ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

IN, LCA1: Decentralized Data Sharing System based on Secure Multiparty Computation

Due on Autumn 2017

D.Froelicher, J.Troncoso-Pastoriza

Max Premi

November 29, 2017

Abstract

Unlynx and Prio are two privacy-preserving data sharing systems, with each it's way to encode, decode and aggregate datas. While Unlynx uses homomorphic encryption based on Elliptic Curve ElGamal and zero knowledge proofs, Prio uses Secret-sharing encoding and *secret-shared non-interactive proofs*(SNIP's), which should perform much better than classic zero knowlege proof in term of computation time, by doing a trade-off execution time/bandwidth.

We consider m servers that consitute the collective authority whose goal is to verifiably compute aggregation functions over data send by n data providers.

Moreover, privacy is assured if at least one client is trusted, and it then assure robustness and scalability, with several other mechanics such as Noise addition for Unlynx while Prio insures robustness if and only if all server are trusted. In both case the number of servers should be significantly smaller than the number of client.

Prio also extends classic private aggregation techniques to enable collection of different class of statistics such as least-square regression.

HERE BE MORE PRECISE This paper presents the implementation of Prio aggregation system, as well as an input validation proof for Unlynx Aggregation system, and then compare both of those to see which one scales better.

Contents

Introduction

Nowadays, tons of data are generated around us and about what we do, and are used to compute statistics. Even if these statistics are colleted with the goal of learning usefull aggregate informations about the users/population, it might end in collecting and storing private data from client.
The need to collect data and to share them in a privacy-preserving way has become crucial in this context. We can for example illustrate this with the problem of Cloud leaking that happened several times in the past years.[ref]
Leaking private user data bring important privacy and security risks. They might be discloded, sold for profit or even be used by agencies for targeting and mass surveillance goals.
A lot of techniques have been developed through the years, by major technology companies such as Google [references], but also researcher in Universities [references].
However, by gaining *privacy*, these protocols sacrifice *robustness* and *scalability*, which are two important aspect to keep in mind while designing a decentralized system. Privacy is necessary so that no leak in the sensitive data happend, and robustness characterise the correctness of the computation. The trade-off between both should be reasonable, as we do not want any data to be leaked, and the server has to correctly compute under given circunstunces. This difficulties lead to the use of technical agreements rather than technical solutions.
Indeed, centralized system are still widely used [ref] as they are way simpler and use a trusted third party. But these trusted parties still is a single point of failure in the system and data provider begin to realize the value of their own datas.
This is why decentralized sytems are becoming more popular, and desirable.
In this paper, we present the implementation of Prio into Unlynx and compare them with input validation proof.

Contributions

Background

1 Unlynx System

- How does unlynx work (with a figure) (system of client, queries and servers)
- How Aggregation is done
- How proof is done

1.1 Model

Unlynx is a privacy-preserving data sharing system developed by LCA1 in collaboration with DeDiS.
It consists of a collective authority (CA) formed by a number m of server $S_1, ...S_m$, and n data providers $DP_1, ...DP_n$ containing sensitive data. These DPs combined represent a distributed database that is used to awnser queries made by a querier Q. The querier and DPs choose one server of the CA to communicate with and can change this choice at any given time.
Functionality: Unlynx should permit SQL queries of the form SELECT SUM(*)/COUNT(*) FROM DP,.... WHERE * AND/OR GROUP BY *, with any number of * clauses.
Privacy and Robustness: Both are assured if at least one server is trusted, as we use the fact that we are allowed to publish ciphertext and their aggregation to show that the computation at the server are actually correct. It leaks nothing as all data are encrypted.

1.2 Threats

Collective authority servers It is assumed an Anytrust model [REF]. It does not requires any particular server to be trusted or to be honest-but-curious. The moment it exists one server that is not malicious, functionality, security and privacy are guaranteed.
Data providers are assumed to be honest-but-curious. The system does not protect against malicious DPs sending false infomations, but a solution will be discuss in Section [INPUT RANGE VALID SECTION].
Queriers are assumed to be malicious, and can collude between themselves or with a subset of the CA servers.
It is also assumed tha all network communication is encrypted and authenticated, by using a protocol like TLS for example.

1.3 Pipeline and proof

The protocol start when a querier wants to retrieve some information about sensitive data. It sends the query to one of the server of the CA. Upon receiving, the server broadcast this query to the other servers in the collective authority.
From here the data are privately and securely processed by the CA, before sending back the result to the querier, encrypter over the public key of the Querrier. During all the steps of the protocol, the server will never get the data in clear.
The pipeline is the following: Encryption, Verification Shuffle, Distributed Deterministig Tag, Collective Aggregation, Distributed Results Obfuscation, Key Switch. At the end of this pipeline, the querier get the data and can decrypt them to get the aggregate statistics he asked for, without any server seeing the data in clear, or knowing from which data provider the data are from.

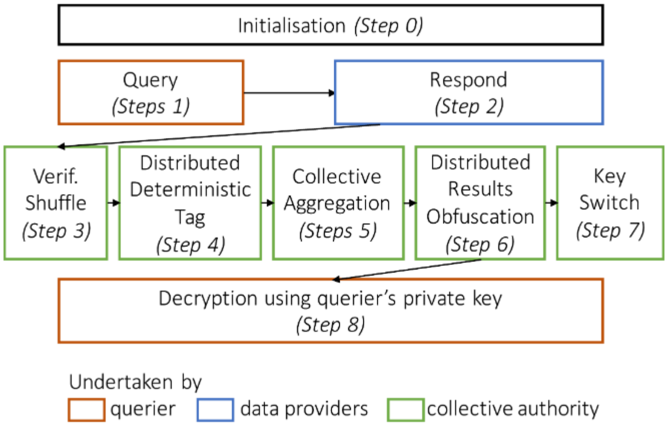
Initialisation *(Step 0)*
Query *(Steps 1)*
Respond *(Step 2)*
Verif. Shuffle *(Step 3)*
Distributed Deterministic Tag *(Step 4)*
Collective Aggregation *(Steps 5)*
Distributed Results Obfuscation *(Step 6)*
Key Switch *(Step 7)*
Decryption using querier's private key *(Step 8)*
Undertaken by
querier
data providers
collective authority

Figure 1: Unlynx query processing pipeline

Proofs are done with zero-knowledge systems, to preserve privacy. There is one for each state of the pipeline, for example when aggregating, it publishes the ciphertexts, and the result of their aggregation, as the data is encrypted using Elliptic curve ElGamal, it leaks nothing about the data alone, but there is no Input range proof , meaning the DPs can send fraudulous data (by giving a very big number, and as it is encoded we cannot verify it) and make the computation wrong. This input validation proof is implemented and described in section [SECTION]

2 Prio System

- How does Prio work (same as before)
- How does Aggregation is done
- New proof system SNIPs

2.1 Model

Prio is also a privacy-preserving data sharing system developed Standford University.
It consists of a collective authority (CA) formed by a number m of server $S_1, ...S_m$,and each data provider holds a private value x_i that is sensitive. Unlike Unlynx, Prio do not encrypt private value x_i that why it is a more challenging aggregation in terms of privacy.
Communication is assumed to be done in secure channels as previously described. **Functionality**: Prio should permit the collective authority to compute some aggregation function $f(x_1, ...x_n)$ over private values of data providers, in a way that leaks as little as possible about these, except what can be inferred from the aggregation itself. **Privacy and Robustness**: Privacy is assured if at least one server is trusted, but robustness is satisfied if and only if all server are trusted, as you cannot be assured that computation at the server are correct.

2.2 Threats

Unlynx is a privacy-preserving data sharing system developed by LCA1 in collaboration with DeDiS.
It consists of a collective authority (CA) formed by a number m of server $S_1, ...S_m$, and n data providers $DP_1, ...DP_n$ containing sensitive data. These DPs combined represent a distributed database that is used to awnser queries made by a querier Q. The querier and DPs choose one server of the CA to communicate with and can change this choice at any given time.
Functionality: Unlynx should permit SQL queries of the form SELECT SUM(*)/COUNT(*) FROM DP,.... WHERE * AND/OR GROUP BY *, with any number of * clauses.
Privacy and Robustness: Both are assured if at least one server is trusted, as we use the fact that we are allowed to publish ciphertext and their aggregation to show that the computation at the server are actually correct. It leaks nothing as all data are encrypted.

2.3 Threats

Collective authority servers In Prion it is needed that one server is not malicious and trusted, so that security and privacy are guaranteed.
Robustness against malicious server seems desirable but doing so would cost privacy and perfomance degradation, which is not desirable.
Data providers are assumed to be malicious. The system protects itself against malicious DPs. All data that does not pass the SNIPs proof, will be discarded.

2.4 Pipeline and proof

do

Prio SNIPs

- Client evaluation

- Consistency checking at the server
- Polynomial identity test
- Multiplication of shares
- Output verification

2.5 Input range validation for ElGamal

do

Implementation

- what is implemented in a little more detailled
- optimization apported by code from github and not aborded in details

Performance evaluation

Explain a little more about comparison and test settings

- Scaling with number of server
- Scaling with number of client
- Scaling with fairly high number of both
- Time and Bandwidth dilemna

El Gamal range input checking

- TO be seen

Conclusion

References