Secure Shufflers for FL

A survey about secure shuffling for Federated Learning

Marc DAMIE

June 24, 2022 (PMR Workshop)

UNIVERSITY OF TWENTE.
Inria

1. Preliminaries

2. Why do you need secure shufflers?

3. What is a secure shuffler?

4. How to implement a secure shuffler?

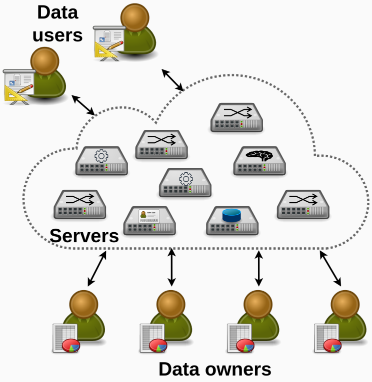
5. How to choose a secure shuffler?

6. Take-away message

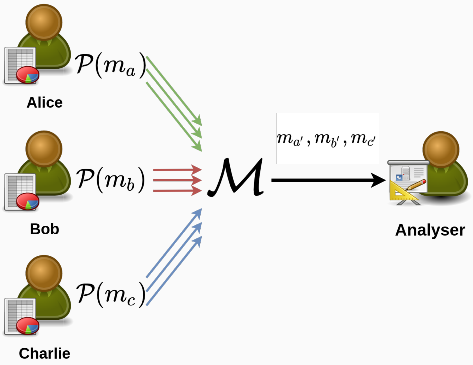
Federated Learning and Terminology

Definition

Federated Learning is a set of techniques enabling a group of data owners to **collaboratively** train a Machine Learning model **without revealing their personal data**.



Naive definition



Secure Shuffler definition

A **secure shuffler** is an entity taking as input a collection of "encoded" messages and outputting the plaintext messages while **hiding their origin**.

Why a survey about secure shufflers?

- **Attracted a lot of attention recently**, especially in the DP community.
- Can be a key component to build **scalable and secure** FL systems.
- The literature **remains vague** about its implementation.
- A **full comparison and definition work** is then needed to understand the concept of secure shuffler and the criteria to choose one.

1. Preliminaries

2. Why do you need secure shufflers?

3. What is a secure shuffler?

4. How to implement a secure shuffler?

5. How to choose a secure shuffler?

6. Take-away message

Secure aggregation

Secure aggregation

Protocol aggregating data while **only revealing the aggregated value**.

Secure aggregation from secure shuffling: Ishai et al. (2006)

Data owners split their private values into k additive shares and send them **anonymously** to an aggregator. The aggregator can only discover the value of the aggregation since the shares **reveal no information**.

About the usefulness of secure aggregation

Can be used to train linear, logistic reg., neural nets, random forests, etc.

Differential Privacy

Shuffle model: Bittau et al. (2017), Cheu et al. (2019)

A DP model enabling (ϵ, δ)-differentially private computation based on three randomized algorithms:

- A local randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$
- A shuffler $\mathcal{S} : \mathcal{Y}^* \to \mathcal{Y}^*$ that randomly permutes his inputs.
- An analyser $\mathcal{A} : \mathcal{Y}^* \to \mathcal{Z}$

- Shuffle DP approaches central DP in terms of privacy-utility trade-off
- Secure shuffling is (implicitly) considered as a problem with scalable solutions requiring less trust to implement...
- But, no paper details the notion of secure shuffler or its implementation

FL protocols with peer-to-peer communications

The anonymization enabled by secure shufflers can also be:

- Interesting for attack mitigation by **limiting the adversary view**.
- Needed **by design** in some fully decentralized algorithms.

1. Preliminaries

2. Why do you need secure shufflers?

3. What is a secure shuffler?

4. How to implement a secure shuffler?

5. How to choose a secure shuffler?

6. Take-away message

A first definition attempt

Remainder: threat models

- *Honest agents*: simply follows the protocol
- *Honest-but-curious agents*: try to **passively** infer private information.
- *Malicious agents*: can perform **any action** to gain additional information.

Intuitive definition

A shuffler $\mathcal{S}$ is a **secure** against a threat model $\mathcal{A}$ if there exists a permutation π such that $\mathcal{S}(m_1, \ldots, m_n) = \pi(m_1, \ldots, m_n)$ with π unknown by any adversary fitting the threat model $\mathcal{A}$.

Beyond the intuition: security properties

- Anonymity
- Correctness
- Client disruption resistance
- Server disruption resistance

Anonymity and Correctness

Anonymity definition by Pfitzmann and Hansen (2010)

A sender P is anonymous, iff P is not distinguishable within the set of potential senders (called anonymity set).

Correctness definition

A shuffler $\mathcal{S}$ is **correct** if its output is composed of all and only the input messages.

- $\mathcal{S}$ **secure shuffler** $\iff$ $\mathcal{S}$ **correct and anonymous**

Client and Server Disruption Resistance

Client disruption resistance

A secure shuffler $\mathcal{S}$ is **resistant against client disruption** if a single malicious data owner is not able to force the protocol abortion.

Server disruption resistance

A secure shuffler $\mathcal{S}$ is **resistant against server disruption** if a single malicious shuffling server is not able to force the protocol abortion.

Inria

1. Preliminaries

2. Why do you need secure shufflers?

3. What is a secure shuffler?

4. How to implement a secure shuffler?

5. How to choose a secure shuffler?

6. Take-away message

Comparison of the techniques in the survey

- **Security**: threat model and security properties

- **Efficiency**: (client & server) computation and communication costs

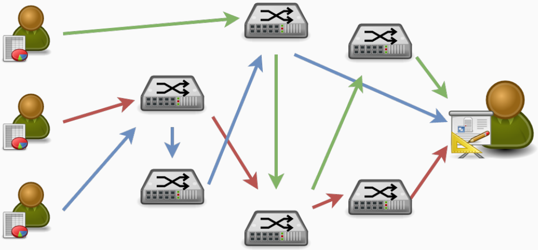
Mix-nets

Advantages:

- Lightweight (millions of msg.)
- Secure vs. $k-1$ malicious nodes per path
- Can already use Tor

Disadvantages:

- Traffic-analysis attacks
- No correctness



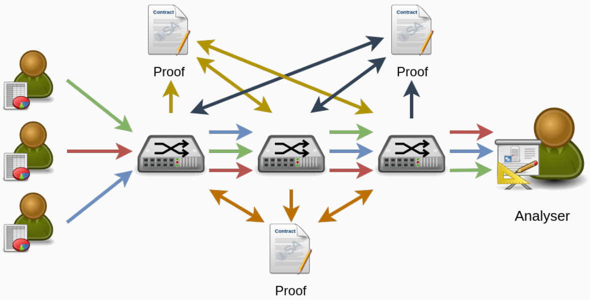
Verifiable shufflers

Advantages:

- Secure vs. $k - 1$ malicious servers
- Correctness
- Auditable results

Disadvantages:

- Scaling limited to thousands of messages
- Trusted setup



DC-nets

- DC-nets $\iff$ Techniques relying on a form of XOR masking
- Each technique has its own advantages and disadvantages
- Some scale up to millions of msg

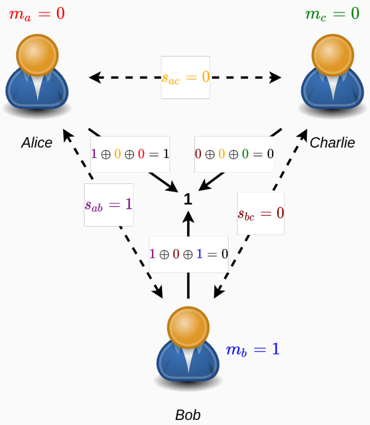


Figure: Dining Cryptographers problem

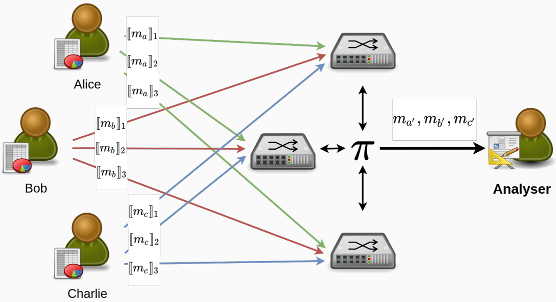
MPC-based shufflers

Advantages:

- Can achieve the highest security guarantees
- Scalable (up to a 1M msg)

Disadvantages:

- Relatively large computation costs
- Secure vs. a proportion of malicious servers (e.g. < 1/3)



What else?

- Few other shufflers (e.g. Trusted Execution Environment)
- **Several orthogonal discussions**: public-key infrastructure, shuffle output broadcasting, etc.
- Table synthesizing the comparison of **20 secure shufflers**

1. Preliminaries

2. Why do you need secure shufflers?

3. What is a secure shuffler?

4. How to implement a secure shuffler?

5. How to choose a secure shuffler?

6. Take-away message

Main selection criteria

- Resources and availability of the data owners
- **Number of messages**
- Number of **independent** shuffling servers available
- Shuffle **correctness** requirement: e.g. mandatory for DP computations.
- **Trust assumption**: how trustworthy are the servers?
- Other minor criteria: auditability, public shuffle output, etc.
- The amount of training data **is not** a criterion, only the number of msg is

Example: Medical surveys

Context

A group of hospitals wants to deploy a privacy-preserving surveying system. The data owners (i.e. survey participants) answer the surveys on their web browser and submit their answers through the shuffler.

Shuffler choice: Verifiable Shuffler

- Scalability is not a problem (unlikely to have millions of submissions)
- Data owners can quickly submit and disconnect.
- Auditability is a nice plus.

Example: On-device recommender system

Context

An open-source application wants to train an on-device recommender system. Several dozens of non-profit organizations agree to deploy shuffling servers. Each user has some data on their local device.

Shuffler choice: Mix-nets

- Can scale up to millions of data owners.
- No explicit correctness or auditability requirement.
- Strong threat model because NPOs are less trustworthy than hospitals.

Inria

Example: AI-assisted diagnostics

Context

A dozen of hospitals wants to collaborate and uses their medical data to train a complex Deep Learning model to assist the practitioners. Each hospital acts at the same time as a shuffling server and a data owner.

Shuffler choice: MPC-based shuffler

- Number of messages is very small $\Rightarrow$ no cost issue
- Correctness is important due to the sensitivity of the use case.
- A weaker threat model is acceptable since hospitals have a reputation issue and should be honest (except if compromised).

1. Preliminaries

2. Why do you need secure shufflers?

3. What is a secure shuffler?

4. How to implement a secure shuffler?

5. How to choose a secure shuffler?

6. Take-away message

Take-away message

- Secure shuffling is a useful tool to deploy **large-scale, private and secure** Federated Learning

- There already exist secure shuffling solutions that enables scalable FL even with **millions of resource-constrained devices**

- One should carefully **study the security properties** of a shuffler before choosing it

Thank you for your attention!