Implementing Arbitrary Maps over Small Finite Domains using Ring Addition and Scalar Multiplication

Andrei Lapets
Reity
Boston, MA
al@reity.org

Abstract

Many secure computation schemes and protocols (such as numerous variants of secure multi-party computation and homomorphic encryption) have favorable performance characteristics when they are used to evaluate addition and scalar multiplication operations on private values that can be represented as ring elements. A purely algebraic argument (with no references to any specific protocol or scheme) can be used to show that the ability to perform these operations is sufficient to implement *any* univariate map that operates on private values when that map's domain is finite. Such implementations of univariate maps can be composed in sequence any number of times. Other forms of composition for such implementations can be realized by using multiplication operations involving ring elements, but it is possible that these can be substituted with scalar multiplication operations within certain secure computation workflows.

1 Introduction

Secure computation schemes and protocols are typically well-suited for evaluating certain kinds of operations involving private values, particularly with respect to cost metrics such as running time and number of communication rounds. This report demonstrates that it is possible to present in a distilled manner – relying only on algebraic structures and their properties – a particular technique for implementing *any* univariate map that has a finite domain using only addition and scalar multiplication operations involving ring elements. Ways in which such implementations may be composed are also discussed. Such a succinct, abstract exposition of the techniques (as opposed to a presentation that is tied to particular applications [1, 12] or families of schemes or protocols [4, 5, 10, 15, 15]) is intended to aid in the broader dissemination of a *simple* (when isolated) algebraic construction, in its implementation and evaluation within a broader variety of existing secure computation frameworks and applications, and in the incorporation of the construction's insights in novel and/or hybrid protocols and applications.

2 Structures, Conventions, and Notation

The techniques presented in this report rely on vectors and matrices in which individual components are elements of rings. In the subsections below, it is assumed that $k, \ell \in \mathbb{Z}^+$.

2.1 Rings and Modules

For any ring R, denote by $\oplus : R \times R \to R$ its binary addition operation (for which $0 \in R$ is the identity) and by $\otimes : R \times R \to R$ its binary multiplication operation (for which $1 \in R$ is the identity).

For two finite rings S and C where $S \cong C$ via a bijection $\iota : S \to C$, denote by $\odot : S \times C \to C$ the scalar multiplication operation for the left S-module C. In such cases, S is also called the *ring of scalars* for the module C. Underlined symbols are used to distinguish the identity elements $\underline{0}, \underline{1} \in C$ from the identity elements $0, 1 \in S$.

Linear secret sharing schemes (LSSS) typically support addition of encrypted (*i.e.*, secret-shared) values and multiplication of encrypted values by an unencrypted (*i.e.*, not secret-shared) scalar without requiring any rounds of communication beyond those that are necessary to distribute inputs and reconstruct outputs [7, 14]. Some partially homomorphic encryption (PHE) schemes support addition of encrypted values and multiplication of encrypted values by unencrypted scalar values [2, 3, 13], and in some fully homomorphic encryption (FHE) schemes [6] these two kinds of operations are less costly than other operations (in particular, multiplication of encrypted values). For some of the schemes in these categories, an appropriately chosen pair of rings S and C can be used to represent scalars and encrypted values (*i.e.*, ciphertexts), respectively, with $\oplus : C \times C \to C$ representing addition of encrypted values and $\odot : S \times C \to C$ representing multiplication of encrypted values by unencrypted scalars.

Some secure multi-party computation (MPC) schemes [7], along with FHE schemes [6, 8], support multiplication of encrypted values. In such schemes, $\otimes : C \times C \to C$ would represent such a (sometimes more costly) multiplication operation.

2.2 Vector and Matrix Notation

A k-component vector $v \in R^k$ of elements from a ring R is represented using the notation $\langle v_1, \ldots, v_k \rangle$, where individual entries in the vector are represented using subscript notation. Vectors in R^k are used interchangeably with their corresponding single-column matrices $R^{k \times 1}$. A *one-hot* vector [9] is any vector $\langle v_1, \ldots, v_k \rangle \in R^k$ in which exactly one component is $1 \in R$ and all other components are $0 \in R$.

Let C be a left S-module. Multiplication of a vector $v \in C^k$ of module elements drawn from C by a matrix $M \in S^{\ell \times k}$ of scalars drawn from S is denoted $M \odot v$ and involves both the scalar multiplication operation $\odot : S \times C \to C$ and the ring addition operation $\oplus : C \times C \to C$:

$$\begin{bmatrix} M_{11} & \ldots & M_{1k} \\ \vdots & \ddots & \vdots \\ M_{\ell 1} & \ldots & M_{\ell k} \end{bmatrix} \odot \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} = \begin{bmatrix} (M_{11} \odot v_1) \oplus \ldots \oplus (M_{1k} \odot v_k) \\ \vdots \\ (M_{\ell 1} \odot v_1) \oplus \ldots \oplus (M_{\ell k} \odot v_k) \end{bmatrix}$$

Similarly, multiplication of v by a matrix $M' \in C^{\ell \times k}$ of module elements drawn from C is denoted $M' \otimes v$ and involves both the multiplication operation $\otimes : C \times C \to C$ and the ring addition operation $\oplus : C \times C \to C$:

$$\begin{bmatrix} M'_{11} & \ldots & M'_{1k} \\ \vdots & \ddots & \vdots \\ M'_{\ell 1} & \ldots & M'_{\ell k} \end{bmatrix} \otimes \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} = \begin{bmatrix} (M'_{11} \otimes v_1) \oplus \ldots \oplus (M'_{1k} \otimes v_k) \\ \vdots \\ (M'_{\ell 1} \otimes v_1) \oplus \ldots \oplus (M'_{\ell k} \otimes v_k) \end{bmatrix}$$

For a matrix $M \in C^{\ell \times \ell}$, $M^\top$ denotes its transpose and $\mathrm{diag}(M)$ denotes the vector $\langle M_{11}, \ldots, M_{\ell\ell} \rangle$.

3 Implementation of Sequentially Composable Univariate Maps

Let X and Y be finite sets and let $m : X \to Y$ be any map. Let two finite rings S and C be such that $|S| \geq 2$, $S \cong C$, and C is a left S-module. It is possible to implement m using only $\oplus : C \times C \to C$ and $\odot : S \times C \to C$.

3.1 Simple Example

Let $X = \{a, b, c\}$ be a set and let $m : X \to X$ be a map such that $m = \{a \mapsto b, b \mapsto b, c \mapsto a\}$. Define a bijection $\eta : X \to C^3$ that maps each element of X to a one-hot vector that represents it:

$$\eta = \Big\{a \mapsto \langle \underline{1}, \underline{0}, \underline{0} \rangle, b \mapsto \langle \underline{0}, \underline{1}, \underline{0} \rangle, c \mapsto \langle \underline{0}, \underline{0}, \underline{1} \rangle\Big\}.$$

For example, $\eta(m(c)) = \eta(a) = \langle \underline{1}, \underline{0}, \underline{0} \rangle$, where $\eta(a)_i$ for $i \in \{1, 2, 3\}$ denote the components of the vector $\eta(a)$. Given m and η, define a matrix $M \in S^{3\times 3}$ by leveraging $\iota^{-1} : C \to S$ in a componentwise manner:

$$M = \begin{bmatrix} \iota^{-1}(\eta(m(a))_1) & \iota^{-1}(\eta(m(b))_1) & \iota^{-1}(\eta(m(c))_1) \\ \iota^{-1}(\eta(m(a))_2) & \iota^{-1}(\eta(m(b))_2) & \iota^{-1}(\eta(m(c))_2) \\ \iota^{-1}(\eta(m(a))_3) & \iota^{-1}(\eta(m(b))_3) & \iota^{-1}(\eta(m(c))_3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

It is then the case that the transformation represented by M over C^3 corresponds to the transformation represented by m over X. For example, the calculation $m(c) = a$ is implemented below:

$$M \odot \eta(c) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \odot \begin{bmatrix} \underline{0} \\ \underline{0} \\ \underline{1} \end{bmatrix} = \begin{bmatrix} (0 \odot \underline{0}) \oplus (0 \odot \underline{0}) \oplus (1 \odot \underline{1}) \\ (1 \odot \underline{0}) \oplus (1 \odot \underline{0}) \oplus (0 \odot \underline{1}) \\ (0 \odot \underline{0}) \oplus (0 \odot \underline{0}) \oplus (0 \odot \underline{1}) \end{bmatrix} = \begin{bmatrix} \underline{1} \\ \underline{0} \\ \underline{0} \end{bmatrix} = \eta(a).$$

3.2 Implementation via Ring Addition and Scalar Multiplication

Let X be a finite set. Let x_i for any $i \in \{1, \ldots, |X|\}$ refer to the ith element in an ordered enumeration of the elements of X. Define a bijection $\eta_X : X \to C^{|X|}$ that maps each element of X to a one-hot vector representing that element such that for $i, j \in \{1, \ldots, |X|\}$, it is the case that

$$\eta_X(x_i)_j = \begin{cases} \underline{1} \text{ if } i = j, \\ \underline{0} \text{ otherwise.} \end{cases}$$

Similarly, let Y be a finite set with a corresponding bijection $\eta_Y : Y \to C^{|Y|}$. Define a matrix $M \in S^{|Y|\times|X|}$ such that for $i \in \{1, \ldots, |Y|\}$ and $j \in \{1, \ldots, |X|\}$, it is the case that

$$M_{ij} = \begin{cases} 1 \text{ if } m(x_j) = y_i, \\ 0 \text{ otherwise.} \end{cases}$$

Given the above definition, M can be interpreted intuitively as a collection of column vectors in which the jth column corresponds to the output $m(x_j) \in Y$ for the input $x_j \in X$:

$$\begin{bmatrix} \iota(M_{11}) & \ldots & \iota(M_{1|X|}) \\ \vdots & \ddots & \vdots \\ \iota(M_{|Y|1}) & \ldots & \iota(M_{|Y||X|}) \end{bmatrix} = \begin{bmatrix} | & & | \\ \eta_Y(m(x_1)) & \ldots & \eta_Y(m(x_{|X|})) \\ | & & | \end{bmatrix}.$$

Then it is the case that for all $x_i \in X$,

$$\eta_Y^{-1}(M \odot \eta_X(x_i)) = m(x_i).$$

Note that for any two maps $m : X \to Y$ and $m' : Y \to Z$ (where X, Y, and Z are finite sets), the implementations of m and m' can be composed sequentially into an implementation of $m' \circ m$ by using matrix multiplication.

4 Non-Sequential Composition via Multivariate Maps

For $n \in \mathbb{N}$ where $n \geq 2$, assume $X_1, \ldots, X_n$ and Y are all finite sets and let $m : X_1 \times \ldots \times X_n \to Y$ be any map. While the technique presented in Section 3 can be used to implement m by treating $X_1 \times \ldots \times X_n$ as a single finite set (albeit of elements that represent tuples), this is not sufficient for composing m with *other* map implementations that may be implemented in parallel and may output one-hot vector representations of elements from the individual sets X_i for $i \in \{1, \ldots, n\}$.

Given two finite rings S and C where $|S| \geq 2$, $S \cong C$, and C is a left S-module, it is possible to prepare an input for an implementation of m by leveraging the ring multiplication operation $\otimes : C \times C \to C$ to implement a bijection $\theta : C^{|X_1|} \times \ldots \times C^{|X_n|} \to C^{|X_1| \cdot \ldots \cdot |X_n|}$ that associates each distinct tuple of one-hot vectors in $C^{|X_1|} \times \ldots \times C^{|X_n|}$ with a corresponding one-hot vector in $C^{|X_1| \cdot \ldots \cdot |X_n|}$. If only one of the one-hot input vectors has components from C (with all other one-hot input vectors having components from S), scalar multiplication via $\odot : S \times C \to C$ suffices.

4.1 Simple Example

Let $X = \{a, b\}$ be a set and let $\eta : X \to C^2$ be a bijection $\{a \mapsto \langle \underline{1}, \underline{0} \rangle, b \mapsto \langle \underline{0}, \underline{1} \rangle\}$ that maps each element of X to a one-hot vector representing that element. Let $\theta : C^2 \times C^2 \to C^4$ be a bijection that associates one-hot vectors in C^4 with pairs of one-hot vectors in C^2:

$$\theta(v, w) = \begin{cases} \langle \underline{1}, \underline{0}, \underline{0}, \underline{0} \rangle \text{ if } v = \langle \underline{1}, \underline{0} \rangle \text{ and } w = \langle \underline{1}, \underline{0} \rangle, \\ \langle \underline{0}, \underline{1}, \underline{0}, \underline{0} \rangle \text{ if } v = \langle \underline{1}, \underline{0} \rangle \text{ and } w = \langle \underline{0}, \underline{1} \rangle, \\ \langle \underline{0}, \underline{0}, \underline{1}, \underline{0} \rangle \text{ if } v = \langle \underline{0}, \underline{1} \rangle \text{ and } w = \langle \underline{1}, \underline{0} \rangle, \\ \langle \underline{0}, \underline{0}, \underline{0}, \underline{1} \rangle \text{ if } v = \langle \underline{0}, \underline{1} \rangle \text{ and } w = \langle \underline{0}, \underline{1} \rangle. \end{cases}$$

Any pair of inputs (each drawn from X) can be converted into a pair of vectors (each one in C^2). For example, suppose the two inputs are $a \in X$ and $b \in X$ where $\eta(a) = \langle 1, 0 \rangle$ and $\eta(b) = \langle 0, 1 \rangle$. It is then possible to calculate $\theta(\eta(a), \eta(b))$ by first multiplying one vector by an appropriate *expansion matrix* in $S^{4 \times 2}$ and the other by a corresponding *repetition matrix* in $S^{4 \times 2}$ (see Section 4.2), and then combining[1] the results by using $\otimes : C \times C \to C$:

$$\operatorname{diag}\left(\left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \odot \begin{bmatrix} \underline{1} \\ \underline{0} \end{bmatrix}\right) \otimes \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \odot \begin{bmatrix} \underline{0} \\ \underline{1} \end{bmatrix}\right)^{\top}\right) = \operatorname{diag}\left(\begin{bmatrix} \underline{1} \\ \underline{1} \\ \underline{0} \\ \underline{0} \end{bmatrix} \otimes \begin{bmatrix} \underline{0} \\ \underline{1} \\ \underline{0} \\ \underline{1} \end{bmatrix}^{\top}\right) = \begin{bmatrix} \underline{0} \\ \underline{1} \\ \underline{0} \\ \underline{0} \end{bmatrix}.$$

If one input is represented by a vector in S^2 (*e.g.*, a by $\langle 1, 0 \rangle \in S^2$), $\odot : S \times C \to C$ alone suffices:

$$\operatorname{diag}\left(\left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) \odot \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \odot \begin{bmatrix} \underline{0} \\ \underline{1} \end{bmatrix}\right)^{\top}\right) = \operatorname{diag}\left(\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \odot \begin{bmatrix} \underline{0} \\ \underline{1} \\ \underline{0} \\ \underline{1} \end{bmatrix}^{\top}\right) = \begin{bmatrix} \underline{0} \\ \underline{1} \\ \underline{0} \\ \underline{0} \end{bmatrix}.$$

4.2 Expansion and Repetition Matrices

Assume X and X' are finite sets. Define the *expansion matrix* $\Gamma_{|X|,|X'|} \in S^{(|X| \cdot |X'|) \times |X|}$ that transforms any one-hot vector of length $|X|$ into a vector of length $|X| \cdot |X'|$ wherein every entry

[1] Transposition and the diag operator are used to implement two versions of the Hadamard product (in terms of $\otimes$ and $\odot$). In practice, the entries in the product of matrices that do not appear in the diagonal need not be computed.

in the original vector is repeated $|X'|$ times:

$$\left(\Gamma_{|X|,|X'|}\right)_{ij} = \begin{cases} 1 \text{ if } \lfloor (i-1)/|X'| \rfloor = j-1, \\ 0 \text{ otherwise.} \end{cases}$$

Define the *repetition matrix* $\Delta_{|X|,|X'|} \in S^{(|X|\cdot|X'|)\times|X|}$ that transforms any one-hot vector of length $|X|$ into a vector of length $|X| \cdot |X'|$ consisting of $|X'|$ concatenated copies of the original:

$$\left(\Delta_{|X|,|X'|}\right)_{ij} = \begin{cases} 1 \text{ if } ((i-1) \bmod |X'|) = j-1, \\ 0 \text{ otherwise.} \end{cases}$$

4.3 Transforming Pairs of Vectors using Ring Multiplication

Given two one-hot vectors $v \in C^{|X|}$ and $v' \in C^{|X'|}$, define a binary operation $\boxtimes : C^{|X|} \times C^{|X'|} \to C^{|X|\cdot|X'|}$ where

$$v \boxtimes v' = \text{diag}((\Gamma_{|X|,|X'|} \odot v) \otimes (\Delta_{|X|,|X'|} \odot v')^\top).$$

It is the case that $v \boxtimes v'$ must be a one-hot vector because (1) the vector on the left-hand side of the $\otimes$ operator in the expression above has exactly one subvector of $\underline{1}$ entries (of length $|X|$) and (2) the vector on the right-hand side of the $\otimes$ operator has $|X|-1$ entries that are $\underline{0}$ between every two $\underline{1}$ entries. Furthermore, for any $w \in C^{|X|}$ where $w \neq v$, $w \boxtimes v' \neq v \boxtimes v'$. Similarly, for any $w' \in C^{|X'|}$ where $w' \neq v'$, $v \boxtimes w' \neq v \boxtimes v'$. Thus, this operation acts as a bijection between the set of pairs of one-hot vectors in $C^{|X|} \times C^{|X'|}$ and the set of one-hot vectors in $C^{|X|\cdot|X'|}$.

4.4 Transforming Pairs of Vectors using Scalar Multiplication

Suppose $v \in S^{|X|}$ is a one-hot vector and $v' \in S^{|X'|}$ is also a one-hot vector. Define a binary operation $\boxdot : S^{|X|} \times S^{|X'|} \to S^{|X|\cdot|X'|}$ where

$$v \boxdot v' = \text{diag}((\Gamma_{|X|,|X'|} \odot v) \odot (\Delta_{|X|,|X'|} \odot v')^\top).$$

If instead it is the case that $v' \in C^{|X'|}$, the above equation defines a binary operation $\boxdot : S^{|X|} \times C^{|X'|} \to C^{|X|\cdot|X'|}$. Via the same argument presented for $\boxtimes$ in Section 4.3, it is the case that $v \boxdot v'$ is a one-hot vector and $\boxdot$ is a bijection between the sets of interest.

4.5 Transforming Collections of Vectors

Let $v_1, \ldots, v_n$ be a collection of one-hot vectors such that $v_i \in C^{|X_i|}$ for $i \in \{1, \ldots, n\}$. By induction, it is sufficient to compute $v_1 \boxtimes \ldots \boxtimes v_n$ to calculate a corresponding one-hot vector in $C^{|X_1|\cdot\ldots\cdot|X_n|}$. For the special case in which $v_1, \ldots, v_n$ is a collection of one-hot vectors such that $v_i \in S^{|X_i|}$ for $i \in \{1, \ldots, n-1\}$ and $v_n \in C^{|X_n|}$, the $\boxdot$ operation is sufficient to calculate the result: $v_1 \boxdot \ldots \boxdot v_n$.

5 Utilization in Secure Computation Schemes

The techniques presented in Section 3 and 4 rely on two algebraic structures: a finite ring S and an isomorphic finite ring C where C is a left S-module. While only the additive and multiplicative identities are used, S and C can be any ring of size greater than or equal to 2. The techniques

are thus compatible with a very broad selection of secure computation schemes, and this is demonstrated in a substantial body of past and ongoing work[2] on developing, improving, and applying variants of these techniques [1, 4, 5, 10, 12, 15].

On its own, the technique in Section 3 implies that all univariate maps with finite domains can be implemented (in a manner that composes) using PHE schemes such as the Pailler cryptosystem [13] and others [2, 3]. It also implies that such maps can be implemented using a variety of MPC schemes, from the most basic to the most sophisticated (including Shamir's secret sharing scheme [14], additive secret sharing schemes [7], any LSSS, and so on), without additional rounds of communication beyond those necessary for distributing shares and reconstructing results. Finally, it implies that in some cases non-linear maps can be computed on encrypted inputs using schemes and protocols that do not rely on cryptographic assumptions, and/or are straightforward to explain to non-experts and to implement in software applications [11].

The techniques presented in Section 4 offer some additional options for composing map implementations that rely one one-hot vectors. These may be of interest within at least three scenarios.

1. The technique in Sections 4.3 (and its generalization in Section 4.5) provides a way to implement multivariate maps at the expense of performing ring multiplications (which correspond to multiplications involving secret-shared or encrypted values). This technique can be used directly within MPC schemes that allow multiplication of secret-shared values [7] and within FHE schemes [6, 8].

2. The technique in Section 4.4 (and its generalization in Section 4.5) provides a template for combining any number of public or plaintext inputs with a single private input and then applying any multivariate map to that input to obtain a private output. This is mostly a convenience, as the public or plaintext inputs and the multivariate map together can be represented using an appropriately defined univariate map. However, the fact that the expressions are nearly identical between Section 4.3 and Section 4.4 (with the exception of the underlying operators used) can be useful in practice.

3. The technique in Section 4.4 provides a template for combining multiple *private* inputs using *scalar* multiplication when specific secure computation workflows are an option (whether using existing schemes and protocols, or using a combination thereof). For example, it may be possible to use a PHE scheme to pass an encrypted accumulator value from one input contributor to another, with each contributor using the technique in Section 4.4 to compute a new private accumulator value by applying a map to the previous private accumulator value and their own unencrypted (but never disclosed and consequently still private) input.

6 Conclusion

For small finite sets X and Y, it is possible to implement any univariate map $m : X \to Y$ by relying only on ring addition and scalar multiplication. Such implementations can be composed arbitrarily many times without modifying the representation of input and output values. Multiplication of ring elements can be leveraged to extend this technique to multivariate maps in general, but scalar multiplication can still be used in place of ring multiplication within certain workflows. These results can be derived independently of any specific secure computation scheme or protocol, as they rely only on algebraic objects and operations that are supported by many such schemes.

[2] Variants of this technique appear in many other works besides the handful that are explicitly cited in this report.

References

[1] A. Abidin, A. Aly, S. Cleemput, and M. A. Mustafa. An MPC-Based Privacy-Preserving Protocol for a Local Electricity Trading Market. In S. Foresti and G. Persiano, editors, *Cryptology and Network Security*, pages 615–625, Cham, 2016. Springer International Publishing.

[2] J. Benaloh. Dense Probabilistic Encryption. In *Selected Areas of Cryptography*, May 1994.

[3] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In J. Kilian, editor, *Theory of Cryptography*, pages 325–341, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[4] E. Boyle, N. Gilboa, and Y. Ishai. Function Secret Sharing. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 337–367, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[5] E. Boyle, N. Gilboa, Y. Ishai, and V. I. Kolobov. Information-theoretic distributed point functions. Cryptology ePrint Archive, Paper 2023/028, 2023. `https://eprint.iacr.org/2023/028`.

[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory*, 6(3), July 2014.

[7] I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, page 643–662, Berlin, Heidelberg, 2012. Springer-Verlag.

[8] C. Gentry, A. Sahai, and B. Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[9] D. Harris and S. Harris. *Digital Design and Computer Architecture, Second Edition*, pages 129–129. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2012.

[10] A. Lapets. Evaluation of Univariate Functions over Small Domains in Secure Multi-Party Computation Protocols, November 2019. Available at `https://github.com/lapets/article-univariate-functions-for-mpc`.

[11] A. Lapets, N. Volgushev, A. Bestavros, F. Jansen, and M. Varia. Secure MPC for Analytics as a Web Application. In *2016 IEEE Cybersecurity Development (SecDev)*, pages 73–74, November 2016.

[12] J. Launchbury, I. S. Diatchki, T. DuBuisson, and A. Adams-Moran. Efficient Lookup-Table Protocol in Secure Multiparty Computation. *SIGPLAN Not.*, 47(9):189–200, September 2012.

[13] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[14] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, November 1979.

[15] S. Wagh. Pika: Secure Computation using Function Secret Sharing over Rings. *Proceedings on Privacy Enhancing Technologies*, 2022:351–377, 10 2022.