Localisation and Sensor Privacy Using the Extended Information Filter and Secure Weighted Aggregation ⋆

Marko Ristic [a], Benjamin Noack [a], Uwe D. Hanebeck [a]

[a] *Intelligent Sensor-Actuator-Systems Laboratory, Institute for Anthropomatics, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany*

Abstract

Distributed state estimation and localisation methods have become increasingly popular with the rise of ubiquitous computing, and have led naturally to an increased concern regarding data and estimation privacy. Traditional distributed sensor navigation methods involve the leakage of sensor information or navigator location during localisation protocols and fail to preserve participants' data privacy. Existing approaches which provide such guarantees, fail to address sensor and navigator privacy in some common model-based non-linear measurement localisation methods forfeiting broad applicability. We define a cryptographically secure weighted aggregation scheme which we apply to the Extended Kalman Filter with range-sensor measurements, and show that navigator location, sensor locations and sensor measurements can remain private during navigation. The security requirements, leakage, and cryptographic proof are given for the private filter and weighted aggregation scheme, and simulations of the filter are used to evaluate the accuracy and performance of the method. Our approach defines a novel, computationally plausible and cryptographically private, model-based localisation filter with direct application to environments where nodes may not be fully trusted and data is considered sensitive.

Key words: System state estimation; Data privacy; Sensor fusion; Kalman filters.

1 Introduction

–

Introduce localisation, filtering and the need for privacy.

Examples of environments where privacy is relevant and concrete examples where lack of privacy could have large costs

Methods for introducing security and privacy include differential privacy methods and encryption methods.

Differential privacy involves using statistical noise as security to make individual users' information cannot be deduced. Often requires a trusted aggregator, although secure aggregation methods exist. always requires noising result such that the outcome is not exact (a problem in localisation).

Encryption schemes involve formal indistinguishability proofs typically over bits or integers. They rely on computationally hard problems involving security parameters of a sufficiently large size; therefore the additional computational requirements of using encryption schemes should be pointed out and what this means in a real-time distributed sensor system. Continuing, explain public-key cryptography applicability to distributed systems; difference to symmetric schemes. Homomorphic encryption power and use case. Why FHE isn't used often, why additive partially homomorphic encryption is.

Advancements in function providing encryption schemes such as homomorphic encryption have also led to several other types of schemes which have found uses in signal processing. Private aggregation schemes allow the secure computation of the sum of encrypted values originating from different parties, leaking only the final result. When considering such multi-party encryption protocols, formal security definitions must now also incorporate the added dangers of colluding malicious parties, and lead to new notions of security. For example Aggregator Obliviousness (AO) is typically proven for private aggregation schemes, while alternatives such as Private Weighted Se-

⋆ This paper was not presented at any IFAC meeting. Corresponding author Uwe D. Hanebeck. Tel. +49–721–608–43909.

Email addresses: marko.ristic@kit.edu (Marko Ristic), noack@kit.edu (Benjamin Noack), uwe.hanebeck@kit.edu (Uwe D. Hanebeck).

cure Aggregator Obliviousness (pWSAO) exist for other specific use-cases.

Another example of function providing encryption, and a generalisation of private aggregation, is called functional encryption (FE) and its distributed extension, multi-client functional encryption (MCFE), which allow the unencrypted result of an arbitrary function to be computed from encrypted inputs. General FE and MCFE are known to be quite computationally expensive (from meeting with ITI and student Johannes - need ref.) but alternatives providing only a subset of possibly computable function exist; for example, inner product encryption.

Several of the aforementioned encryption schemes have found uses in secure localisation, estimation, and control.

1.1 Relevant Literature on Encrypted Localisation and Estimation

Model-free localisation using homomorphic encryption examples include polygon thing, WSN examples which protect against adversaries but in the case of the WSN paper. don't preserve anchor privacy. Importantly, model-based filtering and localisation provide more accurate estimates and these are not applicable there.

Model-based estimation examples include Aristov paper (which requires a linear model, and a hierarchy of sensors), Farokhi paper (which requires the controller compute entirely in encrypted space and send input back to actuator - supporting only the cloud-as-a-service type architectures) and Alexandru paper (which implements a distributed control environment but requires a constant gain matrix K)

pWSAO achieved in Alexandru weighted aggregation, but requires redistributing keys at every timestep resulting in a costly operation, and a complicated communication protocol.

In addition to applying suitable encryption schemes to signal processing tasks, care must be taken when converting sensor output into an encryptable homomorphic format. As is the case with our proposed localisation method, real number sensor output does not trivially encode to integers such that the homomorphic properties provided by an additive encryption scheme over integers keep the underlying real numbers consistent. Methods for handling the encoding of real numbers such that they can be used in homomorphic encryption exist. Google bignum adds power but risks overflow and leaks exponents, Farokhi leaks no information but allows only a single multiplication (extendable to more but each further multiplication limits the real number size and increases the risk of overflow).

Briefly describe navigator scenario and our contributions

Section Summary

1.2 Notation

Notation

2 Problem Statement

Restate the scenario but more formally. Give a concrete example - plane and signal towers.

Exact security guarantees we aim for, as well as the definitions for these guarantees (pWSAO and indistinguishability but in context of localisation as well). Note that learning only the sum in aggregation (as is normal in AO) would, in this case, tell the navigator the average location and measurements of all sensors, which is fine as it does not disclose any exact sensor.

Passive attacks only from sensors to learn navigator position (Otherwise one could do some kind of attack that would send a fake measurement and note the change in its own measurements - possible this would give away the average of other sensors' measurements but unclear). Any largely incorrect inputs from sensors may also be detectable by comparison to alternative navigator onboard sensors (GPS etc.). Justify by saying sensors need to behave for localisation to work in the first place.

Active attacks from navigator to find sensor location allowed, but assume that weights sent to all sensors are the same. In a wireless setting, all sensors would receive all broadcast weights anyway. While special hardware, which may support directional broadcasting or receiving, could be used to locate sensors individually this is beyond the scope of what is considered in our problem.

Point out that learning the aggregation of sensor outputs, which contains measurement and location information also means that the average location and measurement of the sensors may be leaked, and is accepted as a part of the leakage as it is inferrable from the aggregation scheme and any functioning model-based localisation where measurements are not known

Rough computational capabilities expected by parties

Fixed sensor subsets of which only whole subsets can be used at once. Maybe a picture of what this might look like in a high level distributed localisation diagram. Should consider that this sub-grouping would also mean the leakage of the average sensor/measurement of each subset not all sensors at once. This should be considered when choosing sensor subsets and locations.

3 Privacy-Preserving Weighted Aggregation

For achieveing the goal of private localisation defined in Section 2 we require a cryptographic scheme which allows a time-series of homomorphically computed linear combinations of encrypted weights, to be summed by a privacy-preserving aggregation scheme. In our case, at time t, weights $\omega_1^{(t)}, \ldots, \omega_m^{(t)}$ are encrypted and broadcast by the navigator, linear combinations $y_i^{(t)} = \sum_{j=1}^{m} x_{j,i}^{(t)} \omega_i^{(t)}$ are computed by the sensors $i \in \{1, \ldots, n\}$, and aggregation is computed back at the navigator. This has been summarized in Figure 1. In Section 6 we will show how this scheme can be used to compute measurement covariances and measurement vectors homomorphically, and update the Extended Information Filter while preserving privacy.

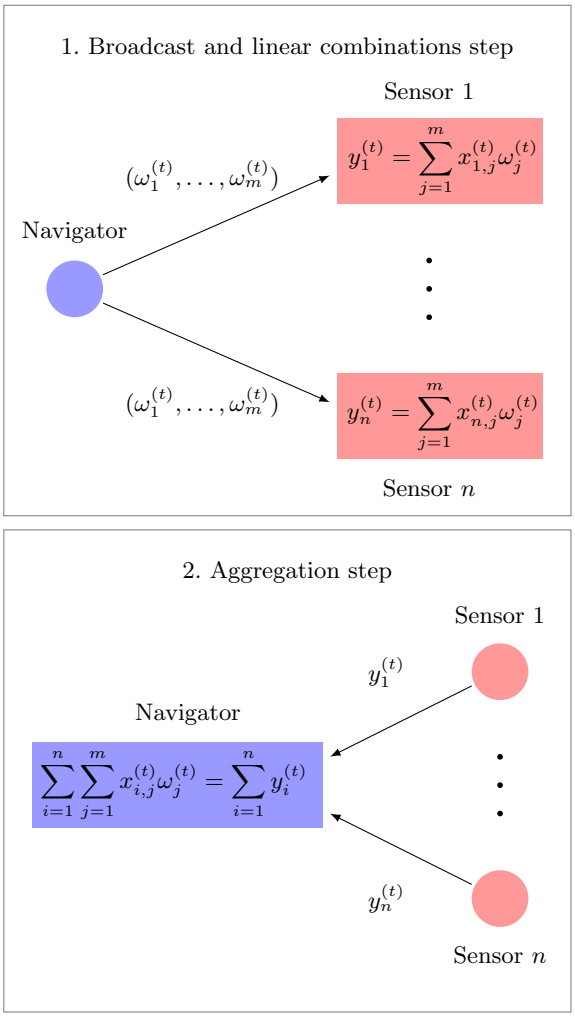

Fig. 1. Required weighted aggregation steps at time t

The security of a cryptographic scheme is typically defined by a security *game*, which captures both the desired privacy guarantees, as well as the capabilities of attackers [2]. Before defining the desired security of our scheme, we note that it has properties of both an additive homomorphic encryption scheme, and a privacy-preserving aggregation scheme. The typical security requirement for a homomorphic encryption scheme is Indistinguishability under the Chosen Plaintext Attack model (IND-CPA) []. Informally, IND-CPA states that an attacker who can choose plaintext messages to be encrypted at will, gains no additional information about an unknown plaintext message when they learn only its encryption. The formal security game for IND-CPA has been given in Appendix A. Privacy-preserving aggregation, without weights, aims for the security goal of Aggregator Obliviousness (AO) []. AO states that no colluding subset of participants *excluding* the aggregator gains additional information about the remaining aggregation values given only their encryptions, while any colluding subset *including* the aggregator learns only their sum. The security game for AO has been given in Appendix B.

The desired security of our weighted encryption scheme has two goals:

Indistinguishable Weights No colluding subset of sensors gains any additional knowledge about the navigator weights ω_j from receiving only their encryptions, and encryptions from any previous timesteps.

Private Linear Combination Aggregation No colluding subset *excluding* the aggregator gains additional information about the remaining sensor values to be weighted $x_{i,j}^{(t)}$, where sensor i is not colluding, given only encryptions of their linear combinations y_i and encryptions from any previous timesteps. Any colluding subset *including* the aggregator learns only the sum of all linear combinations weighted by weights of their choice $\sum_{i=n}^{n} \sum_{j=1}^{m} x_{i,j}^{(t)} \omega_j^{(t)}$. Note that this implies that $\sum_{j=1}^{m} x_{1,j}^{(t)}$ may be learned by the colluding subset given weights $(1, 0, \ldots, 0)$, but that individual sensor values $x_{i,j}^{(t)}$ cannot be learned regardless of weight choice.

From these definitions it is clear that IND-CPA security of weight encryptions is sufficient for the first requirement, while AO is not sufficient for the second. Instead, we give a definition for linear-combination aggregator oblivious Encryption, and the accompanying security game which captures the additional weights, and modified leakage.

3.1 Linear-Combination Aggregator Oblivious Encryption

We define a linear-combination aggregator oblivious encryption scheme as a tuple of the four algorithms (Setup, Enc, CombEnc, AggDec):

Setup(κ) On input of security paramater κ, generate the aggregator's public and private keys pk_0 and sk_0, and the remaining user private keys sk_i, $(1 \leq i \leq n)$.

Enc($\mathrm{pk}_0, \mathrm{sk}_0, \omega$) At time t, the aggregator encrypts weight ω with its public key pk_0, and obtains the encryption $\mathcal{E}_{pk_0}(\omega)$.

CombEnc($\mathrm{pk}_0, \mathrm{sk}_\mathrm{i}, \mathcal{E}(\omega_1), \ldots, \mathcal{E}(\omega_\mathrm{m}), \mathrm{x}_{\mathrm{i},1}, \ldots, \mathrm{x}_{\mathrm{i,m}}$) At time t, user i computes the encrypted linear combination $y_i = \mathcal{E}_{pk_0,sk_i}(\sum_{j=1}^{m} x_{i,j}\omega_j)$ using its secret key sk_i as $y_i = \mathrm{CombEnc}(\mathrm{pk}_0, \mathrm{sk}_\mathrm{i}, \mathcal{E}(\omega_1), \ldots, \mathcal{E}(\omega_\mathrm{m}), \mathrm{x}_{\mathrm{i},1}, \ldots, \mathrm{x}_{\mathrm{i,m}})$.

AggDec($\mathrm{pk}_0, \mathrm{sk}_0, \mathrm{y}_1, \ldots, \mathrm{y}_\mathrm{n}$) At time t, the aggregator computes the aggregation of linear combinations using its private key sk_0, as $\sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j}\omega_j = \mathrm{AggDec}(\mathrm{pk}_0, \mathrm{sk}_0, \mathrm{y}_1, \ldots, \mathrm{y}_\mathrm{n})$.

We define the security notion of Linear-Combination Agregator Obliviousness (LCAO) as the following game between attacker and challenger:

Setup The challenger runs the Setup algorithm and gives pk_0 to the attacker

Queries The attacker can now perform encryptions or submit queries that are answered by the challenger. The three types of actions are:

(1) *Encryption:* The attacker chooses a weight ω and computes an encryption of ω under the aggregator's public key pk_0, obtaining $\mathcal{E}_{pk_0}(\omega)$.

(2) *Combine Queries:* The attacker chooses a tuple $(i, t, \mathcal{E}_{pk_0}(\omega_1), \ldots, \mathcal{E}_{pk_0}(\omega_m), x_{i,1}, \ldots, x_{i,m})$ for a unique pair (i, t) and is given back the encryption of the linear combination $\mathcal{E}_{pk_0,sk_i}(\sum_{j=1}^{m} x_{i,j}\omega_j)$ encrypted under both the aggregator public key pk_0 and thesecret key sk_i.

(3) *Compromise queries:* The attacker chooses i and receives the secret key sk_i. The aggregator's secret key may also be compromised (when choosing $i = 0$).

Challenge Next, the attacker chooses a time t^*, and a subset of users $U \subseteq S$ where S is the complete set of users for which no combine queries for time t^* and no compromise queries are made for the duration of the game. The attacker then chooses two series of tuples

$$\langle (i, t^*, \mathcal{E}_{pk_0}(\omega_1), \ldots, \mathcal{E}_{pk_0}(\omega_m), x_{i,1}, \ldots, x_{i,m}) \mid i \in U \rangle$$

and

$$\langle (i, t^*, \mathcal{E}_{pk_0}(\omega'_1), \ldots, \mathcal{E}_{pk_0}(\omega'_m), x'_{i,1}, \ldots, x'_{i,m}) \mid i \in U \rangle ,$$

and gives them to the challenger. In the case that $0 \in U$ (*i.e.* the aggregator is compromised) and $U = S$, it is additionally required that

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j}\omega_j = \sum_{i=1}^{n}\sum_{j=1}^{m} x'_{i,j}\omega'_j .$$

The challenger then chooses a random bit $b \in \{1, 0\}$ and returns encryptions

$$\langle \mathcal{E}_{pk_0,sk_i}(\sum_{j=1}^{m} x_{i,j}\omega_j) \mid i \in U \rangle$$

if $b = 0$, or

$$\langle \mathcal{E}_{pk_0,sk_i}(\sum_{j=1}^{m} x'_{i,j}\omega'_j) \mid i \in U \rangle$$

otherwise.

More Queries The attacker can now perform more encryptions and submit queries, so long as the queries do not break the requiremets in the Challenge stage. That is, $U \subseteq S$.

Guess At the end, the attacker outputs a bit b' and wins the game only if $b' = b$. The advantage of attacker $\mathcal{A}$ is defined as

$$\mathrm{Adv}^{\mathrm{LCAO}}(\mathcal{A}) \coloneqq \left| \Pr[\mathrm{b}' = \mathrm{b}] - \frac{1}{2} \right| .$$

We say that an encryption scheme meets the security of LCAO if no probabilistic-time adversary has more than negligable advantage in winning this security game.

Our weighted aggregation method relies on the decisional composite residuosity assumption (DCRA) originally defined by Paillier in [3]. The aggregation method can be seen as a combination of the additive homomorphic Paillier encryption scheme, which additionally allows multiplication of an encrypted message with a plaintext value, and the Joye-Libert private stream-aggregation scheme. Both of these schemes base their security on the DCRA, and their descriptions are given briefly below before defining our weighted aggregation scheme.

3.2 Paillier Encryption Scheme

The Paillier encryption scheme is an additively homomorphic encryption scheme which functions over a finite group $\mathbb{G}$, containing

Key generation of the Paillier scheme is performed by choosing two sufficiently large primes p and q, and computing $n = pq$. A generator g is also required for encryption, which is often set to $g = n + 1$ when p and q are of equal bit length []. The public key is then given by (n, g) and secret key by (p, q).

Encryption of a plaintext message $m \in \mathbb{Z}_n$, producing ciphertext $c \in \mathbb{Z}^*_{n^2}$, is computed by

$$c = g^m r^n \pmod{n^2} \tag{1}$$

for a randomly chosen $r \in \mathbb{Z}_n$. The r^n term can be considered as the noise which hides the value $g^m \pmod{n^2}$, which due to the scheme construction, is an easily computable discrete logarithm. The decryption of a ciphertext is computed by

$$m = \frac{L(c^\lambda \pmod{n^2})}{L(g^\lambda \pmod{n^2})} \pmod{n} \tag{2}$$

where $\lambda = \mathrm{lcm}(\mathrm{p}-1, \mathrm{q}-1)$ and $L(u) = \frac{u-1}{n}$.

In addition to encryption and decryption, the following homomorphic functions are provided by the Paillier scheme. $\forall m_1, m_2 \in \mathbb{Z}_n$,

$$\mathcal{D}(\mathcal{E}(m_1)\mathcal{E}(m_2) \pmod{n^2}) = m_1 + m_2 \pmod{n} \tag{3}$$
$$\mathcal{D}(\mathcal{E}(m_1)g^{m_2} \pmod{n^2}) = m_1 + m_2 \pmod{n} \tag{4}$$
$$\mathcal{D}(\mathcal{E}(m_1)^{m_2} \pmod{n^2}) = m_1 m_2 \pmod{n}. \tag{5}$$

The scheme is shown to be indistinguishable under the chosen plaintext attack model (IND-CPA), which is considered the strongest form of security acheivable by a homomorphic encryption scheme [].

3.3 Joye-Libert Privacy-Preserving Aggregation

The Joye-Libert private stream aggregation scheme allows, at each time-step, encrypted time-series values originating from N different measuring parties, to be summed up and decrypted by an aggregator. Time-series values are encrypted by their measuring party's private encryption key, and decrypted by the aggregator using its private key. The scheme functions over the same finite group $\mathbb{G}$ as the Paillier encryption scheme above.

A notable difference in key-generation for the aggregation scheme from a public-key scheme such as Paillier's scheme, is the requirement of an offline generation and distribution step by some trusted party. Similar to the Paillier scheme, key generation first requires choosing two sufficiently large, equal length, primes p and q and computing $n = pq$. In addition to n, a hash function $H : \mathbb{Z} \to \mathbb{Z}_{n^2}$ is defined, and the public key is set to (n, H). The private keys are generated by choosing N keys sk_i, $i \in \{1, \ldots, N\}$ uniformly from $\mathbb{Z}_{n^2}$ and distributing them to all measuring parties, while the last key is set to

$$sk_0 = -\sum_{i=1}^{N} sk_i \pmod{n^2}, \tag{6}$$

and sent to the aggregator.

Encryption of plaintext $m_{i,t} \in \mathbb{Z}_n$ to ciphertext $c_{i,t} \in \mathbb{Z}_{n^2}$ at time t is computed by participant i as

$$c_{i,t} = (n+1)^{m_{i,t}} H(t)^{sk_i} \pmod{n^2}, \tag{7}$$

As with the interpretation of the noise term in the Paillier scheme, here $H(t)^{sk_i}$ can be seen as noise which hides the easily computable discrete logarithm $g^{m_{i,t}} \pmod{n^2}$, where $g = n + 1$.

When all encryptions $c_{i,t}$, $i \in \{1, \ldots, N\}$ are sent to the aggregator, private summation and decryption are computed by the functions

$$c_t = H(t)^{sk_0} \prod_{i=1}^{N} c_{i,t} \pmod{n^2} \tag{8}$$

and

$$\sum_{i=1}^{N} m_{i,t} = \frac{c_t - 1}{n}. \tag{9}$$

Correctness follows from $\sum_{i=0}^{N} sk_i = 0$, and therefore

$$H(t)^{sk_0} \prod_{i=1}^{N} c_{i,t} \pmod{n^2} \tag{10}$$
$$\equiv H(t)^{sk_0} \prod_{i=1}^{N} (n+1)^{m_{i,t}} H(t)^{sk_i} \pmod{n^2} \tag{11}$$
$$\equiv H(t)^{\sum_{j=0}^{N} sk_j} \prod_{i=1}^{N} g^{m_{i,t}} \pmod{n^2} \tag{12}$$
$$\equiv (n+1)^{\sum_{i=1}^{N} m_{i,t}} \pmod{n^2} \tag{13}$$

which results in a solvable discrete logarithm by using (9), as defined in [1], with no more noise term.

The aggregation scheme is proven to achieve aggregator obliviousness (AO), which in that no corrupted colluding subset learns anything about the individual messages of the non-corrupted participants, other than the final sum of messages if the aggregator is corrupted, and is proven in the random oracle model.

4 Our Scheme

Explain it in as an overview

4.1 Proof

Give the reduction proof here for pWSAO and implicit indistinguishability of weights. Alternatively, sketch it out here and give reduction proof in the appendix.

5 Private Localisation

5.1 Integer Encoding for Real Numbers

5.2 Extended Information Filter

6 Private Localisation with Privacy-Preserving Sensors

Explain it in as an overview. How is the aggregation scheme used, what does this require from the measurement model, why can this be a problem for normal distance sensors?

Explain how leakage of the final aggregation sum to the navigator means leakage of the average sensor location and measurement to the navigator. This is the reason for the acceptance of this leakage, as we pointed out in the problem statement section.

6.1 Requirements for Measurement Model

6.2 Localisation Measurement Modification

Show here the weighted integrals that give mean and variance of the new noise. If wanting to show more working, do this in the appendix section, but probably not needed.

Point out here that the further away the sensor is when it makes its distance measurement (the larger the measurement) the more Gaussian the noise and the better the filter. Give flight navigation as an applicable example with typically high distances.

Additionally increased range accuracy may be possible when sensors know the process model of the navigator, allowing them to run their own filter (more accurate than only measurements but not as accurate as the navigator's estimate from multiple sensors) and use their filtered estimated distance as the scaling factor when computing the modified measurement variance.

6.3 Expanding Aggregation for Multi-dimensional Inputs

Give 1D example that's intuitive (with a^2b) and then reduce the equivalent ND case ($A^\top BA$) to a set of weighted sums.

Ensure that timestamps are concatenated with the position so that no aggregation values are blinded by the same noise.

6.4 Algorithm

Piece together the whole algorithm here. Give the algorithm as pseudocode (including encoding and encryption)

7 Results

Decide on what kind of simulations and which plots to make. run times would be nice this time around

Time results can be captured in one graph. Y-axis is time, X-axis is the number of sensors, each line (different colour) will show how the runtime changes as sensors are increased for different Paillier bit-sizes (at least 3: 512, 1024, 2048). Every data point should be the average over some X number of simulations.

Accuracy plots will describe error due to encoding and the average distance of the sensors to the navigator. All plots will use the same ground truth and initial state and covariance estimates (this way average error at each timestep from multiple runs makes sense).

Plot 1 will plot the RMSE of the average of X runs at each encoding size. A fixed layout of 4 mediumly spaced sensor will be used, and a fixed Paillier bit size.

Plot 2 will plot the RMSE as the average distance of sensors changes. Fixed encoding and Paillier bit size. Vary between 4 layouts where 4 sensors are either very close to the centre (and ground truth, and progressively further out)

Plot 3 will accompany plot 3 and display the 4 layouts (arrow for ground truth and points for the sensors).

A well-defined example.

8 Conclusion

Possible future work to consider writing here: Hardware implementations, measurement handling which preservers Gaussian noise, or non-Gaussian noise methods, ways of sending less information form the navigator to the sensors at each time step, active sensor attacker model, different state encryptions received at sensors.

Acknowledgements

Partially supported by the Roman Senate.

References

[1] Marc Joye and Benoît Libert. A Scalable Scheme for Privacy-Preserving Aggregation of Time-Series Data. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, and Ahmad-Reza Sadeghi, editors, *Financial Cryptography and Data Security*, volume 7859, pages 111–125. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[2] J. Katz and Y. Lindell. *Introduction to Modern Cryptography: Principles and Protocols*. Chapman & Hall, 2008.

[3] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 223–238. Springer, 1999.

A Indistinguishability under the Chosen Plaintext Attack (IND-CPA)

B Aggregator Obliviousness (AO)