PUDA - Privacy and Unforgeability for Data Aggregation

HAKAM Mouad, SENANE Zineb, CARDOSO Emerson, SAHKOUN Ludovic, NEUFUSS Cristian

BigSec Final Report

1. Introduction

The paper presents a model for ensuring privacy and unforgeability as regard to the aggregation of data collected from multiple users.

First of all, it is important to define which actors are involved.

- **Users:** Those who produce (valuable) data and provide them encrypted to the aggregator. In this case, users are independent, in other words, they do not interact with each other.

- **Key Dealer:** It is in charge to publish the verification key (used to verify the correctness of the aggregate value) for the set of users, to provide an encryption key for each user (used by the users to encrypt data) and a secret key to the Aggregator

- **Aggregator:** Untrusted party that collects the encrypted data. As the focus is on aggregate data, the aggregator is able to compute the plaintext of an aggregate value (sum for example) of all the values (encrypted data) provided by the users given the corresponding ciphertexts. The output is forwarded to the Data Analyzer.

- **Data Analyzer:** It verifies the correctness of the Aggregator's value and it's the one who actually extracts the useful information about the set of users.

The key point about the Aggregator and the Data Analyzer is that neither of them learns only the value of the aggregation value.
We define these two requirements on which the model is based:

- **Aggregator obliviousness:** ensures that at the end of the protocol execution, the aggregator learns only the sum of the user's data and nothing else.

- **Aggregator unforgeability:** The aggregator cannot forge a valid proof for an aggregate value that was not computed correctly from the user's input.

In addition, the model provides an authentication mechanism (which will be discussed later) to enable the correctness of the aggregate value to be demonstrated.

2. Implementation of PUDA model

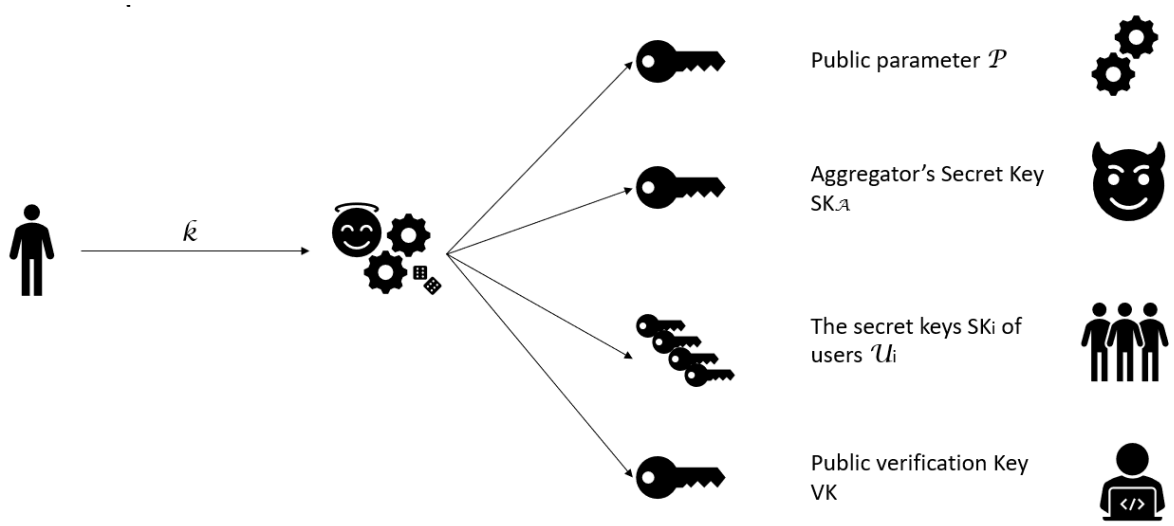
2.1. Setup

First, there is an initial phase of setup executed by the Key Dealer where the security public parameters are defined and used to produce the verification key and the secret keys of the User and the Aggregator.

For each efficient computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, the key dealer KD outputs:$(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. He selects a secret key **a**. Each user Ui selects a random tag key tk_i and forwards g_2^{tki} to the key dealer.

Next, the Key Dealer publishes $VK=(g_2^{\Sigma tki}, g_2^a)$, and sends to each user the secret key g_1^a secret keys of the scheme: $SK_i = (ek_i, tk_i, g_1^a)$.

.



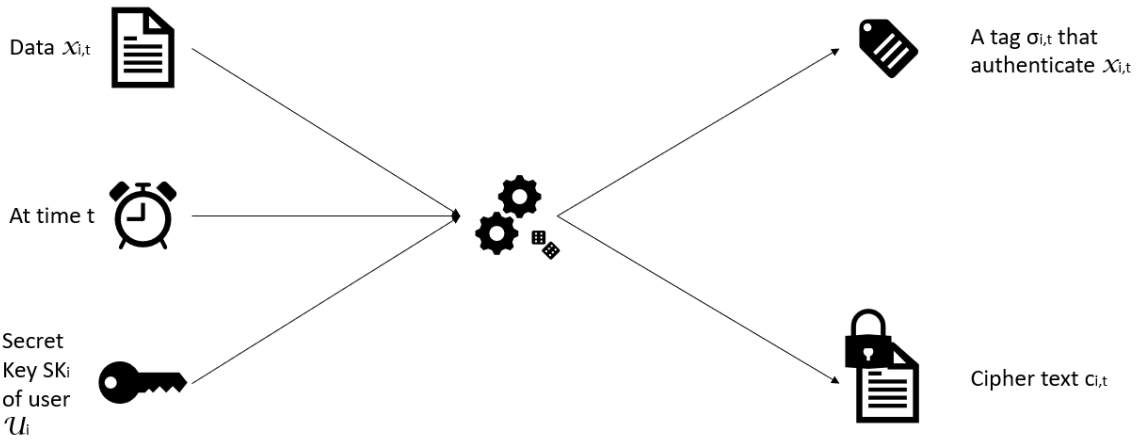
2.2. EncTag

Each user uses a deterministic algorithm that is executed periodically over a determined period of time. Using the User secret key, It will encrypt each data and compute their respective authentication tag.

EncTag(t,$SK_i = (ek_i, tk_i, g_1^a), x_{i,t}$**)=(**$c_{i,t}, \sigma_{i,t}$**)=(**$H(t)^{eki}\ g_1^{xi,t}, H(t)^{tki}\ (g_1^a)^{xi,t}$**) .**

The hash function used by this algorithm is based on the Shi-Chan-Rieffel-Chow-Song Scheme.

User Ui sends these values to the aggregator.

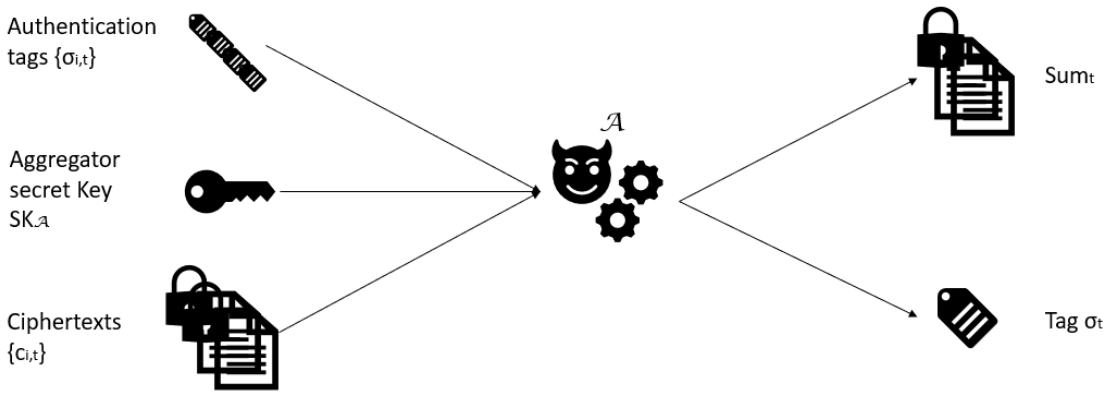


2.3. Aggregate

Once the Aggregator has received the ciphertexts and the authentication tags from the users, it will use an algorithm that for each user will output the aggregate value of the set of encrypted data and will compute a proof of correctness of this result (using the authentication tags). This algorithm relies on homomorphic encryption, which means that the computations can be executed on encrypted data without needing to decipher them.

Aggregate(SK_A, $\{c_{i,t}\}_{UicU}$, $\{\sigma_{i,t}\}_{UicU}$)=(sum_t,σ_t) ; sum_t = sum of data inputs ,σ_t= multiplication of tags.

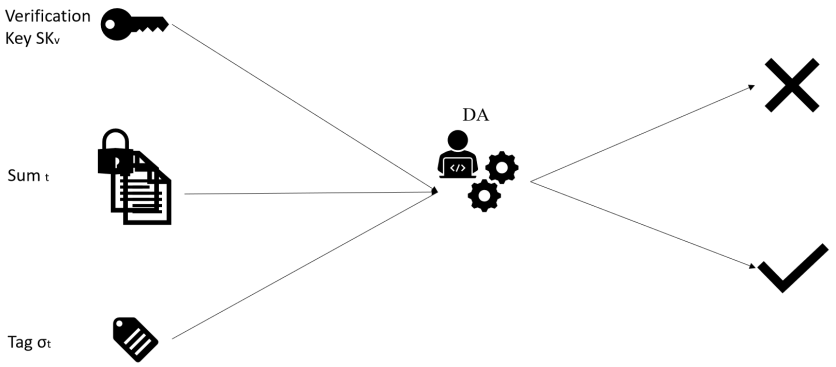
A forwards sum_t,σ_t to the Data Analyzer.



2.4. Verify

Finally, the Data Analyzer will use an algorithm that will use the verification key to establish if the aggregate value computed by the Aggregator is correct or not.

Verify(VK,sum_t,σ_t)=[e(σt , g_2) ?= e(H(t), vk1)e($g_1^{sum\ t}$, vk2)]



NB: All these algorithms are deterministic (given the same output, they will produce the same output) except the one used for deriving the security parameters.

3. PUDA key Assumptions

The security of the PUDA model is measured via oracle access games.
We assume that PUDA protocol ensures aggregator obliviousness if :

$$\mathbf{Pr(Aau) \leq 1/2 + \varepsilon(k).}$$

and ensures aggregate unforgeability if:

$$\mathbf{Pr(Aau) \leq \varepsilon(k).}$$

NB: P(Aau) for the obliviousness game is the probability of predicting the exact b as OracleAO algorithm and P(Aau) for the unforgeability game is the probability of having a forgery(type 1 or 2).

Aggregator Obliviousness: The proposed PUDA model achieves aggregator obliviousness in the random oracle model under the decisional Diffie-Hellman assumption

Aggregate Unforgeability:

- The scheme achieves unforgeability against a Type I Forgery under Bilinear computational Diffie Hellman assumption in the random oracle model
- The scheme guarantees aggregate unforgeability against a Type II forgery under the LEOM assumption in the random oracle

Type I forgery: The aggregator outputs a sum that achieves the verification without even having access to the ciphertexts and tags that are necessary for aggregation and computing the sum (without making a query OEncTag).

Type 2 forgery: The aggregator makes a query to OEncTag to get ciphertexts and tags, and outputs a forged sum that achieves the verification.

4. PUDA Strengths

PUDA's principal strength is that it introduces a model which enables the verification of the correctness of the data, without sacrificing privacy. This is achieved through Aggregator Obliviousness and a novel property termed "Aggregate Unforgeability" which is resilient against Type 1 and Type 2 forgery, under the assumptions mentioned in 3.
The operations to perform the PUDA protocol are furthermore efficient and create a limited overhead in the scale of milliseconds. However, that overhead arises for each data input, which could lead to a computational cost that is not negligible. Finally, the verification of the correctness is possible in public in constant time.

5. PUDA Weakness and limitation

The PUDA model makes a couple of assumptions which can be seen as its limits. For once, it assumes that the Aggregator is the only entity with adversarial behaviour. This is justified by the fact that the Aggregator is the only actor who sees all the messages exchanged during the protocol execution. It also relies on a third-party Key Dealer that has to be trusted. The Key Dealer is responsible for establishing a distributed key system. If the Key Dealer acted maliciously, the whole protocol would be compromised. In order to achieve data unforgeability the aggregator can only query or call Oracle Encryption only once for the same user in the same interval which makes it very difficult for the aggregator to send erroneos sum to the data Analyzer.

6. Related works & Improvements

Considering the weaknesses of the model, it is easy to think that the model can be modified with the aim of improving it. In particular, we want to study a mechanism that allows the correct functioning of the scheme even without pre-assuming that the key dealer and the users are trusted.
Another future work may also exist on extending the PUDA model to deal with privacy even if users are not independent and can interact with each other.