Data Privacy

Protecting Your Data from Anyone

Yubin Xia

Data Privacy

- Data can be used everywhere
  - Risk management
  - Medicine
  - Recommended system
  - …
- Data can be stolen easily
  - Everyone who uses your data can steal it

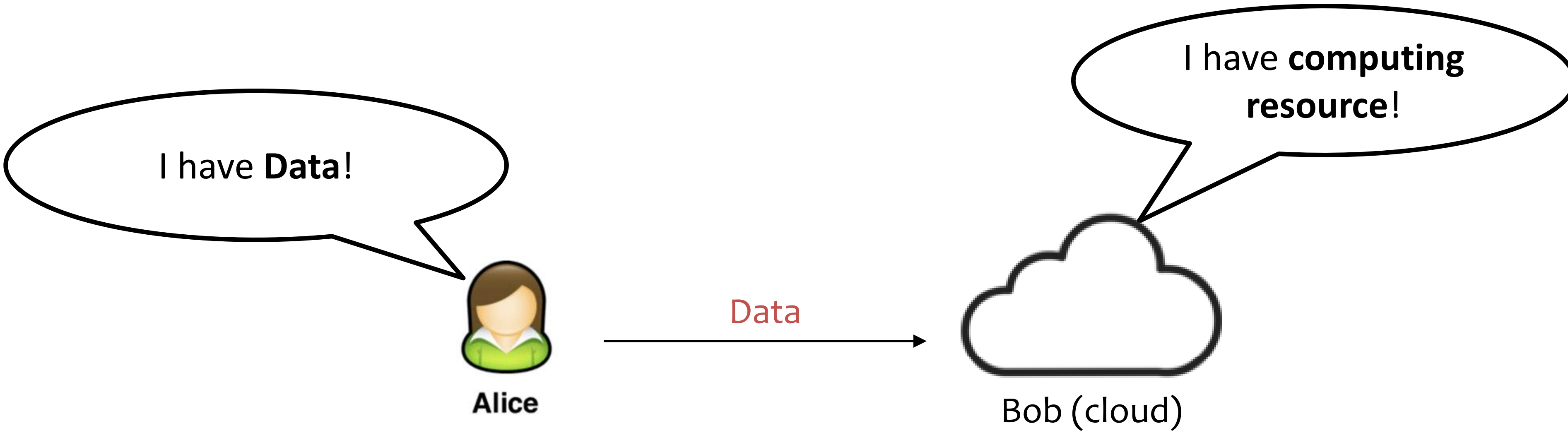
Data Privacy

- What's the target of data privacy system?
  - Allow data **to be used**, and
  - **Protect data from being stolen**
- What will be introduced?
  - Basic data privacy method
    - ZKP, OT, HE, sMPC, TEE, DP
  - Systems which try to enforce data privacy

Trusted Execution Environment

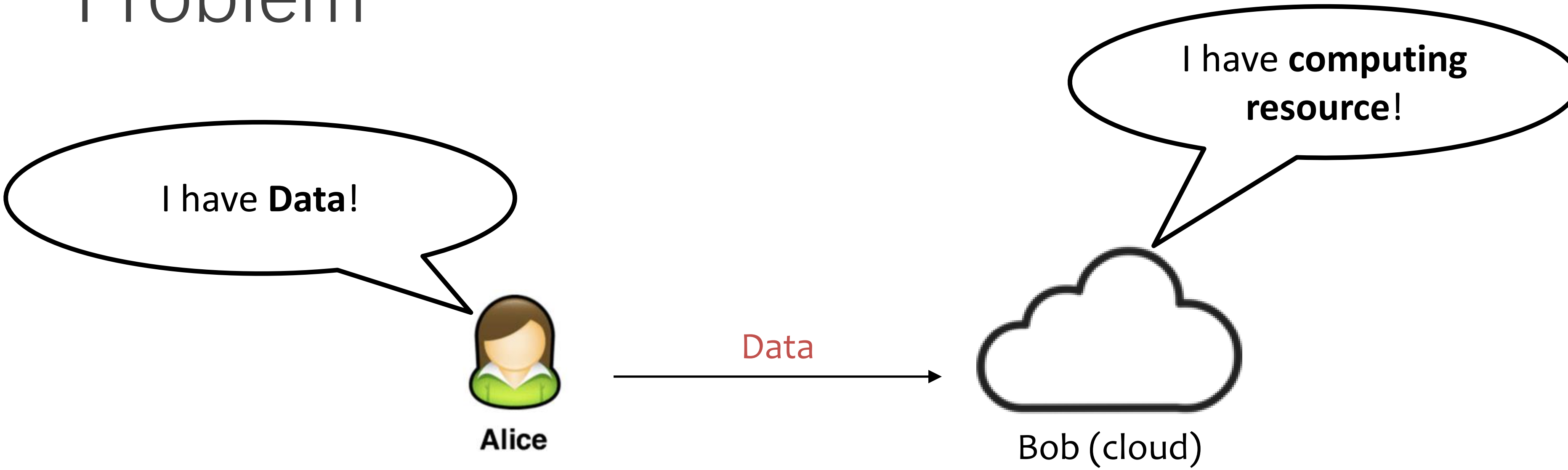
TEE

Problem



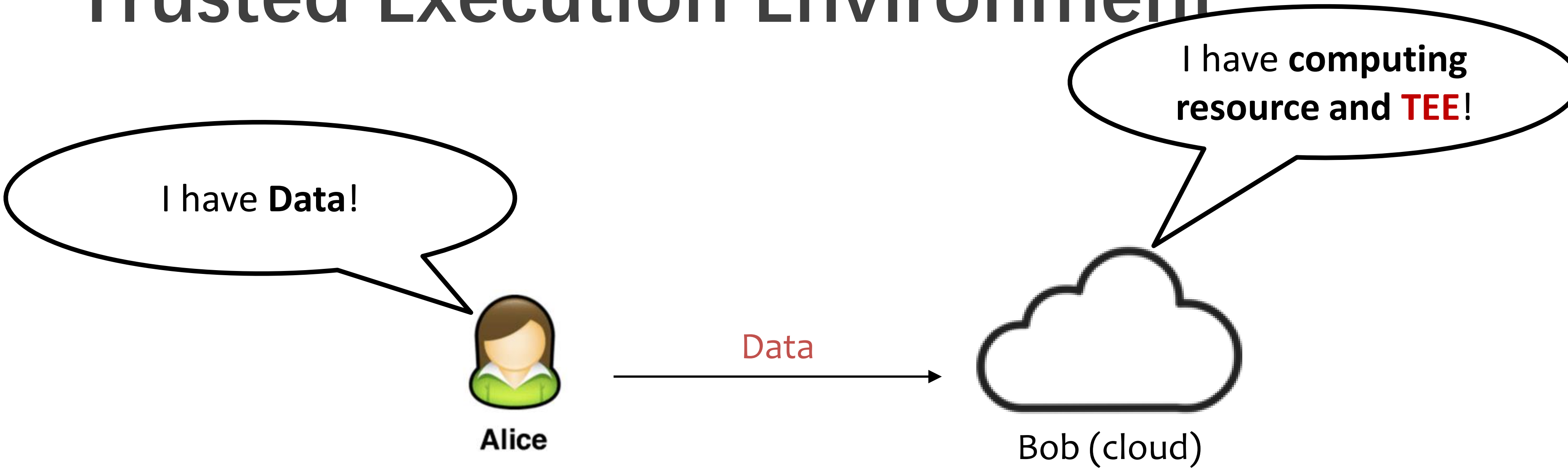
- Alice wants to ask Bob (e.g., a cloud) to perform calculation on her data
- Naïve method: Sending Data to Bob

Problem



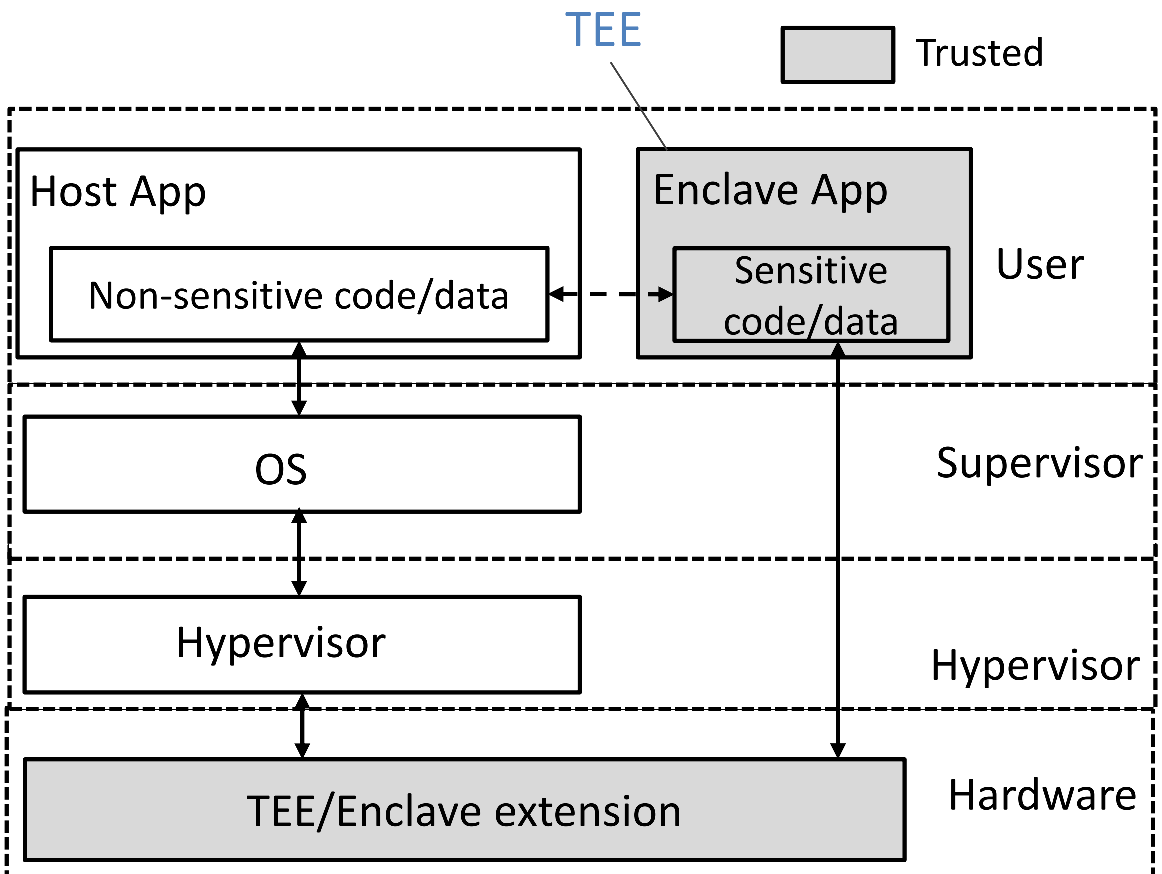
- Alice wants to ask Bob (e.g., a cloud) to perform calculation on her data
- ~~Naïve method: Sending Data to Bob~~

Trusted Execution Environment



- Alice wants to ask Bob (e.g., a cloud) to perform calculation on her data
- ~~Naïve method: Sending Data to Bob~~
- **Bob cloud construct a TEE**

What is TEE



TEE
Trusted
Host App
Non-sensitive code/data
Enclave App
Sensitive code/data
User
OS
Supervisor
Hypervisor
Hypervisor
TEE/Enclave extension
Hardware

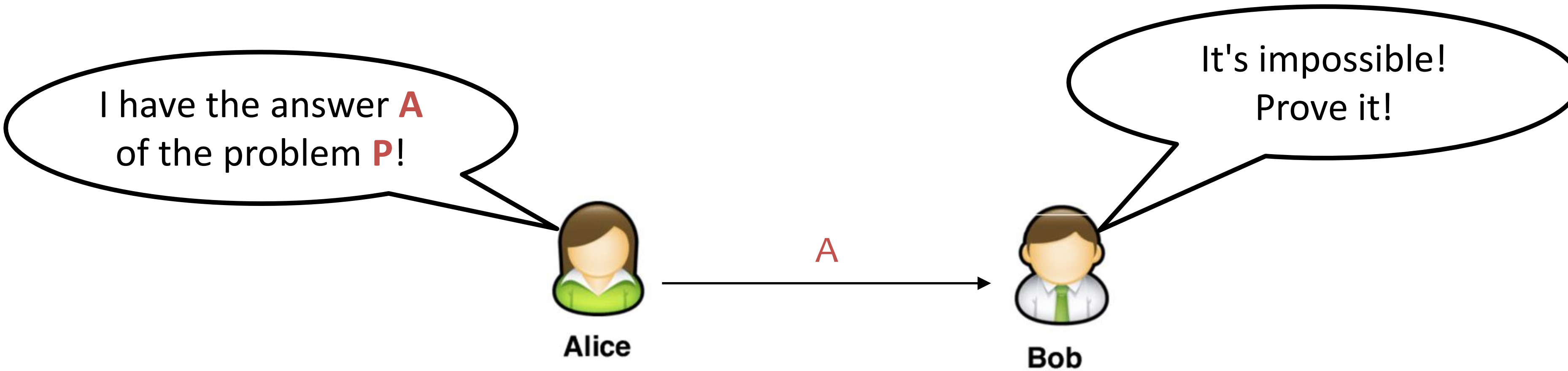
Different TEEs

- Software TEE
  - VM-based TEE
  - Same privilege protection
- ARM TrustZone
- Intel SGX
- AMD SME/SEV
- SANCTUM

Zero-Knowledge Proof

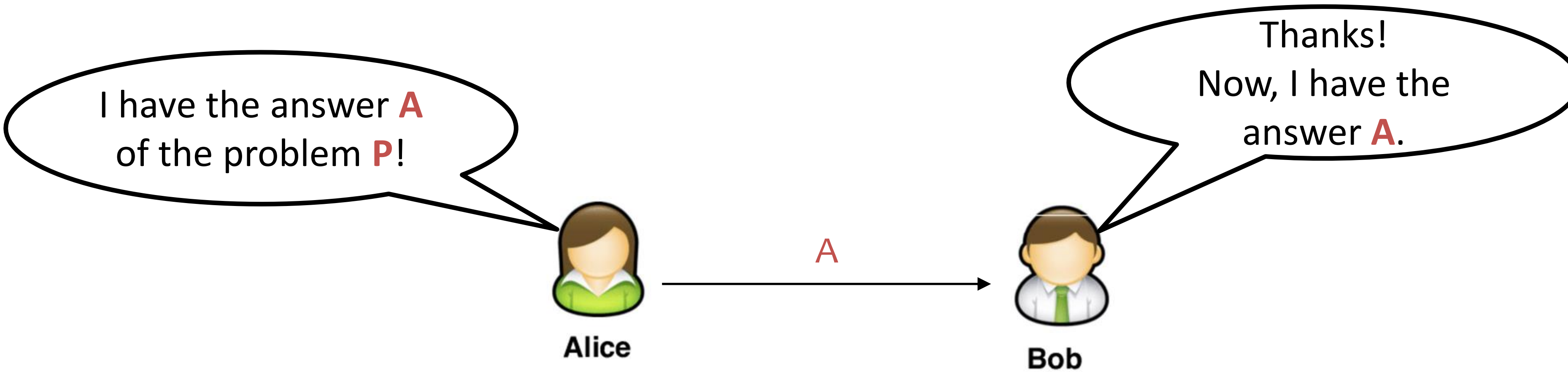
ZKP

Problem



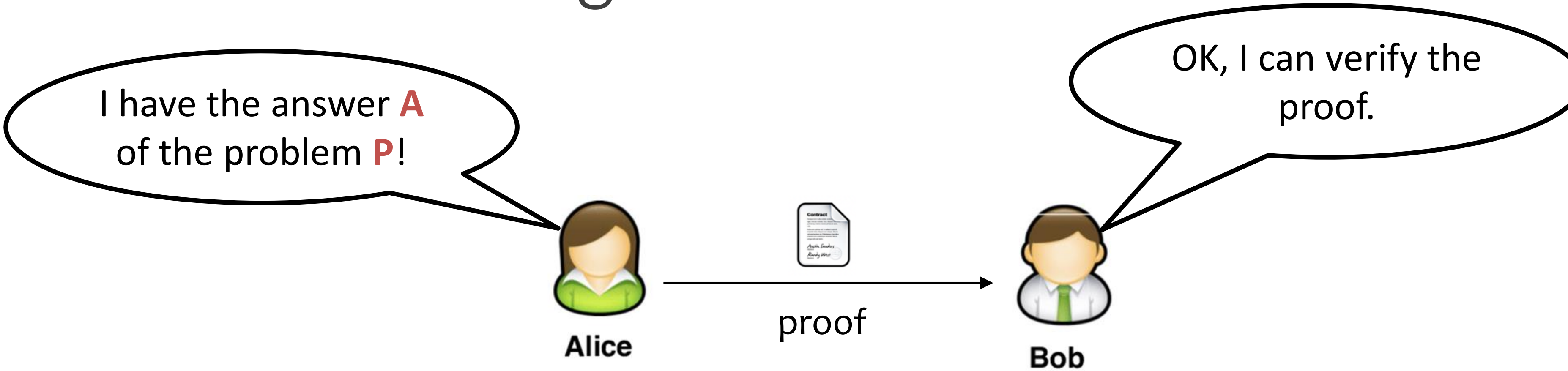
- Alice tries to **prove** to Bob that she **has the answer of a difficult problem** (e.g., a NP problem)
- Naïve method: Sending A to Bob

Problem



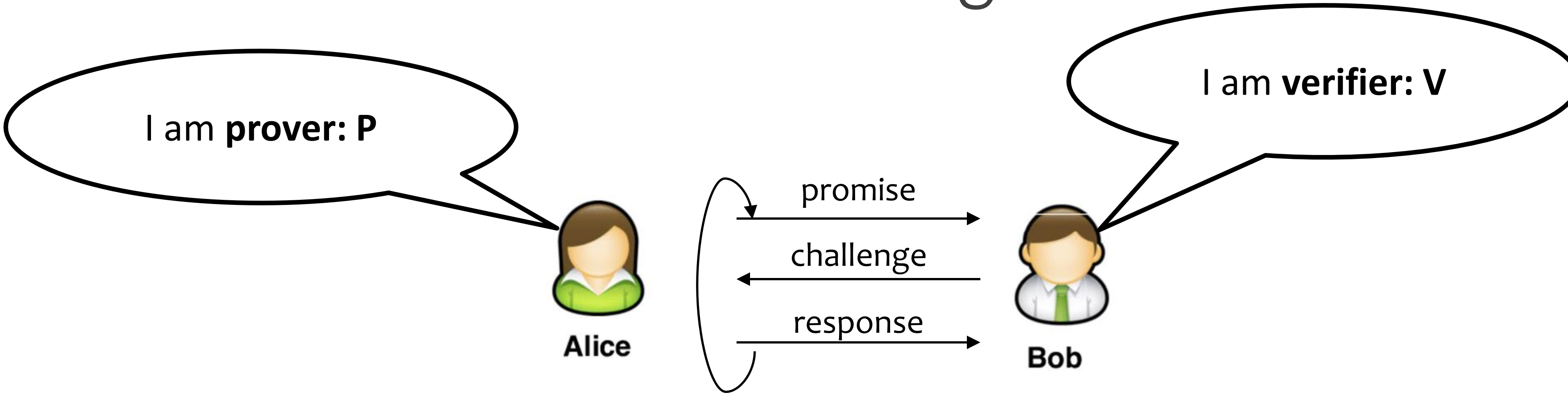
- Alice tries to **prove** to Bob that she **has the answer of a difficult problem** (e.g., a NP problem)
- Naïve method: Sending A to Bob
  - Problem: Bob will get the answer A

Zero-Knowledge Proof



- Alice tries to **prove** to Bob that she **has the answer of a difficult problem** (e.g., a NP problem).
- Zero-Knowledge Proof
  - **Completeness**: Alice can construct the proof if she has A
  - **Soundness**: Alice cannot construct the proof if she doesn't have A
  - **Zero-knowledge**: Bob knows nothing about A

Interactive Zero-Knowledge Proof



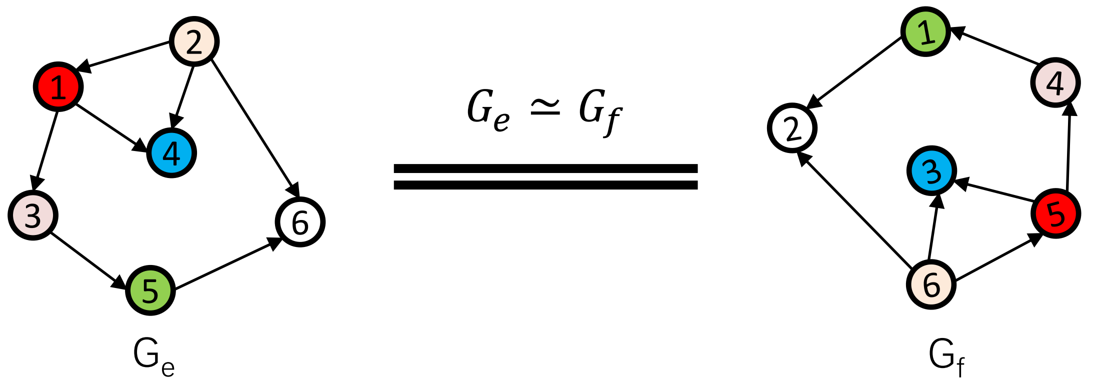
- P has answer $\boldsymbol{x}$ of a problem $\boldsymbol{L}$, and tries to prove it with > 1 iterations:
  - Step-1: P transfers $\boldsymbol{L}$ to $\boldsymbol{L}'$, and promises that $\boldsymbol{L}'$ is transferred from $\boldsymbol{L}$ and she has the answer $\boldsymbol{x}'$
  - Step-2: V challenges P
  - Step-3: P shows the proof of the answer $\boldsymbol{x}'$, which will not leak $\boldsymbol{x}$
  - **V trusts that P has $\boldsymbol{x}$ when P always meets the challenge**

Graph Isomorphism

G_e G_f

- If $\mathbf{G_1 = (V_1, E_1)}$ and $\mathbf{G_2 = (V_2, E_2)}$ are isomorphic, there exist a bijection function $\boldsymbol{\phi}$, that for any $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E_1}$, exist $\boldsymbol{\phi}(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E_2}$

Graph Isomorphism



- If $\mathbf{G_1 = (V_1, E_1)}$ and $\mathbf{G_2 = (V_2, E_2)}$ are isomorphic, there exist a bijection function $\boldsymbol{\phi}$, that for any $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E_1}$, exist $\boldsymbol{\phi}(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E_2}$
- $\mathbf{G_e}$ and $\mathbf{G_f}$ are isomorphic
  - $\boldsymbol{\phi}$ = {1_1->2_5, 1_2->2_6, 1_3->2_4, 1_4->2_3, 1_5->2_1, 1_6->2_2}

Graph Isomorphism w/ Interactive ZKP

G_e and G_f are Isomorphism, $\emptyset = \ldots$

- Select a random bijection function $\boldsymbol{\pi}$
- Calculate $\boldsymbol{G'_f} \simeq \boldsymbol{G_f}$ with $\boldsymbol{\pi}$

$\boldsymbol{G'_f} \simeq \boldsymbol{G_f} \simeq \boldsymbol{G_e}$ →

← Ask Alice to prove $\boldsymbol{G_\sigma} \simeq \boldsymbol{G'_f}$

- Select a random $\boldsymbol{\sigma} \in \{\boldsymbol{e}, \boldsymbol{f}\}$

If $\boldsymbol{\sigma}$ =**f**, send $\boldsymbol{\pi}$
If $\boldsymbol{\sigma}$ = **e**, send $\boldsymbol{\pi} \cdot \emptyset$ →

- Verify that $\boldsymbol{G_\sigma} \simeq \boldsymbol{G'_f}$

Graph Isomorphism w/ Interactive ZKP

G_e and G_f are Isomorphism, $\emptyset = \ldots$

Alice

Bob

- Select a random bijection function $\boldsymbol{\pi}$
- Calculate $\boldsymbol{G'_f} \simeq \boldsymbol{G_f}$ with $\boldsymbol{\pi}$

$\boldsymbol{G'_f} \simeq \boldsymbol{G_f} \simeq \boldsymbol{G_e}$ →

← Ask Alice to prove $\boldsymbol{G_\sigma} \simeq \boldsymbol{G'_f}$

- Select a random $\boldsymbol{\sigma} \in \{\boldsymbol{e}, \boldsymbol{f}\}$

Require too much interactions!

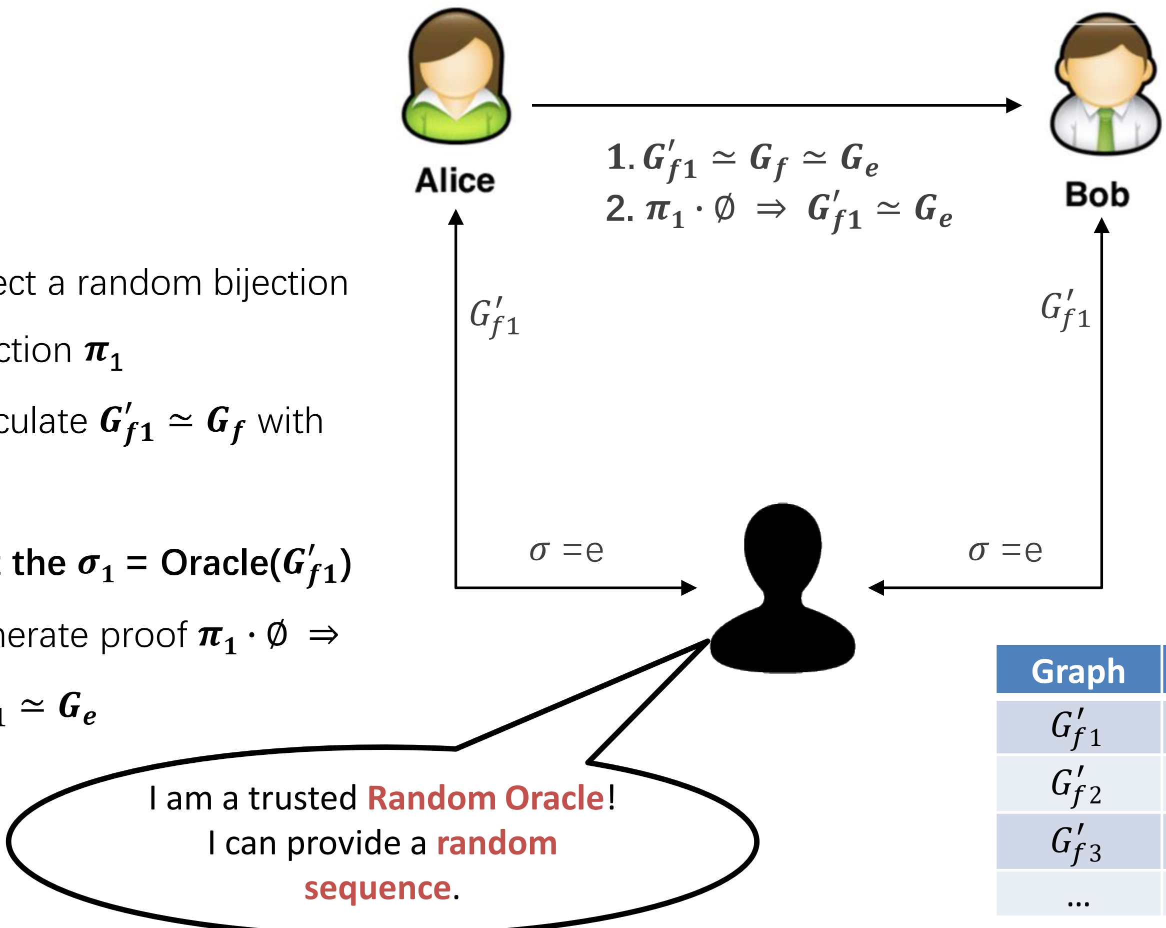
→

- Verify that $\boldsymbol{G_\sigma} \simeq \boldsymbol{G'_f}$

Non-Interactive ZKP

Proof

- Select a random bijection function $\boldsymbol{\pi_1}$
- Calculate $\boldsymbol{G'_{f1} \simeq G_f}$ with $\boldsymbol{\pi_1}$
- **Get the $\boldsymbol{\sigma_1}$ = Oracle($\boldsymbol{G'_{f1}}$)**
- Generate proof $\boldsymbol{\pi_1} \cdot \emptyset \Rightarrow \boldsymbol{G'_{f1} \simeq G_e}$



Verify

- **Get the $\boldsymbol{\sigma_1}$ = Oracle($\boldsymbol{G'_{f1}}$)**
- **Verify $\boldsymbol{G'_{f1} \simeq G_e}$**

Graph	Value
G'_{f1}	σ =e
G'_{f2}	σ =f
G'_{f3}	σ =e
…	…

Oblivious Transfer

OT

Problem

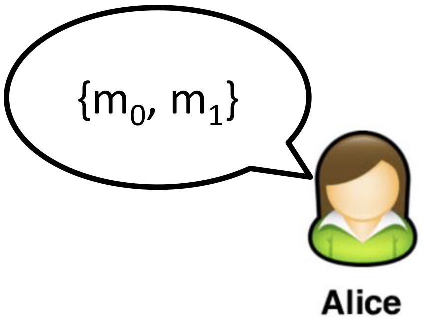
- Alice has **{m_0, m_1}** and Bob wants to get **m_σ**
    - Alice may know the m_σ
    - Bob may get both **m_0** and **m_1**

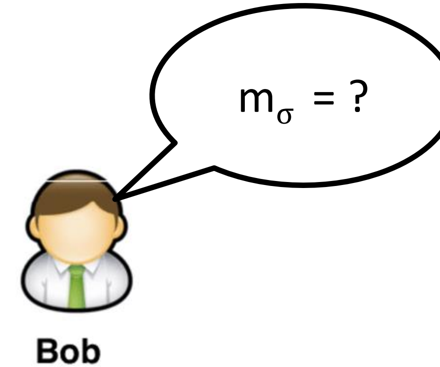
Oblivious Transfer

- Scenario: message transfer
  - A sender has a message list $\{m_0, m_1, \cdots, m_n\}$
  - A receiver wants to get **k** target messages from sender

- Properties: oblivious and secure
  - Oblivious: sender **cannot know which messages are received**
  - Secure: receiver can **only get the target messages**

1-out-of-2 OT





Alice:

- Generate $(\boldsymbol{K_{pub}}, \boldsymbol{K_{prv}})$
- Select random numbers $\boldsymbol{r_0}, \boldsymbol{r_1}$

$\xrightarrow{\boldsymbol{K_{pub}}, \boldsymbol{r_0}, \boldsymbol{r_1}}$

Bob:

- Generate a key $\boldsymbol{k}$
- $\mathbf{V} = \boldsymbol{Enc}(\boldsymbol{K_{pub}}, \boldsymbol{k}) \oplus \boldsymbol{r_\sigma}$, $\boldsymbol{\sigma} \in \{\mathbf{0}, \mathbf{1}\}$

$\xleftarrow{\mathbf{V}}$

Alice:

- $\boldsymbol{k_0} = \boldsymbol{Dec}(\boldsymbol{K_{prv}}, \boldsymbol{V} \oplus \boldsymbol{r_0})$
- $\boldsymbol{k_1} = \boldsymbol{Dec}(\boldsymbol{K_{prv}}, \boldsymbol{V} \oplus \boldsymbol{r_1})$
- $\boldsymbol{C_0} = \boldsymbol{Enc}(\boldsymbol{k_o}, \boldsymbol{m_0})$
- $\boldsymbol{C_1} = \boldsymbol{Enc}(\boldsymbol{k_1}, \boldsymbol{m_1})$

$\xrightarrow{\boldsymbol{C_o}, \boldsymbol{C_1}}$

Bob:

- $\boldsymbol{m_\sigma} = \boldsymbol{Dec}(\boldsymbol{k}, \boldsymbol{C_\sigma})$

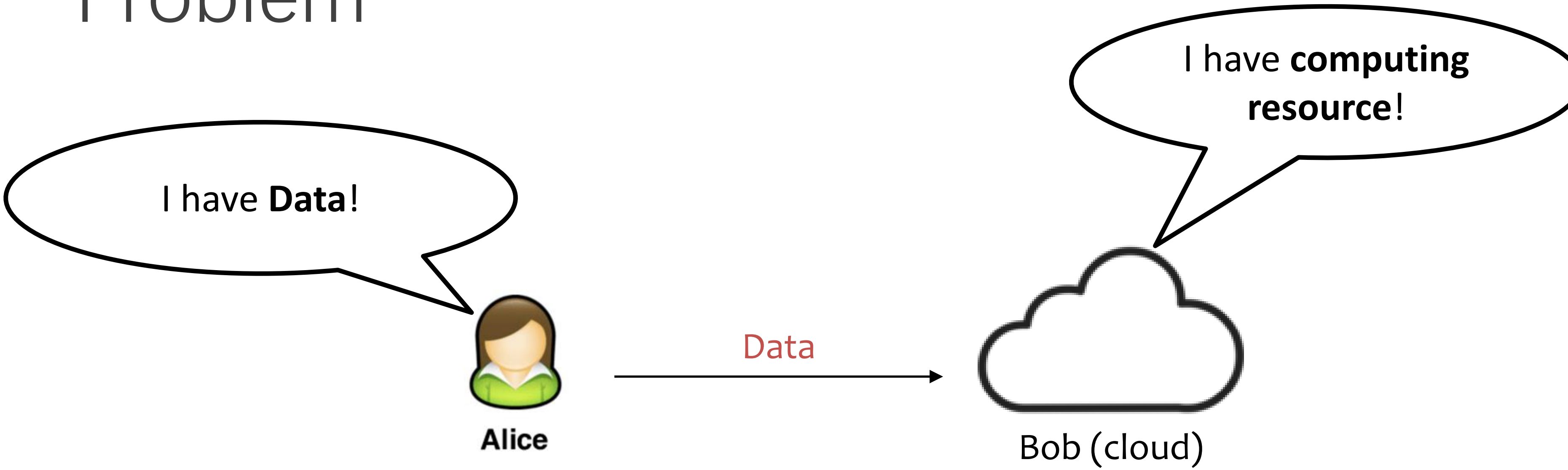
More OT Protocols

- Different numbers of selected messages
  - 1-out-of-2 OT:
  - 1-out-of-n OT
  - k-out-of-n OT
- Implementation method
  - Non-adaptive OT
  - Adaptive OT
  - Publicly Verifiable OT
  - ...

Homomorphic Encryption

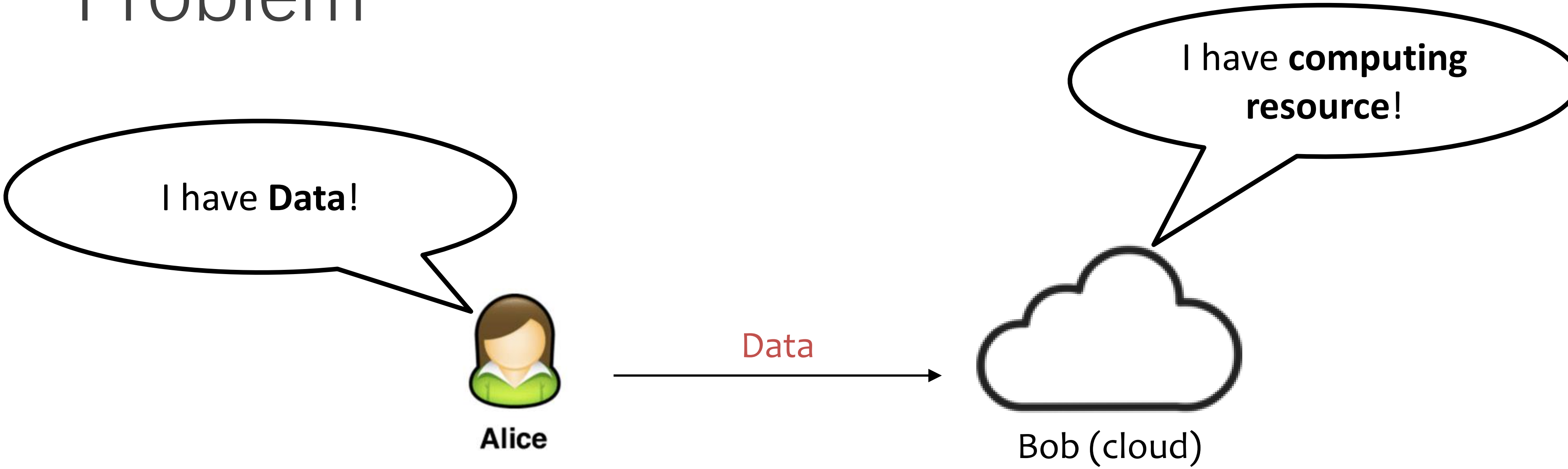
HE

Problem



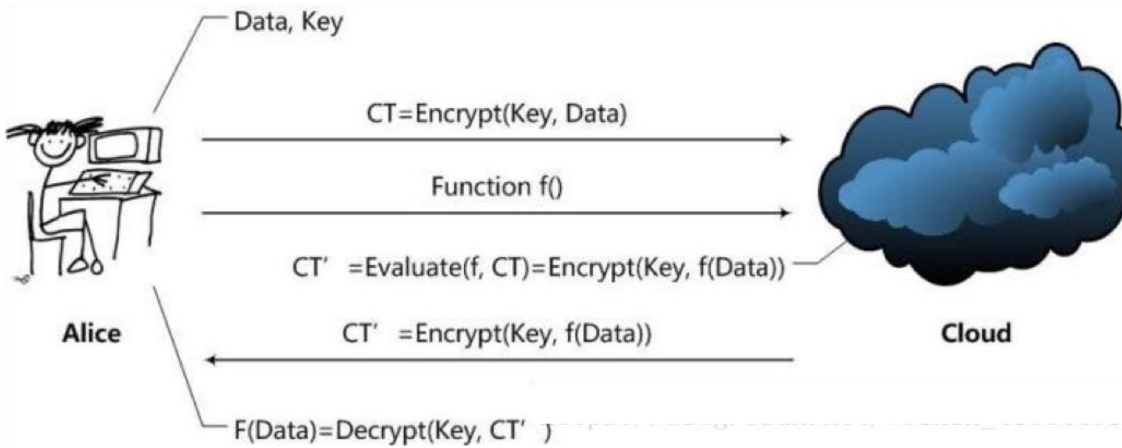
- Alice wants to ask Bob (e.g., a cloud) to perform calculation on her data
- Naïve method: Sending data to Bob

Problem



- Alice wants to ask Bob (e.g., a cloud) to perform calculation on her data
- Naïve method: Sending Data to Bob
  - Bob will get the data

Homomorphic Encryption



- Alice sends ***CT=encrypt(Key, Data)*** and function ***f*** to the Cloud
- Cloud calculates ***CT'= Evaluate(f, CT) = Encrypt(Key, f(Data))***
- Alice gets ***f(Data) = Decrypt(Key, CT')***

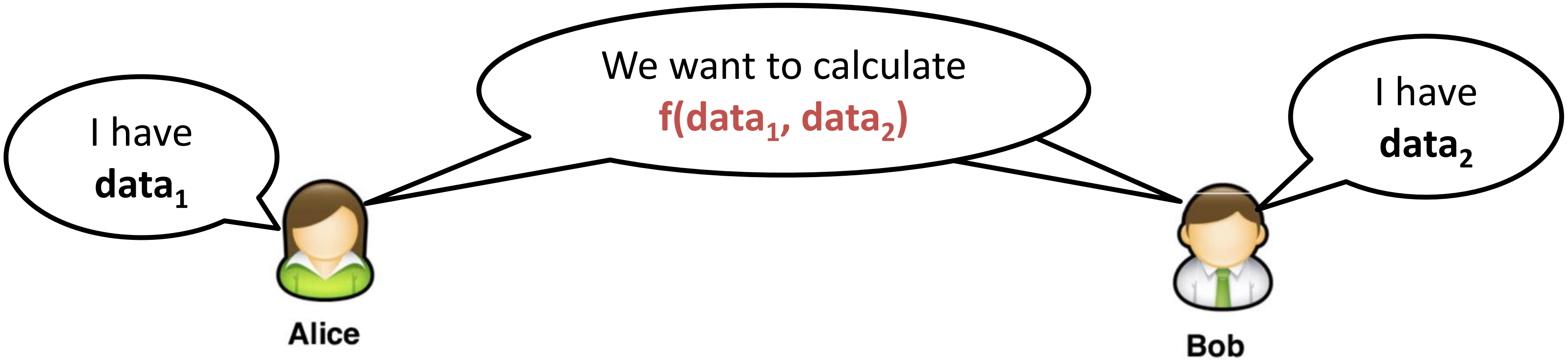
SWHE and FHE

- HE: Homomorphic Encryption
  - $\boldsymbol{Enc\big(f(m_1, m_2)\big) = Eval_f\big(Enc(m_1), Enc(m_2)\big)}$
- SWHE: Somewhat Homomorphic Encryption
  - Support **limited** kinds of operation
  - $\boldsymbol{f(m_1, m_2) = m_1 \cdot m_2}$ (e.g., RSA)
  - $\boldsymbol{f(m_1, m_2) = m_1 + m_2}$
- FHE: Full Homomorphic Encryption
  - Support all kinds of operations
  - Addition and multiplication

Secure Multi-Party Computing

SMPC

Problem



- Multiple parties (at least 2) work together to calculate a function
  - Enforce the **data privacy** for each party

Yao's Protocol

- Two-party computing
- Semi-honest adversary
  - Each party must **follow the protocol**
- Generic protocol
  - Can securely compute **any functionality**
- GC(Garbled Circuits)+OT(Oblivious Transfer)

Millionaire Problem

- Money: Wang has i, Lee has j, i and j are between 1 to 10
- Lee (money: j)
  - Chose a random big integer x
  - K = enc{Key_{pub} of Wang, x}
  - Send c=K-j to Wang
- Wang (money: i)
  - Decrypt with Key_{pri} of Wang ten number: c+1, c+2…c+10, get y1, y2… y10
  - Chose a prime number p
  - Calculate d1 = y1 mod p
  - For n=i to 10, d_n++, other no change
  - Send d1 to d10 to Lee
- Lee
  - Check dj, if dj == x mod p, then i >= j; else i < j

Garbled Circuits

- Represent functions as **Boolean circuits**
  - Basic gates: AND, OR, NOT
  - Adding numbers
  - Comparing numbers
  - Multiplying numbers
  - Computing AES
- Represent input and output as wires

Garbled Circuits

- An encrypted circuits together with a pair of keys **(k_0,k_1)** for every wire so that for any gate, given one key for every input wire:
  - It is possible to compute the key of the corresponding gate output
  - It is **impossible to learn anything else**

Yao's Protocol on GC

- **Input**: **x** and **y** of length **n** from P_1 and P_2
- P_1 generates a garbled circuit **G(C)**
  - $\mathbf{K_L^0, K_L^1}$ are the keys on wire $\mathbf{w_L}$
  - Let $\mathbf{w_1, \cdots w_n}$ be the input wires of P_1 and $\mathbf{w_{n+1}, \cdots w_{2n}}$ be the input wires of P_2
- P_1 sends to P_2 **G(C)** and strings $\boldsymbol{K_1^{x_1}, \ldots, K_n^{x_n}}$
- P_1 and P_2 run **n** OTs in parallel
  - P_1 inputs $\boldsymbol{(K_{n+1}^0, K_{n+1}^1)}$
  - P_2 inputs $\mathbf{y_i}$
- Given all keys, P2 computes **G(C)** and obtains **C(x,y)**

Different sMPC Protocols

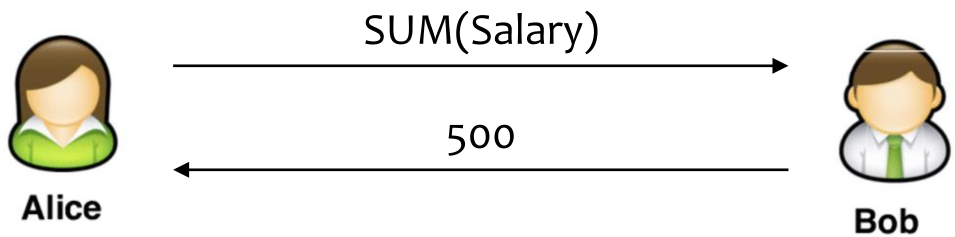
- Two-party
  - Yao's protocol
  - TinyOT protocol
  - Obliv-C

- Multi-party
  - BMR protocol
  - GMW protocol
  - SPDZ protocol

Differential privacy

DP

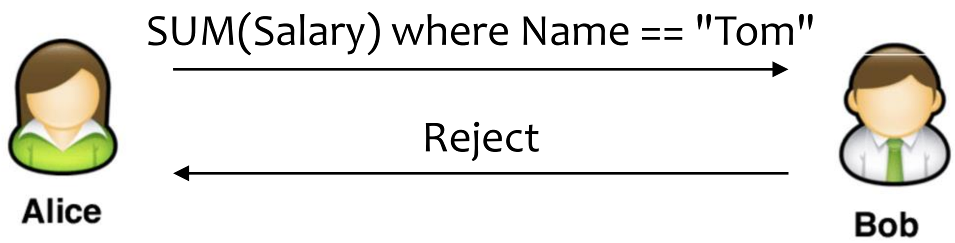
Problem



Name	Salary
Alice	100
Bob	80
Brown	200
Tom	120

- Alice can perform queries on Bob's database, but cannot access a single database entry

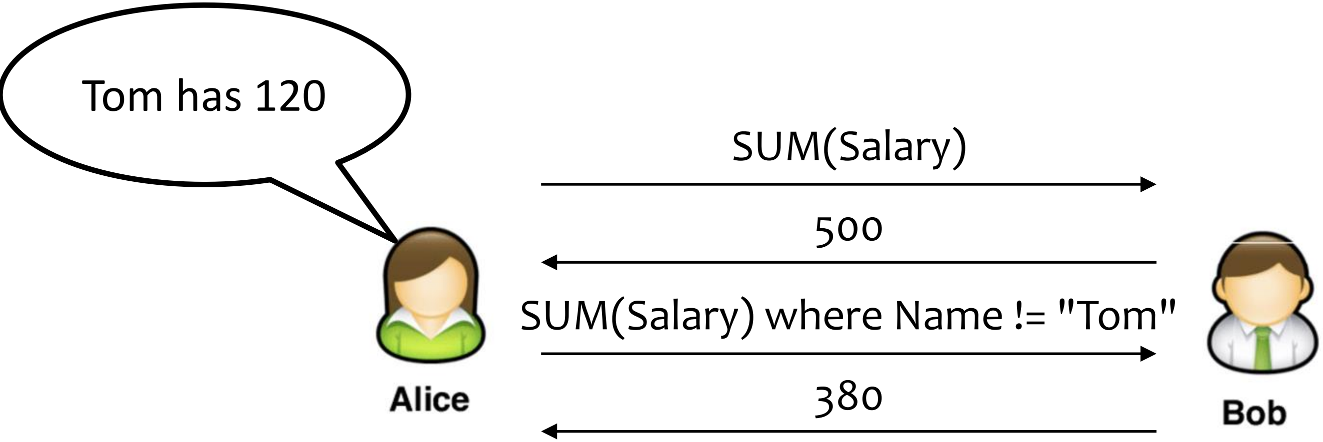
Problem



Name	Salary
Alice	100
Bob	80
Brown	200
Tom	120

- Alice can perform queries on Bob's database, but cannot access a single database entry
  - Naïve method: reject Alice to access single entry

Problem



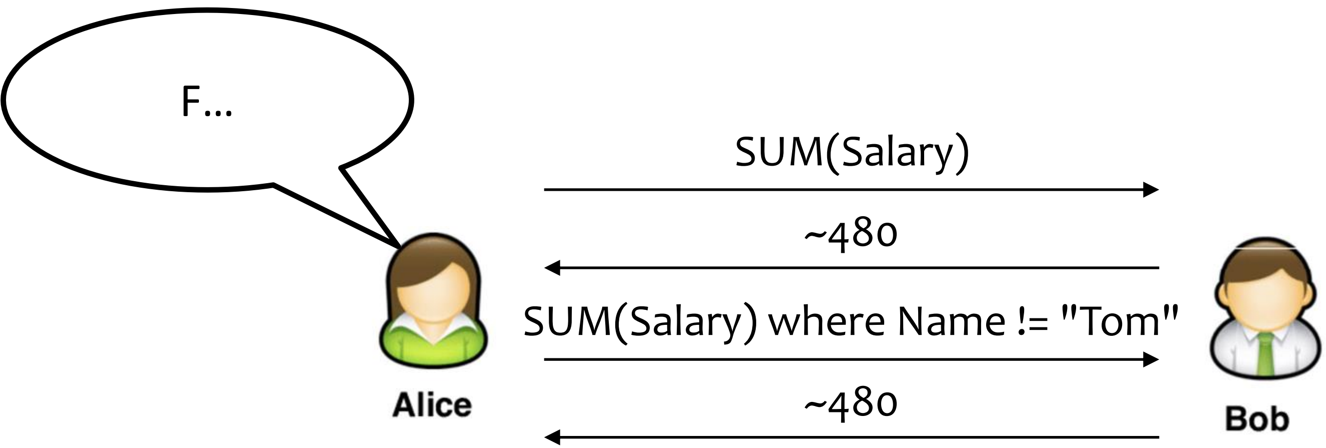
Name	Salary
Alice	100
Bob	80
Brown	200
Tom	120

- Alice can perform queries on Bob's database, but cannot access a single database entry
  - Naïve method: reject Alice to access single entry

Differential privacy

- Allow user to perform a **random function M** on data set $\mathbf{D = \{a_1, \cdots a_n\}}$, but get nothing about any individual entry of **D**
- **M** is $\boldsymbol{\varepsilon}$-DP if:
  - For all datasets $\mathbf{D}$ and $\mathbf{D'}$ that **differ on a single element**, $\mathbf{Pr}(\boldsymbol{M}(\boldsymbol{D}) = \boldsymbol{x}) \le \boldsymbol{e}^{\boldsymbol{\varepsilon}} * \mathbf{Pr}(\boldsymbol{M}(\boldsymbol{D'}) = \boldsymbol{x})$
- Security properties
  - Robustness to post-processing
    - **User can perform any operation on the result of M, and get nothing about the individual entry of D**
  - Composability
  - Group privacy

When DP is Enabled



Name	Salary
Alice	100
Bob	80
Brown	200
Tom	120

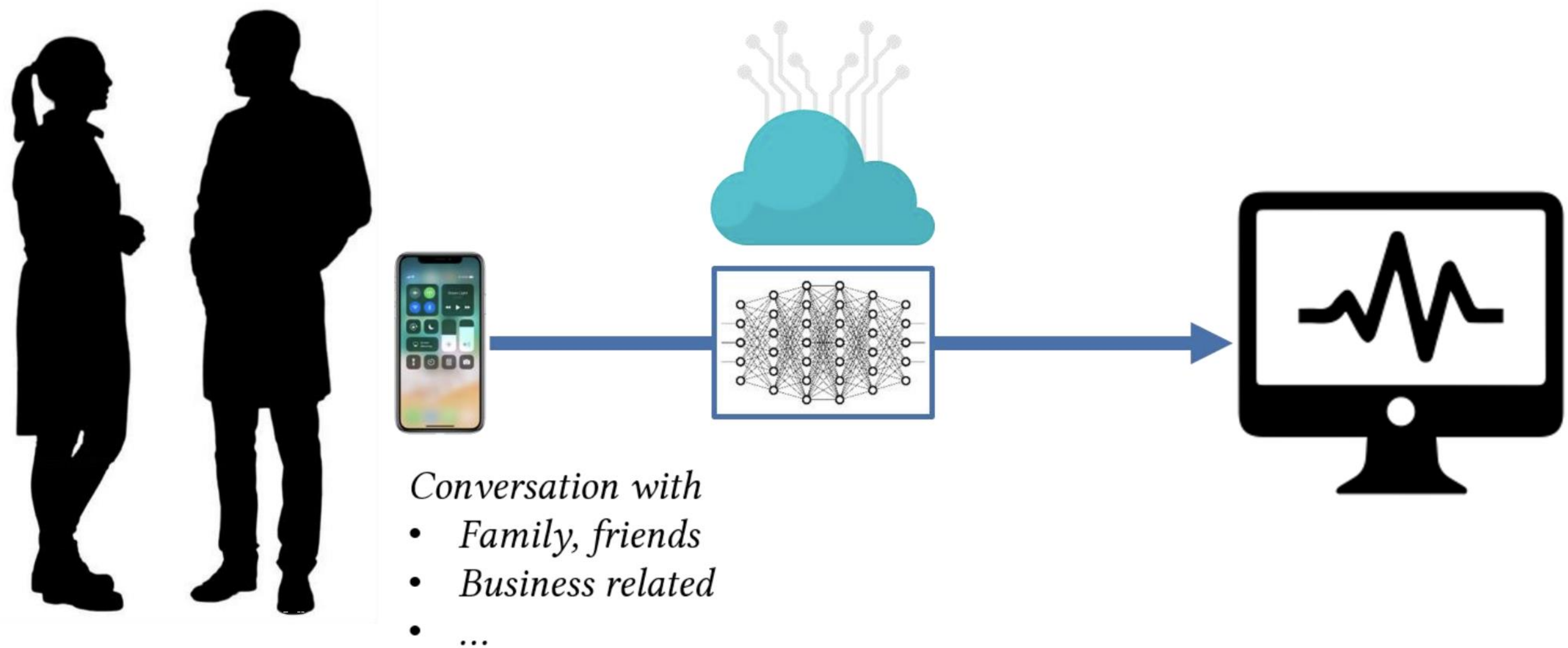
How to Implement a DP Algorithm

- Adding noisy to the function that we want to compute
  - Translate the function **f** to a random algorithm **M**
- Existing Mechanism
  - Laplace mechanism
  - Gaussian mechanism
  - …

Occlumency: Privacy-preserving Remote Deep-learning Inference Using SGX (MobiCom’2019)

OCCLUMENCY

Concerns in Cloud-driven Deep Learning



Sensitive data is exposed to leak / tampering

Cloud Offloading is Inevitable, but Risky

- Practical method to support mobile DL
  - Easily supports a large model with high accuracy
  - Consumes less resources
  - Addresses device heterogeneity
- Privacy concerns!
  - User data can be disclosed
  - Image, video, audio, activities, health/medical data
- More critical for mobile/IoT services
  - Life-immersive: data from users' daily life

Cloud Offloading is Inevitable, but Risky

Cloud

- High computational power
- Sacrifices privacy

On-device

- High privacy protection
- Sacrifices accuracy/speed

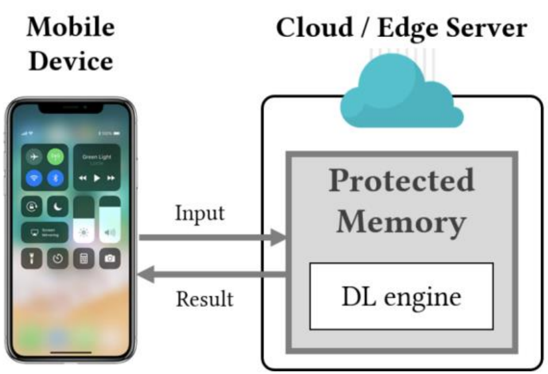
This paper aims to build a **secure cloud**-based solution

to strike the balance between privacy, speed and accuracy

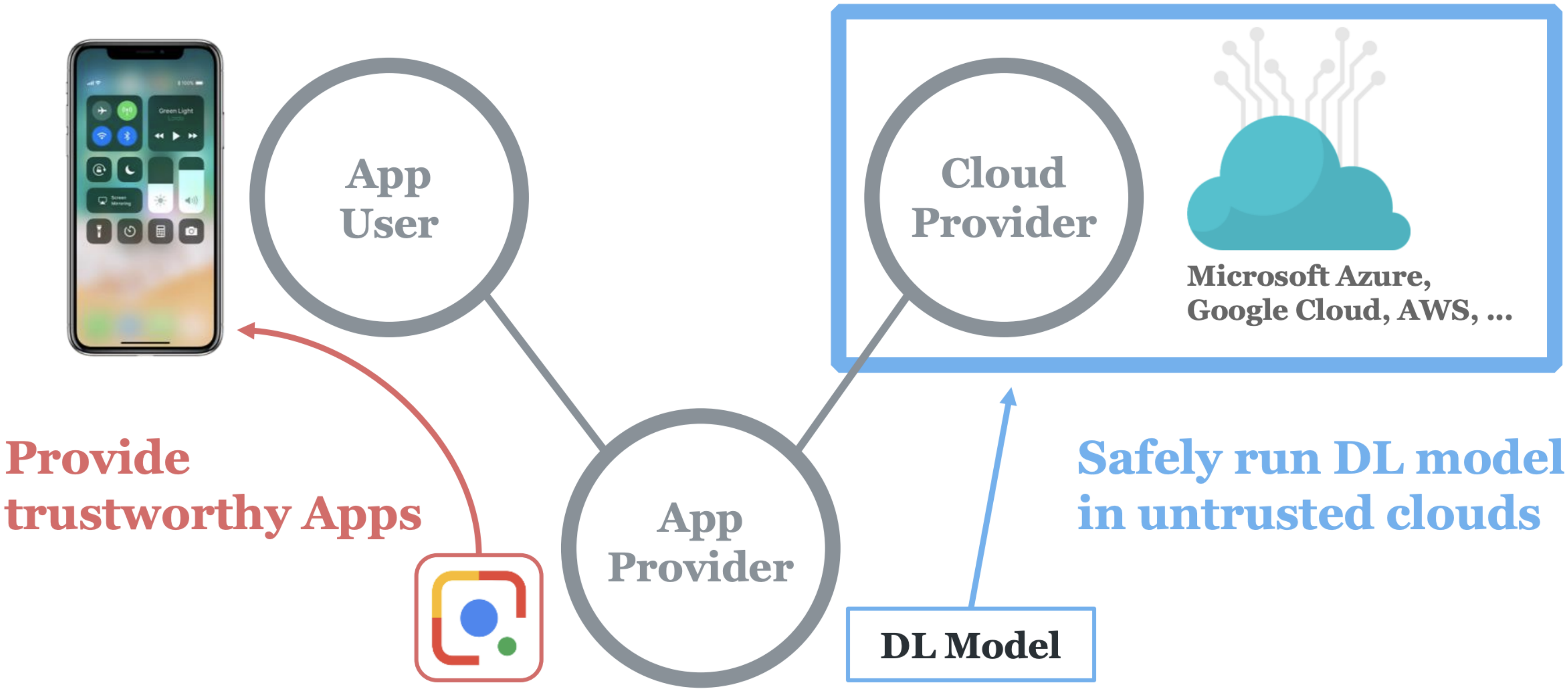
Occlumency

A cloud-driven DL inference system preserving user privacy

- Key approach: SGX enclaves
  - Commodity TEE with the highest protection level
  - Prevent memory access even from OS / hypervisor
- Protects:
  - User data disclosure
  - Inference result manipulation

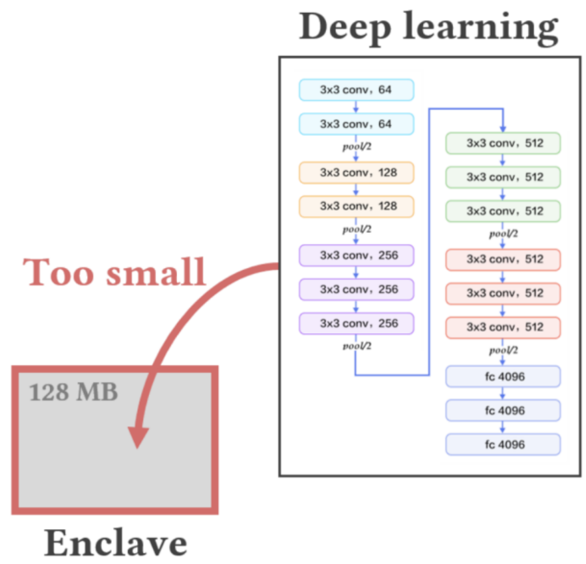


Stakeholders of Occlumency



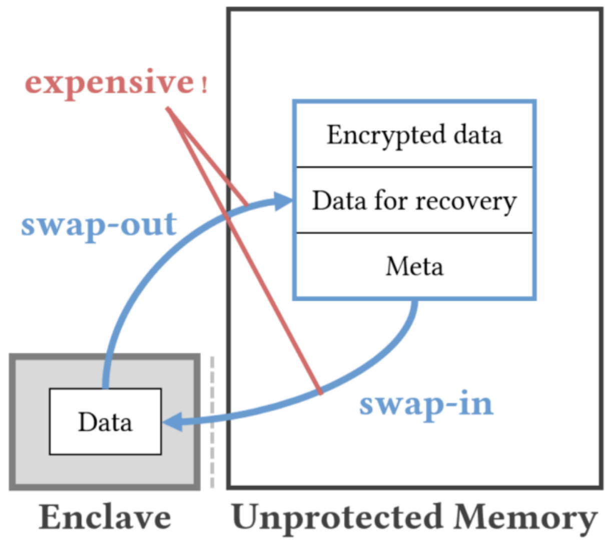
Challenge: Limited Memory Size

- SGX's physical memory is very small (128 MB)
  - DNN requires 100MB ~ 1GB of memory
  - Windows -- fails
  - Linux -- makes frequent page swapping



Challenge: Limited Memory Size

- SGX's physical memory is very small (128 MB)
  - DNN requires 100MB ~ 1GB of memory DNN
    ⇒ Frequent page swapping
- SGX's paging is expensive
  - Swaps-out memory into untrusted memory
  - Involves encryption, redundancy checking, …
- SGX slows down the inference speed
  - Takes **7x longer** latency to infer VGG, YOLO



Enabling DL to Run within Small Memory

Observation -- three dominant memory usages of DNN:

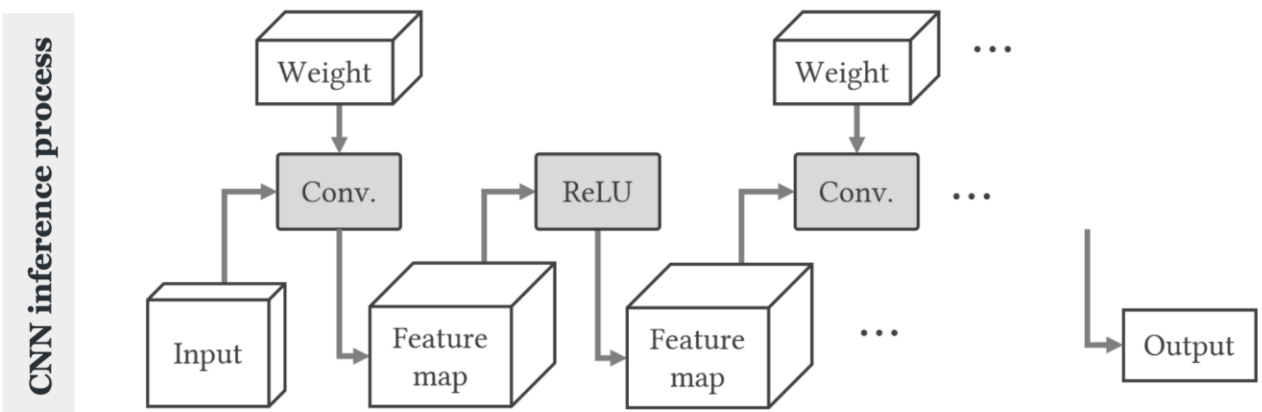
1) Model weights (parameters)

2) Intermediate feature maps

3) Conv. layer computation

Memory Usage of VGG-16

1) Model weights	540 MB
2) Feature maps	61 MB
3) Conv. layer	327 MB
Others	6 MB



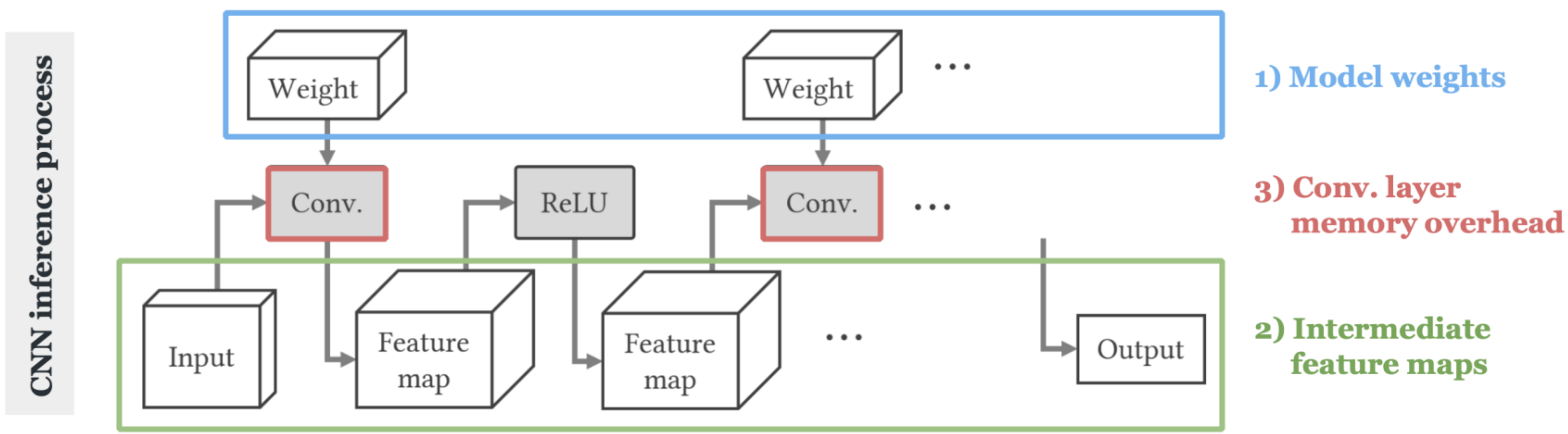
Enabling DL to Run within Small Memory

Observation -- three dominant memory usages of DNN:

1) Model weights (parameters)

2) Intermediate feature maps

3) Conv. layer computation



Approach of Occlumency

1) Model weights (parameters)

2) Intermediate feature maps

3) Conv. layer computation

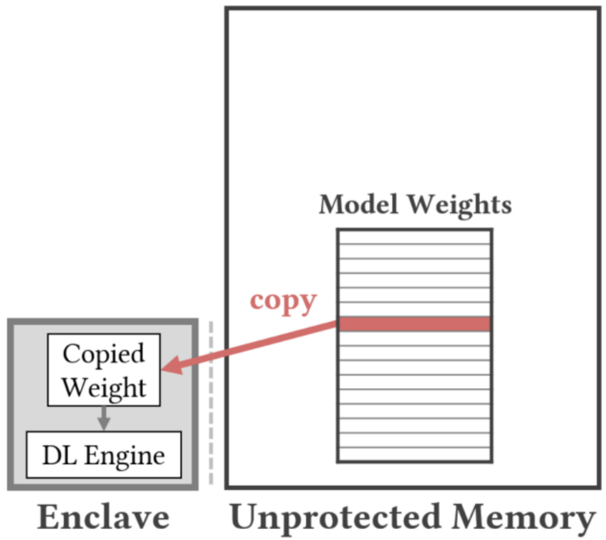
1) On-demand weight loading

2) Memory-efficient FM allocation

3) Partitioned convolution

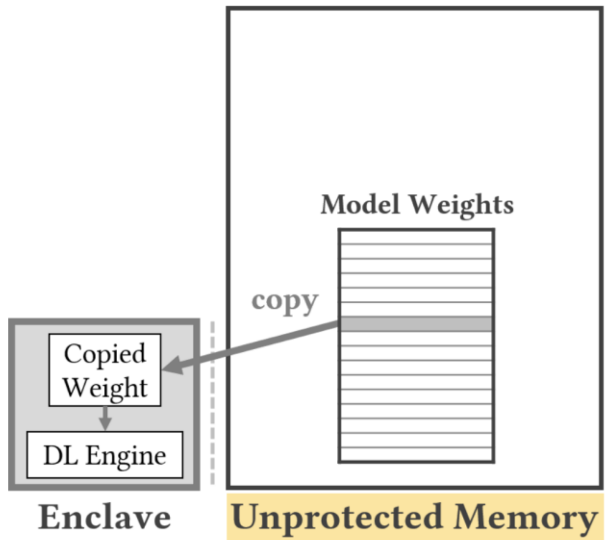
1) On-demand Weights Loading

- Saves memory used to load weights (~500MB for VGG-16)
- Idea: Not protecting model weights
  - Our goal is to protect the user privacy
  - Model weights are irrelevant to user data
- Keeps weight in unprotected memory & copies into enclaves on demand



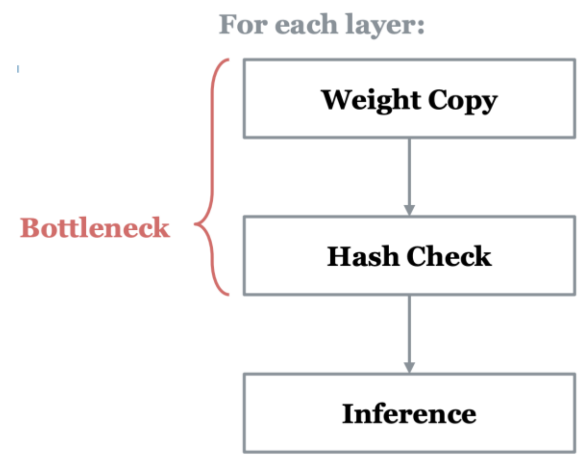
1-1) New Problem 1: Weight Corruption Threat

- Weight may be corrupted
  - Weights are no longer protected in enclaves
  - Weight manipulation attacks will lead to wrong inference results
- Solution: Model integrity checking
  - Hash checking-based weight modification detection
  - Compares hash value of each layer



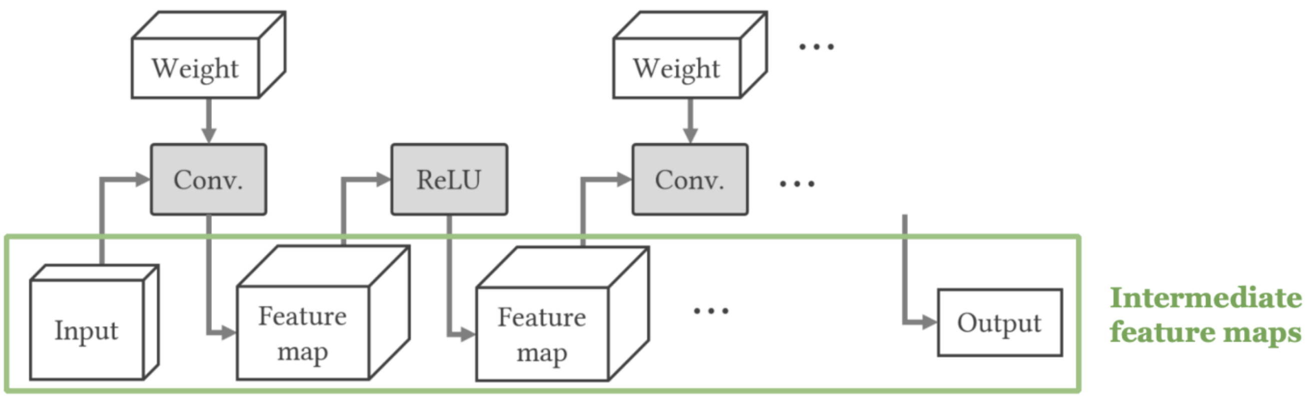
1-2) New Problem 2: Computation Bottleneck

- Additional computation bottlenecks:
  - Weight copying (on-demand weight loading)
  - Hash checking (model integrity checking)
- Parallel pipeline
  - Weight copy / Hash check / Inference
  - Reduces ~17.5% of latency



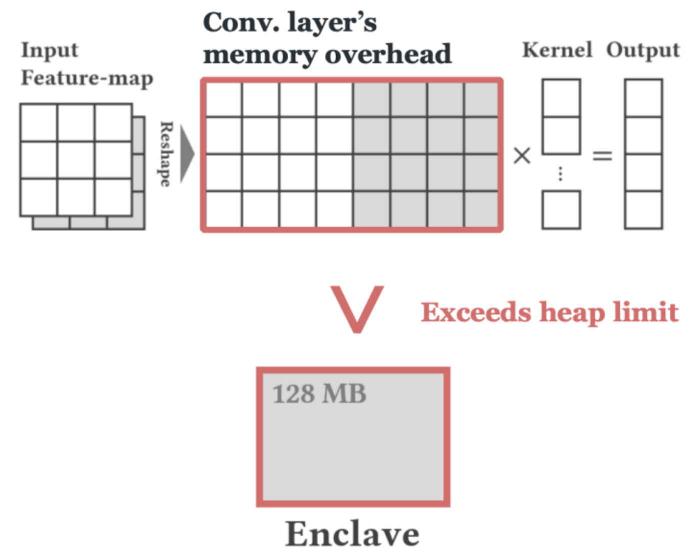
2) Memory-efficient Feature Map Allocation

- Reduces the required memory to load intermediate feature-maps (FM)
- Idea: Releasing unnecessary FMs
  - Profiles when each FM can be released in advance
  - Immediately deallocates FMs already used



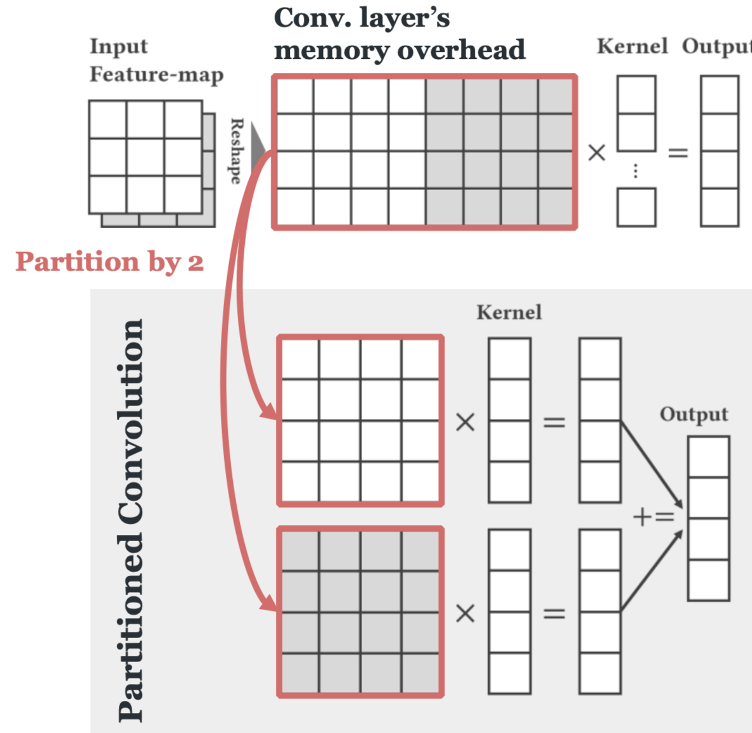
3) Partitioned Convolution

- Reduces memory overhead of Conv. that exceeds the SGX heap limit



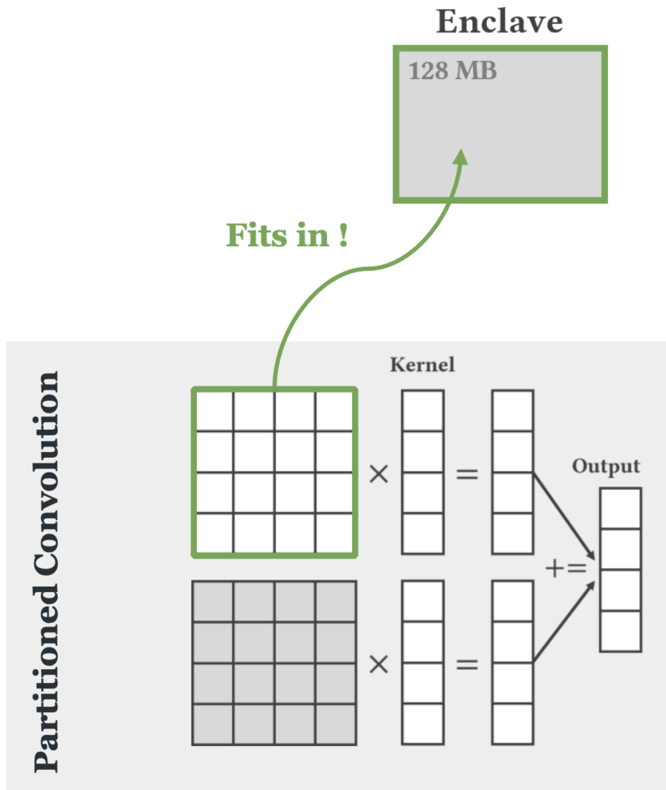
3) Partitioned Convolution

- Reduces memory overhead of Conv. that exceeds the SGX heap limit
- Idea: Breaking down big operations into smaller jobs
  - e.g., Partition by 2 → requires 2x less memory



3) Partitioned Convolution

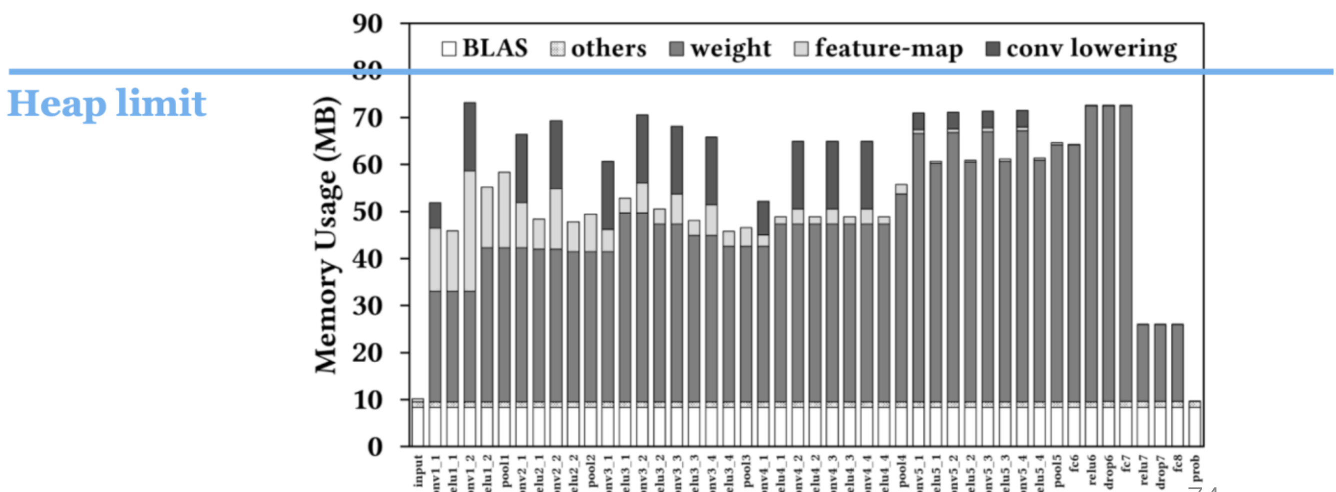
- Reduces memory overhead of Conv. that exceeds the SGX heap limit
- Idea: Breaking down big operations into smaller jobs
  - e.g., Partition by 2 → requires 2x less memory
- Adaptively partitions by 2, 4, 8, ⋯
  - Runs Conv. layer within limited memory size



Memory Usage Evaluation

1) Can Occlumency run DNN within SGX's memory limit?

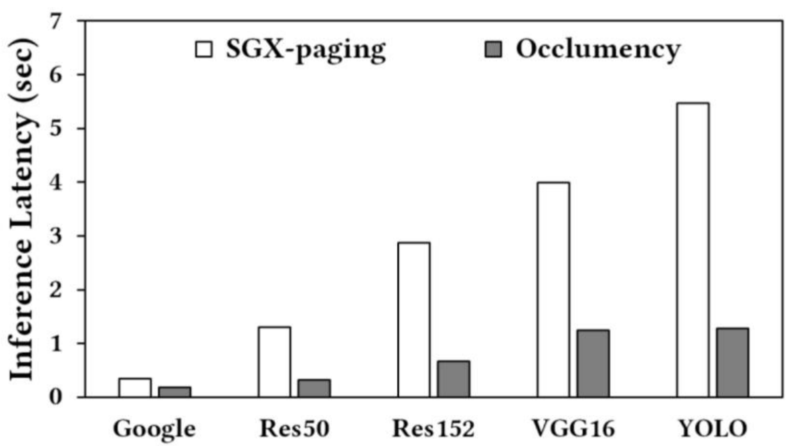
- Occlumency successfully runs DNN models within SGX's heap limit
- Ex) Reduced memory for VGG-19:
  - Original: 980MB
  - Occlumency: 74MB



Inference Latency Evaluation

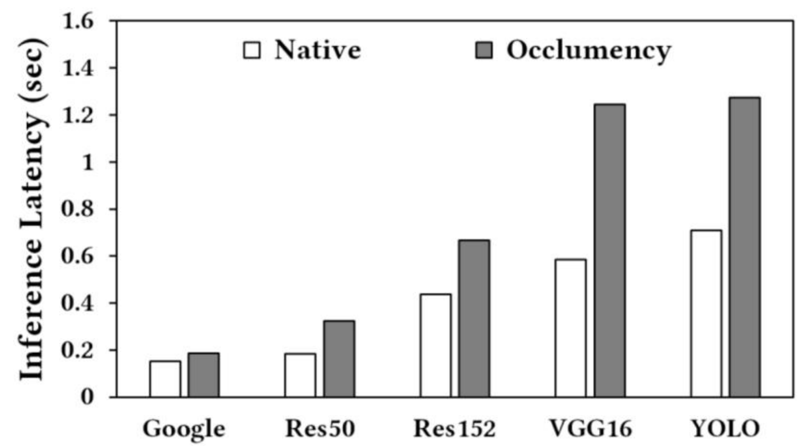
2) How much the Occlumency can enhance the inference speed?

- 3.0 ~ 4.3x faster than **SGX-paging**



3) How much is the overhead of Occlumency?

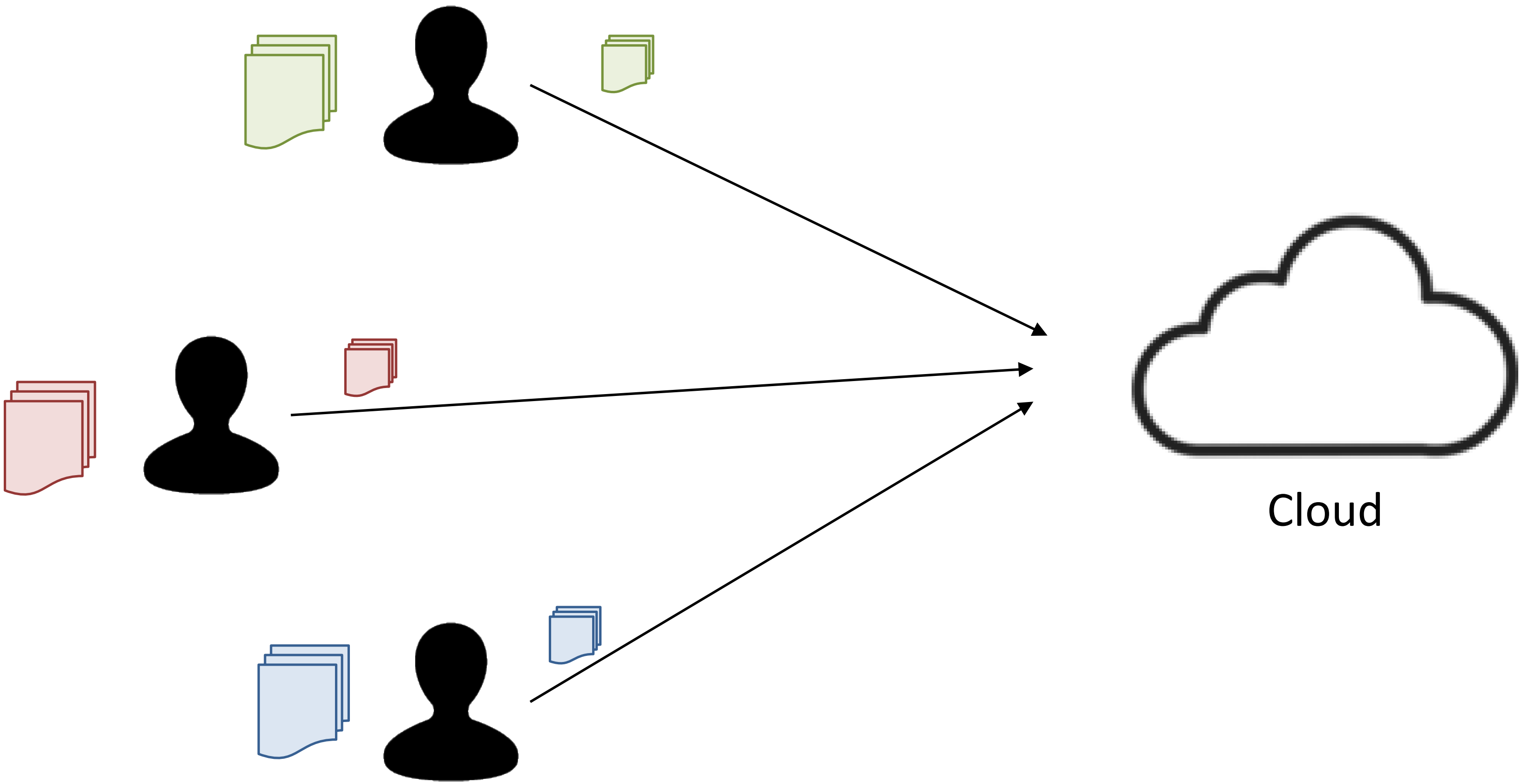
– 72% overhead compared to **Native**



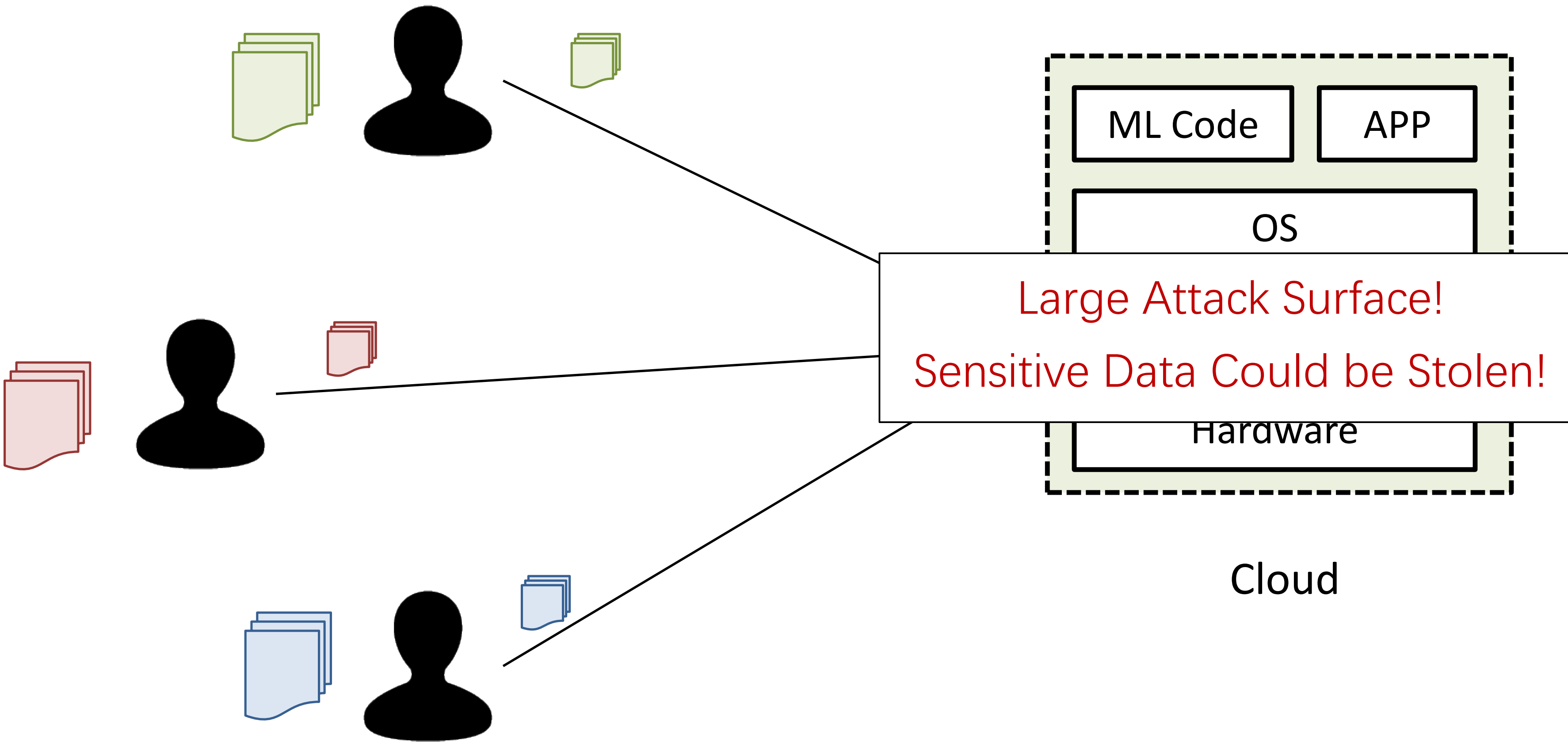
Oblivious Multi-Party Machine Learning on Trusted Processors
(USENIX Security'2016)

OBLIVIOUS MULTI-PARTY MACHINE

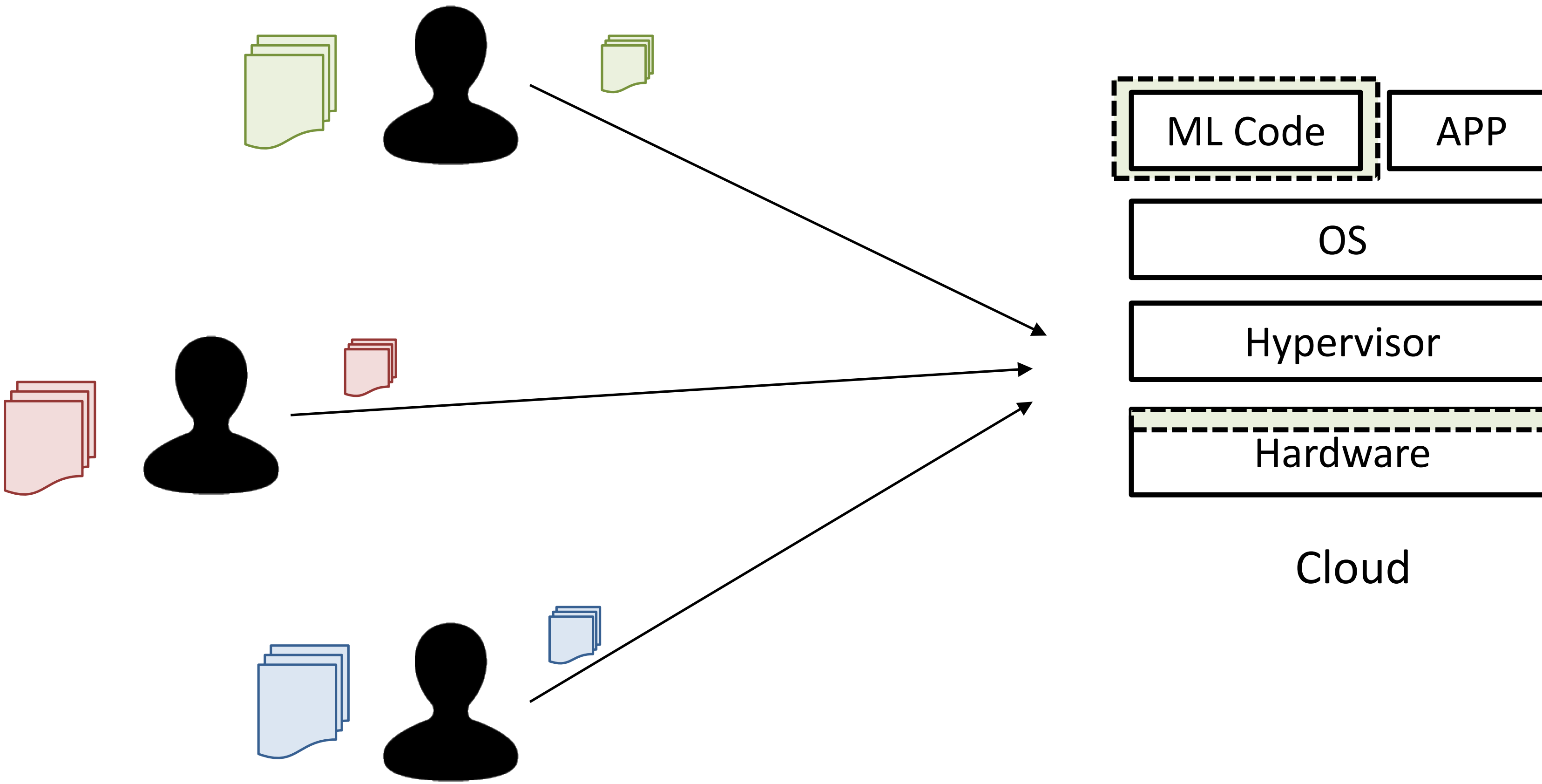
Machine Learning on Cloud



Machine Learning on Cloud



Machine Learning on Cloud with TEE

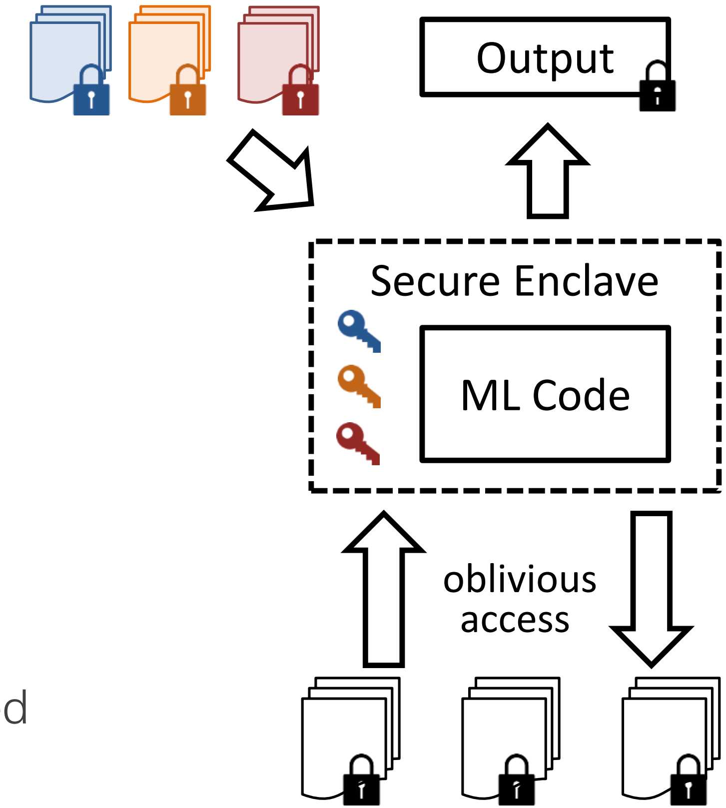


Threat Model

- Any party may be malicious
- The cloud can be malicious
  - **Memory & Network observer**
  - Hardware attackers (on mem bus)
  - Perform **side-channel attacks**
- Assumptions:
  - Code does not leak secrets
  - Do not consider leakage through time or power channels

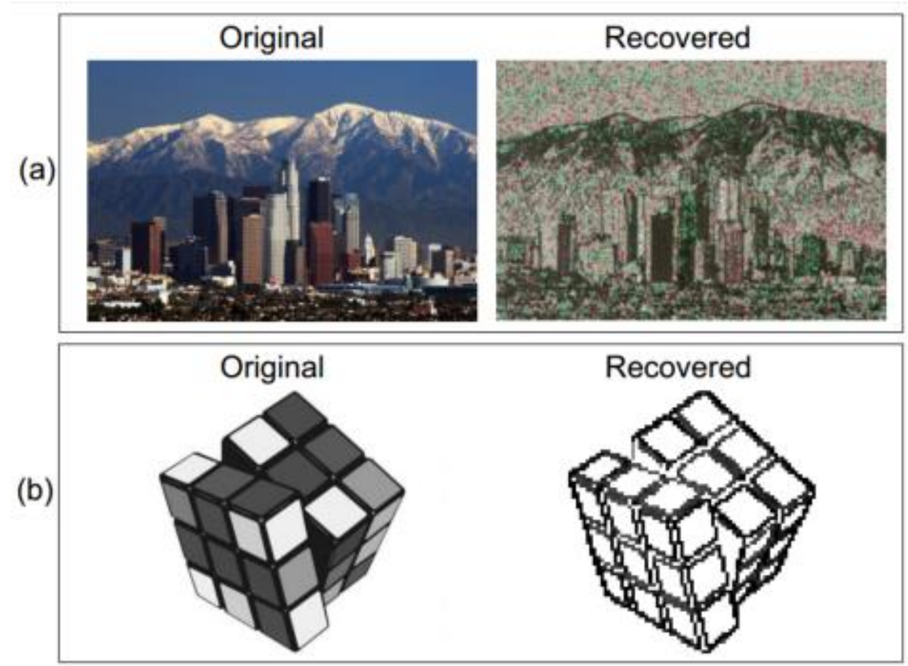
System Design

- Data: encryption
  - Input and output are encrypted
  - Data outside of enclave is encrypted
- Code: secure enclave
  - Trusted processors (SGX)
- Data accesses:
  - Side channel protection
  - Memory, disk, and network are accessed obliviously



SGX's Vulnerabilities

- Side channel attack
  - Malicious OS still controls page fault handler
  - If know the photo processing algorithm, can get the image by monitoring page fault
  - Not 100% accurate, but still good enough



Side-channel Protection

- Memory side-channel
- Security guarantee:
  - **Data oblivious**
  - Given two inputs and a memory trace, one cannot distinguish which one was executed
- Memory accesses only depends on public information
  - E.g., number of instances, number of labels
- Assumption: register-to-register manipulation is data oblivious

Library of Oblivious Primitives LibO

- In assembly:
  - ogreater, omove, oless, oequal
  - oget
    - get the ith array element (hide i)

ogreater()

```
mov     rcx, x
mov     rdx, y
cmp     rcx, rdx
setg    al
retn
```

omove()

```
mov     rcx, cond
mov     rdx, x
mov     rax, y
test    rcx, rcx
cmovz   rax, rdx
retn
```

Oblivious Operation

Non-oblivious

```
int max(int x, int y) {
  if (x > y) return x;
  else return y;
}
```

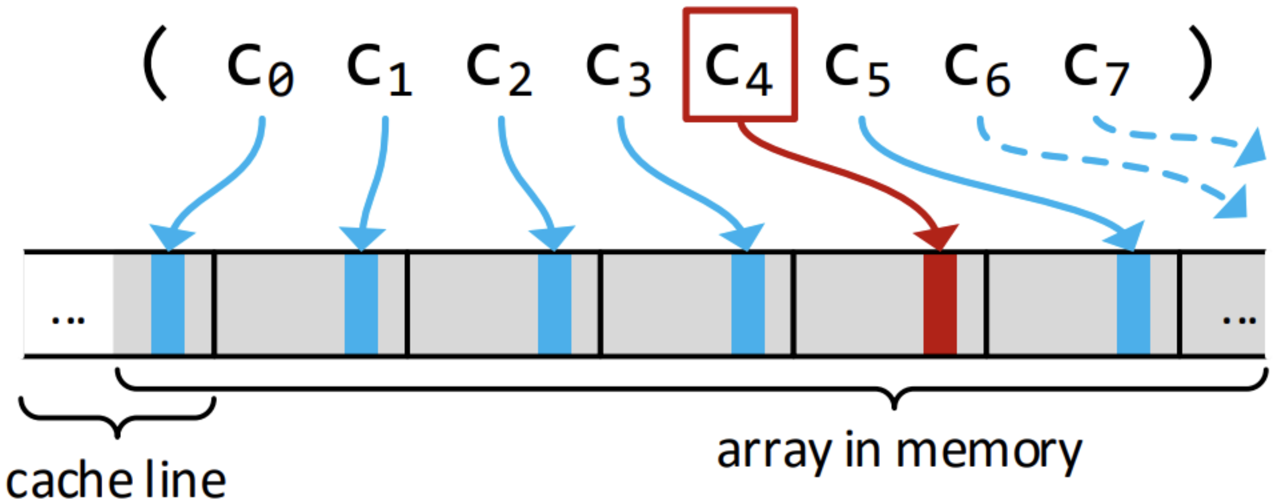
Oblivious

```
int max(int x, int y) {
  bool getX = ogreater(x, y);
  return omove(getX, x, y);
}
```

Optimized Oblivious Array Access

- Naïve method
  - Scan the array
  - Just actually load/store a single element
- Observation
  - Attacker can only trace memory access at cache line granularity
- Optimization
  - Scan arrays at cache line granularity
  - Leveraging AVX2 vector instructions

Optimized Oblivious Array Access



Other Oblivious ML Algorithms

- Decision trees
- Support Vector Machines
- Neural Network
- Matrix Factorization
- K-Means clustering

Evaluation

Algorithm	SGX+enc.	SGX+enc.+obl.	Dataset	Parameters	Input size	# Instances
K-Means	1.91	2.99	MNIST	k=10, d=784	128MB	70K
CNN	1.01	1.03				
SVM	1.07	1.08	SUSY	k=2, d=18	307MB	2.25M
Matrix fact.	1.07	115.00	MovieLens	n=943, m=1,682	2MB	100K
Decision trees	1.22	31.10	Nursery	k=5, d=27	358KB	6.4K

Baseline is processing the data in plaintext without SGX protection.

Different Data Privacy Systems

- Data privacy + ML
  - Sage (SOSP'19)
  - Oblivious multi-party ML (USENIX Security'16)
  - Chiron, …
- Data privacy + Database
  - CryptDB (SOSP'11)
  - EnclaveDB (S&P'18), …
- Data privacy + Data analysis
  - Opaque (NDSI'17)
- More …