A Lightweight Mechanism for Dynamic Secret Sharing of Private Data by Constrained Devices

Daniel Morales, Isaac Agudo, and Javier Lopez

***Abstract*—Outsourced computations are essential for IoT devices, but they can raise privacy issues. Privacy-preserving technologies, such as Secure Multi-Party Computation, can be used to delegate computations on private data from multiple devices while disclosing nothing but the output, but they may come at a prohibitive cost. In particular, Secret Sharing-based Secure Multi-Party Computation requires the device to establish n independent confidential channels for each shared message, one channel per holder. This work proposes a new approach for IoT devices to secretly share private data with a committee of holders by broadcasting a single ciphertext. A straightforward solution is Homomorphic Encryption with Decryption to Shares from *Chillotti et al., 2022*, but it requires Fully Homomorphic Encryption and is not dynamic. Additionally, we propose Oblivious Sharing Re-Encryption, which is a new family of protocols that achieve this lightweight private data sharing without requiring Fully Homomorphic Encryption, and which is also more dynamic. We provide a concrete implementation based on NTRU encryption, together with a security proof and performance analysis. The analysis shows that OSRE outperforms the standard setting with n confidential channels when the device sends more than one message.**

***Index Terms*—Privacy, Secret Sharing, Secure Multi-Party Computation, Internet of Things**

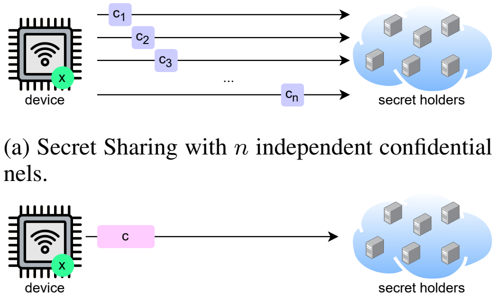
I. Introduction

Outsourcing computation to the cloud is essential for resource-constrained devices such as IoT. This enables cost-effective computing instances [1] and use cases ranging from smart buildings to intelligent transportation to smart healthcare. However, outsourcing computation can introduce new privacy issues, especially when the devices share sensitive data. In [2], one security consideration is *encryption by things*, i.e., having each device encrypt data before uploading it to the cloud. This method can prevent the cloud from accessing the data and protect against data breaches. However, sending encrypted data using traditional mechanisms results in complex key management. Additionally, data must be decrypted before it can be consumed. In these scenarios, privacy-preserving technologies have proven to be very useful because they allow computations to be performed on confidential data.

Among these technologies to enhance privacy, Secure Multi-Party Computation (MPC) is a family of cryptographic protocols that allows a set of parties to jointly compute a function on their private inputs without revealing anything else but the output. Although traditional MPC is set up for direct and interactive computation by the input owners, some works propose MPC in a service provider setting [3]. This delegates the entire computation to a set of computing nodes while ensuring the privacy of the input providers’ data.

Some works have proposed MPC as a solution for constrained devices. For instance, [4] proposed an architecture for performing secure authentication using MPC in smart cities. Similarly, [5] proposed a publicly auditable MPC solution for privacy-preserving computations in industrial IoT.

This work focuses on general-purpose MPC protocols that can compute almost any function on private data. These protocols are mainly based on two technologies: Fully Homomorphic Encryption (FHE) and Secret Sharing (SS)-based MPC. FHE is better suited for cloud environments, where the computing instance receives ciphertexts from the client and computes on them, producing an encrypted output. However, it struggles with settings where there are multiple input providers, because handling decryption keys without compromising privacy is difficult. Typically, solutions involve handling the keys through SS or involving the input providers in some form of interactive MPC for decryption.



(a) Secret Sharing with n independent confidential channels.

(b) Lightweight Secret Sharing.

Fig. 1: Two approaches for Secret Sharing delivery.

SS-based MPC is more flexible, and privacy is guaranteed as long as enough parties in the set of computing nodes remain uncorrupted. More precisely, a secret x is divided into n different shares $\{x_i\}_{i=1}^{n}$, one share per secret holder. Given a corruption threshold $t < n$, only a valid subset of $t+1$ shares can reconstruct x, and functions can be computed directly on the shares. However, this implies a limitation: although increasing the number of SS holders increases security, it also implies an overload on the device when sharing private data. More precisely, if there are n computing nodes, to send a private value x the device must set up n independent confidential channels and send one share of x per channel (see Figure 1a), i.e., n ciphertexts. For a large n, this may be unacceptable for constrained devices that periodically send private data.

The authors are with NICS Lab, University of Málaga, Spain. Emails: {damesca, isaac, javierlopez}@uma.es.

This work focuses on the concept of allowing a constrained device to share a secret with an MPC committee in such a way that there is no need to send n independent values through n independent confidential channels. This alternative is illustrated in Figure 1b, where the device can publicly share a single encryption from where each SS holder can retrieve its own share. To summarize, the following research question is formulated:

Can the dealer of a Secret Sharing scheme send a single ciphertext that allows each holder to recover only its own share of the original secret, and nothing else?

Applications. The proposed approach can be beneficial in scenarios where constrained devices send SS data. For instance, consider a group of smart metering devices that periodically send consumption data to an MPC-based privacy-preserving analytics platform. The platform computes statistics on the aggregated data. By sending only one ciphertext per measurement to the analytics platform, the smart meters can save resources compared to establishing independent confidential channels. Another scenario is privacy-preserving anomaly detection, where an external analysis engine searches for matching rules on SS data packets from network devices. Additionally, privacy-preserving drone geofencing involves a drone periodically sending location data to a monitoring engine that only discloses the location if the drone enters a forbidden flying area.

Contributions. We summarize the main contributions.

- Homomorphic Encryption with Decryption-to-Shares (HEDS) is identified as a method that enables SS of private data by sharing a single ciphertext, thus reducing connections and communication costs. However, it requires FHE and is not dynamic, i.e., formation of committees must be done prior to generate the ciphertext.
- A new general procedure to SS by letting the data owner to send a single ciphertext is proposed, named Oblivious Sharing Re-Encryption (OSRE), which is dynamic regarding the formation of committees.
- A protocol instance of OSRE based on NTRU encryption where a semi-honest proxy can generate the shares of the secret obliviously, by using Additive Homomorphic Encryption (AHE) and deliver them to a committee that is formed after the encryption of the secret thanks to Proxy Re-Encryption (PRE).
- An implementation of our NTRU-based OSRE protocol and a performance evaluation.

Organization. The paper continues as follows: Section II gathers related work on IoT private computing and efficiency in SS. Then, Section III provides a cryptographic background. With respect to Section IV, it introduces both the system and security models. Section V introduces HEDS and how it can be used to deliver shares with a single encryption, and then Section VI introduces OSRE, our main contribution for efficient and dynamic SS. Next, Section VII provides a security proof, and Section VIII discusses about performance and evaluates our implementation. Finally, Section IX presents some conclusions and future work.

II. Related Work

IoT security. As analyzed in [6] the IoT ecosystem presents some vulnerabilities, such as data leakage or improper encryption, which pose high security threats. In addition, other privacy threats such as identification, location tracking, or profiling, lead to critical situations. Some works address confidentiality in IoT networks [7], but they only consider end-to-end encryption, i.e., data must be decrypted at some point in order to be processed. There are works, such as [8], [9], that propose FHE for confidential smart metering aggregation. However, [8] suggests using a gateway encryption key for all smart meters, creating a single point of failure. In contrast, [9] addresses the centralized decryption issue by distributing the decryption key with HSS. Finally, [10] proposes an application for privacy-preserving deep learning using MPC. This allows a model to be trained with encrypted inputs provided by different devices.

Secret Sharing efficiency. SS has been extensively studied and we refer to [11] for a very complete survey. Regarding communication efficiency, some lower bounds are provided and roughly speaking, we have that the size of the share to be sent to one holder must be at least the size of the secret. Since this cannot be circumvented, a common approach to achieve better performance is using Multi-Secret Sharing Schemes (MSSS) [12], where a set of messages $(m_1, ..., m_l)$ is shared with a set of holders H by sending a single share per holder and therefore achieving, at most, a $\times l$ reduction in the overall communication cost. However, this still requires the device to establish n independent confidential channels. Our approach overcomes this by allowing the device to use a single connection, approach that, apart from the work in [13] has never been addressed.

III. Cryptography background

A. Secure Multi-Party Computation

MPC [14] is a theoretical problem where a set of parties $\{P_i\}_{i=1}^n$, each one holding some private data $x^{(i)}$, wish to jointly compute a function $y = f(x^{(1)}, ..., x^{(n)})$ on their private data without revealing anything but the output y. There are several approaches to solve MPC using different building blocks. In this paper, we focus on general-purpose MPC protocols, which are mainly based on SS or FHE. We note that the secret owners may be those involved in performing the computation, or they may delegate it to an MPC computation engine.

B. Secret Sharing

An SS scheme allows a dealer holding a private value x to share it with a set of holders $\{H_i\}_{i=1}^n$, giving each H_i a secret-share x_i of x. There must exist a function $x = open(x_1, ..., x_t)$ that outputs x given a sufficient number of $t \leq n$ shares. Linear SS schemes have the additional property that they can locally apply an additively homomorphism on shares, and some of them also allow multiplication. Additions and multiplications are then mapped to the secret once reconstructed. The most widely adopted SS schemes for MPC are Additive Secret Sharing (ASS) [15] and Shamir Secret Sharing (SSS) [16].

In ASS, the shares are computed s.t. $x = \sum_{i=1}^{n} x_i$, with x_i randomly sampled from $\mathbb{F}_p$. To reconstruct x, all the n shares are needed. Thus, x remains private as long as no more than $t = n - 1$ holders are corrupted.

In SSS, the secret is encoded in a polynomial of degree k, more specifically $a_0 = x$. Then, k random coefficients a_i are sampled and the polynomial is set as $P(y) = a_0 + \prod_{i=1}^{k} a_i y^i$. Then, the share for H_i is computed as $x_i = P(i)$, with $i \in \mathbb{N}$ being H_i's id. Note that at least $k+1$ shares are needed to reconstruct the polynomial by interpolation, thus obtaining x.

Both ASS and SSS lead straightforward to an additively homomorphism, i.e., given shares of x and y, additive shares of the sum can be computed locally by $z_i = x_i + y_i$ s.t. $x + y = z = open(z_1, ..., z_n)$. They also support multiplication, but with additional tricks.

C. Homomorphic Encryption

An FHE [17] scheme is a public key encryption scheme with the additional property that, given two (or more) ciphertexts $c_1 \leftarrow Enc_{pk}(m_1)$ and $c_2 \leftarrow Enc_{pk}(m_2)$, a function can be evaluated on them. More precisely, given $c_3 \leftarrow Eval(f, \{c_1, c_2\})$ and $m_3 = Dec_{sk}(c_3)$, it holds that $m_3 = f(m_1, m_2)$. There are also relaxed and more efficient schemes, called Partially Homomorphic Encryption (PHE), which support only one type of operation on ciphertexts, e.g., addition [18] or product [19].

D. Proxy Re-Encryption

A PRE scheme [20] allows a proxy to re-encrypt a ciphertext c_A encrypted with pk_A into a ciphertext c_B encrypted with pk_B, thus changing who can decrypt it. To do this, the proxy uses a re-encryption key $rk_{A \rightarrow B} \leftarrow ReKeyGen(sk_A, sk_B)$. There are two types of schemes regarding how they compute the re-encryption key. Interactive PRE computes it from (sk_A, sk_B), thus requiring an interactive protocol between Alice, Bob, and the proxy in order not to reveal the secret keys to the other parties [21]. On the other hand, non-interactive PRE computes the key from (sk_A, pk_B), so it can be computed by Alice without revealing her key.

IV. System and Security Model

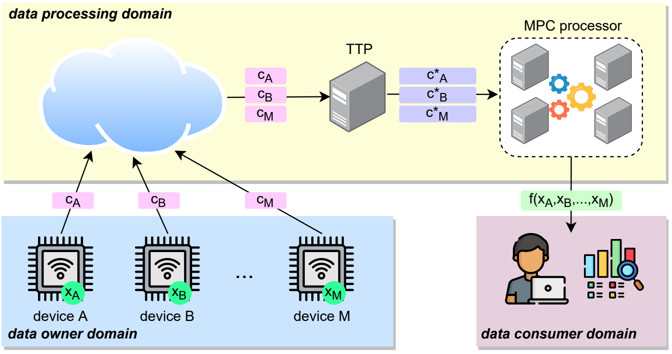


Fig. 2: System model for dynamic lightweight SS delivery.

Figure 2 shows the system model, which involves a set of constrained devices that own some private data, and a data consumer that wants to compute some function on that data. A naive approach without privacy would imply that the devices send the plaintext data to the data consumer who computes the function. In this work, we introduce a *privacy-preserving delegated model* where all the data processing is done in the encrypted domain thanks to MPC. We assume that constrained devices periodically generate encrypted data, under their own public key. We also aim for a *dynamic model*, i.e., the data owner can choose what to do with her encrypted data after it is generated and stored somewhere. Instead of simply retrieving data and sending it to the data consumer, or using PRE to delegate decryption rights, we include a Trusted Third Party (TTP) that is able to delegate data to an MPC processor without exposing the plaintexts, by generating new forms of encryption from the original ciphertexts such that the MPC committee can retrieve their secret shares and nothing else. In the following sections we show that HEDS solves this situation without a third party, but also without dynamism, and that our new protocol that fulfills the general notion of OSRE, instantiated with NTRU and PRE, allows to achieve this scenario with dynamism and with a semi-honest proxy instead of a TTP. We separate the device from the data owner domain, understanding that data generation and encryption is done inside the device, but that the owner domain can perform additional operations. This allows to minimize the work done by the device and therefore its resource consumption. We also assume that ciphertexts generated by the device are stored somewhere accesible by the TTP (or semi-honest proxy), from where the delivery to the MPC committee can be done.

Regarding the security model, we make explicit the distinction between a TTP and a semi-honest proxy. In a naive and general approach, a TTP could just receive m from the device, generate the set of shares $\{m_i\}$, and send each m_i to each share holder H_i in the committee. This would be an outsourcing of the share delivery, but in this work, and more specifically in our construction in Section VI-A, we aim at a semi-honest proxy (that follows the protocol specification) that is able to generate the shares without being able to see them, in a way that each H_i can later retrieve m_i. With respect to the security of the MPC committee, our protocol is agnostic to that, and one can assume both semi-honest or malicious adversaries, and different target corruption thresholds.

V. Lightweight Share Delivery Using HEDS

An HEDS scheme [13] allows Alice to encrypt her message m with a public key pk_H, which is previously generated by a trusted setup that relies on FHE. Each holder H_i owns a share of the secret key sk_{H_i} s.t. $sk_H = \sum_i sk_{H_i}$. Given the ciphertext c generated by Alice, each holder can locally decrypt it with sk_{H_i} and obtain a valid share m_i of the message, i.e., $m = open(m_1, ..., m_n)$. The homomorphic property allows anyone to compute a function on c before decrypting its content to shares. HEDS is straightforwardly achieved for 2-parties in [13] thanks to a property in decryption of linear FHE schemes, and then bootstrapped to n-parties in the setup phase thanks to FHE.

We note that HEDS implicitly enables SS delivery by a single encryption. However, it has a main limitation: the committee of holders (and their key pair) must exist before the device encrypts the message. This may be suitable for static long-term committees, but at the expense of less security in environments with adaptive adversaries. In addition, prior existing static committees hinder dynamism, causing the device to encrypt all the data to be shared with a different public key each time a different committee is selected.

An additional aspect to notice is that, for our application, it would just suffice with Encryption with Decryption to Shares (EDS), because the homomorphism that enables to evaluate on ciphertexts is not needed. However, FHE is also needed for the setup that generates the keys in HEDS, so we leave as an open problem to analyze if EDS can be achieved by a different strategy that also allows to achieve shorter ciphertexts.

VI. Lightweight share delivery using OSRE

One way to alleviate the cost of FHE would be by relying on a TTP that implements a sort of oblivious secret sharing, somehow following the system described in Section IV. The most simplistic approach would be to have an Additive Homomorphic Encryption (AHE) scheme that is dense in the ciphertext space[1] and directly sample random ciphertexts to build the encrypted shares, i.e.,

$$Enc(secret) = \sum_{i=1}^{n} randomC_i = \sum_{i=1}^{n} Enc(randomP_i),$$

from where the sum of the random shares $\{randomP_i\}$ is equal to the original secret.

We now introduce OSRE as a new general procedure that allows dynamic and efficient SS delivery to a committee, from where we will propose a concrete scheme in Section VI-A. OSRE is composed by three general phases:

1) **Device's data delivery.** The device encrypts m using its own long-term public key pub_k and stores $c \leftarrow Enc_{pub_k}(m)$ somewhere accessible by the proxy.
2) **Oblivious sharing.** The proxy retrieves c and generates a set of new ciphertexts $\{c_i^* = Enc_{pub_{k_i}}(m_i)\}$, where pub_{k_i} is a key held by the share holder H_i.
3) **Share decryption.** Each holder decrypts c_i^* using its secret key $priv_{k_i}$, obtaining m_i.

We notice that, in the case that all keys pub_{k_i} are the same, we are in the scenario introduced at the beginning of this section, where ciphertexts are randomly sampled. However, despite this approach does not require a costly setup (as it happens in HEDS) because the holders could just agree on a common key pair for the committee where $priv_{k_i} = priv_{k_j}$ for all i, j, it still has the problem of having to decide the committee beforehand. Therefore, the interesting case would be one where each pub_{k_i} is different (and so is each $priv_{k_i}$) and can be decided later on.

A. Dynamic OSRE based on NTRU and PRE

In this section, we introduce a concrete scheme for OSRE that solves the scenario in Figure 2, where an IoT device sends a ciphertext $c = Enc_{pk_D}(m)$ to a semi-honest proxy (instead of a TTP) that blindly generates secret shares of m (without learning m) and delivers them encrypted to the committee of holders H (which comprises the MPC processor in Figure 2). It is dynamic because H can be selected after c has been generated by the device. For this approach, we rely on AHE to compute the shares obliviously and PRE to let each holder H_i being the only one capable of decrypting c_i^*. Note that density in the ciphertext space is not needed anymore. For all the previous, NTRU encryption scheme is selected as the candidate. For the sake of completeness, the algorithms for NTRU-RLWE [22]–[24] with PRE are introduced below.

Key Generation. Select the following parameters: security parameter λ, ciphertext modulus q, ring dimension N, gaussian key distribution $\mathbf{X}_k$ over the polynomial ring $R = \mathbb{Z}[N]/\langle x^N + 1\rangle$ with distribution parameter σ_e, and an empirically selected assurance measure α to minimize the number of bits needed to represent q. The plaintext space is $M = \{0, 1, ..., p-1\}^N$, where $p \geq 2$ is the plaintext modulus. Then, sample polynomials $\mathbf{f}', \mathbf{g} \leftarrow \mathbf{X}_k$ and set $\mathbf{f} = p\mathbf{f}' + 1$ to satisfy $\mathbf{f} \equiv 1(mod\ p)$. Set $sk := \mathbf{f} \in R$ and $pk := p\mathbf{g}\mathbf{f}^{-1} \in R_q$.

Encryption. Sample random polynomials $\mathbf{s}, \mathbf{e} \leftarrow \mathbf{X}_e$, and compute $\mathbf{c} = \mathbf{h}\mathbf{s} + p\mathbf{e} + \mathbf{m} \in R_q$.

Decryption. Compute $\mathbf{b} = \mathbf{f}\mathbf{c} \in R_q$ and output $\mathbf{m}' = \mathbf{b}(mod\ p)$. Correctness holds as long as there is no wrap-around $mod\ q$, i.e., when $||\mathbf{b}||_\infty \leq q/2$. Then, it holds that $\mathbf{m}' = \mathbf{f}\mathbf{c} = p\mathbf{g}\mathbf{s} + p\mathbf{f}\mathbf{e} + \mathbf{f}\mathbf{m} = \mathbf{f}\mathbf{m} = \mathbf{m}(mod\ p)$.

Re-Encryption Key Generation. Given the source key sk_A and the target key sk_B, the re-encryption key is computed as $rk_{A\rightarrow B} = sk_A \cdot sk_B^{-1}$.

Re-Encryption. Given an NTRU ciphertext $\mathbf{c}_A$ encrypted with pk_A, the new ciphertext is computed as $\mathbf{c}_B = \mathbf{c}_A \cdot rk_{A\rightarrow B} + p\mathbf{e}' \in R_q$.

We first notice that a naive way to achieve OSRE with NTRU would be for additive shares of the secret. Given $c \leftarrow Enc_{pk_D}(m)$, the proxy can sample $n-1$ random shares $\{m_i\}$, and then compute the last share as $c_n = c - Enc_{pk_D}(\sum_i m_i)$. It should be noticed that, while this achieves correctness, it allows the proxy to see $n-1$ shares of m.

On the other hand, a better approach can be achieved for Shamir shares, which is properly introduced in Protocol 1. First, the device D encrypts the message m and sends c_0 to the proxy P. In this approach, for the oblivious sharing phase, the proxy does not see any share in plaintext, but all the coefficients $\{a_i\}$ of the polynomial that encodes the secret except a_0, i.e., the secret. Notice that, without knowledge of a_0, there are still $|\mathbb{F}_p|$ different polynomials that could be valid candidates, and therefore the secret is kept securely hidden from the proxy view. The proxy can construct $Q(i)$, which is a partial polynomial of the secret evaluated with H_i's id, and later achieve the exact share for H_i inside the ciphertext space, i.e., $W(i)$, thanks to the AHE property. Then, PRE allows to re-encrypt each $c_{H_i}^{(W)}$ from pk_D to pk_{H_i} such that

[1] We mean by dense that each random sample from the ciphertext space is a valid ciphertext under a given key in the encryption scheme. It can be easily verified that ElGamal encryption fulfills this.

Protocol 1: Plaintext Shamir OSRE

Participants: the committee of holders H, the device D, the data owner O, the proxy P.
Notation: $\Rightarrow$ means sending data.

1) **Initial data delivery**
   a) D: $a_0 = m$
   b) $D \Rightarrow P$: pk_D, $c_0 \leftarrow Enc_{pk_D}(a_0)$
2) **Oblivious Sharing**
   a) P: samples random $\{a_1, ..., a_k\} \in \mathbb{F}_p$. Then, for each holder H_i, computes $Q(i) = \sum_{j=1}^{k} a_j i^j$, $c_{H_i}^{(Q)} \leftarrow Enc_{pk_D}(Q(i))$, and $c_{H_i}^{(W)} = c_0 + c_{H_i}^{(Q)} = Enc_{pk_D}(W(i))$, where $W(i)$ is H_i's share
   b) $O \Rightarrow P$: $\{rk_{D \rightarrow H_i} \leftarrow ReKeyGen(sk_D, pk_{H_i})\}_{i=1}^{n}$
   c) $P \Rightarrow H_i$: $c'_{H_i} \leftarrow ReEnc_{rk_{D \rightarrow H_i}}(c_{H_i}^{(W)})$
3) **Share decryption**
   a) H_i: $m_i = Dec_{sk_{H_i}}(c'_{H_i})$

each encrypted share c'_{H_i} can be sent to each H_i.

One of the main disadvantages of OSRE against HEDS is that communication is $O(n)$ in the owner side because of the re-encryption keys delivery, despite they are not delivered by the device. However, we identify a way of reaching $O(1)$ communication for the owner, but at the expense of requiring the encryption scheme to enable threshold decryption. It basically requires that the committee H, instead of generating individual key pairs (pk_{H_i}, sk_{H_i}), generate a set of secret keys $\{sk_{H_i}\}$ and a single public key pk_H. Then, each ciphertext c_i will be re-encrypted under the same key $rk_{pk_D \rightarrow pk_H}$. The last step is for the committee to jointly decrypt each c_i such that m_i is only received by H_i. This is possible by each H_j computing a partial decryption $d_{i,j}$ of c_i, such that $m_i = combine(\{d_{i,j}\})$. Equation 1 shows that NTRU allows partial decryption when partial secret keys are additive, i.e., $\mathbf{f} = \sum_i \mathbf{f_i}$. However, linear decryption schemes leak information about the partial decryption keys [25], so we left as future work finding a proper decryption algorithm.

$$\sum_i \mathbf{d}_i (mod\ p) = \sum_i \mathbf{f}_i(p\mathbf{gsf^{-1}} + p\mathbf{e} + \mathbf{m})(mod\ p) = \\ = p\mathbf{gs} + p\mathbf{fe} + \mathbf{fm} = \mathbf{m} \quad (1)$$

VII. Security analysis

This section provides a security analysis of Protocol 1. We remark that the property to be guaranteed is privacy of the secret m with respect to a semi-honest proxy. As introduced before, the protocol needs NTRU because it is additively homomorphic and has PRE. More specifically, NTRU-RLWE has been proven to be IND-CPA both in the original paper [22], [23] and in its PRE version [24]. We remark that IND-CPA in such two cases implies the standard encryption IND-CPA game, but with the addition that the adversary can perform homomorphic operations or re-encryptions on the ciphertext m_b, respectively, which still does not allow to distinguish if the encrypted message was m_0 or m_1. In addition, IND-CCA is not needed because our scheme assumes that the proxy has no oracle access to decryption, i.e., decryption keys are kept by the dealer (for m) and the share holders (for m_i).

For the proof, we follow an **inductive approach** for the oblivious sharing phase. We consider two parameters for the proof, which are the threshold t and the number of parties n, and use the notation $OSRE(t, n)$ for an instance of OSRE where the dealer delivers a message m using a Shamir scheme with threshold t and n holders. To ease notation, we consider in this proof that t is the minimum number of shares needed to reconstruct the secret m.

First, we define $OSRE(1, 1)$ as Protocol 1's base case with $t = 1$ and a single holder H_1, where the ciphertext is $c_{H_1}^{(1)} = Enc_{pk_D}(a_0)$, with $a_0 = m$. The case $OSRE(1, 2)$ implies sending the same $c^{(1)}$ to both holders, where any of them reveals m. The following case, $OSRE(2, 2)$, needs to increase the polynomial degree from 0 to 1, since the degree must always be $t - 1$. Therefore, we can define $c_{H_1}^{(2)} = c_{H_1}^{(1)} + Enc_{pk_D}(a_1 \cdot 1)$, i.e., the evaluation of $P(i) = a_0 + a_1 \cdot i$, with a_1 a randomly sampled coefficient. Next, $OSRE(2, 3)$ computes an additional share for H_3, but security is the same, i.e., 2 shares are enough to open the secret. The general case is presented in Equation 2 to define the ciphertext of a holder H_i in $OSRE(k, n)$.

$$c_{H_i}^{(k)} = c_{H_i}^{(k-1)} + Enc_{pk_D}(a_r \cdot i^{k-1}) \quad (2)$$

Correctness holds because $\sum_i Enc(a_i \cdot x_i) = Enc(\sum_i a_i \cdot x_i)$ is guaranteed by AHE. Regarding security, it can be noticed that $OSRE(k+1, n)$ can be reduced to $OSRE(k, n)$. In addition, $OSRE(k, n)$ presents the same security as $OSRE(k, k)$, as introduced above. Therefore, any step can be reduced to exposing the content in $OSRE(1, 1)$, i.e., breaking the IND-CPA assumption of the base encryption scheme.

Finally, regarding re-encryption, we recall the sequential composition property in [26]. Since both AHE and PRE are IND-CPA, they are easily simulatable in the stand-alone model. This imply that they can be sequentially composed while maintaining the security properties. Even when Protocol 1 performs n re-encryptions, one per share holder, security under sequential composition still holds because each re-encryption is independent of the others.

VIII. Cost analysis and comparison

A. Theoretical cost analysis

This section provides a theoretical comparison of the different schemes proposed. First, regarding asymptotic costs, these are provided in Table I.

We notice that, despite HEDS is more efficient asymptotically, it implies larger ciphertexts because of FHE, and also implies an initial setup to form the committee. On the other hand, OSRE achieves dynamism by relying on a proxy (TTP

TABLE I: Asymptotic cost comparison

Role	HEDS	OSRE (Prot. 1)	OSRE (Prot. 1 with thr. dec.)
Device	$O(1)$	$O(1)$	$O(1)$
Owner	n.a.	$O(n)$	$O(1)$
Proxy	n.a.	$O(n)$	$O(n)$
Share holder	$O(n)$	$O(1)$	$O(n)$

or semi-honest), which takes $O(n)$ operations. We notice that if using HEDS, one could assume that the underlying FHE scheme also supports PRE, and this would add dynamism to HEDS with a re-encryption step with $O(1)$ cost in the owner side. However, the encryption phase done inside the device would still be less efficient. Therefore, assuming that we are selecting a dynamic option that relies on a semi-honest proxy, OSRE is more efficient for the device. Even if it implies more computation in the proxy side, the proxy is normally assumed to be more powerful in terms of resources than the devices. We also notice that OSRE with threshold decryption moves the linear term $O(n)$ from the owner side to the share holders committee side.

The rest of this section is intended to provide some specific communication costs, mainly regarding ciphertext and re-encryption key lengths. For simplicity, the energy savings achieved by using one connection instead of n are omitted here. First, we briefly introduce how lengths are computed for each scheme. To ensure a fair comparison, we set the security parameter $\lambda = 128$, and obtain the size of the parameters (see Table II).

ECIES. This hybrid encryption scheme [27] is used to model n confidential channels, namely standard setting (STD). The channel establishment is based on Diffie-Hellman, hence the device sends 510 bits (a point in the elliptic curve). For message encryption, the recommendation is AES-128-CBC for the ciphertext and HMAC-SHA-256 for the authentication tag (in constrained devices), therefore 384 bits.

NTRU-SVES. The parameters are from [28]. A ciphertext in NTRU is a polynomial with N coefficients $mod\ q$. Therefore, one polynomial can be encoded with $N\lceil log_2(q)\rceil$ bits. As for the re-encryption keys, they are computed as $\mathbf{rk}_{A\to B} = \mathbf{f}_A\mathbf{f}_B^{-1}$. A secret key in NTRU has N coefficients in ternary form, i.e., from $\{-1, 0, 1\}$. Therefore, the length of $rk_{A\to B}$ is $2N$ (2 bits per coefficient).

TFHE. A TLWE sample [29] is determined by the dimension d. Then, a ciphertext is defined as $(a_1, ..., a_d, b) \in LWE_{\mathbf{s},\sigma}((q/p)m) \subseteq R_q^{d+1}$. Each element is typically encoded with 32 bits, therefore the length of a TFHE ciphertext is $32(d + 1)$ bits. Regarding re-encryption, using the FHE implicit approach proposed in [30] we have that $rk_{A\to B} = Enc_{pk_B}(sk_A)$. Therefore, it has the length of one ciphertext.

TABLE II: Length of one ciphertext and re-encryption key with $\lambda = 128$.

Algorithm	Params	Ciph. (bits)	Re-Enc. key (bits)
ECIES	n.a.	510 + 384	n.a.
TFHE	$d = 630$	20192	20192
NTRU-SVES	$N = 439, q = 2048$	4829	878

The intuition behind the comparison is learning from what n the cost of ciphertexts in HEDS/OSRE is lighter than STD in the data owner domain (owner and device). In addition, for the case of OSRE from Protocol 1, even when sending $\{rk\}$ grows faster in n than STD, the intuition says that this is amortized when sending $N_c > 1$ ciphertexts to the same committee. Equation 3 represents the cases for when STD (the first term) is more costly than OSRE/HEDS. Notice that n^* fixes the amount of re-encryption keys that are sent in each scheme, which is n in OSRE, 1 in OSRE with threshold decryption, 0 in HEDS, and 1 in HEDS with PRE.

$$n \cdot PK + n \cdot c_{len}^{symm} \cdot N_c > n^* \cdot rk_{len} + c_{len}^{asymm} \cdot N_c \quad (3)$$

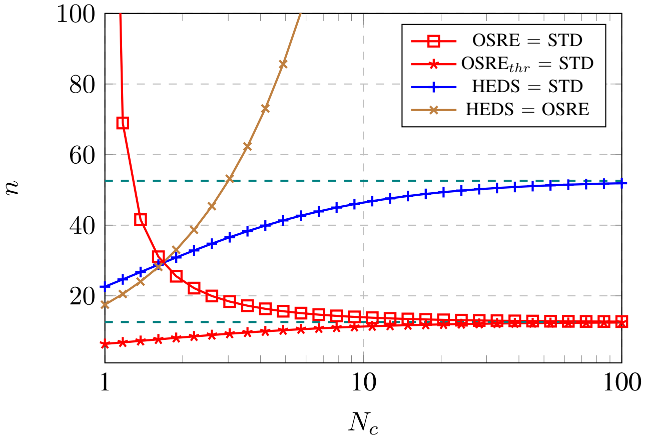
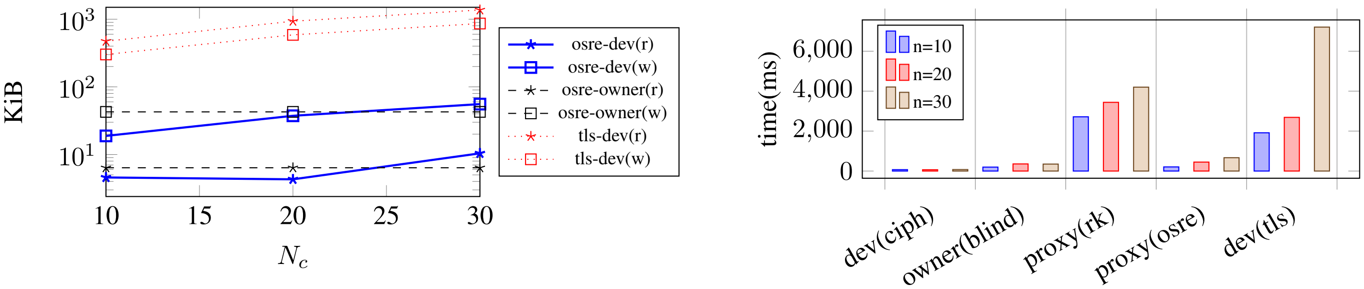


Fig. 3: Comparison of the minimum n needed for scheme A to outperform scheme B ($A = B$), as a function of N_c.

From Equation 3, n can be obtained as a function of N_c, and this is shown in Figure 3 for OSRE (NTRU), OSRE$_{thr}$ (NTRU), HEDS (TFHE), and STD (ECIES).

First, $N_c = 1$ is a special case because OSRE cannot outperform STD, due to the n re-encryption keys. It also happens that HEDS outperforms OSRE, but only for $n \geq 18$, otherwise STD is better. It is clear that the most efficient option is OSRE$_{thr}$, but only for $n \geq 7$, otherwise STD is still the best.

Second, we analyze the amortized case, i.e., when the same re-encryption key is reused for $N_c \geq 1$ ciphertexts. Notice that the area above the curve $A = B$ means that scheme A is more efficient than scheme B. The first conclusion is that OSRE outperforms HEDS in most cases, but those in the area above the curve $HEDS = OSRE$, i.e., when N_c is somehow small. Next, since n varies depending of N_c, we perform an asymptotic analysis to achieve the bounds of the different schemes. First, $OSRE = STD$ presents a lower bound $\lim_{N_c\to\infty} f(N_c) = 12.58$, i.e., OSRE cannot outperform STD with $n < 13$. The bound for $OSRE_{thr} = STD$ is the same, but an upper bound instead of a lower one, which means that OSRE$_{thr}$ always outperforms STD with $n > 13$. It can be concluded then that the benefit of OSRE$_{thr}$ against OSRE is presented for small N_c values. Finally, $HEDS = STD$ presents an upper bound $\lim_{N_c\to\infty} f(N_c) = 52.58$, which means that HEDS always outperforms STD with $n > 52$.



(a) Client side communication cost in Bytes (read/write) to send N_c ciphertexts with a fixed $n = 20$.

(b) Running time for different steps and different values of n.

Fig. 4: Performance evaluation based on simulations.

B. Implementation and evaluation

We have implemented and tested the OSRE protocol, relying on two available libraries for NTRU and its PRE scheme[2], in Java. We have compared it against the STD case, which has been implemented in Java using TLS connections.

There is an important implementation aspect first, which is that NTRU supports homomorphic addition of plaintexts, but these are mapped to elements inside the NTRU ring, i.e., addition is done between polynomials. However, OSRE needs addition inside $\mathbb{F}_p$, which is the field used by SSS. Therefore, we have implemented a special codification method that allows to perform a single addition of two ciphertexts s.t. they can be decoded back to the sum of integers. For that, first notice that NTRU plaintexts M has ternary coefficients, i.e., from $\{-1, 0, 1\}$. Then, given m_b the binary representation of $m \in \mathbb{F}_p$, we let the coefficient of M with degree k to encode the digit $m_b[k]$. NTRU must guarantee a minimum number $dm0$ of coefficients with values of either -1, 0, or 1, therefore we adjust this once m_b is set. What we do next is to perform addition between the polynomials M_1 and M_2 inside the ciphertext space. This emulates binary addition, but leaving the carry to the final step, i.e., if the result $(M_1 + M_2)[k]$ is -1, that means that there is a carry in that digit. Finally, once the result is decrypted, the carry is applied from the LSB to the MSB. We note that, if the polynomial M to encode the message m_b is large enough, different strategies can be applied to decide which cells to use for the message bits, therefore adding variations on how to sparse the digits to avoid attacks based on plaintext knowledge.

Another caveat is that the ReKeyGen algorithm of NTRU PRE in [24] is interactive, between the data owner, the proxy, and the committee of holders, which incurs in larger costs than if it was non-interactive.

Finally, for evaluation, we have simulated a network scenario using docker inside a Ubuntu 22.04.1 LTS virtual machine, operating on an Intel ESXi server with an 8-core CPU (2.2 MHz) and 32 GB of RAM. Note that the intention of the simulation is not showing realistic performance values, but a fair comparison between OSRE and the STD case. All the simulations have been run with 8 threads.

For a fixed $N_c = 1$ the result obtained is very similar to the theoretical analysis, i.e., sending the ciphertext is $O(1)$, but sending the re-encryption keys scales with n, the same as in STD with independent TLS channels. However, our scheme outperfoms STD in terms of communication when $N_c > 1$, as shown in Figure 4a, where the cost for TLS channels grows so much faster than in OSRE (notice that axis Y is logarithmic). This happens because the re-encryption keys are only sent once. This graph has been generated for $n = 20$, but the difference is more significant for larger n values. Finally, Figure 4b presents some running times with different n values. It can be seen how the major latency is introduced by the proxy, overall to derive the re-encryption keys due to the interactivity of the ReKeyGen step. If a non-interactive PRE were used instead, this step would be much more efficient. However, despite this phase is not efficient, for a large n TLS is even slower, due to the parallel management of connections.

IX. Conclusions and Future Work

This work presents a lightweight and dynamic solution to let constrained devices secret sharing data by sending a single ciphertext, hence avoiding the need of n independent confidential channels. Despite this is straightforwardly achieved by HEDS, we introduce a new general approach to achieve protocols named OSRE, which enables more dynamism and does not need FHE. More specifically, the committee of holders can be selected after data has been encrypted by the device with its own public key. Our main contribution is an OSRE scheme for SSS, based on NTRU and PRE. In general, OSRE outperforms HEDS, overall when the amount N_c of ciphertexts sent is large, therefore it is suitable for settings with periodic reports of confidential data. We have provided theoretical bounds, an implementation, and a evaluation, which shows that OSRE beats the STD case with TLS channels in some settings. We leave as future work to find if HEDS can be relaxed to EDS, and also finding a suitable secure threshold decryption scheme for NTRU.

Acknowledgment

This work has been partially supported by the project SecAI (PID2022-139268OB-I00) funded by the Spanish Ministerio de Ciencia e Innovación, and Agencia Estatal de Investigación. The first author has been funded by the Spanish Ministerio de Educación under the National F.P.U. Program (FPU19/01118).

[2] https://github.com/nicslabdev/ntrureencrypt

Funding for open access charge: Universidad de Málaga / CBUA.

References

[1] P. P. Ray, "A survey of IoT cloud platforms," *Future Computing and Informatics Journal*, vol. 1, pp. 35–46, Dec. 2016.

[2] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty Security Considerations for Cloud-Supported Internet of Things," *IEEE Internet of Things Journal*, vol. 3, pp. 269–284, June 2016. Conference Name: IEEE Internet of Things Journal.

[3] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics," 2010.

[4] V. Sucasas, A. Aly, G. Mantas, J. Rodriguez, and N. Aaraj, "Secure Multi-Party Computation-Based Privacy-Preserving Authentication for Smart Cities," *IEEE Transactions on Cloud Computing*, vol. 11, pp. 3555–3572, Oct. 2023. Conference Name: IEEE Transactions on Cloud Computing.

[5] Y. Yang, J. Wu, C. Long, W. Liang, and Y.-B. Lin, "Blockchain-Enabled Multiparty Computation for Privacy Preserving and Public Audit in Industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 18, pp. 9259–9267, Dec. 2022. Conference Name: IEEE Transactions on Industrial Informatics.

[6] M. M. Ogonji, G. Okeyo, and J. M. Wafula, "A survey on privacy and security of Internet of Things," *Computer Science Review*, vol. 38, p. 100312, Nov. 2020.

[7] P. M. Chanal and M. S. Kakkasageri, "Preserving Data Confidentiality in Internet of Things," *SN Computer Science*, vol. 2, p. 53, Jan. 2021.

[8] S. Tonyali, N. Saputro, and K. Akkaya, "Assessing the feasibility of fully homomorphic encryption for Smart Grid AMI networks," in *2015 Seventh International Conference on Ubiquitous and Future Networks*, pp. 591–596, July 2015. ISSN: 2165-8536.

[9] A. Marandi, P. G. M. R. Alves, D. F. Aranha, and R. H. Jacobsen, "Lattice-Based Homomorphic Encryption For Privacy-Preserving Smart Meter Data Analytics," *The Computer Journal*, p. bxad093, Sept. 2023.

[10] Q. Zhang, C. Xin, and H. Wu, "Privacy-Preserving Deep Learning Based on Multiparty Secure Computation: A Survey," *IEEE Internet of Things Journal*, vol. 8, pp. 10412–10429, July 2021.

[11] A. Beimel, "Secret-Sharing Schemes: A Survey," in *Coding and Cryptology* (Y. M. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing, eds.), (Berlin, Heidelberg), pp. 11–46, Springer, 2011.

[12] J. Herranz, A. Ruiz, and G. Sáez, "New results and applications for multi-secret sharing schemes," *Designs, Codes and Cryptography*, vol. 73, pp. 841–864, Dec. 2014.

[13] I. Chillotti, E. Orsini, P. Scholl, N. P. Smart, and B. Van Leeuwen, "Scooby: Improved Multi-party Homomorphic Secret Sharing Based on FHE," in *Security and Cryptography for Networks* (C. Galdi and S. Jarecki, eds.), Lecture Notes in Computer Science, (Cham), pp. 540–563, Springer International Publishing, 2022.

[14] D. Evans, V. Kolesnikov, and M. Rosulek, "A Pragmatic Introduction to Secure Multi-Party Computation,"

[15] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty Computation from Somewhat Homomorphic Encryption," in *Advances in Cryptology – CRYPTO 2012*, pp. 643–662, Springer, 2012.

[16] V. Goyal, Y. Song, and C. Zhu, "Guaranteed Output Delivery Comes Free in Honest Majority MPC," in *Advances in Cryptology – CRYPTO 2020* (D. Micciancio and T. Ristenpart, eds.), Lecture Notes in Computer Science, (Cham), pp. 618–646, Springer International Publishing, 2020.

[17] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek, and N. Aaraj, "Survey on Fully Homomorphic Encryption, Theory, and Applications," *Proceedings of the IEEE*, vol. 110, pp. 1572–1609, Oct. 2022. Conference Name: Proceedings of the IEEE.

[18] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology — EUROCRYPT '99* (J. Stern, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 223–238, Springer, 1999.

[19] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Advances in Cryptology* (G. R. Blakley and D. Chaum, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 10–18, Springer, 1985.

[20] Z. Qin, H. Xiong, S. Wu, and J. Batamuliza, "A Survey of Proxy Re-Encryption for Secure Data Sharing in Cloud Computing," *IEEE Transactions on Services Computing*, pp. 1–1, 2016. Conference Name: IEEE Transactions on Services Computing.

[21] R. Canetti and S. Hohenberger, "Chosen-Ciphertext Secure Proxy Re-Encryption," 2007. Publication info: Published elsewhere. Full version of paper in ACM CCS 2007.

[22] D. Stehlé and R. Steinfeld, "Making NTRU as Secure as Worst-Case Problems over Ideal Lattices," in *Advances in Cryptology – EUROCRYPT 2011* (K. G. Paterson, ed.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 27–47, Springer, 2011.

[23] D. Stehlé and R. Steinfeld, "Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices," 2013. Publication info: Published elsewhere. Submitted. Some of the results in this paper have been presented in preliminary form at Eurocrypt 2011.

[24] D. Nuñez, I. Agudo, and J. Lopez, "NTRUReEncrypt: An Efficient Proxy Re-Encryption Scheme Based on NTRU," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, (Singapore Republic of Singapore), pp. 179–189, ACM, Apr. 2015.

[25] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai, "Threshold Cryptosystems from Threshold Fully Homomorphic Encryption," in *Advances in Cryptology – CRYPTO 2018*, pp. 565–596, Springer International Publishing, 2018.

[26] Y. Lindell, "How to Simulate It – A Tutorial on the Simulation Proof Technique," in *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich* (Y. Lindell, ed.), pp. 277–346, Cham: Springer International Publishing, 2017.

[27] V. Gayoso Martínez, L. Hernández Encinas, and A. Queiruga Dios, "Security and Practical Considerations When Implementing the Elliptic Curve Integrated Encryption Scheme," *Cryptologia*, vol. 39, pp. 244–269, July 2015.

[28] J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, W. Whyte, and Z. Zhang, "Choosing Parameters for NTRUEncrypt," 2015. Publication info: Preprint. MINOR revision.

[29] "TFHE - Security and Parameters." Last accessed: 2024-10-04, https://tfhe.github.io/tfhe/security_and_params.html.

[30] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. Last accessed: 2024-10-04, https://crypto.stanford.edu/craig/.

Daniel Morales is a Ph.D. candidate in the Department of Computer Science at the University of Malaga. His research interests are comprised of developing cryptographic protocols, highly focused on the privacy area, and designing decentralized and trustless infrastructures.

Isaac Agudo is an Associate Professor in the Department of Computer Science at the University of Malaga. He has been involved in several research projects, and is very active in technology transfer with international companies. His main research interests include security and privacy in areas such as blockchain, or smart devices. In particular, he is currently working on privacy preserving access control and information sharing.

Javier Lopez is a Full Professor in the Department of Computer Science at the University of Malaga. His research activities are mainly focused on network security, security protocols and critical information infrastructures, and he leads a number of national and international research projects in these areas.