Securing data in compromised clouds

Ralucu Ada Popa

UC Berkeley

raluca.popa@berkeley.edu  @ralucaadapopa
Massive cloud attacks are relentless

Yahoo 2014:  Equifax 2017:  Capital One 2019:

user records breached
Massive cloud attacks are relentless

Yahoo 2014: 500,000,000
Equifax 2017: 147,000,000
Capital One 2019: 100,000,000

user records breached
Traditional security has a fundamental weakness
Traditional security has a fundamental weakness
Attackers eventually break in
Attackers eventually break in
Assume the attacker will break in

“in the cloud [...] applications need to protect themselves instead of relying on firewall-like techniques”

Werner Vogels, Amazon CTO
Standard use of encryption
Standard use of encryption
Standard use of encryption
Standard use of encryption
Standard use of encryption

encryption in transit

cloud
Standard use of encryption

- Encryption in transit
- Encryption at rest
Standard use of encryption

- Encryption in transit
- Encryption at rest
Use encryption

encryption in transit

cloud

encryption at rest
Use encryption

- Encryption in transit
- Encryption at rest
Use end-to-end encryption
Use *end-to-end* encryption
Use **end-to-end** encryption
Systems in the cloud
Systems in the cloud

complexity
Systems in the cloud

- chat/messaging

complexity
Systems in the cloud

- chat/messaging
- email, file sharing

complexity
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)

complexity
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)

complexity
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)
- machine learning

complexity
My work

- chat/messaging
  - PREVEIL
    - JEDI [USEC19]
    - WAVE [USEC19]
- email, file sharing
  - Verena [IEEESP16]
  - Mylar [NSDI14]
  - CloudProof [Usenix11]
  - Arx [VLDB19], Oblix [IEEESP18], CryptDB [SOSP11], mOPE [IEEESP13], BlindBox [SIGCOMM15], Embark [NSDI16]
  - Opaque [NSDI17]
- database (OLTP)
- database (analytics)
- machine learning
  - Helen [IEEESP19], Delphi [USEC20], Bost et al. [NDSS15]

complexity
My work

- chat/messaging: JEDI [USEC19], WAVE [USEC19], Verena [IEEE SP16]
- email, file sharing: Mylar [NSDI14], CloudProof [Usenix11]
- database (OLTP): Arx [VLDB19], Oblix [IEEE SP18], CryptDB [SOSP11], mOPE [IEEE SP13], BlindBox [SIGCOMM15], Embark [NSDI16], Opaque [NSDI17]
- database (analytics)
- machine learning: Helen [IEEE SP19], Delphi [USEC20], Bost et al. [NDSS15]
End-to-end (E2E) encrypted chat/messaging
End-to-end (E2E) encrypted chat/messaging

Widely adopted industry solutions

[WhatsApp logo]

[Message bubble]

[Reddit logo]
End-to-end (E2E) encrypted chat/messaging

Widely adopted industry solutions

Research on many-to-many (JEDI\[USEC19\]), constrained devices (e.g. IoT WAVE\[USEC19\]), usability
E2E encrypted email and file sharing
E2E encrypted email and file sharing

- More complex than chat: add access, revoke access, edit documents
E2E encrypted email and file sharing

- More complex than chat: add access, revoke access, edit documents
- Challenge: key distribution without affecting usability
E2E encrypted email and file sharing

- More complex than chat: add access, revoke access, edit documents
- Challenge: key distribution without affecting usability
E2E encrypted email and file sharing

- More complex than chat: add access, revoke access, edit documents
- Challenge: key distribution without affecting usability

Research focusing on malicious cloud attackers (Verena [IEEEISP16]), usability, search
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)
- machine learning

complexity
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)
- machine learning

complexity
Computation on encrypted data [RAD78, Gentry09]
Computation on encrypted data [RAD78, Gentry09]
Computation on encrypted data [RAD78, Gentry09]
Computation on encrypted data [RAD78, Gentry09]
Computation on encrypted data [RAD78, Gentry09]

\[ \text{Enc(data)} \]

\[ \text{F}(\text{data}) \]

\[ \text{Enc(\text{F}(\text{data}))} \]
Computation on encrypted data [RAD78, Gentry09]
Computation on encrypted data [RAD78, Gentry09]

Example: Paillier cryptosystem, $F = +$
$Enc(x) = g^x r^n \mod n^2$
$Enc(y) = g^y r^n \mod n^2$
Computation on encrypted data [RAD78, Gentry09]

Example: Paillier cryptosystem, $F = +$

$Enc(x) = g^{xr} \mod n^2$

$Enc(y) = g^{yr} \mod n^2$

(multiply)

$Enc(x) \times Enc(y) = g^{x+y(rr')}n \mod n^2 = Enc(x+y)$
Fully homomorphic encryption [Gentry09]

- enables general functions on encrypted data
- despite much progress, remains orders of magnitude too slow
Approach to build practical systems: co-design systems and cryptography
Encrypted databases: CryptDB \cite{SOSP11}

CryptDB was the first DBMS to process SQL queries on encrypted data.
Encrypted databases: **CryptDB** [SOSP11]

CryptDB was the first DBMS to process SQL queries on encrypted data.
Encrypted databases: CryptDB \cite{SOSP11}

CryptDB was the first DBMS to process SQL queries on encrypted data.
Encrypted databases: CryptDB [SOSP11]

CryptDB was the first DBMS to process SQL queries on encrypted data
Encrypted databases: **CryptDB** [SOSP11]

CryptDB was the first DBMS to process SQL queries on encrypted data.
Encrypted databases: CryptDB [SOSP11]

CryptDB was the first DBMS to process SQL queries on encrypted data.
Encrypted databases: CryptDB [SOSP11]

CryptDB was the first DBMS to process SQL queries on encrypted data.
Encrypted databases: CryptDB [SOSP11]

CryptDB was the first DBMS to process SQL queries on encrypted data.
Encrypted databases: **CryptDB** [SOSP11]

CryptDB was the first DBMS to process SQL queries on encrypted data.
CryptDB in a nutshell
CryptDB in a nutshell

**Observation:** Most SQL can be implemented with a few core operations (e.g., +, =, >)
CryptDB in a nutshell

Observation: Most SQL can be implemented with a few core operations (e.g., +=, =, >)

Tech.#1: Employ an efficient encryption scheme for each operation
CryptDB in a nutshell

Observation: Most SQL can be implemented with a few core operations (e.g., +, =, >)

Tech.#1: Employ an efficient encryption scheme for each operation
  - design your own if needed
CryptDB in a nutshell

**Observation:** Most SQL can be implemented with a few core operations (e.g., +, =, >)

**Tech.#1:** Employ an efficient encryption scheme for each operation
- design your own if needed

**Tech.#2, Onion Encryption:** combine encryptions based on security vs. functionality
CryptDB in a nutshell

**Observation:** Most SQL can be implemented with a few core operations (e.g.,+,=,>)

**Tech.#1:** Employ an efficient encryption scheme for each operation
- design your own if needed

**Tech.#2, Onion Encryption:** combine encryptions based on security vs. functionality
- e.g., Paillier for +, DET for =
CryptDB in a nutshell

Observation: Most SQL can be implemented with a few core operations (e.g., +, =, >)

Tech.#1: Employ an efficient encryption scheme for each operation
   - design your own if needed

Tech.#2, Onion Encryption: combine encryptions based on security vs. functionality
   - e.g., Paillier for +, DET for =

Tech.#3: Redesign the query planner to produce encrypted and transformed query plans
CryptDB in a nutshell

Observation: Most SQL can be implemented with a few core operations (e.g., +, =, >)

Tech.#1: Employ an efficient encryption scheme for each operation
- design your own if needed

Tech.#2, Onion Encryption: combine encryptions based on security vs. functionality
- e.g., Paillier for +, DET for =

Tech.#3: Redesign the query planner to produce encrypted and transformed query plans
- resulting queries did not change the DBMS
CryptDB in a nutshell

**Observation:** Most SQL can be implemented with a few core operations (e.g., +, =, >)

**Tech.#1:** Employ an efficient encryption scheme for each operation
  - design your own if needed

**Tech.#2, Onion Encryption:** combine encryptions based on security vs. functionality
  - e.g., Paillier for +, DET for =

**Tech.#3:** Redesign the query planner to produce encrypted and transformed query plans
  - resulting queries did not change the DBMS
CryptDB in a nutshell

Observation: Most SQL can be implemented with a few core operations (e.g., +, =, >)

Tech.#1: Employ an efficient encryption scheme for each operation
   - design your own if needed

Tech.#2, Onion Encryption: combine encryptions based on security vs. functionality
   - e.g., Paillier for +, DET for =

Tech.#3: Redesign the query planner to produce encrypted and transformed query plans
   - resulting queries did not change the DBMS

Supported all of TPC-C, 27% throughput loss
A rich line of work followed

- **Academic work:**
  
  Cipherbase, CMD, Cryptsis, Autocrypt, Clome, SensorCloud, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer[PKV+14], [CJJ+14], [FJK+15], [K15], Arx, MrCrypt, Monomi, [NKW15],[DDC16],[GSB17],KKN+16], [DCF+20],… > 1000 citations.

- **Industry deployments:**
Lesson: co-design of systems and cryptography
Lesson: co-design of systems and cryptography

A recipe:
1. Focus on a workload. Identify a set of core operations the system needs
2. Identify a suitable encryption building block efficient for each operation
3. Design a planner/compiler that can combine the encryption building blocks based on their constraints and cost model
Lesson: co-design of systems and cryptography

A recipe:
1. Focus on a workload. Identify a set of core operations the system needs
2. Identify a suitable encryption building block efficient for each operation
3. Design a planner/compiler that can combine the encryption building blocks based on their constraints and cost model

For the architecture:
- avoid changing existing applications and cloud systems
Lesson: co-design of systems and cryptography

A recipe:
1. Focus on a workload. Identify a set of core operations the system needs
2. Identify a suitable encryption building block efficient for each operation
3. Design a planner/compiler that can combine the encryption building blocks based on their constraints and cost model

For the architecture:
- avoid changing existing applications and cloud systems
Research challenge: functionality vs security vs performance
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

? complex analytics or ML
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

   complex analytics or ML

2. Sharp security/performance tradeoff. A “rough” sketch:
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

   ? complex analytics or ML

2. Sharp security/performance tradeoff. A “rough” sketch:
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality
   
   complex analytics or ML

2. Sharp security/performance tradeoff. A “rough” sketch:
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality
   
   complex analytics or ML

2. Sharp security/performance tradeoff. A “rough” sketch:
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality
   
   ? complex analytics or ML

2. Sharp security/performance tradeoff. A “rough” sketch:

   - cloud sees all data
   - semantic security (= regular encryption)
   - oblivious (hides access patterns)
   - cloud learns nothing

Leakage from memory addresses accessed
Exploitable depending on attacker strength
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

   ? complex analytics or ML

2. Sharp security/performance tradeoff. A “rough” sketch:

   - too slow for DBs
   - practical

   - cloud sees all data
   - semantic security (= regular encryption)
   - oblivious (hides access patterns)
   - cloud learns nothing
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

? complex analytics or ML

2. Sharp security/performance tradeoff. A “rough” sketch:

<table>
<thead>
<tr>
<th>too slow for DBs</th>
<th>practical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schemes: Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer[PKV+14], [CJJ+14], [FJK+15], [K15], Arx, ...</td>
<td></td>
</tr>
<tr>
<td>Attacks: [NKW15],[DDC16], [GSB17],KKN+16], [DCF+20],...</td>
<td></td>
</tr>
</tbody>
</table>

cloud sees all data | semantic security (= regular encryption) | oblivious (hides access patterns) | cloud learns nothing
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

2. Sharp security/performance tradeoff. A “rough” sketch:

- Schemes: Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer [PKV+14], [CJJ+14], [FJK+15], [K15], Arx, ...
- Attacks: [NKW15], [DDC16], [GSB17], [KKN+16], [DCF+20], ...

- Complex analytics or ML
- Cloud sees all data
- Semantic security (= regular encryption)
- Oblivious (hides access patterns)
- Cloud learns nothing
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

2. Sharp security/performance tradeoff. A “rough” sketch:

- complex analytics or ML

<table>
<thead>
<tr>
<th>too slow for DBs</th>
<th>oblivious algorithms ([GO93], PathORAM, …)</th>
</tr>
</thead>
<tbody>
<tr>
<td>practical</td>
<td></td>
</tr>
<tr>
<td>Schemes: Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer[PKV+14], [CJJ+14], [FJK+15], [K15], Arx, …</td>
<td></td>
</tr>
<tr>
<td>Attacks: [NKW15],[DDC16], [GSB17],KKN+16], [DCF+20],…</td>
<td>semantic security (= regular encryption)</td>
</tr>
</tbody>
</table>

cloud sees all data

oblivious (hides access patterns)

cloud learns nothing
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

2. Sharp security/performance tradeoff. A “rough” sketch:

   - too slow for DBs
     - Schemes: Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer [PKV+14], [CJJ+14], [FJK+15], [K15], Arx, ...
     - Attacks: [NKW15], [DDC16], [GSB17], KKN+16], [DCF+20], ...

   - practical
     - oblivious algorithms
       - ([GO93], PathORAM, …)
       - lower bounds

   - cloud sees all data
   - semantic security
     - (= regular encryption)
   - oblivious
     - (hides access patterns)
   - cloud learns nothing
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

2. Sharp security/performance tradeoff. A “rough” sketch:

- Too slow for DBs
  - Schemes: Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer [PKV+14], [CJJ+14], [FJK+15], [K15], Arx, …
  - Attacks: [NKW15], [DDC16], [GSB17], KKN+16, [DCF+20], …

- Practical
  - Oblivious algorithms
    - ([GO93], PathORAM, …)
  - Lower bounds

- Cloud sees all data
- Semantic security (= regular encryption)
- Oblivious (hides access patterns)
- Cloud learns nothing

Complex analytics or ML
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

2. Sharp security/performance tradeoff. A “rough” sketch:

- complex analytics or ML

- Schemes: Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer[PKV+14], [CJJ+14], [FJK+15], [K15], Arx, …
- Attacks: [NKW15],[DDC16], [GSB17],KKN+16], [DCF+20],…

- oblivious algorithms ([GO93], PathORAM, …)
- lower bounds

- oblivious (hides access patterns)
- semantic security (= regular encryption)
- cloud sees all data

- cloud learns nothing

- practical
- too slow for DBs

- enclaves+ crypto
- Opaque[NSDI17]
- Oblix [IEEEESP18]
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

   ? complex analytics or ML

2. Sharp security/performance tradeoff. A “rough” sketch:

   - Schemes: Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer[PKV+14], [CJJ+14], [FJK+15], [K15], Arx, …
   - Attacks: [NKW15], [DDC16], [GSB17], KKN+16], [DCF+20], …

   - oblivious algorithms ([GO93], PathORAM, …)
   - lower bounds

   - multi-party interaction
   - oblivious algorithms

   - cloud sees all data
   - semantic security (= regular encryption)
   - oblivious (hides access patterns)

   - cloud learns nothing
Research challenge: functionality vs security vs performance

1. Existing building blocks had limited functionality

2. Sharp security/performance tradeoff. A “rough” sketch:

- **complex analytics or ML**
- **semantic security** *(= regular encryption)*
- **oblivious** *(hides access patterns)*
- **cloud sees all data**
- **cloud learns nothing**

### Schemes
- Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer [PKV+14], [CJJ+14], [FJK+15], [K15], **Arx**, ...
- **Schemes:** Cipherbase, [ABE+13], [TKM+13], Seabed [PBC+16], BlindSeer [PKV+14], [CJJ+14], [FJK+15], [K15], **Arx**, ...

### Attacks
- [NKW15], [DDC16], [GSB17], KKN+16, [DCF+20], ...
- **Attacks:** [NKW15], [DDC16], [GSB17], KKN+16, [DCF+20], ...

### Practical
- too slow for DBs
- **too slow for DBs**

### Lower Bounds
- oblivious algorithms ([GO93], PathORAM, …)
- lower bounds
- **oblivious algorithms** ([GO93], PathORAM, …) **lower bounds**

### Multi-party Interaction
- Helen [IEEESP19]
- **Helen** [IEEESP19]

### Enclaves+Crypto
- Delphi [USEC20]
- **Delphi** [USEC20]
- Oblix [NSDI17]
- **Oblix** [NSDI17]
- **Opaque** [IEEESP18]
- **Opaque** [IEEESP18]

### Opaque
- [NSDI17]
- **Opaque** [NSDI17]

### Oblix
- [IEEESP18]
- **Oblix** [IEEESP18]
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)
- machine learning

complexity
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)
- machine learning

complexity
Hardware enclaves 101
Hardware enclaves (Intel SGX)

- Hardware-enforced isolated execution environment
Hardware enclaves (Intel SGX)

- Hardware-enforced isolated execution environment

![Diagram of hardware-enforced isolated execution environment](image)
Hardware enclaves (Intel SGX)

- Hardware-enforced isolated execution environment
Hardware enclaves (Intel SGX)

- Hardware-enforced isolated execution environment
- Data decrypted only on the processor
Hardware enclaves (Intel SGX)

- Hardware-enforced isolated execution environment
- Data decrypted only on the processor
- Protect against an attacker who has root access or compromised OS
Hardware enclaves (Intel SGX)

- Hardware-enforced isolated execution environment
- Data decrypted only on the processor
- Protect against an attacker who has root access or compromised OS

- Cloud offerings: Azure Confidential Computing, Alibaba Cloud
Remote attestation

Client

Server
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange.
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange.
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange

Client

Server

enclave

client code

hash
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange.
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange
Side channels

Enclaves suffer from many side channels:

- cache-timing attacks ([Gotzfried et al17],[Brasser17,…])
- branch predictor based attacks ([Lee et al17],…)
- page fault based attacks ([Xu et al15], …)
- memory bus based attacks (Membuster[USEC20])
- dirty-bit based attacks
Side channels

Enclaves suffer from many side channels:
- cache-timing attacks ([Gotzfried et al17],[Brasser17,…])
- branch predictor based attacks ([Lee et al17],…)
- page fault based attacks ([Xu et al15], …)
- memory bus based attacks (Membuster[USEC20])
- dirty-bit based attacks
Side channels

Enclaves suffer from many side channels:
- cache-timing attacks ([Gotzfried et al.17],[Brasser17,...])
- branch predictor based attacks ([Lee et al.17],...)
- page fault based attacks ([Xu et al.15],...)
- memory bus based attacks (Membuster[USEC20])
- dirty-bit based attacks

Side channels reduce to exploit memory addresses prevented by oblivious computation
Enclaves suffer from many side channels:

- cache-timing attacks ([Gotzfried et al17],[Brasser17,…])
- branch predictor based attacks ([Lee et al17],…)
- page fault based attacks ([Xu et al15], …)
- memory bus based attacks (Membuster[USEC20])
- dirty-bit based attacks

Side channels reduce to exploit memory addresses prevented by oblivious computation

Synergy: enclaves remove expensive network communication of oblivious algorithms
Opaque*: oblivious and encrypted distributed analytics platform

* Oblivious Platform for Analytic QUEries
Query execution

Client

Cloud

Scheduler

Database

1 2 3
Query execution

Client

Scheduler

Opaque
Catalyst
Spark Driver

Cloud

Database

1 2 3
query = SELECT sum(*) FROM table
Query execution

query = SELECT sum(*)
FROM table
query = SELECT sum(*)
FROM table
Query execution

query = SELECT sum(*)
FROM table
query = SELECT sum(*)
FROM table
query = SELECT sum(*) FROM table
Query execution

query = SELECT sum(*)
FROM table
Query execution

```
query = SELECT sum(*) FROM table
```
query = SELECT sum(*) FROM table
query = SELECT sum(*)
FROM table
query = SELECT sum(*)
FROM table
query = SELECT sum(*)
FROM table
Opaque components
Opaque components

Data encryption and authentication
Opaque components

Computation verification

Data encryption and authentication
Opaque components

Distributed oblivious operators
- Oblivious Filter
- Oblivious Aggregation
- Oblivious Join

Computation verification

Data encryption and authentication
Opaque components

Oblivious query planning
- Cost model
- Rule-based opt.
- Cost-based opt.

Distributed oblivious operators
- Oblivious Filter
- Oblivious Aggregation
- Oblivious Join

Computation verification

Data encryption and authentication
Open source

https://github.com/ucbrise/opaque

Adoption: IBM RestAssured, Ericsson, Alibaba
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)
- collaborative machine learning

cloud

复杂性
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)
- collaborative machine learning

complexity
Money laundering detection
Money laundering detection

• Bank wants to detect money laundering using machine learning
Money laundering detection

• Bank wants to detect money laundering using machine learning
Money laundering detection

• Bank wants to detect money laundering using machine learning

• Criminals conceal illegal activities across many banks
Money laundering detection

• Bank wants to detect money laundering using machine learning

• Criminals conceal illegal activities across many banks
Money laundering detection

• Bank wants to detect money laundering using machine learning

• Criminals conceal illegal activities across many banks
Money laundering detection
Money laundering detection

- Want to jointly compute a model on customer transaction data across many banks
Money laundering detection

- Want to jointly compute a model on customer transaction data across many banks
Money laundering detection

• Want to jointly compute a model on customer transaction data across many banks

• Cannot share data because these banks are competing with each other
Two approaches
Two approaches

A different setup tradeoff:
- Hardware enclaves + oblivious algorithms
- Secure multi-party computation
Secure collaborative ML via enclaves

User A

User B

User C
Secure collaborative ML via enclaves

- User A
  - Key

- User B
  - Key

- User C
  - Key

Key management enclave
Secure collaborative ML via enclaves

User A

User B

User C

Each client attests separately and transfers the secret key

Key management enclave
Secure collaborative ML via enclaves

Each client attests separately and transfers the secret key.
Secure collaborative ML via enclaves

User A

User B

Key management enclave

User C

Each client attests separately and transfers the secret key
Secure collaborative ML via enclaves

User A

User B

User C

Key management enclave

Each client attests separately and transfers the secret key
Secure collaborative ML via enclaves

User A

User B

User C

Key management enclave

Each client attests separately and transfers the secret key
Secure collaborative ML via enclaves

Each client attests separately and transfers the secret key
Secure collaborative ML via enclaves

Each client attests separately and transfers the secret key.
Secure collaborative ML via enclaves

Key management enclave

User A

User B

User C

Worker enclaves
Secure collaborative ML via enclaves

User A
- A
- Key management enclave

User B
- B
- Key management enclave

User C
- C
- Key management enclave

Worker enclaves
Secure collaborative ML via enclaves

Key management

Worker enclaves

User A

User B

User C

A

B

C

Secure collaborative ML via enclaves
Secure collaborative ML via enclaves

User A

User B

User C

Key management enclave

Worker enclaves
Secure collaborative ML via enclaves

User A

User B

User C

Worker enclaves

Key management enclave

A

B

C
Secure collaborative ML via enclaves

User A

User B

User C

Key management enclave

Worker enclaves

Run oblivious algorithms; \textbf{mc}^2 work in progress
Secure multiparty computation

(MPC

[Yao82,GMW87,BGW88]  )

Trusted third party
Secure multiparty computation
(MPC [Yao82,GMW87,BGW88])

- Parties emulate a trusted third party via cryptography
Secure multiparty computation (MPC) [Yao82,GMW87,BGW88]

- Parties emulate a trusted third party via cryptography
Secure multiparty computation

(MPC \[\text{[Yao82,GMW87,BGW88]}\])

- Parties emulate a trusted third party via cryptography
Secure multiparty computation (MPC) 

• Parties emulate a trusted third party via cryptography

• No party learns any party’s input beyond the final result
Main challenge: Performance

Generic secure multi-party computation

[SPDZ]
Main challenge: Performance

Generic secure multi-party computation

[SPDZ]

Example: train linear models
Main challenge: Performance

Generic secure multi-party computation

[SPDZ]

Example: train linear models

3 months
Main challenge: Performance

Generic secure multi-party computation
[SPDZ]

Example: train linear models

Our approach:
Systems
ML
Crypto

3 months
Main challenge: Performance

Generic secure multi-party computation

[SPDZ]

Our approach:

ML

Systems

Crypto

Example: train linear models

Helen [IEEEISP’19]

3 months
Main challenge: Performance

Generic secure multi-party computation
[SPDZ]

Our approach:
Systems
Crypto

ML

Example: train linear models

Helen [IEEEESP’19]

3 months

< 3 hours
Main challenge: Performance

Generic secure multi-party computation

[SPDZ]

Our approach:

ML

Systems

Crypto

Example: train linear models

Helen [IEEESP’19]

3 months

< 3 hours

Delphi [USEC20]: secure inference for neural networks
mc²: work in progress
multi-party cryptographic collaboration
mc^2: work in progress

multi-party cryptographic collaboration

An easy-to-use secure collaborative learning platform for the non-expert
mc²: work in progress

multi-party cryptographic collaboration

An easy-to-use secure collaborative learning platform for the non-expert
mc²: work in progress

An easy-to-use secure collaborative learning platform for the non-expert

User specifies Python DSL for learning task which automatically compiles to oblivious collaborative computation in enclaves or in MPC
An easy-to-use secure collaborative learning platform for the non-expert

User specifies Python DSL for learning task which automatically compiles to oblivious collaborative computation in enclaves or in MPC

Open source: https://github.com/mc2-project
  • Secure collaborative XGBoost
An easy-to-use secure collaborative learning platform for the non-expert

User specifies Python DSL for learning task which automatically compiles to oblivious collaborative computation in enclaves or in MPC

Open source: https://github.com/mc2-project
  • Secure collaborative XGBoost
  • Collaboration with ScotiaBank, Azure Confidential, Ericsson, and Ant Financial
An easy-to-use secure collaborative learning platform for the non-expert

User specifies Python DSL for learning task which automatically compiles to oblivious collaborative computation in enclaves or in MPC

Open source: https://github.com/mc2-project
  • Secure collaborative XGBoost
  • Collaboration with ScotiaBank, Azure Confidential, Ericsson, and Ant Financial

Potential societal impact is exciting
Systems in the cloud

- chat/messaging
- email, file sharing
- database (OLTP)
- database (analytics)
- collaborative machine learning

*complexity*
Principles

- Assume attackers will eventually break into the cloud
- Be prepared by processing data in encrypted form
- Co-design systems and cryptography for performance
Principles

- Assume attackers will eventually break into the cloud
- Be prepared by processing data in encrypted form

Co-design systems and cryptography for performance

1. Focus on a workload. Identify a set of core operations the system needs
2. Identify an efficient secure protocol for each operation
3. Design a planner to combine the building blocks based on their constraints and cost model
Principles

- Assume attackers will eventually break into the cloud
- Be prepared by processing data in encrypted form

- Co-design systems and cryptography for performance

1. Focus on a workload. Identify a set of core operations the system needs
2. Identify an efficient secure protocol for each operation
3. Design a planner to combine the building blocks based on their constraints and cost model

Thank you!

raluca.popa@berkeley.edu   @ralucaadapopa