Privacy-Preserving Analytics: Cryptography

Roman Walch
Privacy-Preserving Cryptography

► Traditional Cryptography:
  ● Encryption provides confidentiality
  ● Tags/Signatures provide integrity and authenticity
  ⇒ Secure data storage/transmission

► New picture of cryptography:
  ● Computing with unknown data
  ⇒ Secure function evaluation

► Protocols and Primitives:
  ● Secure Multi-party Computation (MPC)
  ● Fully Homomorphic Encryption (FHE)
Cryptographic Protocols and Primitives

Secure Multi-party Computation
MPC – General Idea

Many processes in our world are solved by using a trusted third party
- Lawyers
- Notaries
- Data markets
- ...

Not always possible to employ this approach
- Lack of trust
- Sensitive data
- ...

Can we cope without a trusted party?
Yao’s Millionaires’ Problem

Hey!

What?

I bet I’m richer than you!

No way!
Yao’s Millionaires’ Problem

I don’t want to say how rich I am!

Alice

Bob
MPC – Secure Function Evaluation

► Generalization of the Problem
  • \( n \) parties
  • Each party has input \( x_i \)
  • Parties jointly evaluate Function: \( F(x_1, \ldots, x_n) \)

► Construct protocol that ensures:
  • Some/All users learn output \( F(x_1, \ldots, x_n) \)
  • Each user learns nothing about inputs of other parties
  • Each user learns nothing about any intermediate values

► Yao’s Millionaires’ Problem [Yao82]:
  • \( F(x_1, x_2) = x_1 > x_2 \)

► MPC has only recently become practical
Example Use Case – Collaborative Learning

- Several Parties train a ML model on combined dataset
- Parties do not want to disclose their dataset
  ⇒ MPC
  - Parties input: dataset
  - Output: ML model
  - Inputs remain private
- Similar: Private classification using MPC
Cryptographic Protocols and Primitives

Fully Homomorphic Encryption
Outsourcing Computation

▶ Distributed systems, cloud computing
  • Outsourcing computation to much more powerful servers
  • Caveat: Have to trust other party

▶ Idea: encrypt data
  • Classical encryption ensures confidentiality of data
  • However, not possible to operate on encrypted data
  • Small change in ciphertext → huge change in plaintext

⇒ use Fully Homomorphic Encryption
Fully Homomorphic Encryption (FHE)

- Operate on encrypted, unknown data
- Without knowing secret decryption key

Fig.: ©TU Graz.
Fully Homomorphic Encryption (FHE)

- Dream: All operations are possible on encrypted data
- Theoretical requirement:
  - Addition and multiplication
  - Arbitrary times
- 2009: Craig Gentry: “A fully homomorphic encryption system” [Gen09]
  - One of the biggest advances in modern cryptography
- Since then:
  - Many variations and improvements
Example Use Case – Outsourcing Computation

- Client wants to use a Server for computations
  - Without revealing data
  - E.g.: “Machine Learning as a Service”

- Workflow:
  - Client encrypts data
  - Server computes on encrypted data ⇒ encrypted result
  - Client can decrypt the result
  - Only client knows secret decryption key
Use Case: Private Set Intersection

[Kal+19]
Motivation

- Two parties want to find common database entries
  - Common customers for targeted advertisement
  - Contact discovery for messaging services (e.g. WhatsApp)

- Trivial solution:
  - Transmit database and compare items
  - $\Rightarrow$ leaks database!

- Use Private Set Intersection (PSI) instead
  - other party learns nothing about items outside intersection
Idea

- Transmit encrypted database and compare items
- For comparison:
  - Encryption under same key required
  - But: cannot give both parties the key (enables decryption of transmitted set)
- Solution:
  - Encrypt using a Oblivious Pseudorandom Function
Oblivious Pseudorandom Function

Encrypt using a Oblivious Pseudorandom Function

Classical Encryption – Pseudorandom Function
\[ y = \text{PRF}_k(x) \]
Oblivious Pseudorandom Function

- Encrypt using an Oblivious Pseudorandom Function
- Evaluate an encryption using MPC

**Classical Encryption – Pseudorandom Function**

\[ y = \text{PRF}_k(x) \]

**MPC Encryption – Oblivious Pseudorandom Function**

\[ y = \text{PRF}_k(x) \]
PSI using OPRF Evaluation

Basic protocol idea:

Party A

Party B
PSI using OPRF Evaluation

Basic protocol idea:
PSI using OPRF Evaluation

Basic protocol idea:

Party A

Party B
PSI using OPRF Evaluation

Basic protocol idea:

Party A 📜

OPRF

Party B 📜
PSI using OPRF Evaluation

Basic protocol idea:

- Party A
- Party B

Set A

Set B

OPRF
PSI using OPRF Evaluation

Basic protocol idea:

Party A

OPRF

Party B
PSI using OPRF Evaluation

Basic protocol idea:
PSI using OPRF Evaluation

Basic protocol idea:
PSI using OPRF Evaluation

Basic protocol idea:

Party A

Party B
Join Safe-DEED’s Professional Partners Community

- Early access to Safe-DEED demonstrators and prototypes
- One use-case discussion with a research partner of your choice, at the Safe-DEED partner’s location
- Contact to other companies interested in privacy enhancing technologies

https://safe-deed.eu/
Limitations and Outlook
Limitations

- Multi-party Computation
  - Huge communication complexity
  - Performance overhead depends on security guarantees
  - Evaluation requires parties to know $F$ (e.g. structure of Neural Network)

- Fully Homomorphic Encryption
  - Huge performance overhead
  - Ciphertext expansion
  - Only addition and multiplication (i.e. ReLU requires approximation)
  - Small multiplicative depth of evaluated function
Conclusion

► Very promising techniques to preserve privacy
  ... but still inefficient

► Performance ...
  • depends on security guarantees
  • good enough for some applications
  • in general not yet sufficient

► Right now lots of research done for MPC, FHE, and PSI
QUESTIONS

www.know-center.at

Roman Walton  Lukas Helminger  Andreas Trügler

Know-Center GmbH
Research Center for Data-Driven Business and Big Data Analytics
Infeldgasse 13/6
8010 Graz, Austria

Firmenbuchgericht Graz
FN 199 685 f
UID: ATU 50387703
