Evaluation of Arithmetic Sum-of-Products Expressions in Linear Secret Sharing Schemes with a Non-Interactive Computation Phase

Miguel de Vega[1], Andrei Lapets[2], Stanislaw Jarecki[3], Wicher Malten[1], Mehmet Ugurbil[1], and Wyatt Howe[4]

[1]Nillion
[2]Reity
[3]University of California, Irvine
[4]University of California, Los Angeles

November 9, 2023

Abstract

Among secure multi-party computation protocols, linear secret sharing schemes often do not rely on cryptographic assumptions and are among the most straightforward to explain and to implement correctly in software. However, basic versions of such schemes either limit participants to evaluating linear operations involving private values or require those participants to communicate synchronously during a computation phase. A straightforward, information-theoretically secure extension to such schemes is presented that can evaluate arithmetic sum-of-products expressions that contain multiplication operations involving non-zero private values. Notably, this extension does not require that participants communicate during the computation phase. Instead, a preprocessing phase is required that is independent of the private input values (but is dependent on the number of factors and terms in the sum-of-products expression).

1 Introduction

In their most basic form, linear secret-sharing schemes (LSSS) [27, 12] allow groups of parties to execute secure multi-party computation (MPC) workflows that involve linear operations on private inputs. The simplicity of such schemes can be highly advantageous in real-world applications [25] because (1) they can be information-theoretically secure (ITS) while still being relatively straightforward to explain to non-experts, (2) implementing them in software does not require skills beyond those of many professional software engineers, and (3) they can support asynchronous participation by parties if only linear operations are applied to inputs. In fact, the latter trade-off has a degree of freedom: *some* non-linear operations involving private values are possible while maintaining a *non-interactive* computation phase that is compatible with asynchronous workflows.

We introduce a novel ITS LSSS-based MPC protocol optimized specifically for non-linear operations involving non-zero values (*i.e.*, multiplicative group elements). This protocol supports the evaluation of expressions that are an arithmetic sum of products via a computation phase that requires no communication. While additive secret sharing is sufficient to demonstrate the protocol's features [23], in this report we present a protocol variant based on Shamir's secret sharing scheme [27] and assume $n = 2t + 1$ semi-honest parties with up to t passive adversaries.

1.1 Masked Factors

We consider an expression to be an arithmetic *sum of products* if it consists of a summation of $A \in \mathbb{Z}^+$ addends or *terms*, wherein each term (referenced by its index $a \in \{1, \ldots, A\}$) consists of $M_a \in \mathbb{Z}^+$ multiplicands or *factors*:

$$\sum_{a=1}^{A} \prod_{m=1}^{M_a} x_{a,m}.$$

We define a *masked factor* $\langle x \rangle_\lambda$ hiding a non-zero secret $x \in \mathbb{Z}_p^*$ with independent uniformly random mask exponent $\lambda \in \mathbb{Z}_{p-1}$ as

$$\langle x \rangle_\lambda = x \cdot g^{-\lambda} \in \mathbb{Z}_p^*,$$

where g is a public generator in $\mathbb{Z}_p^*$. We refer to the element $g^{-\lambda}$ as the multiplicative mask. To simplify notation, we often use $\langle x \rangle$ when λ is implicit.

A masked factor $\langle x \rangle_\lambda$ may be seen as a ciphertext within an encryption scheme with symmetric key λ [4]. This encryption scheme has two main properties. First, it is *multiplicatively homomorphic*. That is, $\langle x_1 \rangle_{\lambda_1} \cdot \langle x_2 \rangle_{\lambda_2} = \langle x_1 \cdot x_2 \rangle_{\lambda_1+\lambda_2}$. Second, it is *perfectly secure* (*i.e.*, ITS). That is, if X, K denote random variables representing the secret and the masked factor, respectively, then $\Pr[X = x | K = \langle x \rangle] = \Pr[X = x]$. In other words, knowledge of the masked factor $\langle x \rangle$ does not change the probability that the secret is x. This can be seen intuitively by considering that for every sample $\langle x \rangle_\lambda \in \mathbb{Z}_p^*$ of K, there is a unique element $g^{-\lambda} \in \mathbb{Z}_p^*$ (and consequently a unique $\lambda \in \mathbb{Z}_{p-1}$) such that $\langle x \rangle_\lambda \equiv x \cdot g^{-\lambda} \pmod{p}$. Thus, no information is leaked about the secret.

1.2 Linear Secret-Sharing Schemes

Since every masked factor is perfectly secure, the key remaining question is how to protect its mask exponent λ. Since the only operations performed on mask exponents are linear (see Equation 1 in Section 2.2), any linear scheme can be used to hide a mask exponent λ. This includes additively homomorphic cryptosystems [10, 24, 8] or somewhat homomorphic encryption (SHE) and fully homomorphic encryption (FHE) schemes such as BGV [7]. In this paper, we focus on linear secret-sharing schemes in order to minimize the number of cryptographic assumptions required, to maintain the simplicity of the protocol's building blocks (enabling easier communication with non-experts and more straightforward implementations), and to prioritize performance. While additive secret sharing would meet these goals most readily [23], we focus on Shamir's secret sharing (SSS) scheme to set the groundwork for extensions in setups with more general (t, n) access structures.

SSS works on finite fields, which poses a problem because the element λ that must be hidden lies in $\mathbb{Z}_{p-1}$. To tackle this, we rely on a prime field $\mathbb{F}_p$ with additional structure. Namely, p is a safe prime so that $p = 2q + 1$, where q is also prime. This way, $p - 1 = 2q$ is the product of two prime numbers and by the Chinese remainder theorem (CRT) there is a ring isomorphism $\mathbb{Z}_{p-1} \cong \mathbb{Z}_2 \times \mathbb{Z}_q \cong \mathbb{F}_2 \times \mathbb{F}_q$. The problem with $\mathbb{F}_2$ is that it does not allow applying SSS to $n > 1$ parties. To address this issue, we work with the extension field $\mathbb{F}_{2^k}$ for $2^k > n$. By convention, we store the bit in $\mathbb{F}_2$ in the independent polynomial coefficient in $\mathbb{F}_2[X]$. The rest of the coefficients are not used and represent $k - 1$ overhead bits. We denote a Shamir sharing of an element $a \in \mathbb{F}_r$ as $[a]^r = \{[a]_1, ..., [a]_n\}$, where $[a]_i$ is party P_i's share (for $i \in \{1, \ldots, n\}$). In this report, we always drop the superscript r when $r = p$ is the safe prime and denote by $[a]$ a Shamir sharing in $\mathbb{F}_p$. We define a sharing $[\![\lambda]\!]$ of an element $\lambda \in \mathbb{Z}_{p-1}$ as a 2-tuple $[\![\lambda]\!] = ([\lambda]^{2^k}, [\lambda]^q) \in \mathbb{F}_{2^k} \times \mathbb{F}_q$ of Shamir sharings. We denote by $[\![\lambda]\!]_i$ party P_i's share. Since $\lceil \log_2(p-1) \rceil = \lceil \log_2(q) \rceil + 1$ and $k = \lceil \log_2(n) \rceil$, this share requires $\lceil \log_2(p-1) \rceil - 1 + \lceil \log_2(n) \rceil$ bits for a $\lceil \log_2(p-1) \rceil$-bit element λ, incurring a small overhead of $k - 1 = \lceil \log_2(n) \rceil - 1$ bits. For example, for $n = 10$ and $n = 100$ nodes, this overhead represents 3 and 6 bits, respectively. We refer to $\mathcal{F}_{\mathsf{REVEAL}}$ as the ideal functionality that reconstructs a secret from its shares using polynomial interpolation. The reconstruction via $\mathcal{F}_{\mathsf{REVEAL}}$ of $\lambda \in \mathbb{Z}_{p-1}$ from a sharing $[\![\lambda]\!] = ([\lambda]^{2^k}, [\lambda]^q)$ implies two $\mathcal{F}_{\mathsf{REVEAL}}$ operations (in $\mathbb{F}_{2^k}$ and $\mathbb{F}_q$) and one CRT invocation.

2 Protocol

The overall protocol consists of two distinct phases: an interactive preprocessing phase that is independent of the values of the inputs (but dependent on the number of terms and the number of factors in each term within the arithmetic sum-of-products expression) and a *non-interactive* computation phase.

2.1 Preprocessing Phase

Presented in Figure 1 is the ideal preprocessing functionality $\mathcal{F}_{\mathsf{PREPROC}}$ for a sum of products $z = \sum_{a=1}^{A} \prod_{m=1}^{M_a} x_{a,m}$. $\mathcal{F}_{\mathsf{PREPROC}}$ generates a sharing $[\![\lambda_{a,m}]\!]$ of a random element $\lambda_{a,m} \in \mathbb{Z}_{p-1}$ for each input at position (a, m) where $a \in \{1, ..., A\}$ and $m \in \{1, ..., M_a\}$. It also generates sharings $[\![\gamma]\!]$ and $[g^{\gamma}]$ for each addend term such that γ_a is subject to the constraint

$$\gamma_a = \sum_{m=1}^{M_a} \lambda_{a,m}.$$

These sharings can be computed using standard MPC protocols [12]. Typically, computing a sharing of a random element requires adding n secret-shared elements, whereas computing $[g^{\gamma}]$ requires a protocol π_{EXP} that multiplies n secret-shared elements [13].

Preprocessing Functionality, $\mathcal{F}_{\mathsf{PREPROC}}$.

$$\{[g^{\gamma_a}]\}, \{[\![\lambda_{a,m}]\!]\} \leftarrow \mathcal{F}_{\mathsf{PREPROC}}(\mathcal{C})$$

1. For every factor input at position (a, m) (where $a \in \{1, ..., A\}$ and $m \in \{1, ..., M_a\}$) in the arithmetic circuit $\mathcal{C}$ implementing the sum of products, the parties compute a sharing $[\![\lambda_{a,m}]\!]$ of a random element $\lambda_{a,m} \in \mathbb{Z}_{p-1}$.
2. For every addend term (corresponding to index $a \in \{1, ..., A\}$), the parties compute a sharing $[g^{\gamma_a}]$ for an element $\gamma_a \in \mathbb{Z}_{p-1}$ such that

$$[\![\gamma_a]\!] = \sum_{m=1}^{M_a} [\![\lambda_{a,m}]\!]. \tag{1}$$

3. Output $\{[g^{\gamma_a}]\}, \{[\![\lambda_{a,m}]\!]\}$.

Figure 1: Ideal functionality for the preprocessing phase.

2.2 Computation Phase

We break down the expression for a sum of products $z \in \mathbb{F}_p$ so that it is written

$$z = \sum_{a=1}^{A} y_a,$$

where for $a \in \{1, ..., A\}$ and $\{x_{a,m}\}, \{y_a\} \in \mathbb{Z}_p^*$,

$$y_a = \prod_{m=1}^{M_a} x_{a,m}.$$

The protocol presented in Figure 2 computes z after the parties broadcast their inputs $\{x_{a,m}\}$ as masked factors $\{\langle x_{a,m}\rangle_{\lambda_{a,m}}\}$. Notice that the computation phase is non-interactive.

Computation Protocol, π.

$$z \leftarrow \pi(\mathcal{C})$$

where preprocessing $\mathcal{F}_{\mathsf{PREPROC}}$ has generated $\{[g^{\gamma_a}]\}, \{[\![\lambda_{a,m}]\!]\}$ for $a \in \{1, \ldots, A\}$ and $m \in \{1, ..., M_a\}$, subject to $\gamma_a = \sum_{m=1}^{M_a} \lambda_{a,m}$.

Input Stage

1. Every party receives a sharing $[\![\lambda_{a,m}]\!]$ of a mask exponent for every input $x_{a,m}$ they contribute to the computation.
2. They reconstruct $\lambda_{a,m}$, then compute and broadcast $\langle x_{a,m}\rangle_{\lambda_{a,m}} = x_{a,m} \cdot g^{-\lambda_{a,m}} \in \mathbb{Z}_p^*$.

Evaluation Stage

3. Parties locally compute the below.
   (a) For each product (*i.e.*, addend term) having index a where $a \in \{1, ..., A\}$,
   $$[y_a] = [g^{\gamma_a}] \cdot \prod_{m=1}^{M_a} \langle x_{a,m}\rangle_{\lambda_{a,m}}. \tag{2}$$
   (b) For the overall sum of products,
   $$[z] = \sum_{a=1}^{A} [y_a]. \tag{3}$$

Output Stage

4. The parties reveal $z \in \mathbb{F}_p$ from its sharing $[z]$.
5. Output z

Figure 2: Protocol for the computation phase with local evaluation stage.

2.3 Correctness and Security

The correctness of the computation phase follows from the fact that Equation 2 holds, as $[g^{\gamma_a}] \cdot \prod_{m=1}^{M_a} \langle x_{a,m} \rangle_{\lambda_{a,m}} = [g^{\gamma_a}] \cdot \prod_{m=1}^{M_a} (x_{a,m} \cdot g^{-\lambda_{a,m}}) = [g^{\gamma_a}] \cdot g^{-\sum_{m=1}^{M_a} \lambda_{a,m}} \cdot y_a = [g^{\gamma_a - \sum_{m=1}^{M_a} \lambda_{a,m}} \cdot y_a] = [y_a]$.

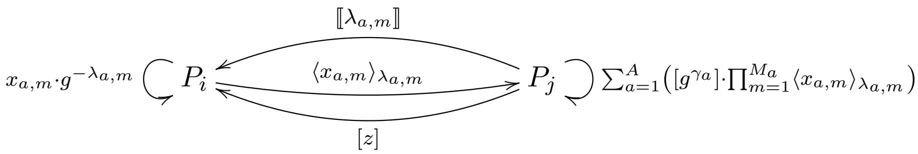

Figure 3: Computation phase interactions between distinct parties P_i and P_j for $i, j \in \{1, \ldots, n\}$.

For security, we argue that protocol π is a secure realization of an *add-of-mults* functionality. Consider an add-of-mults arithmetic circuit where a single addition gate accepts the results of A multiplication gates, where the gate corresponding to index $a \in \{1, \ldots, A\}$ has M_a inputs. If $\tilde{n} = \sum_{a=1}^{A} M_a$ then the add-of-mults functionality is a function $F : \ (\mathbb{Z}_p^*)^{\tilde{n}} \ \to (\mathbb{Z}_p)$ where $F(\{x_{a,m}\}) = \sum_{a=1}^{A} \prod_{m=1}^{M_a} x_{a,m}$. If preprocessing is implemented securely, protocol π is a secure computation of functionality F because a party that provides input $x_{a,m}$ learns only an independent random mask $\lambda_{a,m}$. When this party broadcasts $\langle x_{a,m} \rangle_{\lambda_{a,m}}$, nothing is revealed about $x_{a,m}$ to the other parties because $x_{a,m}$ is in $\mathbb{Z}_p^*$ and $g^{-\lambda_{a,m}}$ acts as a uniform one-time pad in the group $\mathbb{Z}_p^*$ (the one-time pad itself being secret with respect to the other parties by the properties of the secret-shared $[\![\lambda_{a,m}]\!]$). As π is non-interactive, it reveals nothing beyond these broadcast values except the sharing $[z]$ sent in Step 4 of the protocol. Because z is the correct output, it reveals nothing except for the intended output of the function F on the inputs $\{x_{a,m}\}$ corresponding to the above broadcasts. Moreover, $[z]$ leaks nothing because each share value in $[z]$ is distributed (uniformly) according to the choices of the random $\lambda_{a,m}$ values in the preprocessing phase.

3 Related Work

Information-theoretic MPC has a long history, dating back to the popular BGW protocol [3, 9, 26] of the late 1980s, but the round complexity of this MPC protocol and its modern variants [16, 11, 18] is linear in the multiplicative depth of the circuit. Work on constant-round MPC began decades ago [4, 14], and it is now known that any function can be computed in two rounds at a cost that is at least exponential in the circuit depth [19, 20, 15, 2, 1, 22]. Such protocols do not rely on familiar input representations or even on secret sharing techniques, as those require communication to perform multiplications [5]; furthermore, ITS can require an honest majority. Our approach is more similar to work [17, 6] leveraged to deploy a four-round secure Poisson regression protocol [21].

4 Conclusion

We have presented a technique that can be employed within the context of LSSS to enable the evaluation of an arithmetic sum of products via a non-interactive computation phase (at the expense of an input-independent preprocessing phase that can be implemented using standard MPC protocols). This technique maintains the information theoretic security of LSSS, is relatively straightforward to communicate and implement, and is compatible with asynchronous workflows.

References

[1] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. "Degree 2 is Complete for the Round-Complexity of Malicious MPC". In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Cham: Springer International Publishing, 2019, pp. 504–531. ISBN: 978-3-030-17656-3.

[2] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. *Perfect Secure Computation in Two Rounds*. Cryptology ePrint Archive, Paper 2018/894. https://eprint.iacr.org/2018/894. 2018.

[3] Gilad Asharov and Yehuda Lindell. "A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation". In: *J. Cryptol.* 30.1 (Jan. 2017), pp. 58–151. ISSN: 0933-2790.

[4] J. Bar-Ilan and D. Beaver. "Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction". In: *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*. PODC '89. Edmonton, Alberta, Canada: Association for Computing Machinery, 1989, pp. 201–209. ISBN: 0897913264.

[5] Omer Barkol, Yuval Ishai, and Enav Weinreb. "On D-Multiplicative Secret Sharing". In: *J. Cryptol.* 23.4 (Oct. 2010), pp. 580–593. ISSN: 0933-2790.

[6] Dor Bitan and Shlomi Dolev. *Optimal-Round Preprocessing-MPC via Polynomial Representation and Distributed Random Matrix*. Cryptology ePrint Archive, Paper 2019/1024. https://eprint.iacr.org/2019/1024. 2019.

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS '12. Cambridge, Massachusetts: Association for Computing Machinery, 2012, pp. 309–325. ISBN: 9781450311151.

[8] Guilhem Castagnos and Fabien Laguillaumie. *Linearly Homomorphic Encryption from DDH*. Cryptology ePrint Archive, Paper 2015/047. https://eprint.iacr.org/2015/047. 2015.

[9] David Chaum, Claude Crépeau, and Ivan Damgard. "Multiparty Unconditionally Secure Protocols". In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. STOC '88. Chicago, Illinois, USA: Association for Computing Machinery, 1988, pp. 11–19. ISBN: 0897912640.

[10] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. "A Generalization of Paillier's Public-Key System with Applications to Electronic Voting". In: *Int. J. Inf. Secur.* 9.6 (Dec. 2010), pp. 371–385. ISSN: 1615-5262.

[11] Ivan Damgård and Jesper Buus Nielsen. "Scalable and Unconditionally Secure Multiparty Computation". In: *Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO'07. Santa Barbara, CA, USA: Springer-Verlag, 2007, pp. 572–590. ISBN: 3540741429.

[12] Ivan Damgård et al. "Multiparty Computation from Somewhat Homomorphic Encryption". In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662. ISBN: 978-3-642-32009-5.

[13] Ivan Damgård et al. "Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation". In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 285–304. ISBN: 978-3-540-32732-5.

[14] Uri Feige, Joe Killian, and Moni Naor. "A Minimal Model for Secure Computation (Extended Abstract)". In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*. STOC '94. Montreal, Quebec, Canada: Association for Computing Machinery, 1994, pp. 554–563. ISBN: 0897916638.

[15] Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. *Two-Round MPC: Information-Theoretic and Black-Box*. Cryptology ePrint Archive, Paper 2018/909. https://eprint.iacr.org/2018/909. 2018.

[16] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. "Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography". In: *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*. PODC '98. Puerto Vallarta, Mexico: Association for Computing Machinery, 1998, pp. 101–111. ISBN: 0897919777.

[17] H. Ghodosi, Josef Pieprzyk, and Ron Steinfeld. "Multi-party computation with conversion of secret sharing". In: *Des. Codes Cryptography* 62 (Mar. 2012), pp. 259–272.

[18] Vipul Goyal et al. *ATLAS: Efficient and Scalable MPC in the Honest Majority Setting*. Cryptology ePrint Archive, Paper 2021/833. https://eprint.iacr.org/2021/833. 2021.

[19] Y. Ishai and E. Kushilevitz. "Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation". In: *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. FOCS '00. USA: IEEE Computer Society, 2000, p. 294. ISBN: 0769508502.

[20] Yuval Ishai and Eyal Kushilevitz. "Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials". In: *Automata, Languages and Programming*. Ed. by Peter Widmayer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 244–256. ISBN: 978-3-540-45465-6.

[21] Mahimna Kelkar et al. *Secure Poisson Regression*. Cryptology ePrint Archive, Paper 2021/208. https://eprint.iacr.org/2021/208. 2021.

[22] Andrei Lapets. *Implementing Arbitrary Maps over Small Finite Domains using Ring Addition and Scalar Multiplication*. Cryptology ePrint Archive, Paper 2023/1695. https://eprint.iacr.org/2023/1695. 2023.

[23] Nillion. "tinynmc". https://github.com/nillion-oss/tinynmc. 2023. Accessed: 2023-11-07.

[24] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *Advances in Cryptology — EUROCRYPT '99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238.

[25] Lucy Qin et al. "From Usability to Secure Computing and Back Again". In: *Proceedings of the Fifteenth USENIX Conference on Usable Privacy and Security*. SOUPS '19. Santa Clara, CA, USA: USENIX Association, Aug. 2019, pp. 191–210. ISBN: 9781939133052.

[26] T. Rabin and M. Ben-Or. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority". In: *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*. STOC '89. Seattle, Washington, USA: Association for Computing Machinery, 1989, pp. 73–85. ISBN: 0897913078.

[27] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782.