SECURE CRYPTOGRAPHIC TOOLS FOR CLOUD & BLOCKCHAIN APPLICATIONS

Dimitris Papadopoulos
CSE Research and Technology Forum 2021

mail: dipapado@cse.ust.hk

site: [link]
SEARCHING ON ENCRYPTED DATA

- Data outsourced to remote servers usually needs to be encrypted for privacy purposes.
SEARCHING ON ENCRYPTED DATA

- Data outsourced to remote servers usually needs to be **encrypted** for privacy purposes.
Data outsourced to remote servers usually needs to be **encrypted** for privacy purposes.
SEARCHING ON ENCRYPTED DATA

- Data outsourced to remote servers usually needs to be encrypted for privacy purposes.
- Impossible to query it at the server due to encryption!
- Our solution: **Searchable Encryption**

```sql
SELECT * FROM TABLE WHERE DEPT = "HR"
```
Data outsourced to remote servers usually needs to be encrypted for privacy purposes.

**Impossible to query** it at the server due to encryption!

Our solution: **Searchable Encryption**

```sql
SELECT * FROM TABLE
WHERE DEPT = "HR"
```

**token** $tk_{HR}$
Data outsourced to remote servers usually needs to be encrypted for privacy purposes.

Impossible to query it at the server due to encryption!

Our solution: Searchable Encryption

- Data remains always encrypted at the server → Privacy
- Separate tokens for each column or value → Access control
- Based on simple symmetric-key encryption → Efficiency
- Generalizes to data from many users or complex queries

Our scheme published in CCS 2018 is the state-of-the-art-dynamic SE:
- Extracting 1000 records from a dataset of 1M takes less than 10ms
Many decentralized applications require **user-provided randomness** for execution.
Many decentralized applications require user-provided randomness for execution:

- How can we ensure the users’ random values are sampled honestly??
- Our solution: Verifiable (pseudo-)Random Functions (VRF)
Many decentralized applications require user-provided randomness for execution:

- How can we ensure the users’ random values are sampled honestly??
- Our solution: Verifiable (pseudo-)Random Functions (VRF)

\[
\text{Random}(sk, seed) \rightarrow \text{val, proof} \\
\text{Verify}(pk, seed, val, proof) \rightarrow \text{accept/reject}
\]

- Each user computes “random” value val
- Everybody can verify val using pk

→ Our constructions are used in many deployed cryptocurrencies (e.g., ALGORAND with 600M USD market cap)
→ Currently being standardized by the Internet Engineering Task Force
Smart contracts are programs that are stored and executed on the blockchain.
Zero-Knowledge Smart Contracts

- Smart contracts are programs that are stored and executed on the blockchain.
• Smart contracts are programs that are stored and executed on the blockchain
• Input data is provided via blockchain transactions
• Smart contracts are programs that are stored and executed on the blockchain.
• Input data is provided via blockchain transactions.
  → publicly visible to everyone!
  → cannot run smart contracts on sensitive data.
• Smart contracts are programs that are stored and executed on the blockchain.

• Input data is provided via blockchain transactions
  → publicly visible to everyone!
  → cannot run smart contracts on sensitive data

• Our solution: **Zero-knowledge proofs**
ZERO-KNOWLEDGE SMART CONTRACTS

- Smart contracts are programs that are stored and executed on the blockchain.
- Input data is provided via blockchain transactions
  - publicly visible to everyone!
  - cannot run smart contracts on sensitive data
- Our solution: Zero-knowledge proofs
- Data is encrypted on the chain and only revealed to the smart contract owner
- Owner evaluates contract and provides a zero-knowledge proof for the validity
  - nothing is revealed publicly about the data
  - saves effort as verifying the proof can be very fast!

Our scheme published in USENIX 2020 achieves the shortest proofs:
- 160 bytes proof for any smart contract and verify takes 2ms