This Lecture:

- Secure Hardware: Intel SGX, HSM
- Blockchain
- Definition of Differential Privacy
Outsourcing Computation by FHE

Server

Client

Data x
Key sk

c \leftarrow Enc(x)

ct, f

c \leftarrow Eval(f, ct)

c \leftarrow Eval(f, ct)

f(x) \leftarrow Dec_{sk}(ct')
Outsourcing Computation by Secure Hardware

Server

ct \downarrow f \downarrow

Memory

CPU

sk

\[ \text{ct} \]

x \leftarrow \text{Dec}_{sk}(ct)

y := f(x)

ct' \leftarrow \text{Enc}_{sk}(y)

Client

\[ \text{ct} \leftarrow \text{Enc}(x) \]

Data x

Key sk

\[ \text{ct} \leftarrow \text{Enc}(x) \]

\[ \text{ct}, f \]

\[ \text{ct}, f \]

\[ \text{ct} \]

\[ \text{ct} \leftarrow \text{Enc}(x) \]

\[ \text{ct} \leftarrow \text{Enc}(x) \]

\[ \text{ct}' \]

\[ \text{ct}' \]

\[ \text{ct}' \]

\[ y \leftarrow \text{Dec}_{sk}(ct') \]
Intel Software Guard Extension (SGX)

Server

Memory

CPU

Client

"Secure enclave"

Diffie-Hellman key Exchange

\[ b \leftarrow \mathbb{Z}_q \]
\[ \text{output } g^b \]
\[ k := \text{HKDF}(g^{ab}) \]
\[ a \leftarrow \mathbb{Z}_q \]
\[ k := \text{HKDF}(g^{ab}) \]

\[ \text{ct} \leftarrow \text{Enc}_k(x) \]
\[ \text{ct} \leftarrow \text{Enc}_k(y) \]

What could go wrong?
Intel Software Guard Extension (SGX)

**Server**

- Intel
  - (mvk, msk)

**Client**

**Attestation**

**Provisioning**

```
Memory
CPU

(mvk, sk)

\( 6 = \text{Sign}_{msk}(vk) \)
```

```
\( m \)

\( (v, k, 6, 6') \leftarrow \text{Sign}_{sk}(m) \)
```

```
\( m \)

\( (v, k, 6, 6') \)
```
Constraints & Attacks

- Trust in hardware
- Trust in Intel
- Limited memory size: 128 MB
- Replay attacks
- Side-channel attacks: memory access pattern
  - fix: Oblivious RAM (ORAM)
  - overhead $\Theta(\log N)$
Hardware Secure Module (HSM)

\[ ct \equiv Enc_k(m) \]

\[ m \equiv Dec_k(ct) \]

\[ ct' \equiv Enc_k(m) \]

\[ m \equiv Dec_k(ct) \]
Key Agreement

Sample $k_1$, $k_2$, $k_3$ s.t.

$k_1 \oplus k_2 \oplus k_3 = K$

$k_1 \quad k_2 \quad k_3$

$k = k_1 \oplus k_2 \oplus k_3$

FedEx

UPS

USPS
Transactions in Real Life

Alice → Starbucks $3

Starbucks → Bob $$$

A trusted party that maintains a private ledger

Bank of America

1. initiated by sender
2. enough balance in sender's account
Blockchain

- Public ledger that everyone can view & verify
- Maintained by "miners" in a distributed way

**Step 1:** Charlie wants to make a transaction $\text{Charlie} \rightarrow \text{Starbucks} \ 3$
  - broadcasts it to the entire network

**Step 2:** All miners collect all transactions in the network
  - Verify validity
    1. Initiated by sender
    2. Enough balance in sender's account
  - Agree on next block

**Step 3:** Repeat
Transaction Authentication

Alice: \((VKA, SKA) \leftarrow \text{KeyGen}(1^\lambda)\)

Bob: \((VK_B, SK_B) \leftarrow \text{KeyGen}(1^\lambda)\)

Charlie: \((VK_c, SK_c) \leftarrow \text{KeyGen}(1^\lambda)\)

Starbucks: \((VK_s, SK_s) \leftarrow \text{KeyGen}(1^\lambda)\)

\[
\text{Bob} \rightarrow \text{Charlie } B5:
\]

\[
m_1 = (VK_B, VK_c, 5) \quad g_2 \leftarrow \text{Sign}_{SK_B}(m_1)
\]

\[
\text{Charlie} \rightarrow \text{Starbucks } B3:
\]

\[
m_2 = (VK_c, VK_s, 3) \quad g_2 \leftarrow \text{Sign}_{SK_c}(m_2)
\]
Consensus Protocol

\[ TX_1 = \text{Charlie} \rightarrow \text{Starbucks} : 3 \]

\[ m_2 = (v_{k_C}, v_{k_S}, 3) \quad S_2 \leftarrow \text{Sign}_{S_{k_C}}(m_2) \]

\[ TX_2 = \text{Charlie} \rightarrow \text{Alice} : 4 \]

\[ m_3 = (v_{k_C}, v_{k_A}, 4) \quad S_3 \leftarrow \text{Sign}_{S_{k_C}}(m_3) \]

WANT:

1. All miners agree on the same block
2. New block is valid
**Byzantine Agreement**

**Byzantine Fault Tolerance (BFT) Protocol:**

- If $n \geq 3t+1$, then it's possible to reach consensus.
- Assume $t < n/3$, then agree on a valid block.

Any problem?

↓

Agree on a block

( Guaranteed Output Delivery)
Proof of Work (PoW)

Miner 1:

\[
\text{Hash (TX1, nonce)} = 00\ldots01011\ldots0
\]

Find nonce s.t. Hash(block) has \( \geq 30 \) leading 0's.

Consensus Protocol:

Whoever first finds a block that hashes to a value w/ \( \geq 30 \) leading 0's, that block becomes the next block.
Proof of Work (PoW)

Miner 1

Miner 2

Longest Chain Rule: Always adopt the longest chain.

Assuming honest majority of computation power, the longest chain is always valid.
Blockchain

- Efficient verification of sufficient balance: Merkle Tree

- Settlement of a transaction:
  Included in a block which is \( \geq 6 \) blocks deep (\( \sim 1 \) hr)

- Dynamically adjust \# leading 0's s.t. each block takes \( \sim 10 \) min to mine
  Last 1 hr: \( > 6 \) blocks: increase \# leading 0's
  \( < 6 \) blocks: decrease \# leading 0's

- Miners' motivation:
  - transaction fee
  - new coin generated in each block goes to miner

- Extensions
  - Proof of Stake (PoS)
  - Anonymous transactions (zk-SNARGs)
  - Smart contracts
  - Public Bulletin Board
**Differential Privacy**

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Gender</th>
<th>Race</th>
<th>Weight</th>
<th>ZIP</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emily</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fiona</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Want to make the (sensitive) data public/available to others (e.g. for medical study).

**Attempt 1:** "Anonymize" the data.
Delete personally identifiable information (PII): name, DOB, ...

**Attempt 2:** Only answer aggregate statistics queries.
Privacy Guarantee?

Access to the output shouldn't enable one to learn anything about an individual compared to one without access.

Is this possible?
Differential Privacy

\[ X_n \]

\[ \begin{array}{c}
X_1 \\
X_2 \\
\vdots \\
X_m \\
\end{array} \]

\[ D \in X^n \rightarrow M \rightarrow M(D) \]

**Def. E-Differential Privacy** for a randomized mechanism:

∀ neighboring datasets \( D_1 \) & \( D_2 \) (differing in one row),

∀ \( T \in \text{range}(M) \),

\[ \Pr[M(D_1) \in T] \leq e^\epsilon \cdot \Pr[M(D_2) \in T] \]
Differential Privacy

Def: $(\varepsilon, \delta)$ - Differential Privacy for a randomized mechanism:

1. Neighboring datasets $D_1$ & $D_2$ (differing in one row),
2. $\forall T \subseteq \text{range}(M),$

$$\Pr[M(D_1) \in T] \leq e^{\varepsilon} \cdot \Pr[M(D_2) \in T] + \delta$$

Is a bigger $\varepsilon$ better for privacy, or worse?

Is a bigger $\delta$ better for privacy, or worse?