Cryptography

Hari

Cryptography

Cryptography

Secure Communication

- Encryption, signatures
- AES, RSA, etc.
- HTTPS!

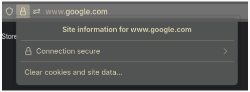

Figure 1: HTTPS Secure Connection

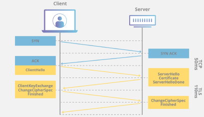

Figure 2: TLS: Cloudflare

Cryptography

Secure Communication

- Encryption, signatures
- AES, RSA, etc.
- HTTPS!

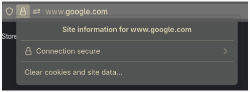

Figure 1: HTTPS Secure Connection

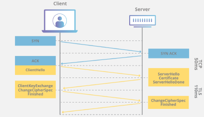

Figure 2: TLS: Cloudflare

Secure **Computation**

- Modern day constructions
- MPC, FE, FHE, ZK, and more
- Voting and Auctions, Verifiable computation

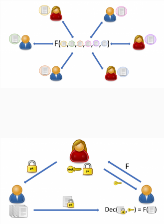

Figure 3: Secure Computation: COSIC

Secret Sharing

Secret Sharing

- **Secret sharing** is a way for a party to split a secret value s into n “shares”.

Secret Sharing

- **Secret sharing** is a way for a party to split a secret value s into n "shares".
- A secret sharing scheme consists of two algorithms: Share and Rec, where
  - Share takes in a secret and outputs a set of shares.
  - Rec takes in a set of shares and outputs the secret, or a failure.

Secret Sharing

- **Secret sharing** is a way for a party to split a secret value s into n "shares".
- A secret sharing scheme consists of two algorithms: Share and Rec, where
  - Share takes in a secret and outputs a set of shares.
  - Rec takes in a set of shares and outputs the secret, or a failure.
- *Correctness:* $\mathrm{Rec}(\mathrm{Share}(s)) = s$.

Secret Sharing

- **Secret sharing** is a way for a party to split a secret value s into n "shares".
- A secret sharing scheme consists of two algorithms: Share and Rec, where
  - Share takes in a secret and outputs a set of shares.
  - Rec takes in a set of shares and outputs the secret, or a failure.
- *Correctness:* $\mathrm{Rec}(\mathrm{Share}(s)) = s$.
- *Privacy:* A set of $n-1$ shares of the secret reveals no information about s.

Secret Sharing

Let s be a secret in $\{0, 1, \ldots, p-1\}$ for a prime p. We do arithmetic $\bmod p$.

- Share: Sample one share $s_1 \leftarrow \{0, 1, \ldots, p-1\}$ uniformly at random. Return $(s_1, s - s_1)$.
- $\mathrm{Rec}(s_1, s_2)$: Return $s_1 + s_2$.

Secret Sharing

For n parties:

- Share: Sample $n-1$ shares $s_1, \ldots, s_{n-1} \leftarrow \{0, 1, \ldots, p-1\}$ uniformly at random. Return $(s_1, s_2, \ldots, s_{n-1}, s - s_1 - s_2 - \ldots - s_{n-1})$.
- $\text{Rec}(s_1, s_2)$: Return $s_1 + s_2 + \ldots + s_n$.

Multiparty Computation

Computation

- We perform computation on data every day all the time!
- Statistics on data
- Finance and trading computations
- Machine learning computation on data

Secure Multiparty Computation

- Can we compute a function of data from multiple parties **securely**?

Secure Multiparty Computation

- Can we compute a function of data from multiple parties **securely**?

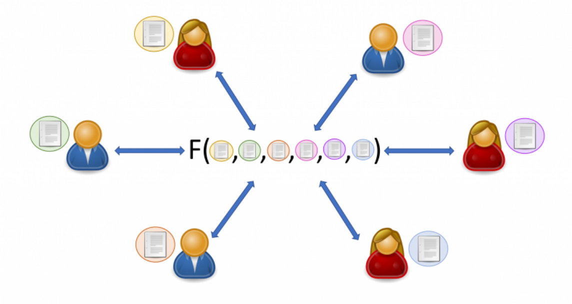

Figure 4: Secure MPC: Cosic

Secure Multiparty Computation

- We have n employees, who would like to compute the average salary without revealing individual salaries.

Secure Multiparty Computation

- We have n employees, who would like to compute the average salary without revealing individual salaries.
- Train a machine learning model without revealing training datasets

Secure Multiparty Computation

- We have n employees, who would like to compute the average salary without revealing individual salaries.
- Train a machine learning model without revealing training datasets

Secure Multiparty Computation

- First two party protocols introduced by Yao
- Generalized to multiple parties by Goldreich, Micali, and Widgerson

Secure Multiparty Computation

- First two party protocols introduced by Yao
- Generalized to multiple parties by Goldreich, Micali, and Widgerson
- Modern day protocols rely on interesting cryptography and a lot of optimizations

Constructing MPC from secret sharing

Expressing computation

- How do we express the computation of a function $\mathcal{F}(x_1, x_2, \ldots, x_n)$?

Expressing computation

- How do we express the computation of a function $\mathcal{F}(x_1, x_2, \ldots, x_n)$?
- In this protocol, we have $\mathcal{F}$ be a polynomial in the variables $x_1, \ldots, x_n$.
- We call this an **arithmetic circuit**: $\mathcal{F}$ is a circuit with addition and multiplication gates.

Expressing computation

- $x_3(x_1 + x_2)$
- $x_2^2 + x_3 x_1$

Expressing computation

- $x_3(x_1 + x_2)$
- $x_2^2 + x_3 x_1$
- We can even write if statements!

```
if (condition == 0) {
 return f(x)
else if (condition == 1) {
 return g(x)
}
```

Expressing computation

- $x_3(x_1 + x_2)$
- $x_2^2 + x_3 x_1$
- We can even write if statements!

```
if (condition == 0) {
 return f(x)
else if (condition == 1) {
 return g(x)
}
```

- $(\texttt{condition} - 1) \cdot f(x) + (\texttt{condition}) \cdot g(x)$

Protocol

- Each party holds some input a. The party secret shares it into a_1 and a_2, and gives a_2 to the other party.

Protocol

- Each party holds some input a. The party secret shares it into a_1 and a_2, and gives a_2 to the other party.
- This is done for each input – so every party holds a secret share for each input. For a value a, we notate this by $[a]$.

Evaluating addition

- For inputs a, b, we want to compute $a + b$.
- One party has a_1, b_1 and the other has a_2, b_2.

Evaluating addition

- For inputs a, b, we want to compute $a + b$.
- One party has a_1, b_1 and the other has a_2, b_2.
- Note that $a_1 + b_1$ and $a_2 + b_2$ are shares for $a + b$! Our prior secret sharing scheme was **linear**.

Evaluating multiplication

- For inputs a, b, we want to compute ab.
- Does $a_1 b_1$ and $a_2 b_2$ work?

Evaluating multiplication

- For inputs a, b, we want to compute ab.
- Does $a_1 b_1$ and $a_2 b_2$ work?
- No! We want shares of $(a_1 + a_2)(b_1 + b_2)$. How do we do this?

Evaluating multiplication

- For inputs a, b, we want to compute ab.
- Does $a_1 b_1$ and $a_2 b_2$ work?
- No! We want shares of $(a_1 + a_2)(b_1 + b_2)$. How do we do this?
- We introduce an extra party to provide data for multiplication.

Beaver triples

- Let the third party sample random values r, s, and set $t = rs$.
- The third party then secret shares the values to $[r], [s], [t]$ and sends the shares r_1, s_1, t_1 and r_2, s_2, t_2 to each party. This is known as a **Beaver triple**.

Beaver triples

- Let the third party sample random values r, s, and set $t = rs$.
- The third party then secret shares the values to $[r], [s], [t]$ and sends the shares r_1, s_1, t_1 and r_2, s_2, t_2 to each party. This is known as a **Beaver triple**.
- Now if the parties want to compute shares of ab from $[a]$ and $[b]$, they do the following:
- The two parties compute shares $[a - r]$ and reveal their shares to each other.
- Similarly, they compute shares of $[b - s]$ and reveal the shares.

Beaver triples

- Let the third party sample random values r, s, and set $t = rs$.
- The third party then secret shares the values to $[r], [s], [t]$ and sends the shares r_1, s_1, t_1 and r_2, s_2, t_2 to each party. This is known as a **Beaver triple**.
- Now if the parties want to compute shares of ab from $[a]$ and $[b]$, they do the following:
- The two parties compute shares $[a - r]$ and reveal their shares to each other.
- Similarly, they compute shares of $[b - s]$ and reveal the shares.

Beaver triples

- The parties now have $x = a - r$ and $y = b - s$. Each party computes

$$xy + xs_0 + yr_0 + t_0$$

$$xy + xs_1 + yr_1 + t_1$$

- Observe that

$$ab = (a - r + r)(b - s + s) = (x + r)(y + s)$$

Beaver triples

- Can we do this without the third party?
- Yes! We can generate beaver triples without a third party using threshold t out of n secret sharing.

More Fun Things

- Our protocol assumes that parties behave honestly – is it secure if parties behave maliciously?
- (No, we need some kind of verifiable secret sharing)

More Fun Things

- Our protocol assumes that parties behave honestly – is it secure if parties behave maliciously?
- (No, we need some kind of verifiable secret sharing)
- How do we show that this protocol is correct and private? What does that mean?