Sieve: Cryptographically Enforced Access Control for User Data in Untrusted Clouds

Frank Wang (MIT CSAIL), James Mickens (Harvard), Nickolai Zeldovich (MIT CSAIL), Vinod Vaikuntanathan (MIT CSAIL)
Motivation

FitBit Cloud Server

Boston Marathon
NY Marathon
Insurance
Motivation

FitBit Cloud Server

Boston Marathon
NY Marathon
Insurance
Motivation

FitBit Cloud Server

Boston Marathon

NY Marathon

Insurance
Motivation

FitBit Cloud Server

type=race

type=running

type=fitness

Boston Marathon

NY Marathon

Insurance
Problem: Curious storage provider or external attacker

FitBit Cloud Server

- type=race
- type=running
- type=fitness
Problem: Curious storage provider or external attacker
Naïve Approach: Encrypt Data under 1 key
Naïve Approach: Encrypt Data under 1 key

FitBit Cloud Server

- type=race
- type=running
- type=fitness

Boston Marathon
NY Marathon
Insurance
Naïve Approach: Encrypt Data under 1 key
Naïve Approach: Encrypt Data under 1 key

How does the user selectively disclose her data?
Another Approach: Encrypt each piece of data individually.

FitBit Cloud Server

- type=race
- type=running
- type=fitness

- Boston Marathon
- NY Marathon
- Insurance
Another Approach: Encrypt each piece of data individually
Another Approach: Encrypt each piece of data individually
Another Approach: Encrypt each piece of data individually
Contributions

• **Sieve**: a new platform that allows users to *selectively* and *securely* disclose their data
  – Sieve protects against server compromise
  – Sieve hides key management from users
  – Reasonable performance
  – Sieve supports revocation
  – Sieves allows users to recover from device loss
  – Good for web services that analyze user data
Outline

• Sieve
  – Protocol
  – Optimizations
  – Revocation
  – Device Loss

• Implementation

• Evaluation
Sieve Overview
Sieve Overview

User

Storage Provider

Web services
Sieve Overview

User

Sieve user client

Storage Provider

Sieve storage daemon

Web services

Sieve data import
Sieve Overview

User

Sieve user client

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial

Storage Provider

Sieve storage daemon

Web services

Sieve data import
Sieve Overview

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial

Sieve data import
Sieve Overview

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Sieve data import

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial

(Year < 2013 AND
Type=Fitness)

Sieve Overview

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial

(Year < 2013 AND Type=Fitness)
Sieve Overview

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Sieve data import

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial

(Year < 2013 AND
Type=Fitness
)

Location=US,
Year=2012,
Type=fitness
Sieve Overview

User

Sieve user client

Storage Provider

Sieve storage daemon

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial

Web services

Sieve data import

Location=US,
Year=2012,
Type=fitness

(Year < 2013 AND
Type=fitness)
Sieve Overview

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Sieve data import

Location=US, Year=2012, Type=fitness

Year=2015, Type=financial

Location=US, Year=2012, Type=fitness

(Year < 2013 AND Type=Fitness)
Threat Model

• Storage provider is a passive adversary
  – Adversary can read all data
  – Follows protocol
• Web services trusted with user data they are given access to
• User and her devices trusted
Our approach: Attribute-based encryption (ABE)

- Assume that user-specific ABE public/private key pair
- Three main functions
Our approach: Attribute-based encryption (ABE)

• Assume that user-specific ABE public/private key pair
• Three main functions

GenerateDecKey

Encrypt

Decrypt
Our approach: Attribute-based encryption (ABE)

- Assume that user-specific ABE public/private key pair
- Three main functions

Policy: (Year < 2013 AND type=Fitness)

- Generate Dec Key
- Encrypt
- Decrypt
Our approach: Attribute-based encryption (ABE)

• Assume that user-specific ABE public/private key pair
• Three main functions

Policy: (Year < 2013 AND type=Fitness)
Our approach: Attribute-based encryption (ABE)

- Assume that user-specific ABE public/private key pair
- Three main functions

Policy: (Year < 2013 AND type=Fitness)

Attributes: Location=US, Year=2012, Type=fitness

GenerateDecKey

Encrypt

Decrypt

(Year < 2013 AND Type=Fitness)
Our approach: Attribute-based encryption (ABE)

- Assume that user-specific ABE public/private key pair
- Three main functions

Policy: (Year < 2013 AND type=Fitness)

Attributes: Location=US, Year=2012, Type=fitness

GenerateDecKey

Encrypt

Decrypt
Our approach: Attribute-based encryption (ABE)

• Assume that user-specific ABE public/private key pair
• Three main functions

Policy: (Year < 2013 AND type=Fitness)

Attributes:
Location=US, Year=2012, Type=fitness

GenerateDecKey

Encrypt

Decrypt

Location=US, Year=2012, Type=fitness

(Year < 2013 AND Type=Fitness)

(Year < 2013 AND type=Fitness)
Our approach: Attribute-based encryption (ABE)

- Assume that user-specific ABE public/private key pair
- Three main functions

Policy: (Year < 2013 AND type=Fitness)

Attributes: Location=US, Year=2012, Type=fitness

GenerateDecKey

Encrypt

Decrypt

Location=US, Year=2012, Type=fitness

(Year < 2013 AND Type=Fitness)
Our approach: Attribute-based encryption (ABE)

- Assume that user-specific ABE public/private key pair
- Three main functions

**Policy:** (Year < 2013 AND type=Fitness)

**Attributes:** Location=US, Year=2012, Type=fitness

**GenerateDecKey**

1. GenerateDecKey
2. Encrypt
3. Decrypt

Note: attributes and policy are in cleartext
Sieve with ABE

User

Sieve user client

Storage Provider

Sieve storage daemon

Web services

Sieve data import
Sieve with ABE

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Sieve data import

ABE Encrypt

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial
Sieve with ABE

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Sieve data import

ABE Encrypt

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial
Sieve with ABE

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Sieve data import

ABE Encrypt

(Year < 2013 AND Type=Fitness)

ABE GenerateDecKey

Location=US, Year=2012, Type=fitness

Year=2015, Type=financial
Sieve with ABE

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Sieve data import

ABE Encrypt

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial

(Year < 2013 AND
Type=Fitness )

ABE GenerateDecKey
Sieve with ABE

User

Storage Provider

Web services

Sieve user client

Sieve storage daemon

Sieve data import

ABE Encrypt

Location=US, Year=2012, Type=fitness

Year=2015, Type=financial

Location=US, Year=2012, Type=fitness

(Year < 2013 AND Type=Fitness )

ABE GenerateDecKey
Sieve with ABE

User

Sieve user client

Storage Provider

Storage daemon

Web services

Sieve data import

ABE Encrypt

Location=US,
Year=2012,
Type=fitness

Year=2015,
Type=financial

(Year < 2013 AND
Type=Fitness )

ABE GenerateDecKey

Location=US,
Year=2012,
Type=fitness

ABE Decrypt
Challenges with ABE

- Performance
- Revocation
- Device Loss
Reduce ABE Operations

• ABE is a public-key cryptosystem so slower than symmetric key cryptography

• Optimizations
  – Hybrid Encryption
  – Storage-based data structure
Hybrid Encryption
Hybrid Encryption
Hybrid Encryption

Data

symmetric
Hybrid Encryption

GUID

Data

symmetric

symmetric

ABE
Hybrid Encryption

Metadata block

 GUID

 symmetric

 Data

 symmetric

 ABE
Hybrid Encryption

Metadata block

GUID

Data

symmetric

ABE

Index

Attr1
Attr2
Attr3
Attr4
Attr5

meta

meta

meta

meta

Index

GUID1
GUID2
GUID3
GUID4
GUID5

data
data
data
data
data
Hybrid Encryption

Only have to perform symmetric key operations in future
Storage-based data structure

• Extension of hybrid encryption
Storage-based data structure

• Extension of hybrid encryption
Storage-based data structure

• Extension of hybrid encryption
Storage-based data structure

• Extension of hybrid encryption
Storage-based data structure

• Extension of hybrid encryption
Challenges with ABE

• Performance
• Revocation
• Device Loss
Revocation

FitBit Cloud Server

- type=race
- type=running
- type=fitness

Boston Marathon
NY Marathon
Insurance
Revocation

FitBit Cloud Server

- type=race
- type=running
- type=fitness

NY Marathon
Boston Marathon
Insurance
Revocation
Revocation

- Web service still has cached keys
- Need to re-encrypt data
Re-encryption with Hybrid Encryption

- Need to re-encrypt metadata and data
  - Easy to re-encrypt metadata block
  - How do we re-encrypt data object?
    - Download, re-encrypt, and upload
    - Requires substantial bandwidth and client-side computation
Solution: Key Homomorphism

• Allows changing key in encrypted data
  – Symmetric cipher that provides *in-place* re-encryption

• Does not learn old key, new key, or plaintext

• More specifics on scheme are in the paper
Full Revocation Process

Metadata Block

symmetric

Data

ABE (attrs, epoch = 0)

symmetric
Full Revocation Process

Metadata Block

 symmetric

Data

symmetric

ABE (attrs, epoch = 0)
Full Revocation Process

Metadata Block

symmetric

ABE (attrs, epoch = 0)

Data

δ(, )

symmetric
Full Revocation Process

Metadata Block

symmetric

ABE (attrs, epoch = 0)

Data

symmetric
Full Revocation Process

Metadata Block

 symmetric

 Data

 ABE (attrs, epoch = 0)

 symmetric

 symmetric

 symmetric
Full Revocation Process

Metadata Block

Data

symmetric

\( \text{ABE (attrs, epoch = 0)} \)

\( \delta \)

symmetric

\( \text{ABE (attrs, epoch = 1)} \)
Full Revocation Process

Metadata Block

Data

ABE (attrs, epoch = 0)

ABE (attrs, epoch = 1)

symmetric

symmetric

symmetric

symmetric
Issue new keys to web services whose data access has been changed and affected by revocation
Challenges with ABE

- Performance
- Revocation
- Device Loss
What if a user loses her device?

- User has ABE private key
- Loss of key requires reset of system
  - Re-encrypting all her data and issuing new keys
- Is there a way for a user to recover from device loss?
Solution: Secret sharing

• User splits her ABE private key across devices
• Requires a threshold to reconstruct secret
  – Reconstruct before using ABE private key
• When a device is lost, gathers devices to reconstruct secret and issue new “shares”
Outline

• Sieve
  – Protocol
  – Optimizations
  – Revocation
  – Device Loss

• Implementation

• Evaluation
Sieve Implementation

Cryptography:
• Libfenc with Stanford PBC for ABE
• AES (no revocation) and randomized counter mode with Ed448 (revocation)
Sieve Implementation

Cryptography:
• Libfenc with Stanford PBC for ABE
• AES (no revocation) and randomized counter mode with Ed448 (revocation)

User

Sieve user client
• ~1400 LoC

Storage Provider

Sieve storage daemon
• ~1000 LoC
• MongoDB and BerkeleyDB

Web services

Sieve data import
• Service-specific
Evaluation

• Is it easy to integrate Sieve into existing web services?
• Can web services achieve reasonable performance while using Sieve?
Evaluation Setup

• Multicore machine, 2.4 GHz Intel Xeon
• Web servers ran on machine’s loopback
  – Minimize network latency
  – Focus on cryptographic overheads
Case Studies

• Integrated with 2 open source web services
  – Open mHealth, health: small data
    • Visualize health data
    • One week’s health data: 6 KB
  – Piwigo, photo: large data
    • Edit and display photos
    • One photo: 375 KB
Easy to integrate with Sieve

• Lines of code required for integration
  – Open mHealth: ~ 200 lines
  – Piwigo: ~ 250 lines
Acceptable performance for Open mHealth and Piwigo

Ed448 with key caching

- Open mHealth
- Piwigo

<table>
<thead>
<tr>
<th>Seconds</th>
<th>Open mHealth</th>
<th>Piwigo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Write</td>
<td>Read</td>
</tr>
<tr>
<td>0</td>
<td>1.5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Legend:
- Write
- Read
Performance gap between AES and Ed448

- **Open mHealth Ed448**
  - Write: 0.5 seconds
  - Read: 0.2 seconds

- **Open mHealth AES**
  - Write: 5 seconds
  - Read: 0.5 seconds
Server per-core throughput is good

• Open mHealth
  – Storage write: 50 MB/s
  – Web service import: 70 users/min (Ed448)

• Piwigo
  – Storage write: 200 MB/s
  – Web service import: 14 photos/min (Ed448)
Revocation performance is reasonable

- Re-encrypt a metadata block (10 attrs): 0.63 s
- Re-key 100 KB data block: 0.66 s
- Generate new 10 attribute key: 0.46 s
Secret sharing is fast

• For 5 shares and threshold of 2:
  – Splitting ABE key requires 0.04 ms
  – Reconstructing key requires 0.09 ms
Summary

• Required < 250 LoC to integrate with case studies
• Read and write data in reasonable amount of time
• Good per-core server throughput for storage writes and application data imports
• Revocation functions take < 1 second
• Secret sharing takes negligible time
Related Work

• Untrusted Servers
  – ShadowCrypt, SUNDR, Depot, SPORC, CryptDB, DepSky, Bstore, Mylar, Privly

• ABE and Predicate Encryption Storage
  – Persona, Priv.io, Catchet (ABE)
  – GORAM (Predicate)

• Access Delegation Schemes
  – OAuth, AAuth, Macaroons
Related Work

• Untrusted Servers
  - ShadowCrypt, SUNDR, Depot, SPORC, CryptDB, DepSky, Bstore, Mylar, Privly

• ABE and Predicate Encryption Storage
  - Persona, Priv.io, Catchet (ABE)
  - GORAM (Predicate)

• Access Delegation Schemes
  - OAuth, AAuth, Macaroons
Related Work

• Untrusted Servers
  - Solve different problems than Sieve

• ABE and Predicate Encryption Storage
  - No complete revocation and/or ability to recover from device loss

• Access Delegation Schemes
  - OAuth, AAuth, Macaroons
Related Work

• Untrusted Servers
  Solve different problems than Sieve

• ABE and Predicate Encryption Storage
  No complete revocation and/or ability to recover from device loss

• Access Delegation Schemes
  Less secure and expressive than Sieve
Conclusions

• Sieve is a new access control system that allows users to *selectively* and *securely* expose their private cloud data to web services
• Efficiently use ABE to manage keys and policies
• Complete revocation scheme compatible with hybrid encryption using key homomorphism
• Easy to integrate and reasonable performance