VeraGreg: A Framework for Verifiable Privacy-Preserving Data Aggregation

Jakub Klemsa, Lukáš Kencl and Tomáš Vaněk
Faculty of Electrical Engineering,
Czech Technical University in Prague
Prague, Czech Republic
{jakub.klemsa, lukas.kencl, tomas.vanek}@fel.cvut.cz

***Abstract*—A lot of effort has been made to devise a scheme for verifiable and privacy-preserving outsourcing of arbitrary computations. However, such schemes rely on Fully Homomorphic Encryption which is still far from practical. In our work, we instead focus solely on encryption schemes with single homomorphic operation, in particular addition. We define a rigorous framework that gives the data originator a possibility to check what values have been incorporated within provided homomorphic aggregate. We also propose a practical scheme that instantiates this framework and prove that it achieves *Indistinguishability under Non-Adaptive Chosen Ciphertext Attack* (IND-CCA1). The definition of our framework led us further to a straightforward modification of the security notions of *Non-Malleability* (NM) and *Adaptive Chosen Ciphertext Attack* (CCA2). Our modification aims at preventing trivial breach which is by principle unavoidable for plain homomorphic encryption. With our enhancement, the notions of security can serve as a novel security goal for any future verifiable homomorphic schemes.**

***Index Terms*—verifiable data aggregation, homomorphic encryption, formal security.**

I. Introduction

In the current age of massive data collection and powerful processing tools, privacy is becoming a serious concern. There are emerging first regulations that intend to limit the scope of data that can be collected about specific person, e.g., General Data Protection Regulation (GDPR, [1]) in the EU. Not only the definition of personal data is very broad, advanced profiling techniques might be further used for extraction of possibly sensitive personal information from seemingly innocent data. Hence it is highly important to limit the scope of the data provided to a service as much as possible—only data strictly necessary for its smooth functionality should be provided.

However, in order to evaluate the data requested by the service, entry-level raw data might need to be processed at a user's device without unveiling to the service. This might be challenging or even impossible for some IoT devices. The only other option would be outsourcing of data storage and/or processing. To outsource arbitrary computation while keeping the data hidden from the service, Fully Homomorphic Encryption (FHE) would be needed. However, current FHE schemes are too demanding for the processing party, hence we focus solely on partially homomorphic schemes, in particular Additively Homomorphic Encryption (AHE). We present a real-world scenario where the service is allowed to learn sums of specific sets of values but not the individual values. We consider a device that measures possibly sensitive user data and has limited resources, hence needs to outsource data storage and processing—such processing where addition operation is sufficient for its functionality.

A smart electricity meter may serve as an example of a real application since it measures sensitive user data—detailed electricity consumption is not something one would like to publish. Ideally, one would not like to share it with the electricity provider either, however, for accounting purposes, the provider needs to know at least certain sums of these values, according to the actual tariff. Hence, the device encrypts the consumption by means of AHE and sends it to the provider for storage. Once needed, the provider sums required ciphertexts and sends the aggregate back to the device for decryption.

At this point, it would be desirable to provide the device with means of verification that the aggregate indeed consists of an adequate set of values, possibly claimed on a separate list. For this purpose, the device incorporates a unique identification (ID) into each ciphertext and sends the unencrypted ID alongside the ciphertext to the provider. The provider then accompanies each aggregate ciphertext with a list of incorporated ID's before sending back to the device. The framework we define must obviously guarantee that the provider is neither able to modify, nor remove the ID's from any ciphertext so that the device is able to detect any forgery.

Once the device is provided with the ability to check what values are included inside an aggregate ciphertext, it can be also given the possibility of refusing some of them. E.g., any single value could be considered as a sum as well, however, it might not be desired to unveil it. For this purpose, each application will define an *aggregating policy* which will state what sets of values are the only allowed for making sums of.

We call the proposed framework the VeraGreg *Framework* which refers to *VERifiable AGGREGates*; see Figure 1 for an illustration.

Paper Outline

First, we mention previous and related work in Section II. In Section III, we define the VeraGreg Framework about which we state basic impossibility results in Section IV. Based

This work was supported by Electrolux through its Student Support Programme.

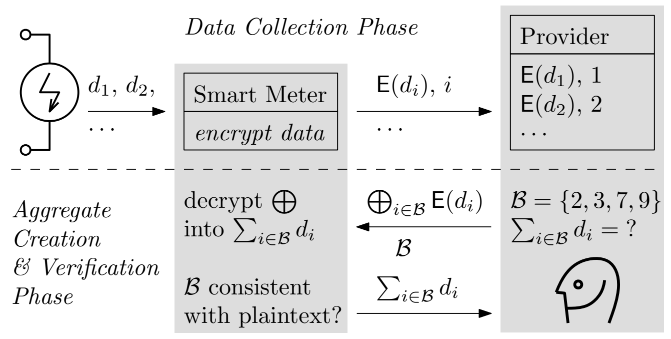
Data Collection Phase
Provider
d_1, d_2, …
Smart Meter
encrypt data
$\mathsf{E}(d_i)$, i
…
$\mathsf{E}(d_1)$, 1
$\mathsf{E}(d_2)$, 2
…
Aggregate Creation & Verification Phase
decrypt $\bigoplus$ into $\sum_{i\in\mathcal{B}} d_i$
$\bigoplus_{i\in\mathcal{B}} \mathsf{E}(d_i)$
$\mathcal{B}$
$\mathcal{B} = \{2,3,7,9\}$
$\sum_{i\in\mathcal{B}} d_i = ?$
$\mathcal{B}$ consistent with plaintext?
$\sum_{i\in\mathcal{B}} d_i$

Fig. 1. Example of VERAGREG utilization in the IoT. Left: classical meter, center: smart meter (aka. device), right: electricity provider.

on these results, we define novel notions of security and show an equivalence analogous to that of NM-CCA2 $\iff$ IND-CCA2. Next we propose an instantiation of the VERAGREG Framework in Section V which we further discuss in Section VI. In Section VII we outline an approach how to prevent the service from learning too much about individual values by solving a system of linear equations. Finally we propose possible future directions in Section VIII.

II. RELATED WORK

Let us remind the properties of plain Fully- and Additively Homomorphic Encryption (FHE and AHE, respectively). AHE is a public key cryptosystem that enables for addition over encrypted data using the public key. I.e., it is possible to combine encrypted values $\mathrm{AHE}(d_1)$ and $\mathrm{AHE}(d_2)$ into a ciphertext that decrypts into $d_1 + d_2$. We will denote the ciphertext addition operation by "$\oplus$" while it holds

$$\mathrm{AHE}^{-1}\big(\mathrm{AHE}(d_1) \oplus \mathrm{AHE}(d_2)\big) = d_1 + d_2. \tag{1}$$

In addition to AHE, FHE allows for multiplication over encrypted data. Note that it is possible to express any computation in terms of addition and multiplication, hence FHE allows for arbitrary computation over encrypted data.

However, any plain homomorphic encryption is inherently malleable—it does not offer yet any means of verification that the computation has been performed honestly. This problem has been studied for FHE—Gennaro et al. [2] introduced a non-interactive protocol for verifiable privacy-preserving computation. Later, improvements (Tang et al. [3]), modifications (Boneh et al. [4]) or other approaches (Liu et al. [5]) have emerged.

Regarding FHE, it had been a long standing open problem whether it exists until the work of Gentry [6]. Since then, several improvements have been proposed, however, neither of them achieves practically usable performance. In contrast, we consider only addition which enables us to use AHE while there exist several practical schemes (e.g., Paillier cryptosystem [7]). Moreover, for a verification that addition has been performed honestly, only a list of ID's of involved values and their multiplicity is sufficient which will allow for a simpler construction.

III. THE VERAGREG FRAMEWORK

We will be using the following notations and symbols:

- for a finite set A, let $A^* = \bigcup_{n=0}^{\infty} A^n$,
- for a group G; $g_1, g_2, \dots, g_n \in G$, let $\langle g_1, g_2, \dots, g_n \rangle$ denote a subgroup generated by $g_1, g_2, \dots, g_n$,
- for a function $f\colon \mathbb{N} \to \mathbb{R}^+$, we say that f is
  - negligible if $\forall c > 0\ \exists k_c\ \forall k > k_c\ f(k) < k^{-c}$,
  - overwhelming if $1 - f$ is negligible, $f \in \mathsf{OW}$,
- let $\div$ denote integer division.

In the framework we define (cf. Figure 1), we consider the data to be elements of an additive Abelian group $(D, +)$. We begin with a couple of definitions of policy-related terms.

Definition 1 (List of ID's)**.** *Let B denote the set of ID's. Any $\mathcal{B} \in \mathbb{Z}^{|B|}$ will be referred to as the* List of ID's*.*

A list item corresponding to $b \in B$ will be denoted by $\mathcal{B}[b]$. The weighted sum of respective data $\sum \mathcal{B}[b_i] \cdot d_i$, where integer multiplication is interpreted as multiple group addition or subtraction, respectively, will be called the Sum of the List*.*

Definition 2 (Policy)**.** *Let B denote the set of ID's. By* Policy *we mean any $\mathcal{P} \subseteq \mathbb{Z}^{|B|}$, i.e., any set of lists. We say a list $\mathcal{B}$ is* compliant *with policy $\mathcal{P}$ if $\mathcal{B} \in \mathcal{P}$.*

Note 1. *We can view $\mathbb{Z}^{|B|}$ as an additive group with standard coordinate-wise addition. Hence any policy can serve as a set of generators of its subgroup. It follows that by using addition and subtraction of received results, the service can compute the same set of values for any policies $\mathcal{P}_{1,2}$ such that $\langle \mathcal{P}_1 \rangle = \langle \mathcal{P}_2 \rangle$, i.e., such policies can be considered equivalent. However, we will discuss possible differences in Section VII.*

The VERAGREG Framework employs five Probabilistic Polynomial Time (PPT) algorithms, here we briefly describe each of them:

Device: Initialization Algorithm – Init

Provided with a security parameter λ, this algorithm generates a public and private keying material for underlying encryption schemes. N.b., Init outputs order λ-bit key, i.e., in order to fulfill the PPT property, λ must be given in unary.

Device: ID Granting Algorithm – Grant

This algorithm inputs the security parameter λ, a sequence of ID's and a piece of data. It outputs a sequence of ID's—which will be used as the input sequence for the next stage—and the actual and unique ID. The sequence of ID's may include, e.g., a part of already granted ID's or a counter. The granting algorithm might also implement part of the policy. We further allow it to output an invalid symbol $\perp$ if, e.g., the counter exceeds certain bounds or the data is invalid in any sense.

Device: Encryption Algorithm – E

This algorithm encrypts the input data while incorporating the provided ID within the output ciphertext so that it can be neither modified, nor removed.

Service: Addition Algorithm – Add

The Addition Algorithm inputs a list of ID's together with a series of respective ciphertexts over which it performs corresponding homomorphic additions or subtractions. It outputs the final aggregate ciphertext.

Device: Decryption Algorithm – D

Last but not least, the Decryption Algorithm inputs a list of ID's which is expected to describe the contents of the second input—the aggregate ciphertext to be decrypted. First, it checks that the list is policy-compliant and terminates with $\perp$ if this is not the case. Then it verifies that the ciphertext was indeed created according to the provided list. It returns the decrypted data if the check passes, $\perp$ otherwise.

Definition 3 (VERAGREG Framework)**.** *Let D denote an additive Abelian group—the data space, $\lambda \in \mathbb{N}$ the security parameter, B the set of ID's, C the ciphertext space, K_P, K_S the public and secret key space, respectively.* VERAGREG Framework *is a* 5*-tuple of PPT algorithms* $(\mathsf{Init}, \mathsf{Grant}, \mathsf{E}, \mathsf{Add}, \mathsf{D})$,

- $\mathsf{Init}\colon \{1\}^* \to K_P \times K_S$,
- $\mathsf{Grant}\colon \{1\}^* \times B^* \times D \to B^* \times B \cup \{\perp\}$,
- $\mathsf{E}\colon K_S \times B \times D \to C$,
- $\mathsf{Add}\colon K_P \times \mathbb{Z}^{|B|} \times C^* \to C$,
- $\mathsf{D}\colon K_S \times \mathbb{Z}^{|B|} \times C \to D \cup \{\perp\}$,

for which it holds: $\forall n \in \mathbb{N}$, $\forall (d_i)_{i=1}^n \in D^n$ with corresponding valid $(b_i)_{i=1}^n$ granted by Grant_λ*, $\forall \mathcal{B} \in \mathbb{Z}^{|B|}$, $\mathcal{B}[b_i] = n_i$, $\mathcal{B}[b] = 0$ for $b \notin \{b_i\}_{i=1}^n$, and a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Init}_\lambda$,*

1) *if $\mathcal{B}$ is policy-compliant,*
$$\Pr\left[\mathsf{D}_{\mathsf{sk}}\Big(\mathcal{B}, \mathsf{Add}_{\mathsf{pk}}\big(\mathcal{B}, \mathsf{E}_{\mathsf{sk}}(b_i, d_i)_{i=1}^n\big)\Big) = \sum_{i=1}^n n_i \cdot d_i\right] \in \mathsf{OW}_\lambda, \tag{2}$$
i.e., the encryption is additively homomorphic,
2) *if $\mathcal{B}$ is not policy-compliant,*
$$\Pr\left[\mathsf{D}_{\mathsf{sk}}\Big(\mathcal{B}, \mathsf{Add}_{\mathsf{pk}}\big(\mathcal{B}, \mathsf{E}_{\mathsf{sk}}(b_i, d_i)_{i=1}^n\big)\Big) = \perp\right] \in \mathsf{OW}_\lambda, \tag{3}$$
i.e., policy-incompliant list is discarded,
3) $\forall \mathcal{B}' \in \mathbb{Z}^{|B|}$, $\mathcal{B}' \neq \mathcal{B}$,
$$\Pr\left[\mathsf{D}_{\mathsf{sk}}\Big(\mathcal{B}', \mathsf{Add}_{\mathsf{pk}}\big(\mathcal{B}, \mathsf{E}_{\mathsf{sk}}(b_i, d_i)_{i=1}^n\big)\Big) = \perp\right] \in \mathsf{OW}_\lambda, \tag{4}$$
i.e., the framework detects any list forgery,
4) *otherwise,*
$$\Pr\big[\mathsf{D}_{\mathsf{sk}}(\cdot, \cdot) = \perp\big] \in \mathsf{OW}_\lambda, \tag{5}$$
i.e., any invalid ciphertext is detected.

IV. FORMAL NOTIONS OF SECURITY

Let us briefly describe some common formal notions of security—these form pairs of a *game* and respective attacker *capabilities*; for a proper formalization we refer to Bellare et al. [8]. The game defines a scenario in which two parties occur: a *Challenger* and an *Adversary*.

In the *Indistinguishability Game* (IND), the goal of the Adversary is to distinguish which of two plaintexts of her choice has been encrypted by the Challenger. The goal of the Adversary in the *Non-Malleability Game* (NM) is to create a valid ciphertext that is related by its plaintext to the challenge one but is not related to any other plaintext.

In each game, the Adversary has different capabilities. In the *Chosen Plaintext Attack* (CPA), the Adversary can encrypt plaintexts of her choice (i.e., has access to an encryption oracle [1] or is provided with the public key). The *Non-Adaptive Chosen Ciphertext Attack* ($\mathsf{CCA1}$) allows the Adversary to query a decryption oracle before the challenge ciphertext is provided. In the strongest variant—the *Adaptive Chosen Ciphertext Attack* ($\mathsf{CCA2}$)—the Adversary is provided with access to a decryption oracle in both phases with only limitation: the oracle refuses to decrypt the challenge ciphertext.

In addition to these "classical" notions, there have emerged some extensions suitable for different scenarios, including homomorphic encryption, e.g., Prabhakaran et al. [9].

Let us state two easy observations due to which we redefine the notions of security with respect to the VERAGREG framework in order to avoid trivial breach and provide certain formal security goal.

Proposition 4. *The* VERAGREG *framework does not resist any "classical"* NM *attack scenario.*

Proof. The Adversary can trivially win the "classical" NM game: she includes the challenge ciphertext into a policy-compliant aggregate together with several custom ciphertexts and accompanies it with corresponding relation, cf. [8, Definition 2.2]. □

Proposition 5. *The* VERAGREG *framework does not resist any "classical"* $\mathsf{CCA2}$ *attack scenario.*

Proof. There is only one limitation of the decryption oracle in the $\mathsf{CCA2}$ attack scenario—it refuses to decrypt the challenge ciphertext in the second phase. Hence it is sufficient to show how this ciphertext can be decrypted, i.e., construct an unlimited decryption oracle. For this purpose, the Adversary creates a policy-compliant aggregate ciphertext consisting of known values (previously encrypted by the encryption oracle) including the challenge ciphertext. Such a ciphertext is not cheated, is policy-compliant and differs from the challenge one, hence the oracle decrypts it. The Adversary then easily calculates the corresponding plaintext. □

The previous proofs use the same reasoning for which no plain homomorphic encryption can achieve any kind of NM or $\mathsf{CCA2}$ security, respectively. However, unlike plain homomorphic encryption, the VERAGREG framework includes an additional verification feature that enables the Challenger to inspect the contents of any ciphertext and possibly detect whether the challenge ciphertext has been included. Taking this possibility

[1]In the VERAGREG framework, encryption uses the private key, hence an encryption oracle is necessary.

into account, we will introduce modified versions of NM and CCA2 which will avoid such a trivial breach.

Note that the CPA scenario does not use decryption oracle at all, in the CCA1 scenario, the decryption oracle is unlimited and only available in the first phase, and regarding the IND game, no related encryptions are created—the only goal is to distinguish. Hence the two of modifications are sufficient. Note that the following definitions are only applicable in the VERAGREG framework.

Definition 6 (L-NM Game). *In addition to the definition of Non-Malleability by Bellare et al. [8, Definition 2.2],* List Non-Malleability *(L-NM) further requires that the output ciphertexts do not include the challenge ciphertext's ID within their lists.*

Definition 7 (L-CCA2 Attack Scenario). *In addition to the definition of Adaptive Chosen Ciphertext Attack by Bellare et al. [8],* List Adaptive Chosen Ciphertext Attack *(L-CCA2) restricts the decryption oracle during the second phase: it refuses to decrypt any ciphertext including the challenge ciphertext's ID within its list.*

In the following theorem, we state an equivalence analogous to NM-CCA2 $\iff$ IND-CCA2 due to Bellare et al. [8].

Theorem 8 (LNM-LCCA2 $\iff$ IND-LCCA2). *An instance of the* VERAGREG *framework is LNM-LCCA2-secure if and only if it is IND-LCCA2-secure.*

Proof. For both implications, the original proofs by Bellare et al. [8, Theorems 3.1, 3.3] work while the first implication requires a slight modification of the Adversary and the second one just a short comment.

$\Rightarrow$ [8, Theorem 3.1]: In Algorithm $A_2^{\mathcal{O}_2}$, the bitwise complement to be encrypted needs to be replaced with some VERAGREG plaintext-group operation, e.g., by changing $\mathcal{E}(\overline{x_c})$ to $\mathcal{E}(x_c + a)$ for some fixed $a \in D$ and respective change of the relation R. The challenge ciphertext's ID is obviously excluded since a fresh encryption is created.

$\Leftarrow$ [8, Theorem 3.3]: In Algorithm $A_2^{\mathcal{D}_{sk}}$, the set of ciphertexts $\mathbf{y}$ (the output of $B_2^{\mathcal{D}_{sk}}$) does not include indeed any ciphertext with the challenge ciphertext's ID inside its list—the assumed success of the L-NM game requires it. Hence the following decryption passes which is the only concern in the L-CCA2 scenario. □

What assumptions need to hold about the five VERAGREG algorithms in order to achieve IND-LCCA2 security—this remains an open question and is subject to ongoing research.

V. A VERAGREG SCHEME

So far we have described the general properties of the VERAGREG framework, let us now propose a concrete VERAGREG scheme (i.e., a description of the five VERAGREG algorithms) which will aim at satisfying the desired properties. In our proposal, we will consider standard modular addition as the additive group D, only addition operation (i.e., non-negative lists of ID's) and binary encoding. For practical reasons, we will also aim at preventing modular overflows by allowing the Addition Algorithm to output $\perp$, shall an overflow occur.

Initialization Algorithm – Init

The Initialization Algorithm computes system parameters based on additional desired properties:

- the length of input piece of data shall be at least δ bits,
- 2^ν of data values shall not overflow after addition.

It follows that the data part must be at least $\nu + \delta$ bits long. The algorithm sets up

$$\mu_1 = \max\{\lambda, \nu + \delta\}, \tag{6}$$

$$\mu_2 = \lambda. \tag{7}$$

Then it initializes an AHE scheme with the security parameter λ while the plaintext additive group must be isomorphic to some modular addition and must be able to handle at least bitlength of

$$\beta_{\text{AHE}} = \nu + \lambda + \mu_1 + \mu_2 \tag{8}$$

in order to prevent overflows. The AHE cryptosystem must comply at least IND-CPA, unlike, e.g., plain RSA. Next, a Symmetric Encryption (SE) scheme is initialized with security parameter λ and ciphertext size at least λ bits. To be specific, these encryption algorithms can be, e.g., Paillier cryptosystem [7] for AHE and AES [10] for SE. See NIST Special Publication 800-57 [11] for key length recommendations.

Finally, Init generates two random $\mu_{1,2}$-bit integers $m_{1,2}$, respectively. It outputs AHE's public key as the public part and AHE's private key together with SE's key and $m_{1,2}$ as the private part.

ID Granting Algorithm – Grant

The ID Granting Algorithm might very much depend on the policy in use. E.g., we can assume that it increments and returns a time-dependent λ-bit counter (the design allows for remembering certain a part of its history, cf. Definition 3). Grant checks whether the input data is shorter than δ bits and returns $\perp$ if not. Note that any valid output must not depend on the data at all.

Encryption Algorithm – E

The Encryption Algorithm receives an ID denoted by b and corresponding data d to be encrypted. It creates a plaintext p using the secret $m_{1,2}$ as follows:

$$p = \big(\text{SE}(b) \cdot m_1 + d\big) \cdot m_2. \tag{9}$$

Finally, it encrypts the plaintext with AHE's public key into a ciphertext $c = \mathsf{E}_{\mathsf{sk}}(b, d) = \text{AHE}\big((\text{SE}(b) \cdot m_1 + d) \cdot m_2\big)$ and outputs this.

Addition Algorithm – Add

The Addition Algorithm exploits the homomorphic property of the underlying AHE to sum the ciphertexts with respective weights according to the provided list; the homomorphic ciphertext addition will be denoted by $\oplus$. If more than 2^ν individual data is to be added, it returns $\perp$.

Note that for a non-negative list $\mathcal{B}$ corresponding to the data $(d_i)_{i=1}^n$, where $\mathcal{B}[b_i] = n_i$, $\mathcal{B}[b] = 0$ for $b \notin \{b_i\}_{i=1}^n$, $\sum_{i=1}^n n_i \leq 2^\nu$, the homomorphic property ensures

$$\mathsf{Add}_{\mathsf{pk}}\big(\mathcal{B}, (c_i)_{i=1}^n\big) = \bigoplus_{i=1}^n n_i \cdot c_i = \bigoplus_{i=1}^n n_i \cdot \mathrm{AHE}(p_i) =$$

$$= \mathrm{AHE}\bigg(\Big(m_1 \cdot \sum_{i=1}^n n_i \cdot \mathrm{SE}(b_i) + \sum_{i=1}^n n_i \cdot d_i\Big) \cdot m_2\bigg) \quad (10)$$

while preventing plaintext overflow, cf. Equation 8, and keeping $m_1 > \sum n_i \cdot d_i$, cf. Equation 6. This ensures that the individual sums are decodable with the knowledge of $m_{1,2}$.

Decryption Algorithm – D

The Decryption Algorithm first checks whether the received list of ID's $\mathcal{B}$ is compliant with given policy $\mathcal{P}$ and returns $\perp$ if not. Then it decrypts the ciphertext with the private key of AHE into $\tilde{p}$ and ensures that $\tilde{p} \bmod m_2 = 0$ while returning $\perp$ if not. Next, it computes $\tilde{b}_{\mathrm{SE}} = \tilde{p} \div (m_1 m_2)$, cf. Equation 10, symmetric encryptions of b_i's and checks if

$$\tilde{b}_{\mathrm{SE}} \overset{?}{=} \sum_{i=1}^n n_i \cdot \mathrm{SE}(b_i) \quad (11)$$

while returning $\perp$ in case of negative result. Finally, it returns the data part extracted as $\tilde{d} = (\tilde{p} \div m_2) \bmod m_1$ while it should hold

$$\tilde{d} = \sum_{i=1}^n n_i \cdot d_i. \quad (12)$$

Algorithm 1 Decryption Algorithm

```
1: function $\mathsf{D}(B, \mathcal{P}; \mathsf{sk}, \mathcal{B}, c)$
2:     if $\mathcal{B} \notin \mathcal{P}$ then return $\perp$
3:     $\tilde{p} = \mathrm{AHE}^{-1}(c)$
4:     if $\tilde{p} \bmod m_2 \neq 0$ then return $\perp$
5:     $\tilde{b}_{\mathrm{SE}} = \tilde{p} \div (m_1 m_2)$
6:     $b_{\mathrm{SE}} = \sum_{b \in B} \mathcal{B}[b] \cdot \mathrm{SE}(b)$
7:     if $\tilde{b}_{\mathrm{SE}} \neq b_{\mathrm{SE}}$ then return $\perp$
8:     $\tilde{d} = (\tilde{p} \div m_2) \bmod m_1$
9:     return $\tilde{d}$
```

VI. Security Analysis

In this section, we analyze the design of our VeraGreg scheme with respect to different notions of security. First, we focus on indistinguishability based on the underlying AHE scheme, second, we discuss the countermeasures with respect to list non-malleability as introduced in Definition 6.

A. Indistinguishability

The following theorem states that our VeraGreg scheme can achieve it for an active non-adaptive adversary (IND-CCA1).

Theorem 9. *If the underlying* AHE *scheme is IND-CCA1-secure, our* VeraGreg *scheme is also IND-CCA1-secure.*

Proof. Let us assume for contradiction that there exists a VeraGreg-IND-CCA1-adversary $\mathcal{A}_V$ with a non-negligible advantage. We will use $\mathcal{A}_V$ to construct an AHE-IND-CCA1-adversary $\mathcal{A}_H$ to beat the AHE-IND-CCA1-challenger $\mathcal{C}_H$, see Figure 2. We will show that $\mathcal{A}_H$ has non-negligible advantage which will conclude the proof.

$$\begin{tikzcd}
\mathcal{C}_H \arrow[r, leftrightarrow, "\substack{\text{AHE-}\\ \text{IND-CCA1}\\ \text{game}}"] & \mathcal{A}_H \arrow[r, leftrightarrow, "\substack{\text{VeraGreg-}\\ \text{IND-CCA1}\\ \text{game}}"] & \mathcal{A}_V \\
\substack{\text{AHE}\\ \text{oracles}} & \substack{\text{VeraGreg}\\ \text{scheme}} & \substack{\text{non-negl.}\\ \text{advantage}}
\end{tikzcd}$$

Fig. 2. AHE-IND-CCA1 and VeraGreg-IND-CCA1 games.

First, let $\mathcal{A}_H$ instantiate our VeraGreg scheme, except for the underlying AHE scheme—$\mathcal{A}_H$ will rather perform any AHE operation by querying $\mathcal{C}_H$'s encryption and decryption oracles. From this point, $\mathcal{A}_V$ will interact with $\mathcal{A}_H$ as if $\mathcal{A}_H$ were a VeraGreg-IND-CCA1-challenger:

- encryption and decryption queries of $\mathcal{A}_V$ are performed by $\mathcal{A}_H$'s internal VeraGreg scheme while all AHE operations are delegated to $\mathcal{C}_H$'s oracles,
- $\mathcal{A}_V$'s challenge plaintexts are processed into VeraGreg plaintexts and forwarded as AHE-IND-CCA1 game challenge plaintexts to the challenger $\mathcal{C}_H$,
- the challenge ciphertext provided by $\mathcal{C}_H$ is forwarded to $\mathcal{A}_V$ (since then only encryption queries are accepted),
- finally, $\mathcal{A}_H$ answers the same decision as $\mathcal{A}_V$.

It follows that the correct answer is the same in both VeraGreg-IND-CCA1 and AHE-IND-CCA1 games, therefore the advantage of $\mathcal{A}_H$ winning the AHE-IND-CCA1 game is the same as that of $\mathcal{A}_V$ winning the VeraGreg-IND-CCA1 game which was assumed to be non-negligible. □

Note that the proposed Paillier cryptosystem was proved by Armknecht et al. [12] to be IND-CCA1-secure, i.e., a VeraGreg scheme employing it is IND-CCA1-secure as well. However, IND-LCCA2-security remains an open question and it is very important since the scheme offers a decryption oracle by default.

B. List Non-Malleability

The VeraGreg's list is the very feature that enabled us to define the extensions L-NM and L-CCA2 that aim at preventing trivial attacks in case of the original NM and CCA2 notions, respectively. Let us remind that our extended definitions disallow the Adversary to submit any ciphertext including the challenge's ID within the attached list. In order to achieve our extended notions of security, certain countermeasures that aim at preventing from cheating the list must be employed in a VERAGREG scheme.

In our VERAGREG scheme, consistency of the attached list is verified by a comparison of two sums, cf. Equation 11. The Adversary might, e.g., directly modify the sum inside the ciphertext or create a different combination that sums into the same value. In order to prevent such cheating, symmetric encryption of ID's and secret $m_{1,2}$ (cf. Equation 9) have been employed. Neither of them can be omitted, indeed:

- if $m_{1,2}$ were known and ID's unencrypted, the Adversary could use the homomorphic property and add or remove ID's at her will;
- if $m_{1,2}$ were secret and ID's unencrypted, it might happen that some subsets of ID's would have the same sum (e.g., for $(b_i = i)_{i=1}^4$, it holds $b_1 + b_4 = b_2 + b_3$), then the Adversary might abuse this property and create an inconsistent ciphertext-list pair;
- if $m_{1,2}$ were known and ID's encrypted, it would be possible to modify the data which is also undesirable.

Hence, in our design, both of these countermeasures are necessary, however, we leave their sufficiency for any form of L-NM as an open question.

VII. COMBINING RESULTS

Let us revisit Note 1 which remarks that one can compute the same results both with policy $\mathcal{P}$ and its closure $\langle\mathcal{P}\rangle$. It might happen that individual values appear in $\langle\mathcal{P}\rangle$—let us illustrate this on an example: if the aggregates $\bigoplus_{i=1}^n c_i$ and $\bigoplus_{i=2}^n c_i$ are policy-compliant, the service can learn d_1 as well by subtraction of respective plaintexts. However, one can check in advance that the subgroup generated by her policy does not harm the original purpose and might modify the policy accordingly. If this is impossible for any reason, one can still use the original policy, however, some sort of blurring of the results needs to be applied. For this purpose, we think of *Differential Privacy* as an ideal candidate—for an exhaustive reading we refer to Dwork et al.: *The Algorithmic Foundations of Differential Privacy* [13].

Differential privacy is neither a method nor an algorithm—it is rather a guarantee that a database access mechanism preserves certain level of privacy. Its primary goal is to give a promise to an involved party that, no matter whether she takes part in the database or not, the results of database queries will appear to be the same, i.e., *any individual "hides" amongst others*. Note that there is a substantial difference between our scenario and the original use case of differential privacy—the data does not come from several parties where each of them desires to hide amongst others, on the contrary, the data comes from single entity. However, our goal is to hide individual values inside aggregates (i.e., amongst other values) which is in the end the same goal as the original one. Applied to the previous example, if the service learns properly blurred values $\sum_{i=1}^n d_i$ and $\sum_{i=2}^n d_i$, it should learn barely anything meaningful about d_1. Hence we encourage for the use of differential privacy if applicable.

VIII. DISCUSSION & FUTURE DIRECTIONS

We introduced the abstract VERAGREG Framework which allows to verify that addition over encrypted data has been performed honestly. For this framework, novel formal notions of security as well as concrete algorithms were suggested.

However, the definition of the framework does not implicitly ensure any formal security guarantee. Hence, it raises the problem of finding assumptions for the five VERAGREG algorithms in order to achieve any novel notion of security. In particular, we proved IND-CCA1 security of our VERAGREG scheme—to make a step towards a stronger guarantee like IND-LCCA2 or any form of L-NM, impossibility results in case of omitting any design feature might help.

In order to achieve practical implementation of the VERAGREG framework, several aspects need to be resolved—ranging from implementation of involved algorithms in constrained devices to usability of differentially private mechanisms in our context. Currently we are preparing an implementation exploiting RSA module for Paillier operations. Last but not least, possible alternative approaches shall be considered and evaluated with respect to specific application.

REFERENCES

[1] "Regulation 2016/679 of the European Parliament (General Data Protection Regulation)," *Official Journal of the European Union*, vol. L119, pp. 1–88, May 2016. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC

[2] R. Gennaro, C. Gentry, and B. Parno, "Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers," in *Annual Cryptology Conference*. Springer, 2010, pp. 465–482.

[3] C. Tang and Y. Chen, "Efficient Non-Interactive Verifiable Outsourced Computation for Arbitrary Functions," *IACR Cryptology ePrint Archive*, vol. 2014, p. 439, 2014.

[4] D. Boneh, G. Segev, and B. Waters, "Targeted Malleability: Homomorphic Encryption for Restricted Computations," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, 2012, pp. 350–366.

[5] X. Liu, R. Deng, K.-K. R. Choo, Y. Yang, and H. Pang, "Privacy-Preserving Outsourced Calculation Toolkit in the Cloud," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[6] C. Gentry, *A Fully Homomorphic Encryption Scheme*. Stanford University, 2009.

[7] P. Paillier *et al.*, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Eurocrypt*, vol. 99. Springer, 1999, pp. 223–238.

[8] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes," in *Annual International Cryptology Conference*. Springer, 1998, pp. 26–45.

[9] M. Prabhakaran and M. Rosulek, "Homomorphic Encryption with CCA Security," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2008, pp. 667–678.

[10] V. Rijmen and J. Daemen, "Advanced Encryption Standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pp. 19–22, 2001.

[11] E. B. Barker, "NIST Special Publication 800-57 Rev. 4. Recommendation for Key Management, Part 1: General," 2016.

[12] F. Armknecht, S. Katzenbeisser, and A. Peter, "Group Homomorphic Encryption: Characterizations, Impossibility Results, and Applications," *Designs, codes and cryptography*, vol. 67, no. 2, pp. 209–232, 2013.

[13] C. Dwork, A. Roth *et al.*, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.