TimeCrypt: Encrypted Data Stream Processing at Scale with Cryptographic Access Control

Lukas Burkhalter, Anwar Hithnawi, Alexander Viand, Hossein Shafagh, Sylvia Ratnasamy
Time Series Data
is Emerging Everywhere

Monitoring, Telemetry, Internet of Things

Problem: High resolution sensitive data! Server compromise results in privacy risks
Data Breaches

> 45 billion records

*since 2004

https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/
Encrypted Data Processing

• Keep data encrypted while in-use → preserve confidentiality and functionality
• Encrypted Databases → relational databases, graph databases, key-value stores
  • E.g., CryptDB [SOSP’11], BlindBox [SIGCOMM’15], Seabed [OSDI’16], Talos [SenSys’15]
Can we enable encrypted data processing for time series workloads?
Challenge I
Scalability and Interactivity
Time Series Databases

Time series workloads are different:
✓ Primarily INSERTS to recent time interval (append)
✓ Statistical queries over time ranges

Requirements:
✓ High throughput writes
✓ Large volumes of data
✓ Support for time-based queries
Time Series Databases

Time series workloads are different:

✓ Primarily INSERTS to recent time interval (append)
✓ Statistical queries over time ranges

Requirements:

✓ High throughput writes
✓ Large volumes of data
✓ Support for time-based queries

Scalability and Latency

- Memory Expansion (~100x)
- Enc/Dec Time (~milliseconds)
- Ciphertext Aggregation (~1000x)
Challenge II
Secure Sharing
Selective Data Sharing
Selective Data Sharing

Duration of Sickness

Time

Policy

Insurance

Doctor

Trainer

Minute
Selective Data Sharing

How to enable users to **selectively** share their encrypted data? Enforce access control semantics **cryptographically**
TimeCrypt in a Nutshell

Data is encrypted end-to-end:

✓ Scalable computation over large volumes of encrypted data
✓ Key time-series data functionalities, analytics, lifecycle operations
✓ Cryptographic access control → selective access to encrypted data
✓ Verifiable computation
Overview and Threat Model

**Semi-Trusted:** Data access according to an access policy

**Trusted:** Full data access

**Untrusted:** Confidentiality + Integrity
Writing Data Streams

Data producer

chunking

Time window

Aggregatable Digest

Sum
Count
...

blob

raw chunk

Encrypted Index

Aggregatable Digest

Server

Storage

blob

Client side

Server side
Aggregatable Digest

• **Additive homomorphic** encryption is the underlying construction

\[ m_1 + m_2 = \text{Dec}(\text{Enc}(m_1) \oplus \text{Enc}(m_2)) \]

How to support statistics and analytics beyond addition?
Aggregatable Digest

- Additive homomorphic encryption is the underlying construction
  \[ m_1 + m_2 = \text{Dec} (\text{Enc}(m_1) \oplus \text{Enc}(m_2)) \]

- Leverage known encoding techniques ⇒ If we can compute sum privately, then we can compute \( f(\cdot) \) privately
  - average, sum, count, variance, min/max (approx.), histograms (approx.), least-squares regression, …
Homomorphic Encryption

**Problem**: Homomorphic encryption based on asymmetric cryptography is expensive (e.g., Paillier, EC-ElGamal)

- **plaintext**: 8 bytes
- **ciphertext**: 768 bytes
- **ciphertext expansion**: 96x
- **plaintext** + **plaintext**: 1028x
- **plaintext** + **ciphertext**: **ciphertext**
- **addition time**: **milliseconds**
- **enc/decryption**: **milliseconds**
TimeCrypt Encryption

Given a key stream: \( k_0, k_1, k_2, k_3, k_4, k_5, \ldots \)

\[ m_0 + m_0 + k_0 + k_1 - k_0 + k_1 - k_2 + k_1 - \ldots + \ldots - k_{N+1} \]

[Castelluccia et al. 05] Symmetric homomorphic encryption

\( +/- \) is addition modulo \( M \)

No ciphertext expansion + fast ciphertext aggregation

\[ m_0 + \ldots + m_N + k_0 - k_1 + k_1 - \ldots + \ldots - k_{(N+1)} \]
Key Stream to Time Encoding

Query statistics from $t_0$ to $t_3$

$m_0 + m_1 + m_2$

The query time range translates to the used keys
Time Interval Access Restriction

Key Stream:

\[ k_0 \quad k_1 \quad k_2 \quad k_3 \quad k_4 \quad k_5 \quad k_6 \quad k_7 \]

\[ t_0 \quad t_3 \quad t_6 \]
Tree-based Key Derivation

The one-way property of the function allows for access control (Interval)

Key Stream:

Master Secret

PRG( )

k₀ k₁ k₂ k₃ k₄ k₅ k₆ k₇

t₀ t₃ t₆
Access Restriction at Resolution Level

How to share aggregated information of a certain granularity?

Per hour aggregates vs Per day aggregates
Only share the outer keys of the desired granularity $k_0, k_2, k_4, \ldots.$
TimeCrypt Implementation

TimeCrypt Client
*(Java, C)*

- Key Management
- Access Policies
- Enc/Decryption
- Serialization / Query logic

TimeCrypt Server node library
*(Java Netty framework)*

- Time Series API
- Aggregation Trees
- Caches

Cassandra storage
Evaluation

Health Dashboard Application

Medical Sensor Data

100 Clients
4 range queries per 1 chunk insert
50Hz data rate/stream, 10s chunks

- **m5.xlarge**
  4-CPU 16GB

- **m5.2xlarge**
  8-CPU 32GB

- **m5.xlarge**
  4-CPU 16GB
System Performance

Throughput under heavy load of 4/1 read-write ratio, 49k streams
Health Dashboard Queries

Latency for statistical queries over one month, based on our health app

120M data records, 241920 chunks (1 chunk/10s)
Summary

• TimeCrypt is an efficient system that augments time series datastores with encrypted data processing capabilities
  • Protects confidentiality of sensitive time series data
  • Supports computation integrity on encrypted data

• TimeCrypt’s Encryption: Efficient construction that couples encrypted data processing with crypto-enforced access-control for time series streams

• Source code available at: https://timecrypt.io/