Universiteit Leiden
The Netherlands

Opleiding Informatica + Wiskunde

Reverse Multiplication Friendly Embeddings over ring extensions and their application to Multiparty Computation.

Jasper van der Zwet

Supervisors:
Eleftheria Makri & Serge Fehr

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

30/06/2025

Abstract

Multi-party computation (MPC) is a field that studies interactive algorithms that allow a group of mutually distrusting parties, providing their own input, to jointly compute a function. Reverse Multiplication Friendly Embeddings (RMFEs) are a tool used in MPC to parallelize mainly binary multiplications. These RMFEs were originally introduced over finite fields. Two types of RMFE constructions were originally shown, a polynomial construction and a more complex construction using algebraic geometry. This thesis introduces some important concepts in MPC and shows how RMFEs can be constructed and (have been) applied to MPC. It is shown in this thesis how the polynomial construction over finite fields can be lifted to Galois rings. This has previously only been done for the more complex algebraic-geometric construction. This thesis ends with an evaluation of RMFEs and their use-cases related to MPC.

Contents

1 Introduction

Imagine there are two millionaires, Alice and Bob, that wish to know who is richer. However, they do not want to reveal their actual wealth to one another. This is an important example of a setting studied in the subfield of secure multi-party computation (MPC) introduced in 1982 by Andrew Yao [Yao82].

MPC is a subfield of cryptography that studies interactive algorithms that allow a group of mutually distrusting parties, providing their own input, to jointly compute a function. These algorithms should apply these inputs without revealing any information about the other participants' private inputs and while guaranteeing correct output. Specifically, we are looking at an arbitrary function $\mathcal{F} : X_1 \times X_2 \times ... \times X_n \to Y$ for non-empty and finite sets $X_1, ..., X_n$. We then consider the situation where n different parties $P_1, ..., P_n$ provide inputs $x_1 \in X_1, ..., x_n \in X_n$. The goal is to compute $\mathcal{F}(x_1, ..., x_n)$ in such a way that the individual inputs remain private. In Yao's example, the parties are the two millionaires. The function here is fairly simple, as it outputs a boolean depending on if $A \geq B$ for A, B the amount of wealth of Alice and Bob respectively. In other words, we have $\mathcal{F} : \mathbb{F}_{2^k} \times \mathbb{F}_{2^k} \to \{0, 1\}$ given by $\mathcal{F}(x, y) = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{if } x \leq y \end{cases}$. Note that many applications include more than two parties and more complex functions. Since it was introduced by Andrew Yao in the 1980s [Yao82], there have been a multitude of approaches to address this challenge. In the last decades, this has developed from a theoretical curiosity to be practically applicable.

MPC has many uses, such as secure auctions [KMS+16, CSTA19], voting [Hir01, NBSK15] and even privacy for training AI systems [Liu24, Zap22]. Although there are some challenging problems with the implementation of MPC in practice, this has been done successfully several times. For more information, the reader can look at paragraph 1.3.1 in [EKR22].

In MPC, functions are represented as a combination of additions and multiplications. Since most tools are linear, addition is often more efficient than multiplication. To see this, recall that linearity of a map $\phi : X \to Y$ guarantees $\forall x, y \in X : \phi(x + y) = \phi(x) + \phi(y)$, but often $\exists x, y \in X : \phi(xy) \neq \phi(x)\phi(y)$. This will be expanded upon in Section 2.2. This thesis will zoom in on a tool addressing efficiency of multiplications by parallelization. These so-called Reverse Multiplication Friendly Embeddings (RMFEs) are pairs of $\mathbb{F}_q$-linear maps (ϕ, ψ) from $\mathbb{F}_q^k \to \mathbb{F}_{q^m}$ and back, which allow for parallelization of multiplications by their property $\forall x, y \in \mathbb{F}_q^k : \psi(\phi(x)\phi(y)) = x * y$. Here, $*$ stands for coordinate-wise multiplication. This notation and the formal definition of RMFEs are introduced in Section 3.

This thesis will introduce a variety of models to the reader in which MPC is studied and zoom in on a couple of these models. This thesis will explain how, when considering these models, RMFEs are applied to increase efficiency [CCXY18, CG20, EXY22, HLW+24]. It will also be shown how RMFEs can be constructed over finite fields [CCXY18] in Section 5. In Section 6, this thesis contributes a lifting of the polynomial construction over finite fields [CCXY18] to Galois rings. This new construction of RMFEs over Galois rings is less complex than the existing construction in the literature by Cramer et al. [CRX21], which lifts the other construction by Cascudo et al. [CCXY18]. Finally, this thesis will evaluate the use of RMFEs in MPC and it will be discussed in what settings RMFEs can be used to increase efficiency and in what settings they cannot be or this has not been shown.

2 Multi-Party Computation

2.1 Defining MPC

An n-party MPC protocol for a function $\mathcal{F} : X_1 \times X_2 \times ... \times X_n \to Y$ over finite sets $X_1, ..., X_n, Y$ is an n-tuple of interactive, randomized algorithms $P_1, ..., P_n$, with certain properties that we discuss below (partially informally). It is assumed that the reader has a basic understanding of (randomized) algorithms, including interactive algorithms, which can exchange messages (besides doing local computations). From now on, we always denote the number of parties in an MPC protocol by n. Logically, one requirement is that for inputs $x_1 \in X_1, ..., x_n \in X_n$, each of these algorithms must output $\mathcal{F}(x_1, ..., x_n)$ (except perhaps with some small probability). An execution involves both 'local computation' of each party and private communication between pairs of parties. For this private communication, we assume that we have access to point-to-point secure communication channels between each pair of parties, possibly with the addition of a broadcast channel. From now on, for simplicity of explanation, we do not distinguish between an algorithm P_i and the party that is running it. We will consider dishonest parties that may run a different algorithm than P_i in an execution of the protocol. We still denote such a party by P_i.
Two security properties that should hold with respect to a threshold t, which represents the maximal number of dishonest parties participating in the MPC protocol, are privacy and correctness.
Privacy: In any execution, the joint view (the set of all messages received and sent by the parties, together with the inputs) of any t parties $\{P_{i_1}, ..., P_{i_t}\}$ should reveal no information on the inputs x_j of $P_j, j \notin \{i_1, ..., i_t\}$ beyond what can be deduced from the inputs $x_{i_1}, ..., x_{i_t}$ and $y = \mathcal{F}(x_1, ..., x_n)$. This should hold even when the t parties are dishonest and arbitrarily deviate from protocol specifications.
Correctness: Similarly, in any execution, all n parties should receive as output $y = \mathcal{F}(x'_1, ..., x'_n)$ with $x'_i = x_i$ for honest parties and x'_i arbitrarily chosen by dishonest parties P_i, even when t parties are dishonest and possibly deviate from protocol specifications. Correctness thus guarantees that replacing their input x_i by another value x'_i is the only action dishonest parties can take to influence the output.
We treat dishonest parties as if they are all corrupted by one entity, called the adversary. We consider some variants in the definition. We consider a passive (or honest-but-curious/semi-honest) adversary. This type of adversary will only passively gather as much information as they can while following the protocol. We may also consider an active (or malicious) adversary. Here, the dishonest parties may deviate arbitrarily from protocol specifications and actively interfere with the protocol. Both of these variants consider an adversary that can corrupt a certain number of parties at will. We thus consider the situation where the parties corrupted by an adversary work in tandem. This leads to the adversary threshold t that was noted previously. This threshold is usually denoted as a fraction of n. An important distinction is denoted by dishonest minority or honest majority ($< n/2$ corrupted parties) versus honest minority or dishonest majority ($\geq n/2$ corrupted parties). Dishonest majority requires more complex and thus computationally expensive protocols. Often, this also leads to an increase in communication complexity. One example which might help the reader see this, concerns validation of output. Assume honest majority. If at least half of the participants state a value x as the correct output, then, assuming they have correct information, this can be concluded to be the correct output. It is not possible to use this in dishonest majority, as up to $n - 1$ parties may be corrupted.

One may consider MPC protocols in the so-called pre-processing model. Such protocols consist of two phases, typically called the pre-processing and the online phase. The pre-processing phase is input-independent and sometimes function-independent, as long as an upper bound on the function complexity, appropriately measured, is available, and thus can be executed before the inputs are known to the parties. The online phase is then to be executed once the inputs are known. The goal is to have the online phase (significantly) more efficient than the pre-processing phase. Such MPC protocols in the pre-processing model are thus useful when the parties have idle time available before the inputs become available and if we want the secure computation to be as fast as possible once the inputs are available. All protocols discussed later in this thesis are protocols in this pre-processing model.
Privacy and correctness were already noted as two important security properties. For subtle reasons beyond the scope of this thesis, instead of security properties, a widely accepted security model called the real-ideal paradigm is used. Using this paradigm, an ideal world is introduced that captures all security constraints. Intuitively, for every dishonest party in the real world, there is a dishonest party in the ideal world, called the simulator, that 'achieves the same thing'. If so, then the protocol is secure. The first implementation of this, though with different notation and terminology, is considered to be a 1984 paper by Goldwasser and Micali [GM84].
In the ideal world, the protocol participants, denoted as parties $P_1, ..., P_n$, securely compute the function $\mathcal{F}$ by privately sending their inputs to a fully trusted third party, denoted as $\mathcal{T}$. Each party P_i sends their input x_i to $\mathcal{T}$, which then calculates $\mathcal{F}(x_1, ..., x_n)$ and sends the result back to all parties. $\mathcal{T}$ is fully trusted, which means that it cannot be corrupted by the adversary, while any other party can. In this model, any dishonest party only learns the output of $\mathcal{F}$, which is why this ideal scenario is considered secure. In the real world, since there is no such functionality, parties communicate using a protocol, usually denoted π.

2.2 Functions as Circuits

Any function the parties need to compute can be and is usually denoted as a circuit. An example of a circuit is shown in figure 1, which is explained in more detail later in this subsection. A circuit C is a connected directed graph with no cycles. It has some strictly positive number of nodes without incoming edges, denoting the inputs. All nodes are labeled input, multiplication, addition, or output. The multiplication and addition nodes are visualized as special symbols. We call these nodes gates (in some cases these are extended to inversions and/or (pseudo-)random gates). Unlabeled nodes are branching nodes. These nodes have only one incoming edge and multiple outgoing edges. Multiplication and addition gates can also have constants as (part of their) input. All nodes without outgoing edges are labeled output.

Notation 1 *Let C be a circuit. Let the integers $k, m > 0$ represent the number of input and output nodes. Then, for any commutative ring R, $\mathcal{F}_C(R) : R^k \to R^m$ is the function that C gives rise to by performing the operations in the circuit over R in the obvious way.*

Note that for computation of finite functions over a finite field or Galois ring (introduced in Section 6), a circuit of multiplication and addition gates is complete, as all circuits represent exactly all multivariate polynomials over the finite field or Galois ring, thus we can indeed model any finite function as such. A circuit is called a binary circuit if it is defined with computation and in- and

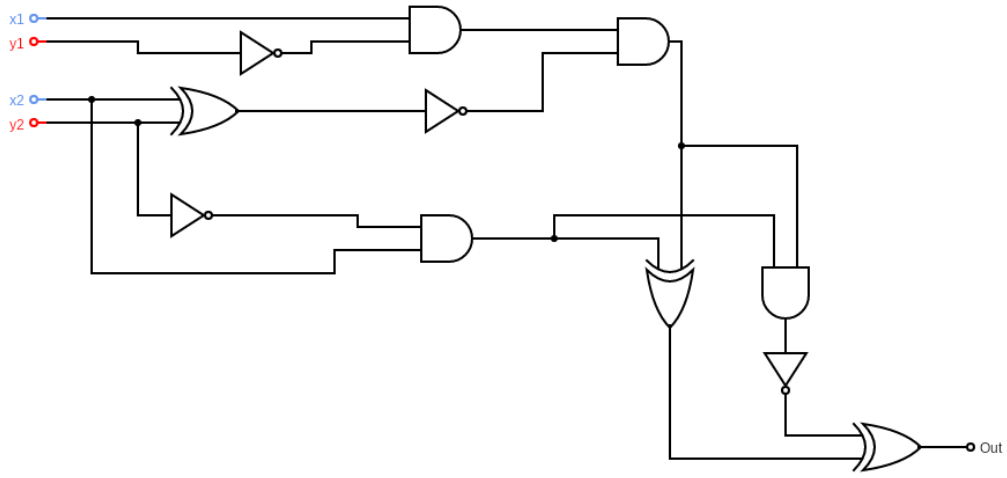

Figure 1: Circuit of Yao's millionaires' problem for 2-bit input. The round gates are AND gates, the gates with the double line at the start are XOR gates. NOT gates, the triangular gates, are added for simplicity, but can be simulated using an XOR gate and an external input 1.

output over $R = \mathbb{F}_2$ and arithmetic if it is defined over another finite field or ring. For binary circuits, multiplication gates are called AND gates and addition gates are called XOR gates.
Denoting the function of Yao's millionaires' problem can be done as follows. Assume that the monetary values of the millionaires are denoted in k bits. In this case, we could form the circuit over $R = \mathbb{F}_2$ by first denoting the two monetary values as x, y with separate bits $x_i, y_i \in \mathbb{F}_2, i = 1, ..., k$. Then calculate for every $i : e_i = x_i \oplus y_i$ ($\oplus$ is the XOR operation). We also calculate $g_i = x_i(1 - y_i)$. This is 1 iff $x_i = 1$ and $y_i = 0$. Then, if we set p_i such that $p_i = 1 \iff$ "all bits more significant than i are equal", which can be written as $p_i = \prod_{j=i+1}^{n}(1 - e_j)$. Then the output of the circuit is $\mathcal{F}_C(\mathbb{F}_2)(x_1, ..., x_k, y_1, ..., y_k) = 1 - \prod_{i=1}^{n}(1 - p_i g_i)$, which is 1 if $x > y$ and 0 otherwise. For clarity, it is shown in figure 1 for $k = 2$, as the circuit is much larger in size for large k.

2.3 Secret Sharing

In this section, secret sharing is introduced. This is a widely used primitive in MPC protocols. Intuitively, this technique allows the parties to split a so-called secret into different shares and distribute these among themselves. The secret can then only be recovered using enough of these shares. Each of these shares is constructed in such a way that it reveals nothing about the secret.

Definition 1 *A $(t, n)-$secret-sharing scheme for a finite and non-empty domain S consists of a randomized sharing algorithm that for a given input $s \in S$ outputs n shares $s_1, ..., s_n$ such that the following two properties hold for any subset $A \subset \{1, ..., n\}$:*

1. *Reconstructability: If $|A| > t$, then s is uniquely determined by the shares s_i with $i \in A$.*
2. *Privacy: If $|A| \leq t$, then the joint distribution of s_i with $i \in A$ does not depend on the choice of s.*

Remark 1 *In practice, it is often additionally required that the sharing algorithm and the reconstruction of s from t shares is efficient, but we do not include this into the definition for simplicity. There are also secret sharing schemes that differentiate between reconstructability and privacy thresholds, not using the same value t for both. Secret sharing is a widely used security approach in MPC, and all the protocols discussed in this thesis incorporate some variation of it.*

Notation 2 *Let $[s]$ denote the list of shares produced from a given secret sharing scheme on the input s. Note that $[s]$ is not fixed, but randomized in some form.*

Definition 2 *A secret sharing scheme is linear if S is a finite field and the sharing $[s] = (s_1, ..., s_n)$ is computed by means of applying a linear map to s and uniformly randomly sampled $r_1, ..., r_q \in S$ for some suitable $q \in \{1, ..., n\}$.*

Remark 2 *In a linear secret sharing scheme, the reconstruction (for any given A) is also in the form of a linear map. In particular, if $[a] = (a_1, ..., a_n)$ is a sharing of a and $[b] = (b_1, ..., b_n)$ is a sharing of b, then $[a] + [b] := (a_1 + b_1, ..., a_n + b_n)$ is a sharing of $a + b$, and with element-wise computation $[a]y$ is a sharing of ay for any $y \in S$. By means of local computation, it is also possible to compute a sharing of $a + z$ from $[a]$ and any $z \in S$.*

Let us look at two important examples of linear secret sharing schemes. Namely Shamir's secret sharing [Sha79] and additive secret sharing.

Shamir's secret sharing scheme is a (t, n)-secret sharing scheme that uses Lagrange interpolation (5.1). Assume the secret $s \in \mathbb{F}_q$ for a prime power $q \geq n$. Here and henceforth, we denote by $\mathbb{F}_q$ the finite field of q elements. Sample uniformly randomly $r_1, ..., r_{t-1} \in \mathbb{F}_q$. Then each party i receives $(i, f(i))$ as their share for $f := s + r_1X + ...r_{t-1}X^{t-1}$. Using interpolation, the secret can then be recovered by any group of at least t parties. In this explanation, the points i are used as interpolation points, but it is possible to use any n distinct points in $\mathbb{F}_q$.

Additive secret sharing refers to any secret sharing scheme that outputs shares $s_1, ..., s_n \in R$ for some commutative ring R of a secret $s \in S$ to the n parties such that we have $\sum_i s_i = s$.

The lay-out of a linear secret sharing-based MPC protocol is as follows. Assume we have n parties $P_1, ..., P_n$ with inputs $x_1, ..., x_n \in \mathbb{F}_q$ and a linear secret sharing scheme $[\cdot]$. Assume furthermore that we have a circuit C that gives rise to the function $\mathcal{F}_C(\mathbb{F}_q)$ that needs to be computed. Let us assume lastly that the pre-processing phase is either not present or already completed. The online phase then commences and is structured into three steps. In the first step, the private inputs $x_i \in \mathbb{F}_q$ are shared among the parties using the secret sharing scheme. Each party now holds a share of every other party's input. In the second step, the computation is performed, represented by the circuit. At every addition gate, each party adds the applicable shares. Let a, b be the inputs for an addition gate, then the parties can locally compute $[a+b] = [a]+[b]$ due to the linearity of the secret sharing scheme. Multiplication of secrets is more complex. This is because $[a \cdot b]$ does not in general equal $[a] \cdot [b]$. A multiplication sub-protocol is introduced here. This sub-protocol varies depending on the protocol and secret sharing scheme. In the pre-processing model, a notion called Beaver triples can be used for this. We explain this notion below. In the third step, when the end of the circuit has been reached, the parties reconstruct the output to receive $\mathcal{F}_C(\mathbb{F}_q)(x_1, ..., x_n)$. This reconstruction is called opening.

In the pre-processing model, the most used, or expanded upon, notion to decrease communication complexity for every multiplication in the online phase is the following.

Definition 3 *A triple of sharings $[a], [b], [c]$ for random and independent a and b, such that $c = ab$, is called a Beaver triple. In the context of an MPC protocol, it is typically also assumed that a and b are unknown to the parties.*

The later protocols in this thesis use an extended version of this, called multiplication quintuples. To understand how these quintuples decrease communication complexity, it can be useful to know how Beaver triples are generated in the pre-processing phase and consumed in the online phase to increase efficiency of the online phase. Beaver triples can be generated in a multitude of ways. For example, for some set-ups using additive secret sharing, every party P_i can generate random values a_i, b_i, then $a := \sum_i a_i, b := \sum_i b_i$. Note that the parties do not have access to the values a, b, since these sums do not need to be computed for the generation of the random values. Then, using a technique called oblivious transfer (OT), sharings of ab can be securely computed. There are other ways, using other techniques, such as homomorphic encryption (HE) or oblivious linear evaluation (OLE). An example of Beaver triple generation can be found in the 2008 paper by Beerliová and Hirt [BTH08]. If we assume we have access to Beaver triples, a multiplication sub-protocol can look something like this.

Input: $[x], [y]$ two sharings of values $x, y \in \mathbb{F}_q$, a Beaver triple $([a], [b], [c])$.
Output: $[xy]$
Compute $[d] \leftarrow [x] - [a]$ and $[e] \leftarrow [y] - [b]$. Reconstruct (usually called 'open') $d \leftarrow [d]$ and $e \leftarrow [e]$ using the opening sub-protocol, which depends on the type of secret sharing scheme. Then, we compute:

$$[w] \leftarrow [a]e + [b]d + de + [c]$$

We see that

$$\begin{aligned} w &= ae + bd + de + c \\ &= a(y - b) + b(x - a) + (x - a)(y - b) + ab \\ &= ay - ab + bx - ba + xy - ya - xb + ab + ab \\ &= xy. \end{aligned}$$

Thus, indeed $[w]$ is a sharing of xy and can be denoted as $[xy]$. This allows us to compute a multiplication with only two rounds of communication (opening of $[d]$ and $[e]$). Note that, for obvious security reasons, we cannot open $[x]$ or $[y]$, but opening $[d]$ and $[e]$ only gives access to the values $x - a$ and $y - b$. Having access to these values does not allow one to compute the value x or y, even with unlimited computational power, as both x, y and a, b are unknown. This type of security guarantee is called statistical security. The other main security model is called computational security, which guarantees security against any adversary as long as they do not solve a problem believed to be infeasible in polynomial time.

3 RMFEs and Elementary Properties

This thesis will focus on protocols using a recently developed tool, first introduced in 2018 by Cascudo et al. [CCXY18] and separately by Block et al. [BMN18]. The name Reverse Multiplication Friendly Embedding (RMFE) and the notation below were introduced by Cascudo et al.

Definition 4 *Let $\mathbb{F}$ be a finite field. Let $a, b \in \mathbb{F}^n$. Then $a * b$ denotes the Schur product given by $a * b = (a_1 \cdot b_1, ..., a_n \cdot b_n)$.*

Definition 5 *Let q be a power of a prime and let $\mathbb{F}_q$ be the field of q elements. Let $k, m \in \mathbb{Z}_{\geq 1}$ be integers. A pair (ϕ, ψ) is called a $(k,m)_q-$reverse multiplication friendly embedding (RMFE) if $\phi : \mathbb{F}_q^k \to \mathbb{F}_{q^m}$ and $\psi : \mathbb{F}_{q^m} \to \mathbb{F}_q^k$ are two $\mathbb{F}_q$-linear maps satisfying:*

$$x * y = \psi(\phi(x) \cdot \phi(y)), \forall x, y \in \mathbb{F}_q^k. \tag{1}$$

Claim 1 *Let (ϕ, ψ) be an RMFE of any kind. Then ϕ is injective and ψ is surjective.*

Proof 1 *Let $x \in \mathbb{F}_q^k$ such that $\phi(x) = 0$. Then, by the definition of an RMFE, we have for any $y \in \mathbb{F}_q^k$: $\psi(\phi(x) \cdot \phi(y)) = x * y$, which is equal to $\psi(0) = 0$, since ψ is linear. This holds for any $y \in \mathbb{F}_q^k$, thus also for $y' := (1, 1, ..., 1)$. This gives us that $x * y' = x = 0$. ϕ is thus injective, as it has trivial kernel.*
Let $x \in \mathbb{F}_q^k$, then we see that $\psi(\phi(x) \cdot \phi((1, 1, ..., 1))) = x * (1, 1, ..., 1) = x$. This holds for any $x \in \mathbb{F}_q^k$, thus ψ is surjective. □

The original goal of RMFEs is as follows. Let C be any circuit with n inputs. Instead of taking $x_1, ..., x_n \in \mathbb{F}_q^k$ and computing $\mathcal{F}_C(\mathbb{F}_q^k)(x_1, ..., x_n)$, thus effectively computing $\mathcal{F}_C(\mathbb{F}_q)$ k times, a suitable RMFE allows us to compute C over $\mathbb{F}_{q^m}$ with n inputs $\phi(x_1), ..., \phi(x_n) \in \mathbb{F}_{q^m}$. Intuitively, the RMFE thus allows us to parallelize the computation. However, for ψ applied element-wise to the outputs,

$$\psi(\mathcal{F}_C(\mathbb{F}_{q^m})(\phi(x_1), ..., \phi(x_n))) \neq \mathcal{F}_C(\mathbb{F}_q^k)(x_1, ..., x_n).$$

This is partially because $\psi(\phi(x) \cdot \phi(y) \cdot \phi(z))$ is not necessarily equal to $x * y * z$ for any $x, y, z \in \mathbb{F}_q^k$. To apply this method, every time we encounter a multiplication gate during computation of C over $\mathbb{F}_{q^m}$, we apply $\phi \circ \psi$ to the output of this gate. To realize that this works, let again $x, y, z \in \mathbb{F}_q^k$. Then, for $\tau := \phi \circ \psi$, we have

$$\tau\Big(\tau\big(\phi(x) \cdot \phi(y)\big) \cdot \phi(z)\Big) =$$
$$\tau\bigg(\phi\Big(\psi\big(\phi(x) \cdot \phi(y)\big) \cdot \phi(z)\Big)\bigg) =$$
$$\tau(\phi(x * y) \cdot \phi(z)) = \phi((x * y) * z) = \phi(x * y * z).$$

Furthermore, we do not in general have $\forall x \in \mathbb{F}_q^k : \psi(\phi(x)) = x$. To retrieve the intended result in $\mathbb{F}_q^k$, we can use ϕ^{-1}, which exists since ϕ is injective. We will show injectivity below. We thus compute a function $\mathcal{F}'_C(\mathbb{F}_{q^m})$, which is given from $\mathcal{F}_C(\mathbb{F}_{q^m})$ in the natural way by applying τ after every multiplication in C. Then we see, by applying ϕ^{-1} element-wise to the outputs,

$$\phi^{-1}(\mathcal{F}'_C(\mathbb{F}_{q^m})(\phi(x_1), ..., \phi(x_n))) = \mathcal{F}_C(\mathbb{F}_q^k)(x_1, ..., x_n).$$

Note that we cannot form a circuit that gives rise to $\mathcal{F}'$ in the natural way from C without the use of additional multiplication gates. For this reason, it is useful to view $\mathcal{F}'$ as a function given rise to by the original circuit C with the addition that τ is applied after every multiplication instead of viewing a new circuit C' such that $\mathcal{F}_{C'}(\mathbb{F}_{q^m}) = \mathcal{F}'_C(\mathbb{F}_{q^m})$. As mentioned, this would include additional multiplication gates and thus induce the issues discussed above.
The above is the basis of the protocol by Cascudo et al. from 2018 [CCXY18], which is discussed further in Section 4.1.

Remark 3 *Note that to guarantee $\psi(\phi(x)) = x$, we can introduce the additional constraint that $\phi((1, 1, ..., 1)) = 1$, since then $\psi(\phi(x)) = \psi(\phi(x) \cdot \phi((1, 1, ..., 1))) = x * (1, 1, ..., 1) = x$. This is an additional assumption made in the protocol by Escudero et al. in 2022 [EXY22]. This protocol is over Galois rings, a generalization of finite fields, which we will get into later.*

4 Application of RMFEs to MPC

4.1 Efficient RMFE-based protocol for binary circuits

The first application of RMFEs, in the paper they were originally introduced in by Cascudo et al. [CCXY18], was to efficiently perform MPC over a binary circuit in the active security, honest majority model (specifically $t < n/3$) with a new trade-off. As the new trade-off, they construct a way to trade the size of the field for amortized communication complexity, whereas in previous research the amortized communication complexity was traded for the adversary threshold t. Since most commonly used protocols, for example those that make use of Shamir's secret sharing scheme, require the field size to be at least n, there is a need to view the inputs of the binary circuit as elements of a larger field. We must embed them in $\mathbb{F}_q$ for a prime power q and $q > n$. This incurs an overhead of $O(\log q) >= O(\log n)$ in communication complexity, as around $\log(q)$ bits are used - exactly $\log(q)$ if $q = 2^k$ for some k - instead of only the 1 that was needed originally. Cascudo et al. [CCXY18] create a protocol using RMFEs by applying a protocol by Beerliová and Hirt [BTH08] as a semi-black box. The protocol by Beerliová and Hirt was, at the time, (one of) the leading protocol(s) in efficiency for dishonest minority ($t < n/3$) and active security, having communication complexity of $O(n)$ per multiplication. As defined in Section 3, they introduce a protocol that computes $\mathcal{F}'_C(\mathbb{F}_{2^m})(\phi(x_1), ..., \phi(x_n)), x_1, ..., x_n \in \mathbb{F}_2^k$ instead of computing $\mathcal{F}_C(\mathbb{F}_2^k)(x_1, ..., x_n)$. They applied the protocol by Beerliová and Hirt for computation of this circuit, except for at the multiplication gates, where they introduced a new subprotocol. As mentioned in Section 3, it is necessary to apply $\tau := \phi \circ \psi$ to the output. Note that ϕ and ψ are only $\mathbb{F}_2$-linear, not $\mathbb{F}_{2^m}$-linear. For any sharing $[r]$ of a value $r \in \mathbb{F}_{2^m}$, a sharing of $\tau(r)$ can thus not be constructed using local computations. For this reason, a subprotocol is needed. They also introduced a sub-protocol using a so called Zero-Knowledge Proof that the input provided is in $\mathrm{Im}(\phi)$, as dishonest parties may attempt to give a value in $\mathbb{F}_{2^m}$ as input that is outside of this image. This thesis will not go into detail about this. The interested reader is redirected to the original paper by Cascudo et al. [CCXY18], where they show that these two issues can be reduced to the following: "construct a secure multi-party protocol that outputs a sharing of a random element in a prescribed $\mathbb{F}_2$-subspace of $(\mathbb{F}_{2^m})^v$. That is, given an $\mathbb{F}_2$-vector space $V \subseteq \mathbb{F}_{2^m}^v$, the protocol should output $([r_1], ..., [r_v])$ where $(r_1, ..., r_v)$ is uniformly random in V, and $[x]$ denotes a sharing of x with the secret sharing scheme used in the protocol for C'." [CCXY18], after which they show how to construct such

a protocol. Note that C' is their notation for C computed over $\mathbb{F}_{2^m}$. After the computation of the circuit, including these subprotocols, the desired output in $\mathbb{F}_q^k$ was retrieved, as in Section 3, by applying ϕ^{-1}. This protocol performs $k >= O(\log n)$ multiplications in parallel, removing the $O(\log n)$ overhead and creating a protocol with amortized complexity of $O(n)$ per gate per instance compared to the $O(n \log n)$ it would regularly cost.
In Remark 7 of the paper by Cascudo et al. [CCXY18], the following practical parameters are introduced for use with their protocol specifications.

1. For all $1 \leq r \leq 9$, there exists a $(2r, 6r-3)_2$-RMFE, obtained by concatenation of $(2,3)_2$ and $(r, 2r-1)_8$-RMFEs.

2. For all $1 \leq r \leq 33$, there exists a $(3r, 10r-5)_2$-RMFE, obtained by concatenation of $(3,5)_2$ and $(r, 2r-1)_{32}$-RMFEs.

The existence of $(2,3)_2$, $(r, 2r-1)_8$, $(3,5)_2$ and $(r, 2r-1)_{32}$-RMFEs follows from Lemma 4 in this paper by Cascudo et al. The possibility of concatenation follows from Lemma 5 in the same paper. In Section 5, the lemmas are shown and proven.

4.2 Translating to Dishonest Majority

In their proceeding paper from 2020 [CG20], Cascudo and Gundersen introduced a new protocol to extend to dishonest majority. To accomplish this, they combined their approach with the SPDZ protocol. The original version of SPDZ was introduced by Damgard et al. in 2012 [DPSZ12] and coined as SPDZ a year later by Damgard et al. [DKL+13]. The SPDZ protocol operates over a large field, which intuitively explains why RMFEs can be used. Furthermore, Cascudo and Gundersen show that their protocol can be transformed to efficiently compute one single binary circuit C (thus over $\mathbb{F}_2$) - compared to the amortized result of running multiple instances of the same circuit - with overhead depending on the circuit structure. They accomplish this by forming a new circuit C' that is to be computed over $\mathbb{F}_{2^m}$, taking multiple multiplication gates that can be computed in parallel from the original circuit and combining them into one multiplication gate in C'. All multiplications that can be done in parallel are considered to be in one so-called layer and all multiplications that can be computed in parallel directly after this round of multiplications are considered the next layer. Under the assumption that all the output bits of a given layer are used in the next layer and only there, they state that one can expect an overhead of about 2 bits of communication per multiplication gate.

4.2.1 The SPDZ Protocol: A General Idea

In SPDZ, to guarantee active security in the dishonest majority model, Message Authentication Codes (MACs) are introduced. Originally, SPDZ is defined over an arithmetic circuit over some large finite field $\mathbb{F}_q$. We take $\alpha \in \mathbb{F}_q$ and call it the global key. This key is then secret shared among the parties with an additive secret sharing scheme. During the protocol, for any value $x \in \mathbb{F}_q$ in the computation, the parties not only receive a share of x, but also of $\alpha \cdot x$. This $\alpha \cdot x$ then acts as a MAC for x. This provides statistical security, as when an adversary attempts to tamper with a value x, for example it 'pretends the value is $x + \varepsilon$' for some $\varepsilon \in \mathbb{F}_q$, by sending incorrect values as shares, they would also need to produce a MAC $\alpha(x + \varepsilon)$ corresponding to the original value. This

is equivalent to having to guess $\alpha \in \mathbb{F}_q$, as they only hold a share of α and do not know its value. The probability of guessing correctly is then $\frac{1}{q}$, which is considered negligible if we are operating over a large field.
As one can see, the probability of guessing correctly is quite large if we are working over a smaller sized field, like $\mathbb{F}_2$ for example. To counteract this, one can embed into a larger field. However, this creates quite a bit of overhead. RMFEs were again used to address this overhead, similarly to 4.1, by embedding multiple smaller inputs into a larger field, such that the adversary needs to guess an element in this larger field.

4.2.2 The Protocol

For the protocol by Cascudo and Gundersen [CG20], only the following two facts following from Lemma 4 and 5 by Cascudo et al. [CCXY18] were used. These lemmas and their proof are shown in Section 5 of this thesis and denoted as Lemma 1 and Lemma 2.

Corollary 1 (Lemma 1, Lemma 2) *For all $r \leq 33$, there exists a $(3r, 10r-5)_2$-RMFE.*

Proof 2 *Lemma 1 gives us that there exists a $(3,5)_2$-RMFE ($k=3$) and a $(r, 2r-1)_{32}$-RMFE for $r \leq 32+1 = 33$. Lemma 2 then gives us a $(3r, 10r-5)_2$-RMFE.* □

Corollary 2 (Proof of Lemma 1, Lemma 2) *For all $r \leq 16$, there exists a $(2r, 8r)_2$-RMFE.*

Proof 3 *Clearly, the proof of Lemma 1 can also be used to show the existence of a $(k, 2k)_q$-RMFE for any $1 \leq k \leq q+1$. This gives us that there exist $(2,4)_2$- and $(r, 2r)_{16}$-RMFEs for $r \leq 16$. Lemma 2 then gives us a $(2r, 8r)_2$-RMFE.* □

The motivation for Corollary 2 is that it gives access to extension fields with a size of a power of 2. Cascudo and Gundersen state that this might be desirable. Of course, this is in theory less efficient than the RMFEs in Corollary 1, thus it is clear that some overhead is introduced.
These RMFEs are used specifically to form 'mixed authenticated sharings'. The sharing of some value x is then the combination of 'data shares', consisting of additive shares of $x \in \mathbb{F}_q$ and 'MAC shares', consisting of additive shares of $\alpha \cdot \phi(x)$ for some global key $\alpha \in \mathbb{F}_{q^m}$ in the extension field. This global key is also additively shared from the start of the protocol, with the shares denoted as $\alpha^{(i)}$. Due to the additivity of ϕ, two of these 'mixed' sharings can still be added locally, as can a public vector and a sharing, to receive a new sharing.
The mixed sharing scheme is denoted as $\langle \cdot \rangle$, with the sharing of $x \in \mathbb{F}_2^k$ as follows:

$$\langle x \rangle := \left((x^{(i)}, ..., x^{(n)}), (m^{(1)}(x), ..., m^{(n)}(x)) \right).$$

Every party P_i holds an additive share $x^{(i)} \in \mathbb{F}_2^k$ of x and a share $m^{(i)}(x)$, which is called a MAC share, such that $\sum_{i=1}^n m^{(i)}(x) = \alpha \cdot \sum_{i=1}^n \phi(x^{(i)}) = \alpha \cdot \phi(x)$ for α as above. It is indeed possible to add such sharings locally by defining

$$\langle x \rangle + \langle y \rangle = \langle x + y \rangle = \left((x^{(1)} + y^{(1)}, ..., x^{(n)} + y^{(n)}), (m^{(1)}(x) + m^{(1)}(y), ..., m^{(n)}(x) + m^{(n)}(y)) \right)$$

and it is also possible to locally compute $\langle x \rangle + a$ for a given public vector a by defining

$$\langle x \rangle + a = \left((x^{(1)} + a, ..., x^{(n)} + a), (m^{(1)}(x) + \alpha^{(1)} \cdot \phi(a), ..., m^{(n)}(x) + \alpha^{(n)} \cdot \phi(a)) \right).$$

These MACs are not checked until the output gate. Here, the parties check the values using random linear combinations of partially opened values after opening the global key α. Partial opening in this context means that the additive sharing of the values for x in the explanation above are opened, thus x is computed from the $x^{(i)}$. This ensures statistical security with probability 2^{-m}. As one might expect, multiplication gates are complicated and require some extra attention.

4.2.3 Dealing With Multiplication

One of the standard ways of efficiently computing multiplication gates is the use of Beaver triples. However, as we will see, the standard method is not sufficient for the protocol by Cascudo and Gundersen [CG20]. Of course, for the computation of the partial sharing for $1 \leq i \leq n : (x * y)^{(i)}$ of $x * y$, Beaver triples can be applied, but when looking at the MAC shares $m^{(i)}(x * y)$ of $\alpha \cdot \phi(x * y)$, an issue is encountered. Assume we have obtained a triple $\langle a \rangle, \langle b \rangle, \langle a * b \rangle$. The parties then partially open $d^{(i)}$ of $d = x - a$ and $e^{(i)}$ of $e = y - b$. We then see:

$$\alpha \cdot \phi(x * y) = \alpha \cdot \phi(a * b) + \alpha \cdot \phi(a * e) + \alpha \cdot \phi(b * d) + \alpha\phi(d * e).$$

The issue here is that $\alpha \cdot \phi(a * e) = \alpha \cdot (\phi \circ \psi)(\phi(a)\phi(e))$, but this is not in general equal to $(\phi \circ \psi)(\alpha \cdot \phi(a)\phi(e))$, since $\phi \circ \psi$ is not in general $\mathbb{F}_{q^m}$-linear. We can thus not easily compute an additive sharing of $\alpha \cdot \phi(x * y)$ based on the sharings of $\alpha \cdot \phi(x)$ and $\alpha \cdot \phi(y)$.
For this, another sharing scheme is introduced, originating from SPDZ, denoted as $[\cdot]$. This is a sharing of a value $r \in \mathbb{F}_{2^m}$, given by

$$[r] = \left((r^{(1)}, ..., r^{(n)}), (m^{(1)}(r), ..., m^{(n)}(r))\right).$$

Each party has an additive share $r^{(i)}$ of r and an additive share $m^{(i)}(r)$ of $\alpha \cdot r$. Thus $\sum_{i=1}^{n} m^{(i)}(r) = \alpha \cdot \sum_{i=1}^{n} x^{(i)}$. By adding to the pre-processing, besides $\langle a \rangle, \langle b \rangle, \langle a * b \rangle$, the values $\langle \psi(r) \rangle, [r]$, for a random element $r \in \mathbb{F}_{q^m}$, one can compute $\langle x * y \rangle$. This proceeds as follows. Compute and partially open σ from the $\sigma^{(i)}$ of

$$[\sigma] = [\phi(x)\phi(y) - \phi(a)\phi(b) - r].$$

The construction of $[\sigma]$ is shown in 4.2.4. Assuming we have σ, we can then apply ψ to find

$$\psi(\sigma) = \psi(\phi(x)\phi(y) - \phi(a)\phi(b) - r) =$$
$$\psi(\phi(x)\phi(y)) - \psi(\phi(a)\phi(b)) - \psi(r) = x * y - a * b - \psi(r).$$

Then finally we obtain $\langle x * y \rangle \leftarrow \psi(\sigma) + \langle a * b \rangle + \langle \psi(r) \rangle$, from $\langle a * b \rangle$ and $\langle \psi(r) \rangle$, both present in the pre-processed quintuple. For the pre-processing, Cascudo and Gundersen thus introduce a quintuple $(\langle a \rangle, \langle b \rangle, \langle a * b \rangle, \langle \psi(r) \rangle, [r])$ for every multiplication instead of a triple $([a]_*, [b]_*, [c]_*)$ for some other secret sharing scheme $[\cdot]_*$.

4.2.4 Construction of σ

For the construction of σ, Cascudo and Gundersen [CG20] introduce two more operations on the shared values. The first is the Schur product of a sharing $\langle x \rangle$ of $x \in \mathbb{F}_2^k$ and a public vector $a \in \mathbb{F}_2^k$.

$$a * \langle x \rangle = \left((\phi(a) \cdot \phi(x^{(1)}), ..., \phi(a) \cdot \phi(x^{(n)})), (\phi(a) \cdot m^{(1)}(x), ..., \phi(a) \cdot m^{(n)}(x))\right).$$

We have $\phi(a) \cdot \phi(x^{(i)})$ as the data shares, which are shares of $\phi(a) \cdot \phi(x)$ and the MAC shares are clearly shares of $\alpha \cdot \phi(a) \cdot \phi(x)$. However, as Cascudo and Gundersen point out: "the data shares are not distributed uniformly in $\mathbb{F}_{2^m}$ because ϕ is not surjective, so one cannot say this equals $[\phi(a) \cdot \phi(x)]$." Note that their wording is slightly incorrect, as ϕ is not in general surjective. What is true however, as they also point out, is that for another shared element $[z], z \in \mathbb{F}_{2^m}$, we have $a * \langle x \rangle + [z] = [\phi(a) \cdot \phi(x) + z]$. This is because after adding a random element in $\mathbb{F}_{2^m}$, the result can once again be any element of $\mathbb{F}_{2^m}$, one could say 'uniformly distributed-ness is restored'. The second operation that was introduced was the sum of a sharing $\langle x \rangle$ of an element $x \in \mathbb{F}_2^k$ and a sharing $[y]$ of an element $y \in \mathbb{F}_{2^m}$, which was defined as $\langle x \rangle + [y] = [\phi(x) + y]$ in the obvious way. It is now time to show how $[\sigma]$ is constructed. Recall that we have given $\langle x \rangle, \langle y \rangle$ and a quintuple $(\langle a \rangle, \langle b \rangle, \langle a * b \rangle, \langle \psi(r) \rangle, [r])$. First, partially open $d = x - a$ and $e = y - b$. Then construct

$$
\begin{gathered}
d * \langle y \rangle + e * \langle x \rangle - \phi(d) \cdot \phi(e) - [r] = [\phi(d) \cdot \phi(y) + \phi(e) \cdot \phi(x) - \phi(d) \cdot \phi(e) - r] = \\
[\phi(x - a) \cdot \phi(y) + \phi(y - b) \cdot \phi(x) - \phi(x - a) \cdot \phi(y - b) - r] = \\
[\phi(x) \cdot \phi(y) - \phi(a) \cdot \phi(y) + \phi(y) \cdot \phi(x) - \phi(b) \cdot \phi(x) - (\phi(x) - \phi(a)) \cdot (\phi(y) - \phi(b)) - r] = \\
[2\phi(x) \cdot \phi(y) - \phi(x)\phi(y) - \phi(a) \cdot \phi(y) + \phi(a)\phi(y) - \phi(b) \cdot \phi(x) + \phi(x)\phi(b) - \phi(a)\phi(b) - r] = \\
[\phi(x) \cdot \phi(y) - \phi(a)\phi(b) - r] = [\sigma].
\end{gathered}
$$

Thus indeed by only partially opening $d = x - a$ from $\langle x \rangle - \langle a \rangle$ and $e = y - b$ from $\langle y \rangle - \langle b \rangle$, we can construct $[\sigma]$ using $\langle a \rangle, \langle b \rangle$ and $[r]$, the three yet unused values of the quintuple $(\langle a \rangle, \langle b \rangle, \langle a * b \rangle, \langle \psi(r) \rangle, [r])$, having used $\langle a * b \rangle$ and $\langle \psi(r) \rangle$ in 4.2.3

5 Construction of RMFEs over Fields

The first explicit construction of RMFEs was given by Cascudo et al. in 2018 [CCXY18]. They gave a polynomial construction in Lemma 4, with a way to concatenate RMFEs in Lemma 5 and went into the construction of an asymptotic family of RMFEs using algebraic geometry in the rest of the section. In this thesis a method is given to extend the polynomial construction to the more general Galois rings. This has been done for the asymptotic family of RMFEs by Escudero et al. in 2022 [EXY22], but, as far as the author could find, not directly for the simpler polynomial construction. This is done in this thesis to provide intuition to the reader. This construction uses the notion of Lagrange interpolation quite heavily. For this reason, we introduce this in the following subsection.

5.1 Lagrange Interpolation

Let q be a prime power and let k be a strictly positive integer. Let $x_1, ..., x_k \in \mathbb{F}_q$ be pairwise distinct. For any $y_1, ..., y_k \in \mathbb{F}_q$, there exists a unique polynomial $f \in \mathbb{F}_q[X]$ of degree at most $k - 1$ such that $f(x_i) = y_i, \forall i \in \{1, ..., k\}$. Lagrange interpolation, which is described below, shows how to find this unique polynomial.

Definition 6 *Let $k \in \mathbb{Z}_{>0}$. Let $x_1, ..., x_k \in \mathbb{F}_q$ be distinct. Let for $1 \leq j \leq k$*

$$
\ell_j(X) := \prod_{\substack{1 \leq m \leq k \\ m \neq j}} \frac{X - x_m}{x_m - x_j} \tag{2}
$$

be the Lagrange basis for polynomials of degree $\leq k - 1$.

Lemma 1 *Let $f \in \mathbb{F}_q[X]$ be a polynomial of degree at most $k-1$. Let $x_1, ..., x_k \in \mathbb{F}_q$ be distinct and let $y_1 = f(x_1), ..., y_k = f(x_k)$. Then*

$$f(X) = \sum_{j=1}^{k} y_j \ell_j(X). \tag{3}$$

Proof 4 *Let $1 \leq i \leq k$. We see*

$$\sum_{j=1}^{k} y_j \ell_j(x_i) = \sum_{j=1}^{k} y_j \left(\prod_{\substack{1 \leq m \leq k \\ m \neq j}} \frac{x_i - x_m}{x_m - x_j} \right) = y_i,$$

since $\forall j \neq i$: $\prod_{\substack{1 \leq m \leq k \\ m \neq j}} (x_i - x_m) = 0$ and for $j = i$, we have $\prod_{\substack{1 \leq m \leq k \\ m \neq j}} (x_i - x_m) = \prod_{\substack{1 \leq m \leq k \\ m \neq j}} (x_j - x_m)$, thus $\ell_j(x_j) = 1$. This holds for all i, thus f and $\sum_{j=1}^{k} y_j \ell_j$ are equal at k distinct points, while both have degree $k-1$. They are therefore equal.

This manner of finding f with given x_i and $y_i = f(x_i)$ is Lagrange interpolation.

Remark 4 *It is in fact possible to extend Lagrange interpolation to include an extra point, often referred to as 'the point at infinity', which is the coefficient of the degree $k-1$ term for polynomials of degree at most $k-1$. In the construction in the next subsection, we will show this.*

5.2 Polynomial Construction

Notation 3 *We write $\mathbb{F}_q[X]_{\leq m} := \{f \in \mathbb{F}_q[X] : deg(f) \leq m\}$ for $m \in \mathbb{Z}_{>0}$.*

Using the fact that this interpolation method induces an isomorphism of $\mathbb{F}_q$-vector spaces between $\mathbb{F}_q[X]_{\leq k-1}$ and $\mathbb{F}_q^k$ for any integer $1 \leq k \leq q+1$, a simple polynomial construction was introduced in [CCXY18], which is given below with additional clarification for completeness.

Notation 4 *Let for any $m \in \mathbb{Z}_{>0}$, ∞_m be a formal symbol such that $f(\infty_m) := b_m$, for $f = \sum_{i=1}^{m} b_i X^i \in \mathbb{F}_q[X]_{\leq m}$.*

Lemma 2 (Lemma 4, Cascudo et al., 2018) *For all $1 \leq k \leq q+1$, there exists a $(k, 2k-1)_q$−RMFE.*

Construction 1 (Lemma 2) *Let $a_1, .., a_k \in \mathbb{F}_q \cup \{\infty_{k-1}\}$ be pairwise distinct. Let $\alpha \in \mathbb{F}_{q^{2k-1}}$ such that $\mathbb{F}_{q^{2k-1}} = \mathbb{F}_q(\alpha)$. Note that this is equivalent to the statement that $\mathbb{F}_{q^{2k-1}}$ has basis $\{1, \alpha, \alpha^2, ..., \alpha^{2k-2}\}$ as an $\mathbb{F}_q$ vector space.*
Define

$$\mathcal{E}_1 : \mathbb{F}_q[X]_{\leq k-1} \to \mathbb{F}_q^k, f \mapsto (f(a_1), ..., f(a_k)) \tag{4}$$

$$\mathcal{E}_2 : \mathbb{F}_q[X]_{\leq 2k-2} \to \mathbb{F}_{q^{2k-1}}, f \mapsto f(\alpha) \tag{5}$$

$$\mathcal{E}_1' : \mathbb{F}_q[X]_{\leq 2k-2} \to \mathbb{F}_q^k, f \mapsto (f(a_1'), ..., f(a_k')) \tag{6}$$

with $a_i' := \begin{cases} a_i & \text{if } a_i \in \mathbb{F}_q \\ \infty_{2k-2} & \text{if } a_i = \infty_{k-1} \end{cases}$.
This is visualized in Figure 2.

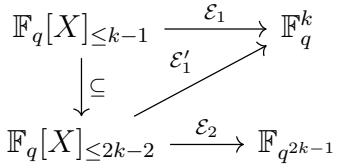

Figure 2: A diagram of $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_1'$

Claim 2 *$\mathcal{E}_1$ and $\mathcal{E}_2$ are isomorphisms of $\mathbb{F}_q$-vector spaces.*

Proof 5 *Clearly, $\mathcal{E}_1$ is linear. To see this, note that $(f+g)(\infty_{k-1}) = f(\infty_{k-1}) + g(\infty_{k-1})$ and $(\lambda f)(\infty_{k-1}) = \lambda f(\infty_{k-1})$. For any point in $\mathbb{F}_q$, it is apparent that this also holds.*
Note that injectivity and surjectivity of $\mathcal{E}_1$ would follow directly from Lagrange interpolation if one were to exclude ∞_{k-1}. We show that it also holds when we include it.
Injectivity of *$\mathcal{E}_1$ Let $f \in \mathbb{F}_q[X]_{\leq k-1}$ such that $\mathcal{E}_1(f) = 0$. Then we see $\forall i \in \{1, ..., k\} : f(a_i) = 0$. If $\nexists i : a_i = \infty_{k-1}$, then f has k distinct roots, but degree $k-1$, thus $f = 0$. Else if $\exists i : a_i = \infty_{k-1}$, then f has $k-1$ distinct roots, but degree $k-2$, thus $f = 0$. Thus $\mathcal{E}_1$ is injective.*
Surjectivity of *$\mathcal{E}_1$ Since $dim_{\mathbb{F}_q}(\mathbb{F}_q[X]_{\leq k-1}) = dim_{\mathbb{F}_q}(\mathbb{F}_q^k) = k$ and $\mathcal{E}_1$ is injective, it is thus surjective and thus with linearity an isomorphism of $\mathbb{F}_q$-vector spaces.*
Finally, $\mathcal{E}_2$ is trivially an isomorphism of $\mathbb{F}_q$-vector spaces, since we have basis $\{1, \alpha, ..., \alpha^{k-2}\}$ for $\mathbb{F}_{q^{2k-1}}$. $\square$

Claim 3 *Set $\phi := \mathcal{E}_2 \circ \mathcal{E}_1^{-1}$ and $\psi := \mathcal{E}_1' \circ \mathcal{E}_2^{-1}$. Then (ϕ, ψ) is a $(k, 2k-1)_q-$RMFE.*

Proof 6 *Clearly, these maps are $\mathbb{F}_q-$linear, thus we need only show (1).*
Let $x, y \in \mathbb{F}_q^k$. Let $f, g \in \mathbb{F}_q[X]_{\leq k-1}$ such that $\mathcal{E}_1(f) = x, \mathcal{E}_1(g) = y$. We then see:

$$\begin{aligned}\psi(\phi(x) \cdot \phi(y)) &= \psi(\mathcal{E}_2(\mathcal{E}_1^{-1}(x)) \cdot \mathcal{E}_2(\mathcal{E}_1^{-1}(y))) = \psi(\mathcal{E}_2(f) \cdot \mathcal{E}_2(g)) \\ &= \psi(f(\alpha)g(\alpha)) = \psi(fg(\alpha)) = \mathcal{E}_1'(\mathcal{E}_2^{-1}(fg(\alpha))) = \mathcal{E}_1'(fg) \\ &= (fg(a_1'), ..., fg(a_k')) = (f(a_1)g(a_1), ..., f(a_k)g(a_k)) \\ &= \mathcal{E}_1(f) * \mathcal{E}_1(g) = x * y.\end{aligned}$$

Note that for all $1 \leq i \leq k$, $fg(a_i') = f(a_i)g(a_i)$, since $fg(\infty_{2k-2}) = b_{k-1}c_{k-1} = f(\infty_{k-1})g(\infty_{k-1})$, for $f = \sum_{i=1}^{k-1} b_i X^i, g = \sum_{i-1}^{k-1} c_i X^i$. $\square$

The construction above gives an embedding for an embedded field of dimension close to twice k. It would intuitively be desirable to reduce the factor $\frac{m}{k}$ for a general $(k, m)_q-$RMFE. There is an assymptotic family of RMFEs, constructed using quite a bit of algebraic geometry, which addresses this problem, the details will not be addressed in this thesis. The interested reader is referred to [CCXY18]. Another main limitation is the limit of $k \leq q+1$. The second limitation is easily addressed with the following lemma, which is Lemma 5 in [CCXY18], which allows us to concatenate certain RMFEs.

Lemma 3 *Let (ϕ_1, ψ_1) be a $(k_1, m_1)_{q^{m_2}}-$RMFE and let (ϕ_2, ψ_2) be a $(k_2, m_2)_q-$RMFE. Define*

$$\begin{aligned}\phi : \mathbb{F}_q^{k_1 k_2} &\to \mathbb{F}_{q^{m_1 m_2}} \\ x = (x_1^{(1)}, x_1^{(2)}, ..., x_1^{(k_2)}, ..., x_{k_1}^{(k_2)}) &\mapsto \phi_1(\phi_2(x_1), ..., \phi_2(x_{k_1})),\end{aligned}$$

where we naturally identify $\mathbb{F}_q^{k_1k_2}$ *with* $(\mathbb{F}_q^{k_1})^{k_2}$ *and write for* $1 \leq i \leq k_1$*:* $x_i := (x_i^{(1)}, ..., x_i^{(k_2)}) \in \mathbb{F}_q^{k_2}$*. Also define*

$$\psi : \mathbb{F}_{q^{m_1m_2}} \to \mathbb{F}_q^{k_1k_2}$$
$$u \mapsto \psi_1(u) = (u_1, ..., u_{k_1}) \mapsto (\psi_2(u_1)^{(1)}, ..., \psi_2(u_1)^{(k_2)}, ..., \psi_2(u_{k_1})^{(k_2)}),$$

with analogous notation. Then (ϕ, ψ) *is a* $(k_1k_2, m_1m_2)_q-RMFE$*.*

Proof 7 (Lemma 3) *Clearly,* ϕ, ψ *are* $\mathbb{F}_q$*-linear. Let* $x, y \in \mathbb{F}_q^{k_1k_2}$*. Let* $x_i^{(j)}, y_i^{(j)}$ *be as above for* $1 \leq i \leq k_1, 1 \leq j \leq k_2$*. We have:*

$$\begin{aligned}\psi_1(\phi(x) \cdot \phi(y)) &= \psi_1(\phi_1(\phi_2(x_1), ..., \phi_2(x_{k_1})) \cdot \phi_1(\phi_2(y_1), ..., \phi_2(y_{k_1}))) \\ &= (\phi_2(x_1), ..., \phi_2(x_{k_1})) * (\phi_2(y_1), ..., \phi_2(y_{k_1})) \\ &= (\phi_2(x_1)\phi_2(y_1), ..., \phi_2(x_{k_1})\phi_2(y_{k_1})).\end{aligned}$$

Thus, we conclude:

$$\begin{aligned}\psi(\phi(x) \cdot \phi(y)) &= (\psi_2(\phi_2(x_1)\phi_2(y_1)), ..., \psi_2(\phi_2(x_{k_1})\phi_2(y_{k_1}))) \\ &= ((x_1 * y_1)^{(1)}, ..., (x_1 * y_1)^{(k_2)}, ..., (x_{k_1} * y_{k_1})^{(k_2)}) = x * y.\end{aligned}$$

Thus (ϕ, ψ) *is a* $(k_1k_2, m_1m_2)_q$*-RMFE.* □

6 Polynomial Construction over Galois Rings

Definition 7 *Let* $t \in \mathbb{Z}_{>0}$ *and let* p *a prime. Let* $f \in \mathbb{Z}/p^t\mathbb{Z}[X]$ *be a monic polynomial of degree* d *that is irreducible modulo* p*. Then* $R_t(d) := (\mathbb{Z}/p^t\mathbb{Z}[X])/f(X)$ *is called a Galois ring.*

Remark 5 *Note that a Galois ring is both local and finite. This gives it very similar characteristics to finite fields, since for a Galois ring* $R_t(d)$*, we have residue field* $R_t(d)/(p) \cong \mathbb{F}_{p^d}$*. For example, Galois rings with the same cardinality are isomorphic, which justifies leaving* f *out of the notation* $R_t(d)$*. Furthermore, any element outside of* (p) *is invertible.*

This polynomial construction can be lifted to Galois rings. In this section, the following Lemma is addressed.

Lemma 4 (Own Contribution) *Let* $t \in \mathbb{Z}_{>0}$ *and let* p *a prime. Let* $1 \leq k \leq p^d + 1$*. Then there exists a* $(k, 2k-1)_{p^t}$*-RMFE over the Galois ring* $R_t(d)$*.*

For this, we first need to extend the definition of an RMFE.

Definition 8 (RMFE over a Galois ring) *Let* $k, m \in \mathbb{Z}_{>0}$*. Let* $\phi : R_t(d)^k \to R_t(dm)$ *and* $\psi : R_t(dm) \to R_t(d)^k$ *be* $R_t(d)$*-module homomorphisms such that*

$$\psi(\phi(x) \cdot \phi(y)) = x * y, \forall x, y \in R_t(d)^k. \tag{7}$$

Then (ϕ, ψ) *is a* $(k, m)_{p^t}$*-RMFE over* $R_t(d)$*.*

6.1 Lifting Lagrange Interpolation

It is clearly possible to interpolate to a polynomial $f \in R[X]$ using the formula for Lagrange interpolation for any commutative ring R, as long as for interpolation points $a_1, ..., a_k$, the differences $a_m - a_j \in R$ are invertible for all $m, j \in \{1, ..., k\}, m \neq j$.

Lemma 5 *Let $R_t(d)$ be a Galois ring over a prime p. Let $a_1, ..., a_k$ be $1 \leq k \leq p^d$ distinct points in pairwise distinct residue classes modulo p. Then*

$$\mathcal{E}_1 : R_t(d)[Y]_{\leq k-1} \to R_t(d)^k \tag{8}$$
$$f \mapsto (f(a_1), ..., f(a_k)) \tag{9}$$

is a surjective homomorphism of $R_t(d)$-modules.

Remark 6 *Note that since $R_t(d)/(p) \cong \mathbb{F}_{p^d}$, we have p^d distinct residue classes. We may thus indeed choose p^d distinct points in pairwise distinct residue classes modulo p.*

Proof 8 *Since for any $1 \leq i, j \leq k$: $a_i - a_j \notin (p)$, we know $a_i - a_j$ is invertible. Thus, for any $y_1, ..., y_k \in R_t(d)^k$, we can find an f such that $f(a_1) = y_1, ..., f(a_k) = y_k$, namely using the formula for Lagrange interpolation. $\mathcal{E}_1$ is thus surjective.* $\square$

In fact, it follows trivially that this map is an isomorphism of $R_t(d)$-modules.

Lemma 6 *Let $1 \leq k \leq p^d$ and $a_1, ..., a_k \in R_t(d)$ pairwise distinct with pairwise differences $(a_i - a_j) \notin (p)$. Then $\mathcal{E}_1$ is an isomorphism of $R_t(d)$-modules.*

Proof 9 *This follows trivially, as $\mathcal{E}_1$ is surjective by Lemma 5 and injectivity follows from the fact that $R_t(d)[Y]_{\leq k-1}$ and $R_t(d)^k$ have equal cardinality.* $\square$

Remark 7 *The following is an alternative way to show injectivity, without the need to use the fact that the sets have equal cardinality. Let $f \in R_t(d)[Y]_{\leq k-1}$ such that $\forall 1 \leq i \leq k : f(a_i) = 0$. Since we have distinct elements $a_1, \ldots, a_k$ whose differences are not divisible by p, their reductions modulo p are distinct in $\mathbb{F}_{p^d}$. Let $[f] := (f \mod p) \in \mathbb{F}_{p^d}[Y]$. Then $[f]$ has degree at most $k-1$ and satisfies $[f]([a_i]) = 0$ for k distinct elements $[a_i] := (a_i \mod p) \in \mathbb{F}_{p^d}$. Therefore, $[f] = 0$. All coefficients of f are thus elements of (p). We can then write $f(Y) = pf'(Y)$ for some $f' \in R_t(d)[Y]_{\leq k-1}$. Note that for all $i : f'(a_i) \in (p)$, since $p \cdot f'(a_i) = f(a_i) = 0$ and all elements outside of (p) are units. Repeating the same argument for f', we conclude that $f(Y) = p^e g_e(Y)$ for any $e \geq 1$ and a corresponding $g_e \in R_t(d)[Y]_{\leq k-1}$. However, we can repeat this process until $e = d$, at which point $p^d = 0$ in $R_t(d)$. We can thus conclude that if $f(a_i) = 0$ for all i, then $f = 0$. Therefore, $\mathcal{E}_1$ is injective, and surjective by Lemma 5, thus an isomorphism.*

Just as in the regular field case, we may include the 'point at infinity' to extend the definition of $\mathcal{E}_1$.

Notation 5 *For any $m \in \mathbb{Z}_{>0}$, we expand the definition of the formal symbol ∞_m such that $f(\infty_m) := b_m$, for $f = \sum_{i=1}^{m} b_i X^i \in R_t(d)[X]_{\leq m}$.*

Lemma 7 *Let $1 \leq k \leq p^d + 1$ and $a_1, ..., a_k \in R_t(d) \cup \{\infty_{k-1}\}$ pairwise distinct such that $(a_i - a_j) \notin (p)$ for all $i, j \in \{1, ..., k\}, i \neq j, a_i \neq \infty_{k-1}, a_j \neq \infty_{k-1}$. Then*

$$\mathcal{E}_1 : R_t(d)[Y]_{k-1} \to R_t(d)^k \tag{10}$$

$$f \mapsto (f(a_1), ..., f(a_k)) \tag{11}$$

is an isomorphism of $R_t(d)$-modules.

Proof 10 *It remains to show bijectivity in the case where $\exists i : a_i = \infty_{k-1}$. Let $f \in R_t(d)[Y]_{\leq k-1}$ such that $\mathcal{E}_1(f) = 0$. Then f has degree $k-2$ and $k-1$ roots in distinct residue classes, thus, using the preceding, we conclude $f = 0$. Thus $\mathcal{E}_1$ is injective.*
Assume without loss of generality that $a_1 = \infty_{k-1}$. Let $(y_1, \ldots, y_k) \in R_t(d)^k$. Then set for $1 \leq i \leq k-1$, $y_i' = y_{i+1} - y_1 y_{i+1}^{k-1}$. Then, using Lagrange interpolation, we have a polynomial $g \in R_t(d)[Y]_{\leq k-2}$ such that for $1 \leq i \leq k-1 : g(a_{i+1}) = y_i'$. There is then a polynomial $g'(Y) = g(Y) + y_1 Y^{k-1}$ of degree at most $k-1$. We see that for $i \neq 1 : g'(a_i) = g(a_i) + y_1 a_i^{k-1} = y_i' + y_1 a_i^{k-1} = y_i - y_1 a_i^{k-1} + y_1 a_i^{k-1} = y_i$. Furthermore, $g'(\infty_{k-1}) = g'(a_1) = y_1$. Thus $\mathcal{E}_1(g') = (y_1, \ldots, y_k)$. We conclude that $\mathcal{E}_1$ is surjective. $\mathcal{E}_1$ is thus an isomorphism of $R_t(d)$-modules.

6.2 The Existence of a Generator

To continue our lift of the polynomial construction, we need a generator α such that $R_t(d)(\alpha) = R_t(dm)$. Let us first denote Hensel's lemma, which will be needed in our proof.

Lemma 8 (Hensel's Lemma) *Let (p) be a maximal ideal of a commutative ring R and let $f \in R[X]$ be a polynomial such that its highest degree term's coefficient a_d is not in (p). If $f = a_d g h$ modulo p for monic polynomials $g, h \in R[X]$ that are coprime modulo p, then for any $k \in \mathbb{Z}_{>0}$, there exist monic polynomials g_k, h_k such that*

$$\begin{aligned} f &\equiv a_d g_k h_k \mod p^k, \\ g_k &\equiv g \mod p, \\ h_k &\equiv h \mod p. \end{aligned}$$

Furthermore, g_k, h_k are unique modulo p^k.

This Lemma has a special case, which is the required result and also often called Hensel's lemma. When $g = X - r$ (or h), coprimality then gives us that r is a simple root of $f \mod p$. In this case, r can be lifted to a simple root $r_k \in R/(p^k)$ such that $r_k \equiv r \mod p$ and r_k is a simple root of $f \mod p^k$ (in other words $g_k = X - r_k$ (or h_k)).

Lemma 9 *Let $R_t(d)$ be a Galois ring over a prime p. Let $m \in \mathbb{Z}_{>0}$. $\exists \alpha \in R_t(dm)$ such that $R_t(dm) = R_t(d)(\alpha)$. In other words, there exists a generator α that extends $R_t(d)$ to $R_t(dm)$.*

Proof 11 *We know that $R_t(d)/(p) \cong \mathbb{F}_{p^d}$ and $R_t(dm)/(p) \cong \mathbb{F}_{p^{dm}}$. For the residue fields, we know that there exists $\alpha' \in \mathbb{F}_{p^{dm}}$ such that $\mathbb{F}_{p^{dm}} = \mathbb{F}_{p^d}(\alpha')$. Let $\bar{\mu}$ be the minimal polynomial of α' over $\mathbb{F}_{p^d}$. Then $\bar{\mu}$ is irreducible of degree m. We can lift $\bar{\mu}$ to a monic irreducible polynomial $\mu \in R_t(d)[X]$ of the same degree m, such that $\mu = \bar{\mu} \mod p$. Using Hensel's Lemma, specifically the special case where $g = X - \alpha'$, we then see that there exists an $\alpha \in R_t(dm)$, a root of μ, such that $R_t(d)(\alpha) \cong R_t(d)[X]/\mu(X)$. Thus $R_t(dm) = R_t(d)(\alpha)$, as both are free $R_t(d)$-modules of rank m.* $\square$

This gives us an isomorphism of $R_t(d)$-modules for any $1 \leq k \leq p^d$ and α such that $R_t(d(2k-1)) = R_t(d)(\alpha)$:

$$\mathcal{E}_2 : R_t(d)[Y]_{\leq 2k-2} \to R_t(d(2k-1)) \tag{12}$$
$$f \mapsto f(\alpha), \tag{13}$$

since α is a generator with minimal polynomial of degree $2k-1$.

6.3 In Conclusion

Claim 4 (Lemma 4) *Let $\phi := \mathcal{E}_2 \circ \mathcal{E}_1^{-1}$ and $\psi : \mathcal{E}_1' \circ \mathcal{E}_2^{-1}$, for $\mathcal{E}_1'$ the natural extension of $\mathcal{E}_1$ to domain $R_t(d)[Y]_{\leq 2k-2}$. Then (ϕ, ψ) is a $(k, 2k-1)_{p^t}$-RMFE.*

Proof 12 *ϕ, ψ are isomorphisms of $R_t(d)$-modules by construction. Let $x, y \in R_t(d)^k$. Then*

$$\psi(\phi(x) \cdot \phi(y)) = x * y$$

analogously to proof 6. □

We thus see that for any $1 \leq k \leq p^d$ there exists a $(k, 2k-1)_{p^t}$-RMFE over $R_t(d)$.

6.4 Practical Parameters

Escudero et al. [EXY22] are one of the first to present a protocol over more general rings $\mathbb{Z}/2^t\mathbb{Z}$. They use the asymptotic family of RMFEs $(\phi_m, \psi_m), m \in \mathbb{Z}_{>0}$ denoted as:

$$\phi_m : (\mathbb{Z}/2^t\mathbb{Z})^{k_m} \to R_t(m), \psi_m : R_t(m) \to (\mathbb{Z}/2^t\mathbb{Z})^{k_m}.$$

The existence of such a family has been proven by Cramer et al. in 2021 [CRX21]. In the paper by Escudero et al., they use explicit values: $(k_m, m) = (21, 65)$ and $(k_m, m) = (42, 135)$. Using the construction shown previously in this section, it is not necessary to understand the more complex lifting procedure shown in this paper. We can obtain RMFEs with these values as follows:

1. A $(21, 65)_{2^t}$-RMFE is obtained by concatenating a $(3, 5)_{2^t}$-RMFE with a $(7, 13)_{2^{5t}}$-RMFE over Galois rings.

2. A $(42, 135)_{2^t}$-RMFE is obtained by concatenating a $(3, 5)_{2^t}$-RMFE with a $(14, 27)_{2^{5t}}$-RMFE over Galois rings.

Naturally, we assume $42 \leq 2^t + 1$, as this is necessary for the construction. Note further that concatenation over Galois rings is possible, since we can identify $R_t(d)^{k_1 k_2}$ with $(R_t(d)^{k_1})^{k_2}$ just as naturally as we could for finite fields, thus Lemma 3 and its proof can be analogously applied to Galois rings. They assume further that $\phi((1, 1, ..., 1)) = 1$, thus providing $\psi(\phi(x)) = x$. This is not addressed in more detail by Escudero et al. [EXY22]. We will see later in Section 7 that there is a need to slightly change our construction to have this work.

7 Coral: A Practical Application of (R)MFEs

The first practical implementation of RMFE-based MPC protocols was provided with the name Coral [HLW+24] in 2024. They note that the usage of RMFEs is in fact limited, as implementations also make use of so called Multiplication Friendly Embeddings (MFEs), which have to be compatible with the applied RMFE. They note, referencing Escudero et al.'s paper from 2022 [EXY22]: "For instance, the compact $(21, 65)_2$-RMFE (also employed in their complexity analysis) with an expansion ratio of m/k $= 65/21 \approx 3.1$ can be constructed by concatenating $(3, 5)_2$-RMFE and $(7, 13)_{32}$-RMFE. Nonetheless, the construction in Section 2.2.3 demonstrates the absence of a compatible MFE for this RMFE." [HLW+24]. They show a table of example (R)MFE parameter sets, of which the best $(k, m)_2$-RMFE ratio is $m/k = 42/14 = 3$. There are more practical difficulties they ran into, such as having to explicitly construct an isomorphism $\mathbb{F}_{(2^{m_1})^{m_2}} \to \mathbb{F}_{2^{m_1 m_2}}$. Coral will be expanded upon in the remainder of this section. Note that many of their optimizations do not optimize communication complexity, which we considered in previous sections, but computation complexity, thus the number of operations needed to perform the MPC protocol. They tested this using benchmarking tools within the MP-SPDZ library [Kel20].
Coral is a framework with support for packed (Single Instruction Multiple Data) circuits and mixed (both boolean and arithmetic) circuits, where some operations corresponding to gates are performed over $\mathbb{F}_2$ and others over another finite field or Galois ring. It is based on the dishonest majority, active security model. For boolean computation, they build upon the same works this thesis has referenced previously, namely the works by Cascudo et al. in 2018 [CCXY18], Cascudo and Gundersen in 2020 [CG20] and Escudero et al. in 2022 [EXY22]. Before we can go into the specifics, we need to introduce a new mathematical tool, which might make the name RMFE appear more logical to the reader.

7.1 Multiplication Friendly Embeddings

Coral uses not only RMFEs, but also Multiplication Friendly Embeddings (MFEs), defined below.

Definition 9 *Let $\sigma : \mathbb{F}_{q^m} \to \mathbb{F}_q^t$ and $\rho : \mathbb{F}_q^t \to \mathbb{F}_{q^m}$ be $\mathbb{F}_q$-linear maps such that*

$$xy = \rho(\sigma(x) * \sigma(y)), \forall x, y \in \mathbb{F}_{q^m}, \tag{14}$$

then (σ, ρ) is a $(t, m)_q$-MFE.

Similarly to the case of RMFEs, we have the following result:

Lemma 10 ([CCCX09]) *Let $m \geq 2$ be an integer with $q \geq 2m-2$. then there exists a $(2m-1, m)_q$-MFE.*

The construction and proof are similar to the case of RMFEs. Let m, q as in the lemma. Let $\alpha \in \mathbb{F}_{q^m}$ such that $\mathbb{F}_{q^m} = \mathbb{F}_q(\alpha)$. Let $a_1, ..., a_{2m-2} \in \mathbb{F}_q$ be pairwise distinct. Let $\infty_{m-1}, \infty_{2m-2}$ be as in the construction of RMFEs. Define:

$$\mathcal{E}_3 : \mathbb{F}_q[X]_{\leq m-1} \to \mathbb{F}_{q^m}, f \mapsto f(\alpha) \tag{15}$$
$$\mathcal{E}_4 : \mathbb{F}_q[X]_{\leq m-1} \to \mathbb{F}_q^{2m-1}, f \mapsto (f(a_1), ..., f(a_{2m-2}), f(\infty_{m-1})) \tag{16}$$
$$\mathcal{E}_3' : \mathbb{F}_q[X]_{\leq 2m-2} \to \mathbb{F}_{q^m}, f \mapsto f(\alpha) \tag{17}$$
$$\mathcal{E}_4' : \mathbb{F}_q[X]_{\leq 2m-2} \to \mathbb{F}_q^{2m-1}, f \mapsto (f(a_1), ..., f(a_{2m-2}), f(\infty_{2m-2})) \tag{18}$$

Note that these maps are very similar and sometimes overlap with the maps $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_1'$ from the construction of RMFEs over fields. Set $\sigma := \mathcal{E}_4 \circ \mathcal{E}_3^{-1}$ and $\rho := \mathcal{E}_3' \circ (\mathcal{E}_4')^{-1}$. Then (σ, ρ) is a $(2m-1, m)_q$-MFE. The proof can be found in the original paper by Cascudo et al. from 2009 [CCCX09].
In the 2024 paper introducing Coral, it is claimed that MFEs can be concatenated similarly to RMFEs. They state that "a $(t_1, m_1)_q$-MFE and a $(t_2, m_2)_{q^{m_1}}$-MFE can be concatenated to produce a $(t_1 t_2, m_1 m_2)_q$-MFE" [HLW+24]. However, they do not provide a proof and they refer to [CG20], where only the case of RMFEs is proven. It is not hard to show that this statement is correct however.

Lemma 11 *Let (σ_1, ρ_1) be a $(t_1, m_1)_q$-MFE and let (σ_2, ρ_2) be a $(t_2, m_2)_{q^{m_1}}$-MFE. Define:*

$$\sigma : \mathbb{F}_{q^{m_1 m_2}} \to \mathbb{F}_q^{t_1 t_2}, x \mapsto (\sigma_1(u_1), ...\sigma_1(u_{t_2})), \sigma_2(x) = (u_1, ..., u_{t_2}) \tag{19}$$

$$\rho : \mathbb{F}_q^{t_1 t_2} \mapsto \mathbb{F}_{q^{m_1 m_2}}, (x_1^{(1)}, ..., x_1^{(t_1)}, x_2^{(1)}, ..., x_2^{(t_1)}, ..., x_{t_2}^{(t_1)}) \mapsto \rho_2(\rho_1(x_1^{(1)}, ..., x_1^{(t_1)}), ..., \rho_1(x_{t_2}^{(1)}, ..., x_{t_2}^{(t_1)})). \tag{20}$$

Then (σ, ρ) is a $(t_1 t_2, m_1 m_2)_q$-MFE.

Note that $\sigma_2 : \mathbb{F}_{(q^{m_1})^{m_2}} \to \mathbb{F}_q^{t_2}$, thus $\sigma : \mathbb{F}_{(q^{m_1})^{m_2}} \to (\mathbb{F}_q^{t_2})^{t_1}$. We can identify $\mathbb{F}_{(q^{m_1})^{m_2}} \cong \mathbb{F}_{q^{m_1 m_2}}$ and $(\mathbb{F}_q^{t_2})^{t_1} \cong \mathbb{F}_q^{t_1 t_2}$. For the second isomorphic relation, we can trivially view elements of $(\mathbb{F}_q^{t_2})^{t_1}$ as elements of $\mathbb{F}_q^{t_1 t_2}$, however, as briefly alluded to prior, we do need to define a specific isomorphism $\mathbb{F}_{(q^{m_1})^{m_2}} \to \mathbb{F}_{q^{m_1 m_2}}$ for practical application. This is briefly discussed later in this thesis. For now, assume that they are equal.

Proof 13 *Let $x, y \in \mathbb{F}_{q^{m_1 m_2}}$. Let $\sigma_2(x) = (u_1, ..., u_{t_2})$ and $\sigma_2(y) = (v_1, ..., v_{t_2})$ We have:*

$$\begin{aligned}
\rho(\sigma(x) * \sigma(y)) &= \rho((\sigma_1(u_1), ..., \sigma_1(u_{t_2})) * (\sigma_1(v_1), ..., \sigma_1(v_{t_2}))) \\
&= \rho((\sigma_1(u_1) * \sigma_1(v_1), ..., \sigma_1(u_{t_2}) * \sigma_1(v_{t_2}))) \\
&= \rho_2(\rho_1(\sigma_1(u_1) * \sigma_1(v_1)), ..., \rho_1(\sigma_1(u_{t_2}) * \sigma_1(v_{t_2}))) \\
= \rho_2(u_1 v_1, ..., u_{t_2} v_{t_2}) &= \rho_2((u_1, ...u_{t_2}) * (v_1, ..., v_{t_2})) = \rho_2(\sigma_2(x) * \sigma_2(y)) = xy.
\end{aligned}$$

Thus (ρ, σ) is indeed a $(t_1 t_2, m_1 m_2)_q$-MFE. □

7.2 Use of (R)MFEs in Coral

For boolean computation, the framework builds upon the previously best-performing (theoretical) RMFE-based protocol by Escudero et al. from 2022 [EXY22], which evaluates a circuit C over $\mathbb{Z}/p^t\mathbb{Z}$. Note that boolean computation is equivalent to setting $p^t = 2$, thus we are evaluating over $\mathbb{Z}/2\mathbb{Z} = \mathbb{F}_2$. For a $(k, m)_2$-RMFE (ϕ, ψ), similarly to Section 4, $\mathcal{F}_C'(\mathbb{F}_{2^m})$ is computed with inputs $\phi(x_i)$, for inputs x_i of P_i. After computation, the outputs are retrieved by applying ψ instead of ϕ^{-1}. Note that this construction yields correct results, since it is assumed that $\phi((1, ..., 1)) = 1$ (thus $\forall x \in \mathbb{F}_2^k : \psi(\phi(x)) = x$), just like in the original protocol by Escudero et al. In this paper [HLW+24], it is, in contrast to the paper by Escudero et al., actually substantiated that such an RMFE, with $\phi((1, ..., 1)) = 1$, exists. Instead of constructing using the $(k-1)$-th degree term as an extra interpolation point, they construct without an extra interpolation point and require that one of the a_i is 0 (this requirement is not necessary in our form of construction, but in their form they

replace the $(k-1)$-th degree term with the constant term). This, of course, does mean that the requirement shifts to $k \leq q$ from $k \leq q+1$, but this is a strong enough result for this application. This indeed gives us that $\phi((1,...,1)) = 1$ trivially, as the constant polynomial 1 now evaluates to 1 in all interpolation points, where-as previously we had for $p(X) = 1 \in \mathbb{F}_q[X] : p(\infty_{k-1}) = 0$. Similarly to previous applications, we need a 'repacking operator' $\tau := \phi \circ \psi$, which is applied after every multiplication gate.

MFEs are used only in the pre-processing phase. Similarly to the SPDZ approach, an input, which is embedded using an RMFE (ϕ, ψ) such that $\phi(x) \in \mathbb{F}_{2^m}$, is authenticated using a MAC key $\alpha \in \mathbb{F}_{2^m}$. An additive sharing is created of $\alpha \cdot \phi(x)$. A more efficient way to generate these sharings was introduced by Escudero et al. in 2022 [EXY22]. Essentially, they used an MFE (σ, ρ) to parallelize the calculation and compute an additive sharing coordinate-wise of $\sigma(\alpha) * \sigma(\phi(x)) \in \mathbb{F}_2^t$, which is decoded back to a sharing in $\mathbb{F}_{2^m}$. This turns out to be more efficient. The communication cost drops from $O(m^2)$ to $O(t)$. The interested reader is referred to Section 4.1 in the paper by Escudero et al. [EXY22], as this requires understanding of Oblvious Transfer (OT) and Oblivious Linear Evaluation (OLE), which is beyond the scope of this thesis.

By the preceding, it is clear that RMFE output should align with MFE input. This is one of the main contributions of the Coral framework. The Coral framework is, similarly to previous works, mainly focused on boolean computation, thus the case that $q = 2$. In Lemma 10, when taking $q = 2$, we must have $m = 2$, as $m \geq 2$ and $q \geq 2m - 2$. Thus, since all MFEs are constructed as a finite concatenation of this MFE, if we have any $(t, m)_2$-MFE, then m is even. This leads to what was already stated about compatibility at the start of this section. There exists a $(21, 65)_2$-RMFE, but, since 65 is odd, it is not compatible with any MFE over $q = 2$. Applying such compatible combinations of RMFEs and MFEs, is one of the efficiency gains of the framework.

There is one last unaddressed issue for practical application of (R)MFEs, which is the need for an explicit construction of an isomorphism between $\mathbb{F}_{2^{m_1 m_2}}$ and $\mathbb{F}_{(2^{m_1})^{m_2}}$. These are only isomorphic, but assumed equal in the construction. For this, they use a method introduced by Sunar et al. in 2003 [SSK03], which generates a conversion matrix $M \in \{0,1\}^{m \times m}$, which allows computation of $x \in \mathbb{F}_{2^{m_1 m_2}}$ via $x = Mx', x' \in \mathbb{F}_{(2^{m_1})^{m_2}}$. This thesis will not go into detail on this construction.

There is another practical improvement the framework makes. By setting $\alpha = X \mod P(X)$, for P the generating (irreducible) polynomial of the extension field $\mathbb{F}_{q^m}$, a basis $\{1, X, ..., X^{m-1}\}$ is obtained of $\mathbb{F}_{q^m}$. In this setting, by abusing the fact that we can now treat elements of $\mathbb{F}_{q^m}$ as elements of $\mathbb{F}_q[X]_{\leq m-1}$, the use of $\mathcal{E}_3$ and $\mathcal{E}_2$ does not require any computation and thus can be left out in practice. Note, further, that we use the altered construction of $(k, 2k)_2$-RMFEs without ∞_{k-1} and their concatenations, instead of $(k, 2k-1)_2$, thus now giving us, for $a_1, ..., a_{k-1} \in \mathbb{F}_q$:

$$\phi := \mathcal{E}_1^{-1}, \mathcal{E}_1 : \mathbb{F}_q[X]_{\leq k-1} \to \mathbb{F}_q^k, f \mapsto (f(0), f(a_1), ..., f(a_{k-1})) \tag{21}$$

$$\psi := \mathcal{E}_1' : \mathbb{F}_q[X]_{\leq 2k-1} \to \mathbb{F}_q^k, f \mapsto (f(0), f(a_1), ..., f(a_{k-1})) \tag{22}$$

and for our MFEs:

$$\sigma := \mathcal{E}_4 \tag{23}$$

$$\rho := \mathcal{E}_3' \circ (\mathcal{E}_4')^{-1} \tag{24}$$

with $\mathcal{E}_3', \mathcal{E}_4, \mathcal{E}_4'$ as under Lemma 10.

7.3 Non-mathematical Optimizations

Since the implementation requires extensive calling of mappings, a lot of computations over common small binary fields are performed. Coral applies pre-computation for this, which increases efficiency greatly. Furthermore, for smaller-sized input vectors, look-up tables are used, where-as when look-up table capacity is exceeded, a hash-based cache is used. These two optimizations, together with the more optimized mapping shown in Section 7.2, massively increased throughput (all operations per second) when testing in the NTL library [Sho23] by 111 times for what they call RMFE decryption, which is taken to mean the operations done when applying ψ, to 794 times for MFE encryption, taken to mean the operations done while applying ρ, as shown in Table 3 of the Coral paper [HLW+24].

7.4 Offline Optimizations

In the pre-processing phase, there are also some notable optimizations. Previously, Escudero et al. [EXY22] introduced multiplication quintuples $([a], [b], [\tau(a)], [\tau(b)], [\tau(a)\tau(b)])$, for some secret sharing scheme $[\cdot]$, for efficient and secure computation of multiplication gates. In the Coral framework, different quintuples are generated, namely $([a], [b], [\phi(\psi(a) * \psi(b)], [\tau(a)], [\tau(b)])$.
The generation of such quintuples is more efficient and also uses an important paradigm of MPC, so-called Oblivious Transfer (OT), as a black box, which gives access to a recent advance in this area, so-called VOLE [YWL+20] instead of the previously most used IKNP [IKNP03]. Neither of them will be addressed in this thesis, but VOLE is more recent and generally more efficient [HLW+24].
Of course, the use of different quintuples necessitates an updated online protocol. Multiplication in the framework is performed as follows. Let $[x], [y]$ be authenticated sharings. We need the output $[z]$ such that $\psi(z) = \psi(x) * \psi(y)$. First, we take one of our pre-processed multiplication quintuples $([a], [b], [\phi(\psi(a) * \psi(b)], [\tau(a)], [\tau(b)])$.
We compute:

$$[d] \leftarrow [x] - [a] \text{ and } [e] \leftarrow [y] - [b].$$

We open $d \leftarrow [d], e \leftarrow [e]$. Then we compute:

$$[z] \leftarrow \tau(d)[\tau(b)] + \tau(e)[\tau(a)] + \tau(d)\tau(e) + [\phi(\psi(a) * \psi(b))].$$

We see:

$$\begin{aligned}\psi(z) &= \psi(\tau(d)\tau(b) + \tau(e)\tau(a) + \tau(d)\tau(e) + \phi(\psi(a) * \psi(b))) \\ &= \psi(\tau(d)\tau(b)) + \psi(\tau(e)\tau(a)) + \psi(\tau(d)\tau(e)) + \psi(\phi(\psi(a) * \psi(b)))\end{aligned}$$

and since $\tau = \phi \circ \psi$, this is equal to

$$\begin{aligned}&= \psi(d) * \psi(b) + \psi(e) * \psi(a) + \psi(d) * \psi(e) + \psi(a) * \psi(b) \\ &= \psi(x-a) * \psi(b) + \psi(y-b) * \psi(a) + \psi(x-a) * \psi(y-b) + \psi(a) * \psi(b) \\ &= \psi(x) * \psi(b) - \psi(a) * \psi(b) + \psi(y) * \psi(a) - \psi(b) * \psi(a) + (\psi(x) - \psi(a)) * (\psi(y) - \psi(b)) + \psi(a) * \psi(b) \\ &= \psi(x) * \psi(b) + \psi(y) * \psi(a) - \psi(b) * \psi(a) + \psi(x) * \psi(y) - \psi(a) * \psi(y) - \psi(x) * \psi(b) + \psi(a) * \psi(b) \\ &= \psi(x) * \psi(y).\end{aligned}$$

Thus indeed this outputs $[z]$ such that $\psi(z) = \psi(x) * \psi(y)$ as needed.
The Coral framework introduces some more pre-processing, improving online efficiency further and/or extending the usage to more general ways of computing circuits. The framework works not just over boolean circuits, but also over mixed circuits. For this, a variation on daBits and edaBits is introduced, which uses RMFEs.
A daBit (Double Authenticated Bit) is traditionally a tuple $([b]_2, [b]_q)$ of two authenticated sharings of a bit $b \in \mathbb{F}_2$ respectively shared over $\mathbb{F}_2$ and over $\mathbb{F}_q$ with $q \neq 2$, which can be used to switch from binary to arithmetic in the following manner. Let $[x]_2$ be a shared secret in $\mathbb{F}_2$. Set $[z]_2 := [x]_2 + [b]_2$. Open $z \leftarrow [z]_2$ and view it as an element of $\mathbb{F}_q$. Then we compute:

$$[x]_q \leftarrow z - [b]_q - 2z[b]_q.$$

We see that indeed $[x]_q$ is now a sharing of x over $\mathbb{F}_q$. An edaBit (Extended Double Authenticated Bit) works similarly, except it is a tuple $([r]_q, [r_1]_2, ..., [r_l]_2)$ where the $r_i \in \mathbb{F}_2$ are all l separate bits of an element $r \in \mathbb{F}_{2^l}$. In other words, we have $r = \sum_{i=1}^{l} r_i 2^{i-1}$.
In Coral, both daBits and edaBits exist as packed versions. Take a vector $r \in \mathbb{F}_2^k$, then a packed daBit is a tuple of a vector of authenticated sharings and $\phi(r)$. Similarly, a packed edaBit in Coral is equivalent to k plain edaBits.
Of course, there is a standard way of generating these daBits and edaBits. It is possible to easily generate these packed (e)daBits by constructing the standard way and then converting the boolean sharings to RMFE sharings. This however has high cost. They thus construct it differently, which is another contribution of the paper. This will not be elaborated on in this thesis; the interested reader is referred to the appendix of the paper introducing Coral [HLW+24].
Another extension the Coral framework provides is the handling of linear combinations with constants in $\mathbb{F}_{2^m}$. As we have seen previously, we run into the fact that both ϕ, ψ are $\mathbb{F}_2$-linear, but not $\mathbb{F}_{2^m}$-linear. For this, they use a so-called 'encoding pair', which is a tuple $([r], [\tau(r)])$ of authenticated sharings. This allows computation of linear combinations as follows. Let $[x]$ be an authenticated sharing and let $c \in \mathbb{F}_{2^m}$. We need output $[z]$ such that $\psi(z) = \psi(x) * c$. Take one of our encoding pairs $([r], [\tau(r)])$. Compute $[d] \leftarrow [x] - [r]$ and open $d \leftarrow [d]$. Then we see:

$$[z] \leftarrow (\tau(d) + [\tau(r)]) \cdot \phi(c).$$

Indeed, $\psi(z) = \psi\Big(\big(\tau(d) + \tau(r)\big) \cdot \phi(c)\Big) = \psi\Big(\big(\tau(x - r) + \tau(r)\big) \cdot \phi(c)\Big) = \psi\big(\phi(\psi(x)) \cdot \phi(c)\big) = \psi(x) * c.$
Naturally, this is somewhat costly, as it requires a communication round (opening d). However, as Huang et al. note, it "is critical for common computations. The additional expense is justified by the advantages of RMFE-based MPC and is marginal for large-scale circuit computations." [HLW+24].

8 Evaluation

Let us summarize the use-cases of RMFEs as specified in the four papers previously discussed in this thesis [CCXY18, CG20, EXY22, HLW+24]. In this first paper by Cascudo et al. [CCXY18], RMFEs were used to reduce communication overhead for honest majority, active security protocols over binary circuits that use Shamir's secret sharing over large finite fields. This was to reduce the efficiency loss that was incurred by the need to view bits in the computation as elements of a larger

field. This was explained in Section 4. In the second paper by Cascudo and Gundersen from 2020 [CG20], a framework for dishonest majority was introduced, using a combination of RMFEs and SPDZ-style MACs. However, this still required parties to re-apply the RMFE (applying τ) for every multiplication, which led to two extra communication rounds for every multiplication. In general, for secret sharing-based MPC protocols without an extra overhead like re-encoding, there is a need for only one communication round per multiplication gate. Escudero et al. in 2022 [EXY22] then introduced quintuples in place of the Beaver triples to save one of the two extra communication rounds introduced in Cascudo and Gundersen and used MFEs for more efficient pre-processing, using the generalization to Galois rings from finite fields proven by Cramer et al. in 2021 [CRX21]. Coral [HLW+24] in 2024 then made the first practical implementation based on the protocol by Escudero et al., making improvements. These improvements, as discussed in Section 7, consist of adding the use of RMFE-based daBits and edaBits, bridging the gap between MFEs and RMFEs (i.e. showing that there is a need for them to be compatible and showing efficient combinations) and choosing a suitable representation of values, improving the construction and application of RMFEs and MFEs. Other smaller practical optimizations are made and the paper addresses some practical necessities, such as a concrete map between $\mathbb{F}_{q^{m_1 m_2}}$ and $\mathbb{F}_{(q^{m_1})^{m_2}}$.

Coral [HLW+24] is shown to be more efficient than the leading practical implementation, based on the work of Frederiksen et al. in 2015 [FKOS15], implemented in the MP-SPDZ library [Kel20]. This style of protocol is called Tinier. This implementation was chosen as a comparison, as it has an easily accessible practical implementation and Coral was integrated into the MP-SPDZ library. In the pre-processing phase, as shown in Table 3 of [HLW+24], with a statistical security parameter of $\lambda = 128$ and choices for $k = 14, t = 195$, they compared the communication complexity of the coral framework with this Tinier protocol, which makes heavy use of Oblivious Transfer. The pre-processing phase of such a protocol is referred to as TinyOT. They compared the two most expensive sub-modules, input authentication and triple/quintuple generation. For input authentication, communication complexity of the coral framework in bits is mentioned to be $t/k \approx 13.9$, while the previous best implementation by Escudero et al., even when using VOLE-OT instead of IKNP-OT, has communication complexity for input authentication of $2t/k \approx 27.9$. As mentioned earlier, this thesis does not go into detail about the differences between VOLE-OT and IKNP-OT. It is only necessary to understand that VOLE-OT is more efficient [YWL+20]. The protocols by Escudero et al. using IKNP-OT ($(\lambda+1)t/k \approx 1796.8$) and another protocol called Tiny ($\lambda = 128$), which uses the triple generation technique introduced by Frederiksen et al. [FKOS15], but uses input authentication from Spdz2k [CDE+18], are several times less efficient here. For the triple/quintuple generation, parameters called bucketing parameters are needed. This thesis has not discussed this procedure. The interested reader is referred to Section 4.2 in [HLW+24]. Taking these bucketing parameters as $B = 3$ or 4 and $B_1 = B_2 = 3$, the communication complexities in bits for triple/quintuple generation are presented in Table 1. Here, Coral is more efficient once again. Note, however, that the results for the quintuples are amortized. These are the results divided by the packing size k of the (k, m)-RMFE.

Coral was also compared to the Tinier protocol, including both the pre-processing and the online phase, for the following four circuits: Hamming distance between two 1024-bit strings using an O(1024) size circuit, AES circuit, SHA256 circuit and Bitonic sorting on an array of 128 elements of 32 bits. These were chosen based on a 2017 paper by Wang et al. [WRK17]. Coral has around

Protocol	Communication Complexity (bits)
Tiny [FKOS15]	$6B^2\lambda = 6912$ or 12288
Escudero with IKNP-OT [EXY22], [IKNP03]	$12tB_1B_2^2(\lambda+1)/k \approx 582158.6$
Escudero with VOLE-OT [EXY22], [YWL+20]	$24tB_1B_2^2/k \approx 9025.7$
Coral [HLW+24]	$2(\lambda+2)B + 10t/k \approx 919.3$ or 1179.3

Table 1: Communication complexity in bits for the triple/quintuple generation sub-module in different MPC protocols

10 times less communication complexity compared to the Tinier protocol. When LAN and WAN runtime were compared, Coral reduced WAN runtime by about 6 times, while LAN runtime was reduced by less than 1.5 times for all four circuits. Once again, these are amortized results. They also compared the mixed-circuit functionality of Coral with a Tinier-based protocol, which had similar results, with a little less improvement (communication improvement ranging from 1.6 to 2.9x, LAN from 1.1 to 2.9 and WAN from 2.7 to 4.9).

RMFE-based MPC protocols can thus be concluded to be generally more efficient, but only in an amortized setting. The results are amortized over 140 instances. This is to reduce the waste of buffered processing material. Unfortunately, no information is given on results or expected results with other numbers of instances. When using binary circuits over a malicious security model and when an amortized setting is applicable, RMFE-based protocols can thus possibly be used to increase efficiency. However, it is necessary to first experiment with amortization over other numbers of instances. For mixed circuits, RMFE-based protocols are shown to be the most efficient as well. The improvement is less than for strictly binary circuits. For this reason, it is advisable that research for different methods continues. It is feasible that for specific mixed-circuit implementations, depending on the structure of the mixed circuit, RMFE-based protocols are not the most efficient option. Efficiency also depends on the throughput of edaBit generation, which in turn depends on available threads/hardware and whether LAN or WAN is applicable, as mentioned by the authors of Coral "In LAN, Coral underperforms with a single thread but multithreading mitigates this disadvantage" [HLW+24]. For honest majority, RMFEs remain a theoretical idea by Cascudo et al. in 2018 [CCXY18] without practical implementation, remaining a topic of further research. For purely arithmetic circuits, there are no practically applicable efficient RMFE-based protocols in the literature as of yet. This is because the main advantage of RMFEs has been reducing overhead incurred by embedding bits into larger fields. It is possible that there is a use for RMFEs in arithmetic circuits where field size is less than n, as embedding into a field of size larger than n then might have benefits similar to the benefits first found by Cascudo et al. in 2018 [CCXY18] described in Section 4. This is left as future research. For arithmetic circuits over fields with size larger or equal to n, it is not apparent where the main benefit of RMFEs could be applied. The author of this thesis would thus not advise the use of RMFEs for (research of) this type of protocol, unless prompted by another discovery requiring embedding to an even larger field. As for which parameters to use, it is desirable to decrease the ratios m/k and t/m, representing the blow-up of field/Galois ring sizes for the used (R)MFEs. However, note that m must be even and there is a trade-off between larger values of k (thus a need for a circuit with possibility to parallelize to a higher degree and/or the need for amortization over a larger number of instances) and smaller ratios. This trade-off should be carefully considered when researching RMFE-based protocols.

References

[BMN18] A. R. Block, H. K. Maji, and H. H. Nguyen. Secure computation with constant communication overhead using multiplication embeddings. Cryptology ePrint Archive, Paper 2018/395, 2018.

[BTH08] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-secure MPC with linear communication complexity. *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21*, page 213–230, 2008.

[CCCX09] I. Cascudo, H. Chen, R. Cramer, and C. Xing. Asymptotically good ideal linear secret sharing with strong multiplication over any fixed finite field. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 466–486, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[CCXY18] I. Cascudo, R. Cramer, C. Xing, and C. Yuan. Amortized complexity of information-theoretically secure MPC revisited. *Annual International Cryptology Conference*, page 395–426, 2018.

[CDE+18] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. SPDZ2k: Efficient MPC mod 2^k for dishonest majority. Cryptology ePrint Archive, Paper 2018/482, 2018.

[CG20] I. Cascudo and J. S. Gundersen. A secret-sharing based MPC protocol for boolean circuits with good amortized complexity. *TCC*, page 652–682, 2020.

[CRX21] R. Cramer, M. Rambaud, and C. Xing. Asymptotically-good arithmetic secret sharing over $\mathbb{Z}/p^\ell\mathbb{Z}$ with strong multiplication and its applications to efficient MPC. *CRYPTO*, 2021.

[CSTA19] J. Cartlidge, N. P. Smart, and Y. Talibi Alaoui. MPC joins the dark side. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, page 148–159, New York, NY, USA, 2019. Association for Computing Machinery.

[DKL+13] I. Damgård, M. Keller, E. Larraia, V. Pastro1, P. Scholl, and N. Smart. Practical covertly secure MPC for dishonest majority – Or: Breaking the SPDZ limits. In *Crampton, J., Jajodia, S., Mayes, K. (eds) Computer Security – ESORICS 2013. ESORICS 2013. Lecture Notes in Computer Science, vol 8134*, 2013.

[DPSZ12] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. *Advances in Cryptology– CRYPTO 2012*, page 643–662, 2012.

[EKR22] D. Evans, V. Kolesnikov, and M. Rosulek. *A Pragmatic Introduction to Secure Multi-Party Computation*. NOW Publishers, 2022.

[EXY22] D. Escudero, C. Xing, and C. Yuan. More efficient dishonest majority secure computation over $\mathbb{Z}_{2^k}$ via Galois rings. *CRYPTO*, page 383–412, 2022.

[FKOS15] T. K. Frederiksen, M. Keller, E. Orsini, and P. Scholl. A unified approach to MPC with preprocessing using OT. In *ASIACRYPT (1) (Lecture Notes in Computer Science, Vol. 9452)*, page 711–735. Springer, 2015.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[Hir01] M. Hirt. *Multi party computation: Efficient protocols, general adversaries, and voting.* Hartung-Gorre, 2001.

[HLW+24] Z. Huang, W. Lu, Y. Wang, C. Hong, T. Wei, and W. Chen. Coral: Maliciously secure computation framework for packed and mixed circuits. Cryptology ePrint Archive, Paper 2024/1372, 2024.

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 145–161, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Kel20] M. Keller. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 1575–1590, New York, NY, USA, 2020. Association for Computing Machinery.

[KMS+16] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 839–858, 2016.

[Liu24] T. Liu. Research on privacy techniques based on multi-party secure computation. In *2024 3rd International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS)*, pages 912–917, 2024.

[NBSK15] Divya G. N., V. P. Binu, and G. Santhosh Kumar. An improved e-voting scheme using secret sharing based secure multi-party computation. *CoRR*, abs/1502.07469, 2015.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM, Volume 22, Issue 11*, pages 612–613, 1979.

[Sho23] V. Shoup. NTL: A library for doing number theory., 2023. https://libntl.org/.

[SSK03] B. Sunar, E. Savas, and Ç. K. Koç. Constructing composite field representations for efficient conversion. *IEEE TRANSACTIONS ON COMPUTERS*, 52:1391 – 1398, 2003.

[WRK17] X. Wang, S. Ranellucci, and J. Katz. Authenticated garbling and efficient maliciously secure two-party computation. Cryptology ePrint Archive, Paper 2017/030, 2017.

[Yao82] A. Yao. Protocols for secure computations (extended abstract). *23rd Annual Symposium on Foundations of Computer Science. IEEE Computer Society Press.*, pages 160–164, 1982.

[YWL+20] K. Yang, C. Weng, X. Lan, J. Zhang, and X. Wang. Ferret: Fast extension for coRRElated oT with small communication. Cryptology ePrint Archive, Paper 2020/924, 2020.

[Zap22] S. Zapechnikov. Secure multi-party computations for privacy-preserving machine learning. *Procedia Computer Science*, 213:523–527, 2022. 2022 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: The 13th Annual Meeting of the BICA Society.