Gemini PhD Seminar

Reconciling Data Privacy and Utility via Attested Smart Contract Execution

Dat Tien Le (dattl@ifi.uio.no), Frank Eliassen (frank@ifi.uio.no)
Outlines

1. Motivation
2. System goals and system designs
3. Challenges and solutions
Era of IoT Data

Traffic patterns for different M2M/IoT applications.

Era of IoT Data

Information from the Internet of Things:
We have gone beyond the decimal system

Today data scientist uses **Yottabytes** to describe how much government data the NSA or FBI have on people altogether.

In the near future, **Brontobyte** will be the measurement to describe the type of sensor data that will be generated from the IoT (Internet of Things)

**Yottabyte**
This is our digital universe today
= 250 trillion of DVDs

**Exabyte**
This will be our digital universe tomorrow...

**Zettabyte**
1.3 ZB of network traffic by 2016.

1 EB of data is created on the internet each day = 250 million DVDs worth of information.

The proposed **Square Kilometer Array telescope** will generate an EB of data per day.
A subject-controlled data market

In the future data will be owned and controlled by the subjects of the data (citizens, other stakeholders)

• New regulations (GDPR)
• New technologies (such as DLT/blockchain)
Can centralized solution fit for future?

https://www.databreachtoday.com/fitbit-hack-what-are-lessons-a-8793
https://www.cnet.com/how-to/150-million-myfitnesspal-accounts-were-hacked-heres-what-to-do/
System goals

• Data **ownership**
• **Fine-grained** access control
• **Autonomous** Access Control Management
• **Confidential** policy agent
• Efficient and scalability
• Auditability
• Distributed
System design

Fig. 1. An overview of a privacy-preserving data management framework.
Challenge 1

• Untrusted data storage nodes
• Untrusted data consuming nodes
• Data locality optimization
Solution 1

- Intel Software Guard eXtension (SGX)
  - is a trusted execution environment (TEE) platform.
  - provide memory encryption engine.
  - hardware enforced security.
Solution 1

- Provide secret region “enclave” protected from kernel and HW-based attacks
Enclave ID. Same code load in resulted in same ID. Similar to Internet certificate. So user can check if they talk with the trusted Enclave.

Same Enclave ID in same platform will have the same sealing key. Sealing isolated between different enclaves.

1. Enclave built & measured
2. Enclave requests REPORT (HW-signed blob that includes enclave identity information)
3. REPORT sent to server & verified
4. Application Key sent to enclave, first secret provisioned
5. Enclave-platform-specific Sealing Key (EGETKEY) generated
6. Application Key encrypted via Sealing Key & stored for later (offline) use
Solution 1

- **Encrypted data** from sensor.
- Decrypted **key provisioned only** to Attested Trusted Execution Environment (TEE).
  - Decrypted Key and raw data never leave TEE.
- Data consumer job can run in host node **with guaranteed confidential result** from the host (with TEE).
  - Support data locality.
Challenge 2

- Confidential policy agent
  - Smart contract, with confidential inner states.
  - Does not leak user’s policy and data price.
Solution 2

• Ekiden: a platform for confidentiality-preserving smart contract from UC Berkeley.
  – Smart contract run inside TEE.
  – Private attribute: **encrypted** on blockchain, **decrypted** inside TEE.
Solution 2

PseudoCode 1 PolicyContract

private keyMaterials;
private ACPolicy;

procedure SetPolicy(pkU, k, wl)
    if Owner(k) = null then
        Owner(k) = pkU;
    if Owner(k) = pkU then
        ACPolicy.Update(k, wl);

procedure HandleDataRequest(AttCertProg, k)
    MProg ← GetMeasurement(AttCertProg);
    if isValid(AttCertProg) ∧ MProg ∈ ACPolicy[k] then
        GrantAccess(AttCertProg, k);

procedure GrantAccess(AttCertProg, k)
    dk ← getDecryptionKey(keyMaterials, k);
    pkProg ← getPK(AttCertProg);
    channelId ← EstablishSecureChannel(pkProg);
    Send(channelId, dk);
Challenge 3

• Scalable key management
  – Support millions of data owners.
  – Fine-grained policy access: different time epochs data, different categories.

• Expensive on-chain storage: 1KB cost 0.032 ETH on Ethereum.
Solution 3

• Key-Aggregate Encryption
  – Any subsets of cipher-text encrypted by the same public key can decrypted using a constant-size decryption key.
  – To encrypt plaintext $p$: use $\text{pubk}$ and an index $i \in [1,n]$.
  – Upon receive decryption request of subset: $S \subseteq \{1, 2, \ldots, n\}$, a party with the master-secret key can generate a single key to decrypt this subset.
Solution 3

- Key-Aggregate encryption
  - Only store the master-key and codified user policy in smart contract.
  - Data consumer can request any subsets of users and users’ categories via the policy contract.
Solution 3

- Key-Aggregate encryption
Implementation result

![Graph 1: KAC encryption time and ciphertext size](image1)

![Graph 2: Size of key materials](image2)

Fig. 3. KAC encryption time and ciphertext size

Fig. 4. Size of key materials
Implementation result

Fig. 5. Key Derivation Time

Fig. 6. Key Reconstruction Time
Thank you !