.

L	R	AND
0	0	0
0	1	0
1	0	0
1	1	1

$\Rightarrow$

L	R	AND	
K_L^0	K_R^0	$E(K_L^0, E(K_R^0, 0))$	$= C_{00}$
K_L^0	K_R^1	$E(K_L^0, E(K_R^1, 0))$	$= C_{01}$
K_L^1	K_R^0	$E(K_L^1, E(K_R^0, 0))$	$= C_{10}$
K_L^1	K_R^1	$E(K_L^1, E(K_R^1, 1))$	$= C_{11}$

protocol: Alice has input b_L, Bob has input b_R

1. Alice samples keys $K_L^0, K_L^1, K_R^0, K_R^1$ and computes $C_{00}, C_{01}, C_{10}, C_{11}$, sends them all to Bob in random order, along with $K_L^{b_L}$

2. Bob does OT with Alice to get $K_R^{b_R}$

   Alice input: K_R^0, K_R^1 Bob input: b_R

3. Bob tries to decrypt each of $C_{00}, C_{01}, C_{10}, C_{11}$, but only succeeds with one of them. The only ciphertext it can decrypt is the output! Bob sends output to Alice.

Alice's view: OT protocol & output

Bob's view: Ciphertexts, OT protocol, random keys $K_L^{b_L}, K_R^{b_R}$, output

Both can be simulated using security of Encryption + OT.

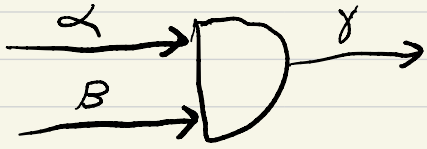
Same idea applies for XOR

Application: Romantic Cryptography

Garbling a whole circuit

How can Alice use the idea above to garble a full circuit instead of just one gate?

Idea: chain garbled gates!



Instead of encrypting 0/1, encrypt the inputs to the next gate! (except encrypt 0/1 for output wire)

α	β	γ
0	0	0
0	1	0
1	0	0
1	1	1

⟹

α	β	γ
K_α^0	K_β^0	$E(K_\alpha^0, E(K_\beta^0, K_\gamma^0))$
K_α^0	K_β^1	$E(K_\alpha^0, E(K_\beta^1, K_\gamma^0))$
K_α^1	K_β^0	$E(K_\alpha^1, E(K_\beta^0, K_\gamma^0))$
K_α^1	K_β^1	$E(K_\alpha^1, E(K_\beta^1, K_\gamma^1))$

protocol:

1. Alice Garbles full circuit $f: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, sends Bob 4 CTs for each gate, along with all n keys corresponding to bits of Alice's input.
2. Bob does n OTs, one for each bit of its input
3. Bob decrypts one ciphertext for each gate from inputs to outputs, ends up with circuit's 0/1 output. sends result to Alice.

Improving Efficiency of Garbled Circuits

- There is a lot of research in making garbled circuits more efficient, and many optimizations to reduce protocol costs, we'll mention a few here.

- Half-gates optimization allows Alice to send only 2 cts per gate instead of 4.

- As described here, OT is performance bottleneck, but this is no longer the case in practice because of OT extension. OT extension shows how to extend a few real OTs so you can do a much larger number of OTs for the same cost. That is, you do group operations for a few OTs and then only need symmetric crypto for the larger number of OTs

- Free XOR optimization allows Alice to send nothing at all for XOR gates, so cost of protocol really depends primarily on the number of AND gates in circuit

Idea: - Alice picks a random value R at beginning of protocol.
- instead of choosing random K^0, K^1 for each gate, she chooses random K^0 and sets $K^1 \leftarrow K^0 \oplus R$.
- Now XORs can be done with no crypto ops!

define: $K_\alpha^0 \oplus K_\beta^0 = K_\gamma^0$

then $K_\alpha^0 \oplus K_\beta^1 = K_\alpha^0 \oplus K_\beta^0 \oplus R = K_\gamma^0 \oplus R = K_\gamma^1$

$K_\alpha^1 \oplus K_\beta^0 = K_\alpha^0 \oplus R \oplus K_\beta^0 = K_\gamma^0 \oplus R = K_\gamma^1$

$K_\alpha^1 \oplus K_\beta^1 = K_\alpha^0 \oplus R \oplus K_\beta^0 \oplus R = K_\gamma^0$

Final note: optimized garbled circuits are very nice, but for many applications it may prove more efficient to build a custom protocol rather than evaluating the functionality as a circuit (or even thinking of it as a generic 2PC)