Multiplicative Partially Homomorphic CRT Secret Sharing

(Preliminary Version)

Shlomi Dolev, IEEE Fellow
*D*ept. of Computer Science
Ben-Gurion University of the Negev
Beersheba, Israel
dolev@cs.bgu.ac.il

Yaniv Kleinman
*D*ept. of Computer Science
Ben-Gurion University of the Negev
Beersheba, Israel
yanivkl@post.bgu.ac.il

***Abstract*—A new CRT-based positive (non-zero) secret-sharing scheme with perfect information-theoretic (PIT) security and multiplicative homomorphism is presented. The scheme is designed to support the evaluation of multiplications of non-zero secrets of multiplicative groups.**

Our CRT-based scheme is partially homomorphic, supporting homomorphic multiplications. Nevertheless, our scheme has the potential to be regarded as fully homomorphic for practical scenarios, such as bounded-sized multi-cloud databases.

***Index Terms*—Secret Sharing, Perfect Information Theoretic, Homomorphism, Chinese Remainder Theorem.**

I. Introduction

Secret sharing and secure multiparty computation are essential in securing data and computation. Perfect information-theoretic (PIT) secure multiparty computation is preferred over computational. The computational solution for a fully homomorphic[1] solution has the benefit of being performed by a single (usually honest but curious, see, e.g., [9]) server. On the other hand, the computation-based solution is • based on an unproven candidate for a one-way function, • is required to protect an encryption/decryption key, and • has no redundancy as the server can erase the entire data. Thus, the distributed PIT secure-multi-party computation is employed in many scenarios and efficient versions, where the communication overhead, which is originally quadratic in the number of participants, is of great interest. Ultimately, the addition homomorphism of secret sharing can be extended to be homomorphism of both addition and multiplication, allowing communication-less distributed secure multiparty computations.

We present (CRT-based) secret sharing that supports multiplicative homomorphism. In practice, there is a rising need for cloudifying storage and computing. Users of cloud services want to be sure that their sensitive data is not exposed to the service provider. Cloud computing allows companies to focus on their primary objective, while cloud providers handle all of the storage and computing infrastructure.

In more specific terms, so far, secure migration of data and computation to the cloud are based on the following:

- Fully Homomorphic Encryption(FHE), e.g., [8]. Also, a distributed case of (partially computational secure) FHE, e.g., [14].
- Partially (typically addition) Homomorphic Secret Sharing Schemes, e.g., [1], [6], [10].
- Secure Multiparty Computation with communication between the participants, e.g., [11].

The first approach, where data is encrypted to hide its content, suffers from some serious drawbacks:

- Encryption is done using a key that needs to be stored.
- Encryptions are mainly based on the (unproven) hardness of (one-way) computing problems. Those problems are assumed to imply the need for a brute-force search for the encryption keys. Unfortunately, the most popular functions, DH (based on Discrete Log) and RSA (based on Integer Factorization), can be broken using Shor's algorithm; designed for emerging quantum computing. Furthermore, the recently proposed (by NIST) post-quantum functions are also not proven to be hard for (quantum) computers to solve.
- Encryption can be very time-consuming, for example, calculating powers on numbers done in RSA.

The other approaches, besides encryption, are distributed Secret Sharing (SS) and distributed Secure Multiparty Computations (SMC or SMPC) based on PIT secure proofs. These methods are based on mathematical proofs that ensure the secret data is being secured as long as the adversary does not have a sufficient number of shares defined by a threshold to recover the secret.

[1]Homomorphism is a map between two algebraic structures of the same type that preserve the operation of the structures [3]. A homomorphic operation can be addressed by the equation: $f(x \oplus y) = f(x) \oplus f(y)$, where $'\oplus'$ is a binary mathematical operation in this case. Homomorphism is a property of the function f. Homomorphism can be limited to several operations; in this case, the scheme is only partially homomorphic.

The first secret-sharing schemes were presented by Shamir [6] and Blakley [10]. Later many schemes were presented, including schemes based on the Chinese Remainder Theorem (CRT), e.g. Asmuth-Bloom's scheme [1]. The above schemes are designed to respect a threshold of t out of n shares to reconstruct the secret, where $t \leq n$ shares are enough to reconstruct the secret fully. The terminology for using such a reconstruction threshold is called a *threshold scheme*.

The above schemes are partially or not homomorphic, meaning they support bounded (addition and multiplication) operations' performance or do not support one (either addition or multiplication) or both operations. We present a new Secret Sharing Scheme inspired by the Asmuth-Bloom scheme. Our scheme is multiplicative homomorphism (namely, partially homomorphic) and has the potential to allow fully homomorphic operations under some restrictions and modifications. Shamir's and Asmuth-Bloom's schemes are perfect secret-sharing schemes, and so is our proposed base multiplicative scheme. The fully homomorphic extension of our scheme is only statistically secure rather than PIT secure.

II. Preliminaries

Secret Sharing Schemes are typically defined by two parameters (n, t). n is the number of participants, and t is the reconstruction threshold. The reconstruction threshold states the minimum number of participants needed to reconstruct the secret. The secret is a bit string or an integer denoted by S. (Ramp) Secret Sharing Schemes can have two thresholds, t and s. The additional parameter $s \leq t$ is a secrecy bound. The secrecy bound states the maximum number of participants who cannot learn new information about the secret. Schemes in which $s < t$ are referred to as Ramp Schemes:

Definition 1. *Ramp Scheme*: A scheme that uses a secrecy bound s.
In most of the threshold schemes, the secrecy bound is $s = t - 1$ whereas in ramp schemes, the secrecy bound can be $1 \leq s < t$.

For a group of participants G, we have three ranges in this kind of scheme:

1) $|G| \leq s$: No information leak at all.
2) $s < |G| < t$: Might be a partial information leak.
3) $|G| \geq t$: Fully reconstruction.

Another important notation is the access structure. An access structure is a set of all the qualified groups of participants who are authorized and should be capable of reconstructing the secret data S. The access structure is marked $\mathcal{A}$. In the sequel, we restrict the possible secret's value. Distribution of the secret S is signed with $\mathcal{D}$. In our scheme, we have no restrictions on the distribution other than there can not be a zero in the multiplicative group. The secret's (possible) domain is signed with $\mathcal{S}$ whereas the participant $i'th$ shared part's (possible) domain is signed with $\mathcal{S}_i$.

Definition 2. A *Perfect Secret Sharing Scheme* should satisfy the following two conditions:

- *Correctness*: Any qualified group of participants in $\mathcal{A}$ can reconstruct the secret.
- *Perfect Privacy*: No unqualified group of participants in $\bar{\mathcal{A}}$ can get any information about the secret. More formally: Given any $S' \in \mathcal{S}$:
  - $Pr[S = S'] = p$.
  - For every group $G \in \bar{\mathcal{A}}, |G| < n$ and $S' \in \mathcal{S}$ where $S'_{i_j} \in \mathcal{S}_{i_j}, j \in G \subset [n]$, we have:
$$Pr[S = S' | S_{i_j} = S'_{i_j} \forall j \in G] = p$$

Definition 3. A *Ramp Secret Sharing Scheme* is a ramp scheme that should satisfy the following two conditions:

- *Correctness*: Any qualified group of participants in $\mathcal{A}$ can reconstruct the secret.
- *Perfect Ramp Privacy*: For every group of participants $G, |G| \leq s$, given a secret S with distribution $\mathcal{D}$, the secret distribution stays the same, meaning the probability of the secret being equal to a specific element $S' \in \mathcal{S}$ stays the same even when knowing the shared secret data held by the participants of G. More formally: Given any $S' \in \mathcal{S}$:
  - $Pr[S = S'] = p$.
  - For every $S' \in \mathcal{S}$ and $S'_{i_j} \in \mathcal{S}_{i_j}, 1 \leq j \leq s$, with $1 \leq i_1 < i_2 < \cdots < i_s \leq n$, we have:
$$Pr[S = S' | S_{i_1} = S'_{i_1}, \ldots, S_{i_s} = S'_{i_s}] = p$$

The following sum up the relations between the scheme factors:

1) $1 \leq s < t \leq n$
2) The gap from s to t does not have to be one.
3) It is possible that a group of participants smaller than t could reconstruct the data.
4) It is possible that a group of participants larger than s would not learn anything about the data.

In order to move on and examine some Secret Sharing Schemes, we need to introduce additional notations. The secret share of the $i'th$ participant is S_i. To note that a calculation is performed under modulo p or inside $\mathbb{Z}_p$, we use the notation $[\cdot]_p$. When referring to the multiplicative group of integers modulo M, we will use U_M. In some more complex schemes, as in our case, some moduli are performed by the same participant – in particular, one modulo for one part of the secret and another modulo for another part of it. In order to define those moduli, we will use m_i, which is the $i'th$ number to perform the modulus calculations on. We can also address m_{i+j}, which is the $(i + j) (\mathrm{mod}\ r)$ number to perform the

modulus calculations on. We added the $(\mathrm{mod}\ r)$ since we are using r different modulus calculations in those schemes, and we want the cyclic property. The r parameter is driven from n and t. When n is equal to t, then r can also be equal to n, but for example, when $n > t$, then r is usually greater than n because more redundancy is needed in schemes supporting threshold smaller than the number of participants.

III. Related Work

Some of the known SMPC homomorphic methods relevant to this research are:

Simple additive SSS – In this scheme, the sum of shared values reconstructs the secret. One such scheme has a threshold of $t = n-1$, meaning it is an n out of n scheme. Let S be the secret, n be the number of participants, and $\mathbb{Z}_p$ be the field on which the calculations are being made [5].

- Distribution phase:
  The dealer randomly chooses the group elements $S_1, S_2, ..., S_{n-1} \in \mathbb{Z}_p$. The dealer then computes $S_n = S - \sum_{1 \le i < n} S_i$. Finally, the dealer sends S_i to participant i for $1 \le i \le n$.
- Reconstruction phase:
  Let G be a group of participants gathered to reconstruct the secret. The participants compute the secret S by summing all their secret shares.
  $RF(\bigcup_{p_i \in G} S_i) : \begin{cases} if\ |G| = n \Rightarrow S = \sum_{1 \le i \le n} S_i (\mathrm{mod}\ p) \\ if\ |G| < n \Rightarrow \perp \end{cases}$
- Additive homomorphism:
  Let S_1 and S_2 be secrets and S_{1_i} and S_{2_i} the secrets' shares of the i'th participant. $S_1 + S_2 = \sum_{1 \le i \le n} S_{1_i} + \sum_{1 \le i \le n} S_{2_i} = \sum_{1 \le i \le n} S_{1_i} + S_{2_i}$. Each participant can perform $S_{1_i} + S_{2_i}$ and send the sum for the reconstruction phase. Therefore, the scheme is an additive homomorphic scheme.

Simple multiplicative homomorphic SSS – This scheme is very similar to the additive one, but in this case, the product of the shared values gives the secret. Moreover, there are some restrictions; Zero or numbers that are not co-prime to p are not allowed to be used. One such scheme has a threshold of $t = n - 1$, meaning it is an n out of n scheme. Let S be the secret, n be the number of participants, and U_p be the group on which the calculations are being made. [2]:

- Distribution phase:
  The dealer picks $n - 1$ uniformly random nonzero elements $S_i, 1 \le i < n$, from U_p. The dealer then calculates $S_n = S \cdot (\prod_{1 \le i < n} S_i)^{-1}$. Finally, the dealer sends S_i to participant i for $1 \le i \le n$.
- Reconstruction phase:
  Let G be a group of participants gathered to reconstruct the secret. The participants compute the secret S by multiplying all their secret shares.
  $RF(\bigcup_{p_i \in G} S_i) : \begin{cases} if\ |G| = n \Rightarrow S = \prod_{1 \le i \le n} S_i (\mathrm{mod}\ p) \\ if\ |G| < n \Rightarrow \perp \end{cases}$
- Multiplicative homomorphism:
  Let S_1 and S_2 be secrets and S_{1_i} and S_{2_i} the secrets' shares of the i'th participant. $S_1 \cdot S_2 = \prod_{1 \le i \le n} S_{1_i} \cdot \prod_{1 \le i \le n} S_{2_i} = \prod_{1 \le i \le n} S_{1_i} \cdot S_{2_i}$. Each participant can perform $S_{1_i} \cdot S_{2_i}$ and send the product to the reconstruction phase. Therefore, the scheme is a multiplicative homomorphic scheme. All the multiplications are correct and do not form a zero or a number with a common divider with p because U_p is a multiplicative group.

Ramp additive homomorphic SSS – [4]. This scheme is based on the ideas of Asmuth-Bloom SSS [1] as mentioned earlier in the background. The scheme is an n out of n scheme with a security factor of s. This security factor means there is no information leak unless there are at least $s + 1$ secret sharing parts. Using such a security factor s is called a *ramp scheme*. Let S be the secret, n the number of participants, and $\mathbb{Z}_{\text{prod}}$ the group on which the calculations are being made.

- Distribution phase:
  The dealer chooses a set of integers $(\text{prod}, m_1, m_2, ..., m_n)$ such that:
  1) $m_1 < m_2 < ... < m_n$ and $S < \text{prod} = M_n = \prod_{i=1}^{n} m_i$
  2) $gcd(m_i, m_j) = 1 (\forall i \ne j)$

  The dealer randomly chooses s integers $(r_1, ..., r_s)$ in $\mathbb{Z}_{\text{prod}}$, and computes $S_{\text{mix}} = [S + \sum_{i=1}^{n} r_i]_{\text{prod}}$.
  The dealer computes and distributes the shared set of each participant i:

  $$S_i = (S_{\text{mix}} (\mathrm{mod}\ m_i), r_1 (\mathrm{mod}\ m_{i+1}), \ldots, r_s (\mathrm{mod}\ m_{i+s})).$$

- Reconstruction phase:
  Let G be a group of participants gathered to reconstruct the secret. The participants compute the secret S by solving the CRT equations as seen in "Fig.1".

- Additive homomorphism:
  Let S_1 and S_2 be secrets. $S_{1_{\text{mix}}}, r_{1_1}, ..., r_{1_s}$ are the blinded secret and all the blinding randoms of S_1. $S_{2_{\text{mix}}}, r_{2_1}, ..., r_{2_s}$ are the blinded secret and all the blinding randoms of S_2. Each participant i has the secret shares of each blinded secret and blinding randoms:
  $S_{1_{mix_i}}, r_{1_{1_i}}, ..., r_{1_{s_i}}, S_{2_{mix_i}}, r_{2_{1_i}}, ..., r_{2_{s_i}}$.
  Now we will show that this scheme is additive homomorphic:
  $S_1 + S_2 = S_{1_{\text{mix}}} - \sum_{1 \le j \le s} r_{1_j} + S_{2_{\text{mix}}} - \sum_{1 \le j \le s} r_{2_j} = S_{1_{\text{mix}}} + S_{2_{\text{mix}}} - \sum_{1 \le j \le s} r_{1_j} + r_{2_j}$. Each participant can perform $S_{1_{mix_i}} + S_{2_{mix_i}}$ and $r_{1_{j_i}} + r_{2_{j_i}}$ for each $1 \le j \le s$ and send the sum of all the needed parts to the

$$RF\begin{pmatrix}\bigcup\limits_{p_i\in G} S_{i,0}\\ \bigcup\limits_{p_i\in G} S_{i,1}\\ \dots\\ \bigcup\limits_{p_i\in G} S_{i,s}\end{pmatrix}:\begin{cases} if\ |G|=n: \begin{pmatrix} S_{\text{mix}}=CRT[S_{1,0},...,S_{n,0}]_{\text{prod}}\\ r_1=CRT[S_{1,1},...,S_{n,1}]_{\text{prod}}\\ \dots\\ r_s=CRT[S_{1,s},...,S_{n,s}]_{\text{prod}}\end{pmatrix}\\ \Rightarrow S=[S_{\text{mix}}-\sum_{i=1}^{s} r_i]_{\text{prod}}\\ if\ s+1\le|G|<n\Rightarrow partial\ information\\ if\ |G|<s+1\Rightarrow\perp\end{cases}$$

Fig. 1. Ramp additive homomorphic SSS - Reconstruction phase

reconstruction phase, where using a CRT solver algorithm. Therefore, the scheme is an additive homomorphic scheme.

IV. Motivation for the New Scheme

In this work, we want to introduce a novel SSS with the feature of multiplicative homomorphism. There are already schemes that allow homomorphic multiplication, one of them found in the related work Section(III). However, under some disclaimers, our scheme can be extended to support more features, such as threshold reconstruction and additive homomorphism, as suggested in the following sections.

V. Method Explanation

The scheme is an n out of n scheme with a security factor of s, meaning that without having at least $s+1$ secret sharing parts, there is no information leak. Using such a security factor s is called a *ramp scheme*. Let $S \in U_{\text{prod}}$ be the secret, n be the number of participants, and U_{prod} be the group in which the calculations are being made.

- Distribution phase:
  The dealer chooses a set of pairwise co-primes $m_1, m_2, ..., m_n$ and calculate prod $= \prod_{1\le i\le n} m_i$ such that:

  1) $m_1 < m_2 < ... < m_n$ and $S <$ prod $= M_n$.

  2) $\gcd(m_i, m_j) = 1, \forall i \neq j$.

  The dealer randomly chooses s integers $r_1, ..., r_s$ in U_{prod}, and computes $S_{\text{mix}} = [S \cdot \prod_{1\le i\le s} r_i]_{\text{prod}}$. The dealer computes and distributes the shared set of each participant $1 \le i \le n$:

  $$S_i = (S_{\text{mix}}(\text{mod } m_i), r_1(\text{mod } m_{i+1}), \dots, r_s(\text{mod } m_{i+s})).$$

- Reconstruction phase:
  Let G be a group of participants gathered to reconstruct the secret. The participants compute the secret S by solving the CRT equations, following the steps of the reconstruction function seen in "Fig.2".

A. Auxiliary Claims

Corollary 1. *Given the multiplicative group $A = U_{m_1\cdot m_2\cdot\dots\cdot m_n} = U_{M_n}$ there is an isomorphism to the direct product of M_n pairwise co-primes dividers, meaning $B = U_{m_1} \times U_{m_2} \times ... \times U_{m_n}$.*

Corollary 2. *Define $M_n = m_1 \cdot m_2 \cdot \dots \cdot m_n$ where $m_1, m_2, \dots, m_n$ are pairwise co-primes. Given element $\alpha \in A = U_{M_n}$ of a finite group:*
$\forall\gamma \in U_{M_n}, \exists\beta \in U_{m_n}$ s.t. $\alpha\cdot\beta = \gamma$. Meaning that all elements can be the product of α and another element in the group.

Corollary 3. *Let G be a finite group. Given two elements $g_1, g_2 \in G$ chosen randomly, uniformly and independently, the multiplication $g_1 \cdot g_2 = g\prime \in G$ is a randomly uniformly element of G.*

Corollary 4. *Let G be a finite group. Given two elements $g_1, g_2 \in G$, where g_1 is taken from a uniform distribution and g_2 is from some distribution $\mathcal{D}$. g_1 and g_2 are chosen independently. The multiplication $g_1 \cdot g_2 = g\prime \in G$ is a randomly uniform element of G.*

Proofs for all the corollaries can be found in Appendix (A)

B. Correctness

In order to use a scheme, one must know that the scheme is always correct.

Theorem 1. *The multiplicative scheme can always be reconstructed given a group of participants $G \in \mathcal{A}$.*

Proof. Given a group of participants $G \in \mathcal{A}$. The scheme is of threshold $t = n-1$, meaning $|G| = n$. The correctness of this scheme relies on the fact that all elements in U_{M_n} have an inverse. Each $r_i, 1 \le i \le s$ can be reconstructed from the n shares of its modulus using CRT as $r_i \in U_{M_n}$. CRT can reconstruct numbers to the product of modulus pairwise co-primes equations, meaning one solution in $\mathbb{Z}_{M_n}$ because all the modulus are the pairwise co-primes factors of M_n. After reconstructing $r_i, \forall 1 \le i \le s$, we can calculate r_i^{-1}. That is

$$RF\left(\begin{array}{c}\bigcup\limits_{p_i\in G} S_{i,0}\\ \bigcup\limits_{p_i\in G} S_{i,1}\\ \cdots\\ \bigcup\limits_{p_i\in G} S_{i,s}\end{array}\right) : \begin{cases} if\ |G| = n : \begin{pmatrix} S_{\text{mix}} = CRT[S_{1,0},...,S_{n,0}]_{\text{prod}} \\ r_1 = CRT[S_{1,1},...,S_{n,1}]_{\text{prod}} \\ \cdots \\ r_s = CRT[S_{1,s},...,S_{n,s}]_{\text{prod}} \end{pmatrix} \\ \Rightarrow S = [S_{\text{mix}} \cdot \prod_{i=1}^{s} r_i^{-1}]_{\text{prod}} \\ if\ s+1 \le |G| < n \Rightarrow partial\ information \\ if\ |G| < s+1 \Rightarrow \perp \end{cases}$$

Fig. 2. Ramp multiplicative homomorphic SSS - Reconstruction phase

the reason why we *must* take the randoms from U_{M_n} and not from $\mathbb{Z}_{M_n}$. To get the secret, we calculate:

$$S_{\text{mix}} \cdot \prod_{i=1}^{s} r_i^{-1} = S \cdot \prod_{i=1}^{s} r_i \cdot \prod_{i=1}^{s} r_i^{-1} = S.$$

That can be done because multiplication is associative in U_{M_n}. □

C. Multiplicative Homomorphism

Theorem 2. *The multiplicative scheme is multiplicatively homomorphic.*

Proof. To show that the scheme is multiplicative homomorphic, we will show that when taking k secrets, distributing them, multiplying them as shares on the participants' side and finally reconstructing the result, the result will be correct. □

An expanded explanation of this proof can be found in Appendix (B).

D. Security Analysis

Since we want to use the scheme to store and make operations on secret data, we want to ensure that the scheme holds the security properties we need.

Theorem 3. *The multiplicative scheme is a Perfect ramp secret sharing scheme and is a perfect secret sharing scheme in case* $s = n - 1$.

For the simplicity of notation and sizes, we will use $m_i, \forall 1 \le i \le n$ as primes and not the general case of pairwise co-primes. To get some *intuition*, we will start by analyzing the basic case of $s = 1$. The domain of S_{mix} and r_1 is U_{M_n} and the size of the domain is $|U_{M_n}| = \varphi(M_n) = \prod_{1\le i\le n} \varphi(m_i)$. For any elements $r_1', S'_{\text{mix}}, S' \in U_{M_n}$, we have:
$Pr[r_1 = r_1'] = \frac{1}{\varphi(M_n)}$ as r_1 is randomly uniformly chosen.
$Pr[S = S'] = p$ as S is chosen from some distribution $\mathcal{D}$.
$Pr[S_{\text{mix}} = S'_{\text{mix}}] = \frac{1}{\varphi(M_n)}$ as r_1 is randomly uniformly chosen, and the secret S is of some distribution, based on Corollary 4.

The probabilities of r_1 and S_{mix} to be equal to r_1' and S'_{mix}, respectively, do change knowing some i'th participant data since $s = 1$:

- $Pr[r_1 = r_1' | r_1 (\text{mod } m_{i+1}) = r_1' (\text{mod } m_{i+1})] = \frac{1}{\frac{\varphi(M_n)}{\varphi(m_{i+1})}} = \frac{\varphi(m_{i+1})}{\varphi(M_n)}$[2],
- $Pr[S_{\text{mix}} = S'_{\text{mix}} | S_{\text{mix}} (\text{mod } m_i) = S'_{\text{mix}} (\text{mod } m_i)] = \frac{1}{\frac{\varphi(M_n)}{\varphi(m_i)}} = \frac{\varphi(m_i)}{\varphi(M_n)}$.

Yet, we claim that the conditional probability of the secret S to be equal to S' does not change:

$$Pr[S = S' | \begin{array}{l} r_1 (\text{mod } m_{i+1}) = r_1' (\text{mod } m_{i+1}), \\ S_{\text{mix}} (\text{mod } m_i) = S'_{\text{mix}} (\text{mod } m_i)] \end{array} = p$$

We can write the conditional probability as shown in "Fig.3".

Now

$$\begin{array}{l} Pr[S_{\text{mix}} (\text{mod } m_i) = S'_{\text{mix}} (\text{mod } m_i)] = \\ Pr[r_1 \cdot {r_1'}^{-1} (\text{mod } m_i) = S' \cdot S^{-1} (\text{mod } m_i)] = \frac{1}{\varphi(m_i)} \end{array} \quad (1)$$

and

$$\begin{array}{l} Pr[r_1 (\text{mod } m_{i+1}) = r_1' (\text{mod } m_{i+1})] = \\ Pr[r_1 \cdot {r_1'}^{-1} (\text{mod } m_{i+1}) = 1 (\text{mod } m_{i+1})] = \frac{1}{\varphi(m_{i+1})}. \end{array} \quad (2)$$

Since the events in "(1)" and "(2)" are clearly independent, we obtain:

$$Pr[\begin{array}{l} r_1 (\text{mod } m_{i+1}) = r_1' (\text{mod } m_{i+1}) \quad \wedge \\ S_{\text{mix}} (\text{mod } m_i) = S'_{\text{mix}} (\text{mod } m_i) \quad] = \frac{1}{\varphi(m_{i+1})} \cdot \frac{1}{\varphi(m_i)}. \end{array}$$

Since S, S' and r_1, r_1' are independent:

$$\begin{array}{l} Pr[S = S' \wedge r_1 (\text{mod } m_{i+1}) = r_1' (\text{mod } m_{i+1}) \wedge \\ \quad S_{\text{mix}} (\text{mod } m_i) = S'_{\text{mix}} (\text{mod } m_i)] = \\ Pr[S = S' \wedge r_1 \cdot {r_1'}^{-1} (\text{mod } m_{i+1}) = 1 (\text{mod } m_{i+1}) \wedge \\ \quad r_1 \cdot {r_1'}^{-1} (\text{mod } m_i) = S' \cdot S^{-1} (\text{mod } m_i)] = p \cdot \frac{1}{\varphi(m_{i+1})} \cdot \frac{1}{\varphi(m_i)}. \end{array}$$

and,

Finally:

$$\frac{Pr[S=S' \wedge r_1 (\text{mod } m_{i+1}) = r_1' (\text{mod } m_{i+1}) \wedge S_{\text{mix}} (\text{mod } m_i) = S'_{\text{mix}} (\text{mod } m_i)]}{Pr[r_1 (\text{mod } m_{i+1}) = r_1' (\text{mod } m_{i+1}) \wedge S_{\text{mix}} (\text{mod } m_i) = S'_{\text{mix}} (\text{mod } m_i)]} = p,$$

[2]same as a secret shared in Mignotte's scheme, which is the equivalence class of $r_1 (\text{mod } m_{i+1})$

$$Pr[S = S' | r_1(\text{mod } m_{i+1}) = r'_1(\text{mod } m_{i+1}), S_{\text{mix}}(\text{mod } m_i) = S'_{\text{mix}}(\text{mod } m_i)]$$

$$= \frac{Pr[S = S' \wedge r_1(\text{mod } m_{i+1}) = r'_1(\text{mod } m_{i+1}) \wedge S_{\text{mix}}(\text{mod } m_i) = S'_{\text{mix}}(\text{mod } m_i)]}{Pr[r_1(\text{mod } m_{i+1}) = r'_1(\text{mod } m_{i+1}) \wedge S_{\text{mix}}(\text{mod } m_i) = S'_{\text{mix}}(\text{mod } m_i)]}.$$

Fig. 3. Conditional probability of $S = S'$ with the knowledge of i'th participant knowledge

as required.

The security analysis proof for the general case can be found in Appendix (C)

VI. Conclusion Remarks

The addition expansion of our multiplicative method. Returning to our main goal - performing fully homomorphic operations on secret data. This goal is equivalent to being able to calculate any polynomial function on secret data. All polynomials can be represented as the addition of products called monomials, with no parentheses.
In the proposed scheme, the dealer must know the function we would like to calculate in advance. Given a function represented as the addition of monomials, the dealer chooses the randoms so that the product of all monomials' randoms is equal. For example, for a function represented as the addition of monomials:

$$f(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 + x_2 \cdot x_3 + x_1 \cdot x_3 \cdot x_4.$$

An example of randoms for the given equation:

$$r_1 = 6, r_2 = 11, r_3 = r_1 = 6, r_4 = 11 \cdot [6^{-1}]_{M_n}.$$

As this example shows, there are many dependencies between the randoms. In order to break some of these dependencies, we offer two methods:

1) Each occurrence of a variable can be hidden with a different random number. Using this method, the number of variables to be stored equals all occurrences of the variables in the equation.
2) We can add one variable to each monomial with the value 1. Then we can give each variable an actual random number to hide the variable value and, in the end, fix the product of the randoms with this independent random number multiplied by the 1 variable.

Based on the methods described above, one can create a random selection for the function that needs to be calculated - depending on the trade-offs between flexibility and performance. Moreover, a deeper analysis needs to be done about the security of each method.

Addition under the limitation of calculations with a threshold. As defined in our proposed scheme, $m_1 < m_2 < \cdots < m_n$ are the co-primes that the modulus operations are performed on the participants' side. So, when the calculations are done under the limit of m_1, all results and intermediate results are elements of U_{M_n}. This means that all the calculations are perfectly secured by the security analysis performed in Section V-D.

Unbounded calculations. In this case, the security analysis needs to deal with the different scenarios. If, for example, a calculation or an intermediate calculation S_{res} is not from U_{M_n}, then there is a modulo m_i such that $m_i | S_{\text{res}_{\text{mix}}}$, meaning that the participant with the m_i modulo used for S_{mix} can know the equivalence class of the real secret[3]: $S = S + k \cdot m_i, \exists k$.

Arithmetic circuit extension. We can calculate all the randoms needed for any given function in advance by parsing the equation. We can ensure that each addend's randoms are equal for each addition operation in the function. To achieve what we described here and avoid leakage, we suggest the following guidelines:

1) Before each addition, we are fixing the second multiplier with 1 (similar to the fix suggested for additions).
2) Add random and subtract the same random from the two (or more, to have no influence on the actual result, as we ensure that the total sum of these new values is zero) addends of each addition operation.

The first guideline is just an expansion of the first method described in the addition expansion of our original scheme. The second guideline is a way of dealing with leakage that may appear - Since both addends are hidden with the same random, it is possible to know if they are from the same equivalence class or not (if we limit the calculation to be $S_{res} < m_1$ then it is even possible to know if they are equal or not). The addition and subtraction of the same random do not change the value of the function, but they avoid revealing information on possible equality. Since each operation (addition and subtraction) requires a later multiplication by 1 (to allow the addition), and the computation is done once (over the same servers) leakage of (equality) information is avoided.

Byzantine fault tolerance. It is possible to make our scheme a threshold scheme with threshold $t < n$, as explained in [5], in Section 2.4. Therefore, it is possible to create a byzantine fault tolerance algorithm for our scheme by taking ideas from [12] and [13]. The most trivial way is to transform our scheme to a threshold scheme with $t < n$ using redundancy [5]. Then during the reconstruction phase, check all results reconstructed from groups of a sufficient amount of participants ($|G| \geq t$).

[3]All the randoms are from U_{M_n}; therefore, they do not change the effect of zero divisors.

After reconstructing the results, take the result from the majority of the groups (if there is no majority, then there were more adversaries than we can handle), see e.g., [12] and the references therein for more efficient schemes.

VII. References

[1] C. Asmuth, J. Bloom, A modular approach to key safeguarding, IEEE Transactions on Information Theory (Volume: 29, Issue: 2, Mar 1983) 208 - 210.

[2] Dor Bitan, Shlomi Dolev, Optimal-round preprocessing-MPC of polynomials over non-zero inputs via distributed random matrix. Wireless Netw (2022).

[3] Stanley Burris, H.P. Sankappanavar, A Course in Universal Algebra (2012).

[4] Oğuzhan Ersoy, Thomas Brochmann Pedersen, Emin Anarim, Homomorphic extensions of CRT-based secret sharing, Discrete Applied Mathematics (Volume 285, 15 October 2020) 317-329.

[5] Yildirim, İsmail Fatih, SecurePL: A compiler and toolbox for practical and easy secure multiparty computation, Master Thesis Sabanci University (2008).

[6] Adi Shamir, How to share a secret, Communications of the ACM (Volume 22, Issue 11, 01 November 1979), 612–613.

[7] Leslie G. Valiant, Why is Boolean complexity theory difficult?, Proceedings of the London Mathematical Society Symposium on Boolean function complexity (1992), 84–94.

[8] Craig Gentry, Fully homomorphic encryption using ideal lattices, Proceedings of the forty-first annual ACM symposium on Theory of computing (May 2009), 169-178.

[9] Shlomi Dolev and Arseni Kalma, Verifiable Computing Using Computation Fingerprints Within FHE. NCA 2021: 1-9.

[10] George Robert Blakey, Safeguarding cryptographic keys, Federal Information Processing Standard Conference Proceedings, 48 (1979), 313–317.

[11] Michael Ben-Or, Shafi Goldwasser, Avi Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, Proceedings of the twentieth annual ACM symposium on Theory of computing (January 1988), 1-10.

[12] Oded Goldreich, Dana Ron, Madhu Sudan, Chinese Remaindering with Errors, IEEE Transactions on Information Theory (Volume: 46, Issue: 4, July 2000), 1330 - 1338.

[13] Asaf Cohen, Shlomi Dolev, Nir Tzachar, Efficient and Universal Corruption Resilient Fountain Codes, IEEE Transactions on Communications (Volume: 61, Issue: 10, October 2013), 4058 - 4066.

[14] Shlomi Dolev, Stav Doolman, Blindly Follow: SITS CRT and FHE for DCLSMPC of DUFSM (Extended Abstract), CSCML 2021, 487-496

Appendix A
Auxiliary Claims Proofs

Corollary 1. Given the multiplicative group $A = U_{m_1 \cdot m_2 \cdot \cdots \cdot m_n} = U_{M_n}$ there is an isomorphism to the direct product of M_n pairwise co-primes dividers, meaning $B = U_{m_1} \times U_{m_2} \times ... \times U_{m_n}$.

Proof. Let us define $f : A \to B$ in the following way:

$$f(\alpha) = \langle \alpha(\text{mod } m_1), \alpha(\text{mod } m_2), ..., \alpha(\text{mod } m_n) \rangle. \quad (3)$$

We need to show that $f(\alpha_1 \cdot \alpha_2) = f(\alpha_1) \cdot f(\alpha_2)$.

$$f(\alpha_1 \cdot \alpha_2) = \langle (\alpha_1 \cdot \alpha_2)(\text{mod } m_1), (\alpha_1 \cdot \alpha_2)(\text{mod } m_2), ..., (\alpha_1 \cdot \alpha_2)(\text{mod } m_n) \rangle. \quad (4)$$

$$\begin{aligned} f(\alpha_1) \cdot f(\alpha_2) = \langle \alpha_1(\text{mod } m_1), \alpha_1(\text{mod } m_2), ..., \alpha_1(\text{mod } m_n) \rangle \cdot \\ \langle \alpha_2(\text{mod } m_1), \alpha_2(\text{mod } m_2), ..., \alpha_2(\text{mod } m_n) \rangle = \\ \langle (\alpha_1 \cdot \alpha_2)(\text{mod } m_1), (\alpha_1 \cdot \alpha_2)(\text{mod } m_2), ..., (\alpha_1 \cdot \alpha_2)(\text{mod } m_n) \rangle. \end{aligned} \quad (5)$$

The last equality in "(5)" is achieved by the definition of multiplication in a direct product. □

Corollary 2 Define $M_n = m_1 \cdot m_2 \cdot \cdots \cdot m_n$ where $m_1, m_2, \ldots, m_n$ are pairwise co-primes. Given element $\alpha \in A = U_{M_n}$ of a finite group:
$\forall \gamma \in U_{M_n}, \exists \beta \in U_{m_n}$ s.t. $\alpha \cdot \beta = \gamma$. Meaning that all elements can be the product of α and another element in the group.

Proof. Given $\alpha \in U_{M_n}$, let's assume in contradiction that there is an element $\gamma_1 \in U_{M_n}$ such that: $\alpha \cdot \beta \neq \gamma_1, \forall \beta \in U_{m_n}$. We know that U_{M_n} is a multiplicative group, so every multiplication of elements in the group forms an element in the group. Since the source and the domain of the multiplication are of the same finite size, there is at least one element $\gamma_2 \in U_{M_n}$ such that:

$$\alpha \cdot \beta_1 = \alpha \cdot \beta_2 = \gamma_2, \beta_1 \neq \beta_2. \quad (6)$$

The element α has an inverse α^{-1} because U_{M_n} is a group. Applying α^{-1} on "(6)" we get:

$$\alpha^{-1} \cdot \alpha \cdot \beta_1 = \alpha^{-1} \cdot \alpha \cdot \beta_2 = \alpha^{-1} \cdot \gamma \Rightarrow \beta_1 = \beta_2 = \alpha^{-1} \cdot \gamma. \quad (7)$$

in contrast to $\beta_1 \neq \beta_2$ which means that our assumption was incorrect:

$$\begin{aligned} &\nexists \gamma_1 \in U_{M_n}, \forall \beta \in U_{m_n} \\ &\text{s.t. } \alpha \cdot \beta \neq \gamma_1 \Rightarrow \forall \gamma \in U_{M_n}, \exists \beta \in U_{m_n} \\ &\text{s.t. } \alpha \cdot \beta = \gamma. \end{aligned}$$

□

Corollary 3 Let G be a finite group. Given two elements, $g_1, g_2 \in G$ chosen randomly, uniformly and independently. The multiplication, $g_1 \cdot g_2 = g\prime \in G$ is a randomly, uniformly element of G.

Proof. We need to show that each $g\prime$ can be chosen with the same probability, which means $\frac{1}{|G|}$.

$$Pr[g_1 \cdot g_2 = g'] = Pr[\bigcup_{g_0 \in G} \{g_1 = g_0, g_2 = g_0^{-1} \cdot g'\}] =^1$$

$$\sum_{g_0 \in G} Pr[g_1 = g_0, g_2 = g_0^{-1} \cdot g'] =^2$$

$$\sum_{g_0 \in G} Pr[g_1 = g_0] Pr[g_2 = g_0^{-1} g\prime] =^3$$

$$\sum_{g_0 \in G} \frac{1}{|G|} \cdot \frac{1}{|G|} = |G| \cdot \frac{1}{|G|^2} = \frac{1}{|G|}.$$

The 1 equality is achieved by the fact that each event of $g_0 \in G$ is different.
The 2 equality is achieved because of the independence of g_1 and g_2.
The 3 equality is achieved because g_1 and g_2 are chosen randomly uniformly. □

Corollary 4 Let G be a finite group. Given two elements $g_1, g_2 \in G$, where g_1 is taken from a uniform distribution and g_2 is from some distribution $\mathcal{D}$. g_1 and g_2 are chosen independently. The multiplication $g_1 \cdot g_2 = g\prime \in G$ is a randomly uniform element of G.

Proof. Same proof as Corollary(3), until equality 3.

$$Pr[g_1 \cdot g_2 = g'] = \cdots = \sum_{g_0 \in G} Pr[g_1 = g_0] Pr[g_2 = g_0^{-1} g\prime] =^3 \frac{1}{|G|}.$$

The 3 equality is achieved as g_0 goes over all elements in G. The same holds with g_0^{-1} as each element has a different inverse. So finally, when going over all elements in G, we receive that each element g' has the same probability. □

APPENDIX B
MULTIPLICATIVE HOMOMORPHISM EXPANDED PROOF

Let n be the number of participants, k the number of secrets and s the secrecy bound.

- Let $S_1, S_2, \ldots, S_k \in U_{M_n}$ be secrets that the user wants to know later about their product.
- Apply the distribution phase of the scheme on each secret. Note that $S_{i_j}, 1 \leq i \leq k, 1 \leq j \leq n$ is the part of the $S_i, 1 \leq i \leq k$ secret held by the j'th participant. Calculating:

$$S_{1_1} = (S_{1_{\text{mix}}} (\text{mod } m_1), r_{1_1} (\text{mod } m_2), \ldots, r_{1_s} (\text{mod } m_{s+1}))$$
$$\ldots$$
$$S_{k_1} = (S_{k_{\text{mix}}} (\text{mod } m_1), r_{k_1} (\text{mod } m_2), \ldots, r_{k_s} (\text{mod } m_{s+1}))$$
$$S_{1_2} = (S_{1_{\text{mix}}} (\text{mod } m_2), r_{1_1} (\text{mod } m_3), \ldots, r_{1_s} (\text{mod } m_{s+2}))$$
$$\ldots$$
$$S_{k_2} = (S_{k_{\text{mix}}} (\text{mod } m_2), r_{k_1} (\text{mod } m_3), \ldots, r_{k_s} (\text{mod } m_{s+2}))$$
$$\ldots$$
$$S_{1_n} = (S_{1_{\text{mix}}} (\text{mod } m_n), r_{1_1} (\text{mod } m_1), \ldots, r_{1_s} (\text{mod } m_s))$$
$$\ldots$$
$$S_{k_n} = (S_{k_{\text{mix}}} (\text{mod } m_n), r_{k_1} (\text{mod } m_1), \ldots, r_{k_s} (\text{mod } m_s))$$

- Multiply all shares of each participant - this operation can be done by the participants themselves as needed:

$$S_{\text{res}_1} = (\prod_{1 \leq i \leq k} S_{i_{\text{mix}}} (\text{mod } m_1), \ldots, \prod_{1 \leq i \leq k} r_{i_s} (\text{mod } m_{s+1}))$$
$$S_{\text{res}_2} = (\prod_{1 \leq i \leq k} S_{i_{\text{mix}}} (\text{mod } m_2), \ldots, \prod_{1 \leq i \leq k} r_{i_s} (\text{mod } m_{s+2}))$$
$$\ldots$$
$$S_{\text{res}_n} = (\prod_{1 \leq i \leq k} S_{i_{\text{mix}}} (\text{mod } m_n), \ldots, \prod_{1 \leq i \leq k} r_{i_s} (\text{mod } m_s))$$

- Reconstructing all the results of products shares in order to find the result of the product of all secrets $S_{\text{res}} = \prod_{1 \leq i \leq k} S_i$:

$$S_{\text{res}_{\text{mix}}} = CRT[S_{\text{res}_{1_0}} (\text{mod } m_1), \ldots, S_{\text{res}_{n_0}} (\text{mod } m_n)]_{M_n}$$
$$r_{\text{res}_1} = CRT[r_{\text{res}_{1_1}} (\text{mod } m_2), \ldots, r_{\text{res}_{n_1}} (\text{mod } m_1)]_{M_n}$$
$$\ldots$$
$$r_{\text{res}_s} = CRT[r_{\text{res}_{1_s}} (\text{mod } m_{s+1}), \ldots, r_{\text{res}_{n_s}} (\text{mod } m_s)]_{M_n}$$

Each CRT equation has all the pairwise co-prime modulus $m_i, 1 \leq i \leq n$ results, meaning that the numbers are reconstructed perfectly in modulo $\mathbb{Z}_{M_n}$. Also, the calculations are correct from Corollary 1.

APPENDIX C
SECURITY ANALYSIS PROOF

Theorem 3 The multiplicative scheme is a Perfect ramp secret sharing scheme and is a perfect secret sharing scheme in case $s = n - 1$.

Proof. Let G be a group of curious participants gathered to reconstruct the secret or leak some information about it, $|G| \leq s$.

Given $S_{\text{mix}}, r_1, \ldots, r_s \in U_{M_n}$, the domain of $S_{\text{mix}}, r_1, \ldots, r_s$ is U_{M_n} and the size of the domain is $|U_{M_n}| = \varphi(M_n) = \prod_{1 \leq i \leq n} (m_i - 1)$ (The Euler function in case M_n is the product of primes from degree 1).
For each element $r'_1, \ldots, r'_s, S'_{\text{mix}}, S' \in U_{M_n}$ the probabilities of $r_1, \ldots, r_s, S_{\text{mix}}$ and S to be equal accordingly are:
$Pr[r_1 = r'_1] = \frac{1}{\varphi(M_n)}$ as r_1 is randomly uniformly chosen.
$\ldots$
$Pr[r_s = r'_s] = \frac{1}{\varphi(M_n)}$ as r_s is randomly uniformly chosen.
$Pr[S = S'] = p$ as S is chosen from some distribution $\mathcal{D}$.
$Pr[S_{\text{mix}} = S'_{\text{mix}}] = \frac{1}{\varphi(M_n)}$ as $r_1, \ldots, r_s$ are randomly uniformly chosen, and the secret S is of some distribution, based on Corollary 3 with induction that can be applied on it and Corollary 4.

The probabilities of $r_1, \ldots, r_s$ and S_{mix} to be equal to $r'_1, \ldots, r'_s$ and S'_{mix} respectively, do change knowing the information held by the group $G = \{i_1, \ldots, i_s\}, 1 \leq i_1 < \cdots < i_s \leq n$, as can be seen in "Fig.4".

However, we claim that the conditional probability of the secret S to be equal to S' does not change:

$$\begin{aligned}
&Pr[S = S' | r_1(\text{mod } m_{i_1+1}) = r'_1(\text{mod } m_{i_1+1}),\\
&\quad \dots,\\
&\quad r_1(\text{mod } m_{i_s+1}) = r'_1(\text{mod } m_{i_s+1}),\\
&\quad \dots,\\
&\quad r_s(\text{mod } m_{i_1+s}) = r'_s(\text{mod } m_{i_1+s}),\\
&\quad \dots,\\
&\quad r_s(\text{mod } m_{i_s+s}) = r'_s(\text{mod } m_{i_s+s}),\\
&\quad S_{\text{mix}}(\text{mod } m_{i_1}) = S'_{\text{mix}}(\text{mod } m_{i_1}),\\
&\quad \dots,\\
&\quad S_{\text{mix}}(\text{mod } m_{i_s}) = S'_{\text{mix}}(\text{mod } m_{i_s})] = p
\end{aligned}$$

In fact, we know that $s < n$ and therefore, we know that each modulo of $\varphi(m_{i_j}), \forall 1 \leq j \leq s$ does not appear n times in different equations. Hence, the total amount of valid options for S to be calculated by the group G is:

$$\prod_{1 \leq j \leq s} \frac{\varphi(M_n)}{\prod_{i \in G} \varphi(m_{i+j})} \cdot \frac{\varphi(M_n)}{\prod_{i \in G} \varphi(m_i)} =$$

$$\frac{\varphi(M_n)^{s+1}}{(\prod_{1 \leq j \leq s} \prod_{i \in G} \varphi(m_{i+j})) \cdot (\prod_{i \in G} \varphi(m_i))}$$

$$\leq {}^{4} \frac{\varphi(M_n)^{s+1}}{\varphi(M_n)^s}$$

As in the intuition part V-D, based on Corollary 1, it is possible to show that the distribution of S does not change, even when knowing the information of the group G, and each element in $S' \in U_{M_n}$ stays with the same probability that S is equal to it. Thus, the scheme is a perfect ramp security for any $s < n$.

The choice of $s = n-1$ yields a perfect secret sharing scheme according to Definition 2 as $\forall G \notin \mathcal{A}$, there is no information leak. □

[4]Since $s < n$, than $\varphi(m_i), \forall 1 \leq i \leq n$ appear at most s times.

$$Pr[r_1 = r'_1 | r_1(\text{mod } m_{i_1+1}) = r'_1(\text{mod } m_{i_1+1}), \dots, r_1(\text{mod } m_{i_s+1}) = r'_1(\text{mod } m_{i_s+1})] =$$

$$\frac{1}{\frac{\varphi(M_n)}{\prod_{i \in G} \varphi(m_{i+1})}} = \frac{\prod_{i \in G} \varphi(m_{i+1})}{\varphi(M_n)}$$

$$\dots$$

$$Pr[r_s = r'_s | r_s(\text{mod } m_{i_1+s}) = r'_s(\text{mod } m_{i_1+s}), \dots, r_s(\text{mod } m_{i_s+s}) = r'_s(\text{mod } m_{i_s+s})] =$$

$$\frac{1}{\frac{\varphi(M_n)}{\prod_{i \in G} \varphi(m_{i+s})}} = \frac{\prod_{i \in G} \varphi(m_{i+s})}{\varphi(M_n)}$$

$$Pr[S_{\text{mix}} = S'_{\text{mix}} | S_{\text{mix}}(\text{mod } m_{i_1}) = S'_{\text{mix}}(\text{mod } m_{i_1}), \dots, S_{\text{mix}}(\text{mod } m_{i_s}) = S'_{\text{mix}}(\text{mod } m_{i_s})] =$$

$$\frac{1}{\frac{\varphi(M_n)}{\prod_{i \in G} \varphi(m_i)}} = \frac{\prod_{i \in G} \varphi(m_i)}{\varphi(M_n)}$$

Fig. 4. Conditional probabilities of secret's shared parts with knowledge of an unauthorized group of participants