On the Interaction Between Linear Codes, Secret Sharing, and Multiparty Computation

Gundersen, Jaron Skovsted

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

ON THE INTERACTION BETWEEN LINEAR CODES, SECRET SHARING, AND MULTIPARTY COMPUTATION

BY
JARON SKOVSTED GUNDERSEN

DISSERTATION SUBMITTED 2020

AALBORG UNIVERSITY
DENMARK

On the Interaction Between Linear Codes, Secret Sharing, and Multiparty Computation

Ph.D. Thesis

Jaron Skovsted Gundersen

Thesis submitted October, 2020

Dissertation submitted: October, 2020

PhD supervisors: Professor Horia Cornean
Aalborg University

Assistant Professor Ignacio Cascudo
IMDEA Software Institute

Ramón-y-Cajal fellow Diego Ruano
Universidad de Valladolid

Professor Olav Geil
Aalborg University

PhD committee: Professor Lars Døvling Andersen (chairman)
Aalborg University

Research Expert Emmanuela Orsini
KU Leuven

Associate Professor Claudi Orlandi
Aarhus University

PhD Series: Faculty of Engineering and Science, Aalborg University

Department: Department of Mathematical Sciences

ISSN (online): 2446-1636
ISBN (online): 978-87-7210-826-1

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2020

Abstract

This thesis is based on four articles on the interaction between linear codes, secret sharing and multiparty computation.

The focus through my Ph.D. has been on achieving new results in these three areas but I have mainly focussed on multiparty computation where the computations take place in a small finite field. This is for instance reflected directly in Paper D where I provide a new efficient protocol for multiparty computation over the binary field. Even though Paper A and C are not directly focusing on multiparty computation the results in these papers are still very relevant for multiparty computation over small finite fields since the papers are considering secret sharing and linear codes; two tools that are often used in multiparty computation. In this context it is worth noticing that, when using linear codes and secret sharing schemes in multiparty computation, the finite field for the codes and the schemes can have a great impact on the efficiency of the multiparty computation protocol. Hence, when I show limitations for secret sharing schemes over small finite fields in Paper A this implies limitations for multiparty computation protocols over small fields, and similarly when I construct good linear codes over small fields in Paper C this can increase efficiency of multiparty computation protocols.

In all four articles, linear codes will be used in one form or another, even if the main focus is on either secret sharing or multiparty computation. Therefore, I also start Part I with an introduction to linear codes. Afterwards, I introduce secret sharing and multiparty computation.

In Part II, the four articles are presented. The first deals with limitations on what one can achieve in terms of privacy and reconstruction in secret sharing. Since secret sharing is used in multiple protocols for multiparty computation these results also provide some limitations for those protocols.

The second article uses linear codes as a tool to achieve a certain functionality known as oblivious transfer in multiparty computation. We show that it is possible to use codes over other fields than previously used, which gives more flexibility and sometimes saves complexity.

The third article considers a particular type of linear codes called matrix-product codes. A matrix-product code is a certain type of code constructed

from shorter constituent codes. If the constituent codes have good parameters the longer matrix-product code is also guaranteed to have good parameters. We show that in some cases one can describe the "square" of the matrix-product code which may be necessary if one wants to use the codes in certain multiparty computation contexts.

The fourth and final article provides a new efficient multiparty computation protocol over the binary field. This is probably the article where linear codes are most absent but since the protocol uses a linear secret sharing scheme, linear codes are included indirectly. This shows very well that these three areas are very connected and that the connection between them is worth studying in more details.

Resumé

Denne afhandling bygger på fire artikler omhandlende samspillet mellem lineære koder, secret sharing og multiparty computation.

Fokusset gennem min Ph.D. har været på at opnå nye resultater indenfor disse tre områder, men jeg har hovedsageligt haft for øje, at opnå resultater om multiparty computation hvor beregningerne foregår i et lille endeligt legeme. Dette afspejles for eksempel direkte i Paper D, hvor jeg giver en ny effektiv protokol for multiparty computation over det binære legeme. Selvom Paper A og C ikke direkte fokuserer på multiparty computation, er resultaterne i disse artikler stadig meget relevante for multiparty computation over små endelige legemer, da artiklerne omhandler secret sharing og lineære koder; to værktøjer, der ofte bruges i multiparty computation. I denne sammenhæng er det værd at bemærke, at når du bruger lineære koder og secret sharing schemes i multiparty computation, kan det endelige legeme for koderne og schemes'ne have stor indflydelse på effektiviteten af multiparty computation-protokollerne. Derfor når jeg viser begrænsninger for secret sharing schemes over små endelige legemer i Paper A medfører dette begrænsninger for multiparty computation-protokoller over små legemer, og ligeledes når jeg konstruerer gode lineære koder over små legemer i Paper C, kan dette øge effektiviteten af multiparty computation-protokoller.

I alle fire artikler vil der i en eller anden form være anvendt lineære koder også selv om hovedfokusset er på enten secret sharing eller multiparty computation. Derfor starter jeg også med i Part I at give en introduktion til linære koder. Herefter introducerer jeg secret sharing og multiparty computation.

I Part II er de fire artikler præsenteret. Den første omhandler begrænsninger for hvad man kan opnå i forbindelse med privathed og rekonstruktion i secret sharing. Eftersom secret sharing benyttes i flere protokoller for multiparty computation giver disse resultater også nogle begrænsninger for disse protokoller.

Den anden artikel benytter lineære koder som et redskab til at opnå en bestemt funktionalitet kendt som oblivious transfer i multiparty computation. Vi viser, at det er muligt at benytte koder over andre legemer end der tidligere er brugt, hvilket giver mere fleksibilitet og nogle gange sparer kom-

pleksitet.

Den tredje artikel betragter en bestemt type lineære koder kaldet matrix-produkt koder. En matrix-produkt-kode er en bestemt type kode konstrueret ud fra kortere koder. Hvis de kortere koder benyttet i konstruktionen har gode parametre, garanteres det også, at den længere matrix-produkt-kode vil have gode parametre. Vi viser, at man i visse tilfælde kan beskrive "kvadratet" af matrix-produkt-koden, hvilket kan være nødvendigt, hvis man vil bruge koderne i visse multiparty computation-sammenhænge.

Den fjerde og sidste artikel giver en ny effektiv multiparty computation-protokol over det binære legeme. Dette er nok den artikel hvor lineære koder er mest fraværende, men eftersom protokollen benytter sig af et lineært secret sharing scheme er der indirekte lineære koder inkluderet herigennem. Dette viser meget godt, at disse tre områder hænger meget sammen og at sammenhængen mellem dem er værd at studere nærmere.

Contents

Preface

This thesis is a collection of papers I have written with my co-authors during my Ph.D. at the Department of Mathematics, Aalborg University.

The thesis is in two parts. The second part consists of four papers I have produced during my Ph.D. study. The first and third paper are published in peer-reviewed journals, the second paper is presented at the "Security and Cryptography for Networks 2018" conference and the fourth will be presented at the "Theory of Cryptography Conference 2020". Both the second and the fourth paper are published in the book series "Lecture notes in Computer Science" by Springer:

A I. Cascudo, J. S. Gundersen and D. Ruano, "Improved Bounds on the Threshold Gap in Ramp Secret Sharing," in *IEEE – Transactions on Information Theory*, 2019.

B I. Cascudo, R. B. Christensen and J. S. Gundersen, "Actively Secure OT-Extension from q-ary Linear Codes," in *Security and Cryptography for Networks — SCN 18*, Springer International Publishing, 2018.

C I. Cascudo, J. S. Gundersen and D. Ruano, "Squares of Matrix-product Codes," in *Finite Fields and Their Applications*, 2020.

D I. Cascudo and J. S. Gundersen, "A Secret-Sharing Based MPC Protocol for Boolean Circuits with Good Amortized Complexity," in *Theory of Cryptography — TCC 20*, Springer International Publishing, 2020.

All the papers consider the interactions between three theoretical areas; Linear codes, secret sharing and multiparty computation. Hence, the first part of the thesis gives small introductions to these three areas. Furthermore, I indicate how these areas are related and rely on each other. At the end of Part I, I put the papers from Part II into the theoretical context described in Part I.

Acknowledgement

I would like to thank my family for supporting me through my Ph.D. study both at home but also on my stay abroad. I appreciate that you went with me and that we also could make a home abroad so I did not have to miss you during my stay.

In this context I am also grateful to CWI and the employees at the Cryptology group for letting me visit your place and for welcoming me.

Another thanks goes to my friends and my colleagues, especially René Bødker Christensen. Thanks for all your time and our discussions. You have helped me several times during the four years.

At last I would like to thanks my supervisors. Olav Geil and Diego Ruano, I am grateful for your support and guidance and Horia Cornean thanks for stepping in when I needed a new supervisor. To Ignacio Cascudo I will say that I am grateful for all the time you have spent on reading and commenting drafts. You have been a great support and I appreciate that you still took your time to supervise me even after you left Aalborg University. I am also grateful for my visit at your new workplace at IMDEA Software Institute. I think that we had some good days and fruitful discussions.

Jaron Skovsted Gundersen
Aalborg University, October 14, 2020

Part I

Introduction

Background

All the papers in Part II are related to cryptography and more specific secure computation. The majority of the papers consider or are related to secure multiparty computation over small fields. As you will see in the papers the main building blocks rely on linear codes and secret sharing, two concepts that are strongly related. Therefore, I will start by giving small introductions to linear codes and secret sharing in Part I. After these introductions I will end Part I with a presentation of some of the main concepts in secure multiparty computation and an overview of the main results in the papers in Part II.

1 Linear Codes

In his paper "A Mathematical Theory of Communication" from 1948, [25], Shannon built the foundation of what today is known as information theory. One of the main motivations for information theory was to study communication and within this area a lot of research has been done on error-correcting codes. Below I will give a small introduction to error-correcting codes. For a deeper understanding of these concepts the reader is referred to [16].

The classical use of error-correcting codes is to improve the reliability of communication in situations where the communication is through noisy channels. If the sender only sends the message some information gets lost because of the errors which occurs due to the noise. But if the sender uses an error-correcting code the receiver is able to correct the errors if not too many errors had occurred. The majority of error-correcting codes used in practice are linear codes.

Generally, linear codes are subspaces of a vector space over a finite field. Typically, we denote a finite field with q elements as $\mathbb{F}_q$ and hence a linear code is a subspace $C \subseteq \mathbb{F}_q^n$ where n is called the length of the code. The dimension k of C as a subspace of $\mathbb{F}_q^n$ is also said to be the dimension of the code and we use the notation $[n,k]_q$ for C to tell which field the code is over and what length and dimension it has.

Because C is a k-dimensional subspace one can fix a basis consisting of k

vectors (or codewords) in $\mathbb{F}_q^n$. Collecting those vectors as rows in a matrix we obtain what is called a generator matrix for the code C. A generator matrix G for a linear code C is often used to encode the message $\mathbf{m} \in \mathbb{F}_q^k$ by computing the matrix-vector product $\mathbf{c} = \mathbf{m}G$. Since G has full rank the map represented by G is an injection with image C. Thus $\mathbb{F}_q^k$ and C are isomorphic as vector spaces.

That C is a k-dimensional subspace also gives rise to the study of another subspace which is the orthogonal complement. We use the conventional notation

$$C^\perp = \{\mathbf{x} \mid \forall \mathbf{c} \in C, \langle \mathbf{c}, \mathbf{x} \rangle = 0\}$$

to denote the orthogonal complement. Here $\langle \cdot, \cdot \rangle$ is the usually inner product and due to some standard linear algebra it is clear that $C^\perp$ is also a subspace and has dimension $n - k$. We call $C^\perp$ the dual code of C and from the above observations it is an $[n, n-k]_q$ code.

Besides the length and dimension another parameter is also interesting in the classical study of linear codes, namely the minimum distance. To define the minimum distance we start by introducing the Hamming weight of a vector $\mathbf{x}$. The Hamming weight $w(\mathbf{x})$ is defined to be the number of nonzero elements in $\mathbf{x}$. Similarly, we define the Hamming distance between two vectors, $\mathbf{x}, \mathbf{y}$, to be the number of entries where the two vectors differ, or equivalently $d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y})$.

The minimum distance of the code C is then defined as

$$d(C) = \min_{\mathbf{x},\mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}} d(\mathbf{x}, \mathbf{y})$$

and if $d(C) = d$ we often refer to the code C as an $[n, k, d]_q$ code. We remark that due to the linearity of linear codes it holds that $d(C) = \min_{\mathbf{x} \in C \setminus \{\mathbf{0}\}} w(\mathbf{x})$. One of the main studies for linear codes have been to find maximal values of k or d when n and q (and the other parameter of k and d) are fixed. The reason why we would like a high dimension is that a high dimension implies that we communicate more data when sending a codeword. However, a high dimension of course implies a low minimum distance. We consider two extreme cases. First, if $k = n$ we necessarily have $d = 1$. This follows since we are actually not encoding (we are simply sending the data, we would like to send). But this also comes with the drawback that we cannot correct any errors since if an error occurs the receiver receives another message. The other extreme case is using the repetition code. In this case $k = 1$ and we are sending n copies of the same message. However, we see that the minimum distance is n in this case and we can determine the message sent by a majority vote. This also means we cannot always be able to correct if $\frac{n}{2} = \frac{d}{2}$ errors occur. To see this consider the situation where $q = 2$ (the possible values are 0 and 1), and $n = 4$. If the sender wants to send the message 1, he sends 1111 using the repetition code. But if $\frac{4}{2} = 2$ errors occur, such as the receiver

receives 0101 it cannot be decided by a majority vote which message was sent. The fact that we can correct if less than $\frac{d}{2}$ errors occur is actually a general result. By decoding here we mean that we correct to the codeword which is closest to the received vector with respect to the Hamming distance.

These two extreme cases show the trade-off between large messages and large capability of error-correction. However, intermediate values for the dimension and minimum distance are often more preferable and a large amount of research has been done trying to find optimal codes.

Up until now we have only considered the classical parameters for error-correcting codes. Other parameters have also turned out to be important for several purposes. This includes for example the minimum distance of the dual code and the minimum distance of the squared code. The squared code is defined as the $\mathbb{F}_q$-linear span of the set consisting of every componentwise product of two codewords in C and these codes and their generalizations to higher powers have recently attained focus in research [5, 21–23]. This is due to the fact that the parameters of these codes have impact on some decoding algorithms for linear codes and several cryptographic protocols. I will describe some of these protocols later on and return to the influence these parameters have on their performance. Furthermore, some other, recently discovered parameters are the relative generalized Hamming weights of code pairs introduced in [15]. Here we consider two codes, C_2 and C_1, where $C_2 \subseteq C_1$. The relative generalized Hamming weights (RGHW) generalizes in some way the minimum distance of a linear code. To see this we present the definition of the i'th RGHW for an $i \in \{1, 2, \ldots, \dim(C_1) - \dim(C_2)\}$ below. It uses the support weight $w_S(D) = |\bigcap_{\mathbf{x} \in D} \operatorname{supp}(\mathbf{x})|$ of a subspace $D \subseteq \mathbb{F}_q^n$ and is defined as

$$M_i(C_1, C_2) = \min\{w_S(D) : D \subseteq C_1, D \cap C_2 = \{0\}, \dim(D) = i\}.$$

The first RGHW reduces to the usual minimum distance if $C_2 = \{0\}$ since the support weight reduces to the Hamming weight when the dimension of the spaces considered are one.

2 Secret Sharing

For a formal definition of secret sharing and many other concepts in this area I refer the reader to Paper A. In this introduction I will only give explanations and descriptions of the concepts.

The classical setup for secret sharing considers n parties and a single dealer (the dealer can be, but is not necessarily, one of the parties). The dealer holds some information or data which he wants to share among the parties. Even though the dealer wants to share the data among the parties he does not trust the individual parties enough to let them receive the data.

So he splits the data in shares and gives the individual parties a share each. Small subsets of shares must not reveal anything about the secret but collecting enough of the shares the collaborating parties should be able to recover the shared data.

When secret sharing was invented in 1979, independently by Adi Shamir and George Blakley [2, 24], only threshold schemes were considered. This means that any t parties have no information about the secret while any $t+1$ parties are able to reconstruct the secret.

2.1 Linear Secret Sharing and Shamir's Scheme

Shamir's secret sharing scheme is probably the most well-known scheme and is also widely used nowadays. As an introduction to how secret sharing works I will give a description of this scheme below.

To give a mathematical description of secret sharing we typically assume that there is a translation from the secret to some element in a finite field (or vector over a finite field). Therefore, when talking about the secret we are from now on referring to the field element $s \in \mathbb{F}_q$. If a dealer wants to secret share $s \in \mathbb{F}_q$ among n parties such that any t parties have no information about the secret and any $t+1$ parties can recover the secret, the dealer can use Shamir's scheme as long $q > n$. The dealer starts by constructing the polynomial

$$f(x) = s + r_1 x + r_2 x^2 + \cdots r_t x^t, \tag{1}$$

where r_i is randomly chosen in $\mathbb{F}_q$ by the dealer for $i = 1, 2, \ldots t$. Afterwards, the dealer distributes the shares $f(\alpha_i)$ to the i'th party where the α_i's are different nonzero elements of $\mathbb{F}_q$ (this is also why we need $q > n$). Using Lagrange interpolation it is known that the evaluations in any $t+1$ points uniquely determine a polynomial of degree at most t. This shows that any $t+1$ parties can determine the polynomial and hence the secret. On the other hand, consider if we had only t shares. Then the parties do not have any further information about s which was not known a priori. This can easily be seen since adding the point $(0, s)$ to the t shares gives a polynomial of the right degree with s as constant term using Lagrange interpolation. However, we can choose any value of s, meaning that the knowledge of t shares is not enough to obtain any information about the secret.

Besides being a threshold scheme Shamir's secret sharing scheme has also another property, namely linearity. Linearity means that linear combinations of sharings have to result in a share for the same linear combination of the corresponding secrets. To see that this holds, we assume that a dealer has secret shared s using f as above and $\tilde{s}$ using another t-degree polynomial $\tilde{f}$ with constant term $\tilde{s}$. Then the parties can compute $af(\alpha_i) + b\tilde{f}(\alpha_i) = (af + b\tilde{f})(\alpha_i)$ for some $a, b \in \mathbb{F}_q$. The equality shows that this is a share for

the i'th party of the t-degree polynomial $af + b\tilde{f}$ which has constant term, and hence secret, $as + b\tilde{s}$.

Linear secret sharing schemes can in general be described by linear codes, a connection which has been well-studied since the 1980's, see for instance [17–19]. There are typically two ways to describe secret sharing with linear codes, the first one uses a single linear code and the latter uses a code pair where the one is a subspace of the other, see for instance [7].

In the first approach we let the first entry (or entries) of the codeword be the secret and distribute the remaining entries to the parties. Due to the error-correction properties of the linear codes we conclude that knowing enough correct shares we can find the codeword and hence the secret.

In the latter approach we consider the codes $C_2 \subsetneq C_1$. We denote by L a subspace such that $C_1 = L \oplus C_2$. Fixing the randomness corresponds to fixing a codeword in C_2 and fixing the secret corresponds to fixing a codeword in L. Hence, when the secret and the randomness is fixed we have a unique codeword in C_1 and we let the entries in the codeword of C_1 be the shares distributed to the parties. Since there is a bijective map between the codewords in C_1 on the one hand and the secret and randomness on the other it is always possible to reconstruct the secret from the codeword (if we have the codeword this means we have access to all the shares). We remark that knowing enough entries in the codeword the parties might be able to compute the remaining entries (using some decoding algorithm for error-correcting codes) and in that way they might be able to reconstruct the secret. Furthermore, the linearity of the secret sharing scheme is inherited from the linearity of the codes.

To see the difference between the two methods we consider Shamir's scheme. In the first approach we can identify the linear code with the generator matrix

$$G = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ 0 & \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \alpha_1^t & \alpha_2^t & \cdots & \alpha_n^t \end{bmatrix}.$$

Multiplying by the vector $\mathbf{m} = (s, r_1, \ldots, r_t)^T$ yields

$$\mathbf{m}G = (f(0), f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n))^T,$$

with the f from (1). We see that the first entry is the secret and the remaining entries corresponds to the shares.

In the other approach we will identify the first code C_1 with the code spanned by the rows in G if we delete the first column. This is also known as the punctured code. The other code $C_2 \subsetneq C_1$ is given by first restricting to all

the codewords having 0 in the first component and removing that coordinate. This is also known as shortening the code. In the case above this corresponds to removing the first column and row of G and this also implies that $L = \text{span}\{(1,1,\ldots,1)\}$, where the vector has n ones. Sharing is similar and is done by the computation

$$\mathbf{m}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & & \vdots \\ \alpha_1^t & \alpha_2^t & \cdots & \alpha_n^t \end{bmatrix} = (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n))^T,$$

where $\mathbf{m}$ is the same as above. One can see that the two methods are similar and in fact they are equivalent. One can always convert the one to the other following the method described above. For more details about this we refer the reader to Appendix A in Paper A.

2.2 Privacy and Reconstruction

The two main concepts in secret sharing are privacy and reconstruction. The dealer wants that when he shares the secret privacy is retained, which means that small subsets of shares reveal no information about the secret, but simultaneously he wants that if enough of the parties pool together their shares they are able to reconstruct. In Shamir's scheme described above we have in some sense the optimal framework since any t parties have no information and any set of $t+1$ parties can recover the secret. This means that there are no subsets having partial information about the secret and furthermore privacy and reconstruction relies completely on how many parties there are cooperating.

Usually, we denote by t the maximum number such that all subsets of t parties have no information about the secret and call t the privacy threshold. Similarly, we denote by r the minimum number such that all subsets of r parties are able to reconstruct the secret (i.e. Shamir's scheme has $r = t+1$) and call r the reconstruction threshold. The difference between these two numbers, $g = r - t$, is referred to as the threshold gap, and if $g = 1$ the scheme is called a threshold scheme. Typically, we want this threshold gap to be small and therefore one can consider threshold schemes as optimal with this perspective.

On the other hand threshold schemes, such as Shamir's scheme, are also kind of limited. We could imagine setups where we would like different access structures, meaning that we would like different subsets of parties of potentially different cardinalities to be able to reconstruct. Furthermore, threshold schemes have some other disadvantages for instance regarding share size.

As we saw in Shamir's scheme the size of the share was the same as the size of the secret. In fact it is unavoidable to obtain smaller shares if we insist on a threshold scheme. For some applications this might be a problem.

Another limitation we have for Shamir's scheme is the requirement that $q > n$. If for example the secret is just a single bit and the number of parties is large we cannot immediately apply Shamir's scheme. We could, however, embed the secret into a larger field with $q > n$ elements but this implies that the shares are much larger than the secret which might be unintended for complexity reasons.

Even though we have the above mentioned limitations we are always ensured to have some amount of privacy and reconstruction in linear secret sharing schemes if we choose the linear codes for the construction wisely. Due to some general results from coding theory it is shown in [7] that we have the following bounds on t and r when the secret sharing scheme is defined from a single code

$$\begin{aligned} t &\geq d(C^{\perp}) - 2 \\ r &\leq n - d(C) + 2. \end{aligned} \tag{2}$$

Furthermore, [7] also shows that when the secret sharing scheme is constructed from a code pair $C_2 \subseteq C_1$, the t and r satisfy the bounds

$$\begin{aligned} t &\geq d(C_2^{\perp}) - 1 \\ r &\leq n - d(C_1) + 1. \end{aligned} \tag{3}$$

In fact it is shown in [11, 14] that we can turn the last inequalities into equalities if we consider the RGHWs instead of the minimum distance.

$$\begin{aligned} t &= M_1(C_2^{\perp}, C_1^{\perp}) - 1 \\ r &= n - M_1(C_1, C_2) + 1. \end{aligned} \tag{4}$$

Unfortunately, the minimum distance of a linear code is often hard to compute and the same holds for the RGHWs of a code pair.

3 Multiparty Computation

Yao formalized two-party computation in his paper [26] from 1982. The concept was generalized to the multiparty setting in 1987 in the paper [12]. In multiparty computation (MPC) we consider n parties who want to compute the output of some function f. Each party holds an input, i.e. the i'th party holds x_i, to the function and are not willing to reveal its input to the other parties. If all the parties could agree on some external trusted entity the parties could send its input to this entity. This entity will then be able to

compute the output $f(x_1, x_2, \ldots, x_n)$ and send back the output to the parties. In multiparty computation we would like to replace this external trusted entity by communication between the n parties but still obtain the same amount of privacy.

Talking about privacy in multiparty computation is a bit tricky since we want the parties to learn something, namely the output of the function. The output can potentially reveal one or some of the parties' inputs completely. This could for instance be in a two-party protocol where the two parties would like to compute $f(x_1, x_2) = x_1 + x_2$ for $x_1, x_2 \in \mathbb{F}_q$. Since the second party knows x_2 and learns $f(x_1, x_2)$ it is easy to determine $x_1 = f(x_1, x_2) - x_2$. The same holds for the first party. But this is unavoidable due to the function we would like to compute.

3.1 Security in MPC

This small example illustrates some of the challenges of talking about privacy in multiparty computation. What we can request is that the parties do not obtain more information about other parties' inputs than what the output of the function together with their own inputs reveal to them. This would be what we would like to achieve with regard to privacy but we can have several other security concerns to take care of and it is often difficult to list all the requirement we want a multiparty computation protocol to achieve. Of course, we want the parties to receive the correct output and we want privacy of the inputs, but depending on the setup we might want additional properties. In an auction it would for instance be bad if a corrupt party could modify another party's input and place it as its own input. There could also be a question about fairness for some situation meaning that if one party gets the output all other parties must also learn it. Furthermore, in other situations we would like to ensure that all parties actually receives an output which is known as guaranteed output delivery. Since you can never be sure that you have listed all the requirements, the security in multiparty computation is often defined in another way using the ideal/real world paradigm. The idea about this paradigm is that we in some way are comparing to the ideal world where we do have a trusted external entity everyone could send its inputs to. This trusted entity takes care of all the computations and sends back the outputs to the parties. In the real world this entity does not necessarily exist so the parties are communicating by following some steps in a protocol. We say that a protocol is secure if it is as secure as the ideal world meaning that an adversary cannot do more harm in the protocol than it would be able to do in the ideal setting.

One security definition using this ideal/real world paradigm is the universal composable (UC) security definition from [3, 4] which is a very common security definition nowadays. Typically, we are using the UC definition

if the protocol might be a part of a larger protocol or if we want to compose the protocol with others. A protocol might be secure as it stands alone but there might be problems if the protocol is used several times, for instance in parallel, or composed with other protocols. If a protocol is UC secure it means that it is safe to use in any context. Below I give a small description of the UC security definition.

In order to prove that a multiparty computation protocol is secure we again consider the two worlds. In the real world we have the protocol and the parties but we also have an adversary taking control over some of the parties. Furthermore, we have the environment which is everything external to the protocol and hence also includes the adversary. The environment can provide inputs and see outputs of the honest parties.

In the ideal world we also have the environment but we do not consider any parties. Instead of the protocol we have an uncorruptable ideal functionality which computes exactly what we would like to compute and instead of the parties we have a simulator.

The simulator then needs to provide the data to the environment by following the steps in the protocol in such a way that the environment cannot distinguish if it is interacting with the simulator in the ideal world or the parties in the real world. The only help the simulator gets to simulate messages to the environment is access to the ideal functionality. This means that the simulator can provide inputs on behalf of the corrupted parties to the ideal functionality and receive the outputs as well. The environment is also providing inputs and sees the outputs on behalf of the honest parties in this setup.

We say that a protocol is secure, if the environment cannot distinguish if it has run the protocol with the parties and adversary in the real world or if it has received messages from the simulator using the ideal functionality in the ideal world. In this way we ensure that an adversary cannot do any more harm or collect any more information than it is capable of in an ideal world.

There are different security definitions depending on how good the simulator needs to be and the power of the environment/adversary. If the two views the environment would see are identically distributed we say that the protocol has perfect security. If the views are so close that the environment has only a very small probability of distinguishing we say that the protocol has statistically security. We then need to specify the security parameter and show that the probability of distinguishing between the two worlds are negligible in the security parameter.

In both cases we have assumed that the adversary and environment have unlimited computation power. If we restrict them to be bounded by polynomial time, then we also talk about computational security.

Besides that we can also prove protocols secure under different assumptions on the adversary's power. If the adversary is only allowed to try to

collect as much information they can, but is not allowed to deviate from the protocol we say that a protocol is secure in the presence of passive (or semi-honest) adversary. If the adversary is allowed to deviate from the protocol we call it an active (or malicious) adversary. The papers collected in this thesis mainly focus on active adversaries. At last we also consider restrictions on how many of the parties the adversary is allowed to corrupt. Typically, we divide protocols into three groups. One where the adversary may corrupt less than $\frac{1}{3}$ of the parties, one where we have honest majority, and the last where we have half or more of the parties corrupted. In this setting we typically allow the adversary to corrupt all but one of the parties, and this setting is called dishonest majority. For security against active adversaries we require that less than $\frac{1}{3}$ of the parties are corrupted if we want perfect security. If up to half of the parties are corrupted we can only obtain statistical security (here we also need to assume that a broadcast channel is available). Similarly, if more than half of the parties are corrupted we can only strive for computational security and in this setting we have to give up some of the security requirements since we cannot guarantee fairness. This means that the adversary can abort the protocol after they have received their outputs but before the honest parties learned their outputs. To guarantee fairness we require that less than half of the parties are corrupted. We remark that this is not the case if we assume passive adversaries since all the parties will follow the instructions in this case. A more restrictive situation is guaranteed output delivery where all the parties must receive the output. This can also be guaranteed as long as we have an honest majority and a broadcast channel is available. If a broadcast channel is not available less than one third of the parties must be corrupted to ensure guaranteed output delivery.

For a more formal and thorough walk-through of multiparty computation we refer the reader to [8]. However, I will present a sketch of a multiparty computation protocol below which indicates a strong relation between secret sharing and MPC.

3.2 An MPC Protocol

The protocol I present is secure against a passive adversary if less than half of the parties are corrupted. Even though my papers are mainly considering active adversaries I decided to present a protocol with passive security here since these protocols are simpler and the reason for presenting the protocol is mainly to get a feeling of how MPC works and the interaction there is between secret sharing and MPC. The protocol can also be found in [8].

In secret sharing based MPC protocols we typically describe the function f as an arithmetic circuit. An arithmetic circuit is allowed to combine two expressions it has already computed via an addition gate or a multiplication gate. The circuit takes the inputs of the parties as inputs and combine those

in a predetermined way until we have the output of the function.

If we want to compute f in the clear the parties could just broadcast their inputs and compute the arithmetic circuit. However, we want to keep the inputs secret so we compute the gates on the sharings instead of on the values.

We will use the notation $\langle x \rangle$ for a sharing of x. This means that each party P_i holds a share $x^{(i)}$ for $i = 1, 2, \ldots, n$ and there exists a linear reconstruction function

$$\rho(x^{(1)}, x^{(2)}, \ldots, x^{(n)}) = \sum_{i=1}^{n} \lambda_i x^{(i)} = x.$$

We call a secret sharing scheme multiplicative if there also exists a linear reconstruction function

$$\rho^*(x^{(1)}y^{(1)}, x^{(2)}y^{(2)}, \ldots, x^{(n)}y^{(n)}) = \sum_{i=1}^{n} \lambda_i^* x^{(i)} y^{(i)} = xy$$

for all $x, y \in \mathbb{F}_q$ and all possible shares of x and y.[1] We remark that if a secret sharing scheme is multiplicative we necessarily have that the privacy threshold $t < \frac{n}{2}$, see for instance [8].

Since we want that the sharings are travelling on the wires we start by fixing a multiplicative secret sharing scheme and let each party secret share its input using this scheme. After the parties have secretly shared their input, meaning that we have $\langle x_i \rangle$ for $i = 1, 2, \ldots, n$, the protocol goes on gate by gate. Addition and multiplication by constants can be carried out with local computations because of the linearity of the secret sharing scheme. For instance, if we needed to compute $x + y$, we can obtain $\langle x + y \rangle$ if all parties sum their corresponding shares, i.e. $x^{(i)} + y^{(i)}$ for $i = 1, 2, \ldots, n$.

Multiplication gates are a bit more complex and require communication between the parties to achieve. If we have $\langle x \rangle$ and $\langle y \rangle$ going into a multiplication gate we start by letting each party compute $\tilde{z}_i = x^{(i)}y^{(i)}$. Then the party considers this value as a secret and shares this between all parties using the same secret sharing scheme as before. Thus we have $\langle \tilde{z}_i \rangle$ for $i = 1, 2, \ldots, n$. Now, we can use the properties of the multiplicative secret sharing scheme. Let $z = xy$ and notice that we have

$$\rho^*(\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_n) = xy.$$

Furthermore, if each party uses ρ^* on the shares $\tilde{z}_j^{(i)}$ we obtain shares of z, i.e.

$$\rho^*(\tilde{z}_1^{(i)}, \tilde{z}_2^{(i)}, \ldots, \tilde{z}_n^{(i)}) = z^{(i)}.$$

[1] Remark that a sharing of $\langle x \rangle = (x^{(1)}, x^{(2)}, \ldots, x^{(n)})$ is just a codeword in C_1 if we are using the code pair construction for the secret sharing scheme. Hence, $(x^{(1)}y^{(1)}, x^{(2)}y^{(2)}, \ldots, x^{(n)}y^{(n)})$ is just a codeword in the squared code which is the reason for using the notation ρ^*

This follows since

$$\begin{aligned}\rho(z^{(1)}, z^{(2)}, \ldots, z^{(n)}) &= \rho(\rho^*(\tilde{z}_1^{(1)}, \ldots, \tilde{z}_n^{(1)}), \ldots, \rho^*(\tilde{z}_1^{(n)}, \ldots, \tilde{z}_n^{(n)})) \\ &= \sum_{i=1}^{n} \lambda_i \left(\sum_{j=1}^{n} \lambda_j^* \tilde{z}_j^{(i)} \right) \\ &= \sum_{j=1}^{n} \lambda_j^* \left(\sum_{i=1}^{n} \lambda_i \tilde{z}_j^{(i)} \right) \\ &= \rho^*(\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_n) \\ &= xy.\end{aligned}$$

Since I now have described how the parties can handle the different gates I only need to describe how they obtain the output. After the execution of a number of addition and multiplication gates the parties will hold a share each for the output. Since we only consider passive security we can just instruct the parties to send this share to all other parties so that each party can compute the output using ρ.

It can be shown using the ideal/real world paradigm that as long as the privacy threshold t of the secret sharing scheme is high enough the protocol is secure. To get a feeling of why this is true consider the messages sent through the protocol. The only messages which are transmitted between the parties are shares of the inputs and shares of $x^{(i)}y^{(i)}$ in the multiplication gates. But under the assumption that less than t of the parties are corrupted these shares looks like random field elements to the adversary. Shamir's scheme with $t = \left\lfloor \frac{n-1}{2} \right\rfloor$ is a multiplicative secret sharing scheme implying that as long as the adversary is able to corrupt less than half of the parties the described protocol is secure. This implies that the protocol is perfectly secure against passive adversaries as long as we have an honest majority if we use Shamir's scheme.

4 Overview

Part II of this thesis consists of four papers I have published during my Ph.D. Below I will put the papers into context of the existing theory and highlight the main contributions for each of the papers. Furthermore, I will point out how the papers use and contribute to each of the three theoretical areas described above.

The main contribution of Paper A is new limitations on the threshold gap and the privacy and reconstruction thresholds in linear secret sharing. We use techniques from coding theory to obtain the results. More specifically, we use the newly obtained results from [11, 14], which I presented in equation

(4), linking the RGHWs of a code pair to the privacy and reconstruction thresholds for the corresponding secret sharing scheme. We combine the equalities with bounds on the RGHWs to obtain the following bounds on the thresholds t, r, and the threshold gap g, where q is the size of the field, n is the number of parties and k_1, k_2 are the dimensions of the codes $C_1 \supsetneq C_2$ used in the construction of the scheme. Furthermore, $\ell = k_1 - k_2$ is the length of the secret vector.

$$\begin{aligned} t &\leq \frac{q^{m+1} - q^m}{q^{m+1} - 1}(k_2 + m + 1) - 1, \\ r &\geq \frac{q^m - 1}{q^{m+1} - 1}n + \frac{q^{m+1} - q^m}{q^{m+1} - 1}(k_1 - m - 1) + 1, \\ g &\geq \frac{q^m - 1}{q^{m+1} - 1}(n + 2) + \frac{q^{m+1} - q^m}{q^{m+1} - 1}(\ell - 2m) \end{aligned} \tag{5}$$

for all $m \in \{0, 1, \ldots, \ell - 1\}$. The results shows that the threshold gap is dependent on several factors such as the share size (q), the size of the secret compared to the size of the shares (ℓ) and the number of parties (n). There have been presented several bounds on the threshold gap in the literature but it is only few which include all these variables. One exception of this is the following bound from [6].

$$g \geq \frac{n + 2}{2q + 1} + \frac{2q}{2q + 1}(\ell - 1), \tag{6}$$

which holds ad long as $\ell \geq 2$. However, we show in the paper that if $\ell \geq 2$ which allows us to choose $m = 0$ and $m = 1$ in (5) one of these choices will improve on the bound in (6).

We also compare to several other bounds showing that our bounds improve on existing bounds both for concrete cases and in an asymptotic setting. We see that especially when the field size is small our bounds improve on existing bounds. One of the conclusions is that the field size (which is essentially the share size) has a huge impact on the threshold gap. I remark that these results also have an impact on several multiparty computations protocols, for instance the protocol presented in section 3.2 since limitations on the privacy threshold or threshold gap in secret sharing implies limitaions for how many corrupted parties we can allow in multiparty computation protocols using the schemes.

Paper B is regarding a specific primitive in multiparty computation known as oblivious transfer (OT). An OT takes place between two parties; a sender and a receiver. In its simplest form the sender inputs 2 messages, m_0 and m_1, to the functionality and the receiver inputs a single bit b. The receiver learns m_b but the sender does not learn anything. Hence, the receiver does not learn anything about the other message m_{1-b} and the sender does not

learn which of the messages the receiver learnt. A natural generalization of this simple OT functionality is that the sender inputs N messages and the receiver chooses $K < N$ of these messages to learn. We call such an OT for a K-out-of-N OT.

OT is a very useful functionality for multiparty computation. In fact it is shown in [13] that any functionality for multiparty computation can be implemented if the OT-functionality is available. Unfortunately, constructions of OT's seems to be expensive which is an obstacle for the theoretical result from [13]. Towards making MPC based on OT's useful in practice OT-extension was introduced. OT-extension protocols simulates a large number of OT's using a much smaller number of what is called base OT's, and in this way we can save some complexity in practice.

In Paper B we considered an actively secure protocol from [20] which made use of binary linear codes to construct many 1-out-of-N OT's. We generalized the protocol so it was not restricted to use binary codes but it could instead use a linear code over any finite field. The parameters of the code have impact on both the complexity and the security of the protocol but also on the possible values of N for the OT. For instance N equals the number of codewords in the code and depends therefore on the dimension of the code and the field size, the minimum distance has an impact of the security, and the length of the code corresponds to the number of base OT's. It is especially the number of base OT's we can lower when going to a q-ary code. However, as we show in the paper this comes with a cost of increasing the complexity in other parts of the protocol.

In the first two papers we need linear codes with "good" parameters in order to obtain the best constructions. The parameters of interest differ however. In the first paper we are mainly interested in codes with high RGHWs in order to obtain good thresholds for the secret sharing schemes while the codes needed in Paper B needs to have "good classical parameters" to optimize our OT-extension protocol. In Paper C we focus on constructing linear codes with a "high" dimension and minimum distance on its square simultaneously. Several multiparty computation protocols rely on linear codes with this property and in fact the sketch of an MPC protocol from section 3.2 requires at least a similar property. To justify this statement without going into details I refer to the inequalities involving t and r which I presented in (2). For a scheme to be multiplicative we require that we are able to reconstruct in the squared code (meaning that the minimum distance of the code C^* needs to be at least 2). Simultaneously, we would like a high dual distance on C but since $d(C^\perp) \leq \dim(C)$ we need to have a high dimension as well. I will refer to another application of codes with high squared distance when discussing Paper D.

Even though such codes seem to be important for many MPC protocols, not much research has been done in producing these codes, especially not

when the field is small. We consider matrix-product codes, a code construction where we obtain longer codes from shorter ones. By considering the structure of generator matrices for matrix-product codes, we show that the square of some matrix-product codes can be determined as another matrix-product code allowing us to give bounds on the parameters of the square from the codes used in the construction. This could for instance be useful if we want to compute a boolean circuit in MPC with many parties. We cannot directly rely on Shamir's scheme as it needs the field size to be larger than the number of parties. Hence, we would need to base our secret sharing scheme used in the protocol on another code over $\mathbb{F}_2$ which has the length equal to the number of parties. A matrix-product code could be a good suggestion for such an application.

The last paper, Paper D, introduces a new multiparty computation protocol in the dishonest majority setting. We follow a similar strategy as the protocol in [10] also known as MiniMAC. MiniMAC uses ideas from the SPDZ protocol [9], where they, as in the protocol from section 3.2, also consider the function as a circuit and use secret sharing. In their case they use additive secret sharing, meaning that the reconstruction function ρ is just the sum of all the shares. Like in section 3.2, addition gates can be carried out without communication. Thus, the main difference is the setup the protocols consider and the way they carry out the multiplication gates. SPDZ and MiniMAC consider the dishonest majority setting so to ensure that the parties do not deviate from the protocol they not only share their inputs but also a so-called MAC. The MAC is used to check that values are opened correctly and the probability of fooling the MAC is therefore important. In SPDZ the probability of fooling the MAC is $\frac{1}{q}$ where q is the field size and hence SPDZ requires that the field size is large. In MiniMAC the techniques from SPDZ are adapted to work over small fields but a direct approach of using MACs in small fields will of course not provide enough amount of security, so [10] suggested to group several inputs together and encode them to a codeword in a good linear error-correcting code. The MACs and the shares are then vectors over a small field and the probability of fooling the MACs depends on the minimum distance of the code.

In the dishonest majority setting we cannot apply the techniques for the multiplication gates described in section 3.2 for several reasons. For instance, we need to ensure that the corrupt parties do not deviate from the protocol and another reason is that there do not exist multiplicative schemes with a privacy threshold that high. Instead the SPDZ protocol uses a well-known technique known as Beaver's trick to carry out the multiplication gate [1]. However, in MiniMAC this trick implies that we end up with codewords in the squared code and hence we need to have a large minimum distance on the square to ensure security as well. Thus for the MiniMAC protocol one needs a code over a small field with high dimension and minimum distance

on the square which are exactly such codes we considered in Paper C.

However, in Paper D we suggest to replace the linear code with a tool which seems well-suited for this type of protocol, a tool called reverse multiplication friendly embedding (RMFE). Instead of mapping the grouped inputs to a codeword we map them to an element of an extension field and use the map from the RMFE such that we can reverse back to the small field after the multiplication gate is carried out in the extension field. In this way we also circumvent the problem with MACs over small field because of this conversion. We compare our protocol to MiniMAC and find that the reverse multiplication friendly embedding is "more compact" in some sense and hence we save some complexity.

References

[1] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology — CRYPTO '91*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 420–432.

[2] G. R. Blakley, "Safeguarding cryptographic keys," *Managing Requirements Knowledge, International Workshop on*, vol. 00, p. 313, 1979.

[3] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," Cryptology ePrint Archive, Report 2000/067, 2000, https://eprint.iacr.org/2000/067.

[4] R. Canetti, "Universally composable security: a new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001, pp. 136–145.

[5] I. Cascudo, "On squares of cyclic codes," *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 1034–1047, 02 2019.

[6] I. Cascudo, R. Cramer, and C. Xing, "Bounds on the threshold gap in secret sharing and its applications," *IEEE Transactions on Information Theory*, vol. 59, no. 9, pp. 5600–5612, September 2013.

[7] H. Chen, R. Cramer, S. Goldwasser, R. de Haan, and V. Vaikuntanathan, "Secure computation from random error correcting codes," *Advances in Cryptology - EUROCRYPT 2007*, pp. 291–310, 2007.

[8] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[9] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology – CRYPTO 2012*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662.

[10] I. B. Damgård and S. Zakarias, "Constant-overhead secure computation of boolean circuits using preprocessing," in *Theory of Cryptography*. Berlin, Heidelberg: Springer, 2013, pp. 621–641.

[11] O. Geil, S. Martin, R. Matsumoto, D. Ruano, and Y. Luo, "Relative generalized hamming weights of one-point algebraic geometric codes," *IEEE Transactions on Information Theory*, vol. 60, no. 10, pp. 5938–5949, October 2014.

[12] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87. ACM, 1987, pp. 218–229.

[13] J. Kilian, "Founding crytpography on oblivious transfer," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. ACM, 1988, pp. 20–31.

[14] J. Kurihara, T. Uyematsu, and R. Matsumoto, "Secret sharing schemes based on linear codes can be precisely characterized by the relative generalized hamming weight," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 11, pp. 2067–2075, 2012.

[15] Y. Luo, C. Mitrpant, A. J. H. Vinck, and K. Chen, "Some new characters on the wire-tap channel of type ii," *IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 1222–1229, March 2005.

[16] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. North-Holland Publishing Company, 1977, vol. 16.

[17] J. L. Massey, "Some applications of coding theory in cryptography," in *Codes and Ciphers: Cryptography and Coding IV*, 1995, pp. 33–47.

[18] J. L. Massey, "Minimal codewords and secret sharing," in *Proceedings of the 6th Joint Swedish-Russian International Workshop on Information Theory*. Citeseer, 1993, pp. 276–279.

[19] R. J. McEliece and D. V. Sarwate, "On sharing secrets and reed-solomon codes," *Commun. ACM*, vol. 24, no. 9, p. 583–584, Sep. 1981.

[20] M. Orrù, E. Orsini, and P. Scholl, *Actively Secure 1-out-of-N OT Extension with Application to Private Set Intersection*. Cham: Springer International Publishing, 2017, pp. 381–396.

[21] H. Randriambololona, "Asymptotically good binary linear codes with asymptotically good self-intersection spans," *IEEE Transactions on Information Theory*, vol. 59, pp. 3038 – 3045, 05 2013.

[22] H. Randriambololona, "An upper bound of singleton type for componentwise products of linear codes," *IEEE Transactions on Information Theory*, vol. 59, pp. 7936 – 7939, 09 2013.

[23] H. Randriambololona, "On products and powers of linear codes under componentwise multiplication," *Contemporary Math.*, vol. 637, 04 2015.

[24] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, November 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176

[25] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, July 1948.

[26] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, ser. SFCS '82. Washington, DC, USA: IEEE Computer Society, 1982, pp. 160–164.

Part II

Papers

Paper A

Improved Bounds on the Threshold Gap in Ramp Secret Sharing

Ignacio Cascudo, Jaron Skovsted Gundersen, and Diego Ruano

The paper has been published in the
IEEE Transactions on Information Theory Vol. 65(7), pp. 4620–4633, 2019.

The layout has been revised.

Abstract

In this paper we consider linear secret sharing schemes over a finite field $\mathbb{F}_q$, where the secret is a vector in $\mathbb{F}_q^\ell$ and each of the n shares is a single element of $\mathbb{F}_q$. We obtain lower bounds on the so-called threshold gap g of such schemes, defined as the quantity $r - t$ where r is the smallest number such that any subset of r shares uniquely determines the secret and t is the largest number such that any subset of t shares provides no information about the secret. Our main result establishes a family of bounds which are tighter than previously known bounds for $\ell \geq 2$. Furthermore, we also provide bounds, in terms of n and q, on the partial reconstruction and privacy thresholds, a more fine-grained notion that considers the amount of information about the secret that can be contained in a set of shares of a given size. Finally, we compare our lower bounds with known upper bounds in the asymptotic setting.

1 Introduction

Secret sharing, introduced independently by Blakley and Shamir [2, 28], is among the most useful primitives in cryptography. A secret sharing scheme allows to distribute the knowledge of a secret among n participants by sending each of them a piece of information (a *share*), in such a way that only certain prescribed subsets of these participants can reconstruct the secret from the joint information they have received. Secret sharing schemes are not only useful as a stand-alone primitive that can be used for secure distributed storage of information, but also play an important role as an element in more complex cryptographic tools, in areas such as threshold cryptography or secure multiparty computation.

In the study of secret sharing schemes and its applications it is often interesting to determine the amount of information about the shared secret that can be derived from pooling together a certain fixed number of shares. We say that a secret sharing scheme has t-privacy if any set of t shares provides no additional information about the secret to what was known a priori. On the other hand, the secret sharing scheme has r-reconstruction if the knowledge of any set of r shares uniquely determines the secret. By abuse of notation, fix t to be the largest integer for which there is t-privacy and fix r to be the smallest integer for which there is r-reconstruction. Then obviously $0 \leq t < r \leq n$, and we define the *threshold gap* as $g = r - t$, which is thus a strictly positive integer. It is usually desirable for applications of secret sharing that the privacy and reconstruction thresholds are as close as possible and hence, that the threshold gap is small. Since this allows to optimize the compromise between security against an adversary who attempts to learn enough shares to gain information about the secret (for which we want to set t large), and resilience against losing a number of shares by corruption or

other reasons (for which we want to set r small).

Secret sharing schemes with threshold gap $g = 1$ are called *threshold secret sharing schemes*. Shamir's secret sharing scheme (see Section 2 for its definition) is the most well-known example of a threshold secret sharing scheme: for any integers t and n with $1 \leq t < n$, one can construct a Shamir secret sharing scheme for n participants with t privacy and $t+1$ reconstruction. However, Shamir's scheme presents some restrictions regarding the size of the secret and shares in terms of n: in first place, both the secret and each of the shares are elements of the same finite field, which means that each of the shares is as large as the secret; in second place, the finite field must have at least $n+1$ elements (remember n is the number of participants) and therefore each share must be at least $\log(n+1)$ bits long.

Typically, in applications of secret sharing we would like the secret to be as large as possible while the shares are small, but it turns out that the two restrictions above are unavoidable for threshold secret sharing schemes, and more in general in secret sharing schemes with small threshold gap.

Consider first the relation between the size of the shares and the size of the secret. It is well-known that in any threshold secret sharing scheme, each share must be at least the same size as the secret (this holds more generally for any perfect secret sharing scheme, i.e., any secret sharing scheme where every set of shares either has full information about the secret or no information about it). And, more generally, if every share is an element of a certain alphabet of size q and the secret is a-priori uniformly distributed in an alphabet of size M, then it necessarily holds that

$$g \geq \log_q M. \tag{A.1}$$

This is a well known bound that is included as a special case of more general results in [4, 19, 25, 26], which relate the size of the secret and shares to various properties of the access and adversary structures of the secret sharing scheme[1]. However, when the only parameter about these structures we consider is the threshold gap, the bound in (A.1) is tight: it can be attained by a generalization of Shamir's scheme frequently known as packed Shamir's scheme, first proposed by Blakley and Meadows [3] (also defined in Section 2) where each share is in a finite field $\mathbb{F}_q$, the secret is in $\mathbb{F}_q^\ell$ for some $\ell \geq 1$ and we have $g = \ell$. We point out that this secret sharing scheme requires that $q \geq n + \ell$, which indicates that a large number of participants n will also introduce a restriction for the threshold gap and the size of secrets and shares, as we will see next.

[1]The access structure is defined as the family of sets of participants which can determine uniquely the secret from the shares they hold while the adversary structure is the family of sets of participants which can obtain no information about the secret (beyond what they know a priori) from their shares.

First, we note that the size of the shares in a threshold scheme is restricted by the number of participants, as a series of results have shown. In first place, it is known that threshold secret sharing schemes where the secret and each share is in the same alphabet are equivalent to maximum distance separable (MDS) codes (MDS codes are those which attain the so-called Singleton bound, see for instance [23]). The length of these codes is upper bounded by the size of the alphabet over which they are defined. Exploiting this connection, one can already show that if $1 \leq t < n-1$ (and $g = 1$, since we are considering threshold schemes), then $n < 2q - 2$ (see [12], Theorem 11.113).

But even in the more general case where we do not assume that the secret is in the same alphabet as the shares (for example even if the secret is just one bit), it was first noticed in the unpublished work [20] (see [7] for the statement and proof) that in any threshold scheme the average bitlength λ^* of the shares is $\Omega(\log(n-t))$. The result was later generalized in [7], where it was shown that for any secret sharing scheme where $t \geq 1$ (no individual participant obtains information about the secret) it necessarily holds that

$$g \geq \frac{n-t+1}{2^{\lambda^*}}.$$

If all shares belong to some alphabet of cardinality q, the bound can be rewritten as

$$g \geq \frac{n-t+1}{q}. \tag{A.2}$$

This bound hence establishes that, for certain values of t and n, there exist limitations on how small the threshold gap can be that depend solely on the size of the shares (and not on the secret). The bound was shown to be tight for $t = 1$ and $t = 2$ (the latter only in the case $q = 2$) in [27].

Later, [6] showed that if $r \leq n-1$, the bound

$$g \geq \frac{r+1}{q}$$

holds, which together with the bound in [7] implies

$$g \geq \frac{n+2}{2q-1} \tag{A.3}$$

as long as $1 \leq t < r \leq n-1$. This last bound had been shown earlier by [7] only in the case where the secret sharing scheme is $\mathbb{F}_q$-linear.

The two kinds of limitations that we have mentioned, represented by Equations (A.1) and (A.3) above are incomparable: the former depends on the relation between the sizes of the secret and shares, while the latter sets limitations on the relation between the size of the shares and the number of

participants. Note that, even though the bound given in Equation (A.1) can be attained by the Blakley-Meadows construction, this requires that $n < q$, and therefore the bound is not necessarily tight when n grows in relation to q (and in fact in general it cannot be attained, by virtue of Equation (A.3)). It is then natural to investigate what bounds one can get which depend on all these parameters simultaneously. In this regard, for $\mathbb{F}_q$-linear schemes where the secret is in $\mathbb{F}_q^\ell$ with $\ell \geq 2$ and each share is in $\mathbb{F}_q$, [7] showed the bound

$$g \geq \frac{n+2}{2q+1} + \frac{2q}{2q+1}(\ell - 1) \tag{A.4}$$

which is tighter than the straightforward combination $g \geq \max\{\ell, \frac{n+2}{2q-1}\}$ of Equations (A.1) and (A.3), when ℓ is large enough.

Another bound depending on both the share size and the relation between the size of the shares and the size of the secret can be deduced from [13]. In the language of all-or-nothing transforms they present a bound which in the setting of secret sharing implies

$$g \geq \frac{r}{q} + 1 - \frac{q-1}{q}\frac{r}{q^\ell - 1}. \tag{A.5}$$

Here one should note that as ℓ increases the bounds tends to $g \geq \frac{r}{q} + 1$. So for large enough ℓ the bound in (A.4) performs better than this bound.[2]

1.1 Contributions

In this paper we focus on $\mathbb{F}_q$-linear secret sharing schemes where secrets are in $\mathbb{F}_q^\ell$ and every share is in $\mathbb{F}_q$. In Section 3, we improve the bound (A.4) given in [7]. More precisely our main result (Theorem 3.2) is a family of bounds given by

$$g \geq \frac{q^m - 1}{q^{m+1} - 1}(n+2) + \frac{q^{m+1} - q^m}{q^{m+1} - 1}(\ell - 2m), \tag{A.6}$$

for $m = 0, 1, \ldots, \ell - 1$, and we show that for any $\ell \geq 2$, there is some m for which this new bound is tighter than (A.4).

We obtain these bounds by proving limitations on the so-called partial privacy and reconstruction thresholds. These are defined as follows: let r_i, for $i = 1, \ldots, \ell$, be the smallest number such that every set of shares of that

[2] We also remark the similarities with the bound from Theorem 4.4 stating that $g \geq \frac{r+1}{q} + \frac{q-1}{q} b_i$, where b_i is an non-negative integer. With $b_i \geq 1$ this bound is tighter than $\frac{r}{q} + 1$ and even for $b_i = 0$ the bound in (A.5) can only be one unit larger and in order to be larger we require a large ℓ.

size gives at least i q-bits of information about the secret and let t_i, also for $i = 1, \ldots, \ell$, be the largest integer such that every set of shares of that size learns less than i q-bits about the secret. We call t_i and r_i the partial privacy and reconstruction thresholds, respectively, and note that $r_\ell = r$ and $t_1 = t$ which means that $g = r_\ell - t_1$.

Relative generalized Hamming weight (RGHW) was first studied in the context of wiretap channel of type II, see [22]. However, when representing a linear secret sharing scheme as a nested code pair, it is shown in [17, 21] that the RGHWs of the pair of nested codes used in the construction determine the partial thresholds. Combining this with the Griesmer bound on the RGHWs implies limitations for t_i and r_i which eventually leads to the bounds in (A.6).

We emphasize that the improvement over (A.4) comes from two sources. The main one is the fact that we use results on the application of Griesmer bounds directly to the RGHWs instead of using a shortening argument to bound r and t and then applying the Griesmer bound to the resulting code as in [7]. In addition, we set a parameter m that determines how we bound each of the summands appearing in the Griesmer bound, while [7] simply set $m = 1$. This provides more flexibility, which for example is beneficial when proving asymptotic bounds (see Theorem 5.1).

In Section 4 we prove some additional results on the relation between the partial privacy and reconstruction thresholds. We remark that this also imply bounds on the RGHWs and therefore might also be relevant in the context of wiretap channel of type II. In this section we follow more or less the same approach as in [7] but generalize some of their results on r and t to the partial thresholds. We derive that as long as $t \geq 1$, we necessarily have

$$r_i \geq \frac{n}{q^{\ell-i+1}} + 1$$

for all $i \in \{1, \ldots, \ell\}$. Note that for $i = \ell$ we obtain $r \geq \frac{n}{q} + 1$. This is a bound that was also shown in [7] and was used to prove the more general inequality (A.2).

Moreover, we can also prove this bound under milder conditions, namely if $t_j \geq j$ for some $j \in \{1, \ldots, \ell\}$, then the same bound

$$r_i \geq \frac{n}{q^{\ell-i+1}} + 1$$

holds, but now for every $i \in \{j, j+1, \ldots, \ell\}$.

This leads to the following generalization of (A.3):

$$g \geq \frac{n+2}{2q-1} + \frac{q-1}{2q-1}(a_i + b_i),$$

where $a_i = t_i - t - i + 1 \geq 0$ and $b_i = r - r_{\ell-i+1} - i + 1 \geq 0$ are two quantities that capture how much the scheme deviates from the situation where $t_1 =$

$t, t_2 = t+1, \ldots, t_\ell = t+\ell-1$ and $r_\ell = r$, $r_{\ell-1} = r-1, \ldots, r_1 = r-\ell+1$, which occurs in the scheme of Blakley-Meadows (also known as packed Shamir), and which would correspond to $a_i = 0$, $b_i = 0$ for all i. At last in this section, we consider an example attaining this bound.

There are several potential uses of partial reconstruction and privacy thresholds in cryptography. For example, the notion of functional secret sharing introduced in [1] considers a scenario where large enough sets of participants can recover certain functions of the secrets and hence the threshold r_i gives us some information about functional secret sharing schemes where the output of the functions of interest consist of i q-bits. On the other hand, considering a relaxed notion of privacy (the threshold t_i) may be interesting in applications where secret sharing is combined with some other privacy amplification technique. For example with the goal of constructing a linear-time encodable secret sharing scheme [11] combines an error correcting code (which can be seen as a secret sharing scheme where small sets of participants can obtain partial information about the secret) with a hash function that destroys this partial information, so that perfect privacy is obtained in the final construction. This combination of "imperfect" secret sharing and privacy amplification may be of interest in secure computation, too. Our bounds on t_i and r_i would set some limitations on those potential applications as well.

Finally, we consider asymptotic secret sharing schemes in Section 5. We adopt the setting considered in [16], define an asymptotic threshold gap (in Equation (A.24)) and provide the asymptotic version of the previous bounds. At the end, we compare our bound with the asymptotic version of the bounds in [7] and investigate how sharp is our bound by comparing it with threshold gaps of secret sharing schemes constructed from algebraic geometric codes (in the case of large fields) and from random linear codes (for small fields).

2 Secret Sharing

In this section, we recall some notions regarding secret sharing schemes and their relationship with linear codes.

Let $S_0, S_1, \ldots, S_n$ be random variables taking values in the finite alphabets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n$. Then we call $\mathbf{S} = (S_0, S_1, \ldots, S_n)$ a vector of random variables. In this paper, we let $\mathcal{I} = \{0, 1, \ldots, n\}$ and $\mathcal{I}^* = \{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$ and for a subset $A \subseteq \mathcal{I}$ we denote by $\mathbf{S}_A$ the vector $(S_i)_{i \in A}$. Notice that $\mathbf{S} = \mathbf{S}_{\mathcal{I}}$. With this notation we define a secret sharing scheme.

Definition 2.1 (Secret Sharing Scheme). *A secret sharing scheme Σ is a vector of random variables*

$$\mathbf{S} = (S_0, S_1, \ldots, S_n) \in \mathcal{S}_0 \times \mathcal{S}_1 \times \cdots \times \mathcal{S}_n,$$

such that

$$H_q(S_0) = \log_q |\mathcal{S}_0|,$$

where H_q is the Shannon entropy with base q.[3] *Further, we require that*

$$H_q(S_0|\mathbf{S}_{\mathcal{I}^*}) = 0.$$

We call S_0 the *secret* and, for $i \in \mathcal{I}^*$, we call S_i the i'th *share*. The scheme has n *participants*, which we identify with the set $\mathcal{I}^*$, and the i'th participant holds S_i, for $i = 1, 2, \ldots, n$.

The requirement that $H_q(S_0) = \log_q |\mathcal{S}_0|$ implies that the random variable S_0 is uniformly distributed in $\mathcal{S}_0$; while it is of course possible to consider secret sharing schemes with a different distribution on the secret space, it was shown in [5] that such scheme could be transformed into one where the distribution of secrets is uniform and with the same reconstruction and privacy thresholds (introduced below). Therefore, this assumption is without loss of generality for our purposes.

The other requirement, that $H_q(S_0|\mathbf{S}_{\mathcal{I}^*}) = 0$, means that the secret is uniquely determined by the set of all the shares with probability 1.

A secret sharing scheme is called *linear* if $\mathbf{S}$ is uniformly distributed on some subspace $V \subseteq \mathcal{S}_0 \times \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ and if $\mathcal{S}_i$ is a $\mathbb{F}_q$-vector space for all $i \in \mathcal{I}$ where $\mathbb{F}_q$ is the finite field with q element. In this paper, we will focus on the schemes, where $\mathcal{S}_i$ is one-dimensional for $i \in \mathcal{I}^*$ and $\mathcal{S}_0$ is ℓ-dimensional. Without loss of generality we can assume that $\mathcal{S}_0 = \mathbb{F}_q^\ell$ and $\mathcal{S}_i = \mathbb{F}_q$ for the $i \in \mathcal{I}^*$.

Linear secret sharing schemes are also characterized by the following property; consider two secrets $\mathbf{s}, \mathbf{t} \in \mathcal{S}_0 = \mathbb{F}_q^\ell$. Let $\mathbf{x} \in \mathbb{F}_q^n$ be a possible share vector for the secret $\mathbf{s}$, i.e. $P((S_0, \mathbf{S}_{\mathcal{I}^*}) = (\mathbf{s}, \mathbf{x})) > 0$, and $\mathbf{y} \in \mathbb{F}_q^n$ a possible share vector for $\mathbf{t}$. Thus, $(\mathbf{s}, \mathbf{x}) \in V$ and $(\mathbf{t}, \mathbf{y}) \in V$. For $a, b \in \mathbb{F}_q$, we have $(a\mathbf{s} + b\mathbf{t}, a\mathbf{x} + b\mathbf{y}) \in V$, proving that

$$P((S_0, \mathbf{S}_{\mathcal{I}^*}) = (a\mathbf{s} + b\mathbf{t}, a\mathbf{x} + b\mathbf{y})) > 0.$$

Therefore, a linear combination of share vectors results in a share vector for the same linear combination of the corresponding secrets. This property makes linear secret sharing schemes very useful for secure multiparty computation and threshold cryptography.

Well known examples of linear secret sharing schemes are Shamir's secret sharing scheme and its generalization by Blakley and Meadows, described below. Assume that $n + \ell \leq q$. Let $\alpha_{0,1}, \alpha_{0,2}, \ldots, \alpha_{0,\ell}, \alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_q$ be pairwise-distinct. Fix an integer $\ell - 1 \leq k \leq n - 1$ and define the vector of random variables $\mathbf{S}$ given by selecting a polynomial uniformly at random

[3] Note that H_q is the Shannon entropy of base q and not the Rényi entropy of order q.

among the set of polynomials in $\mathbb{F}_q[X]$ of degree less than k and defining S_0 as the variable taking the value $(f(\alpha_{0,1}), f(\alpha_{0,2}), \ldots, f(\alpha_{0,\ell})) \in \mathbb{F}_q^\ell$ and each of the S_i's as the variables taking values $f(\alpha_i) \in \mathbb{F}_q$. Note that the condition $n + \ell \leq q$ can be weakened to $n \leq q$ by using an element of an extension field as a single evaluation point for the secret, rather than the elements $\alpha_{0,1}, \alpha_{0,2}, \ldots, \alpha_{0,\ell}$, as was done in for example [9].

Shamir's scheme as defined in [28] is the version with $\ell = 1$ and $\alpha_{0,1} = 0$. Blakley and Meadows' scheme is sometimes referred to as packed Shamir's scheme. It is easy to verify that this scheme is linear.

The following alternative definition of linear secret sharing schemes was given in [10]. For completion we show that the definitions are equivalent in Appendix A.

Let C_1, C_2, and L be linear codes in $\mathbb{F}_q^n$, such that $C_1 = L \oplus C_2$. Further, let $\dim L = \ell$, $\dim C_2 = k_2$, $\dim C_1 = k_1 = k_2 + \ell$, and let $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_\ell\}$ be a basis of L and $\{\mathbf{b}_{\ell+1}, \mathbf{b}_{\ell+2}, \ldots, \mathbf{b}_{k_1}\}$ be a basis of C_2. We define a linear secret sharing scheme from the nested linear codes $C_2 \subsetneq C_1$ in the following manner. Given the secret $\mathbf{s} \in \mathbb{F}_q^\ell$, choose k_2 uniformly random elements in $\mathbb{F}_q$, say $a_1, a_2, \ldots, a_{k_2}$. Then the vector

$$\mathbf{c} = s_1\mathbf{b}_1 + s_2\mathbf{b}_2 + \cdots + s_\ell\mathbf{b}_\ell + a_1\mathbf{b}_{\ell+1} + a_2\mathbf{b}_{\ell+2} + \cdots + a_{k_2}\mathbf{b}_{k_1} \in C_1$$

is called a share vector and the i'th share is defined to be the i'th entry of this vector $\mathbf{c}$. One should notice that, setting the distribution of the secret to be uniform in $\mathbb{F}_q^\ell$, this is indeed a secret sharing scheme according to our definition, since the set of all shares corresponds to a vector in $C_1 = C_2 \oplus L$ which can be projected into a unique element in L.

In secret sharing, we are interested in determining which subsets of participants are able to reconstruct the secret from their shares and which subsets are not. This leads to the definition of privacy and reconstructing sets.

Definition 2.2 (Privacy and Reconstructing set). *Let Σ be a secret sharing scheme given by the vector of random variables $\mathbf{S}$ and let $A \subseteq \mathcal{I}^*$. Then A is a privacy set if*

$$H_q(S_0|\mathbf{S}_A) = H_q(S_0),$$

and A is a reconstructing set if

$$H_q(S_0|\mathbf{S}_A) = 0.$$

As in Definition 2.1, $H_q(S_0|\mathbf{S}_A) = 0$ implies that the secret is uniquely determined by the shares held by the participants in A. On the other hand, $H_q(S_0|\mathbf{S}_A) = H_q(S_0)$ is equivalent to S_0 and $\mathbf{S}_A$ being independent. Therefore, the participants in A have no information about the secret from their shares. Additionally, we can define the information held by the participants in A using the *mutual information*

$$I_q(S_0, \mathbf{S}_A) = H_q(S_0) - H_q(S_0|\mathbf{S}_A). \tag{A.7}$$

This quantity is measured in q-bits and lies between $0 \leq I_q(S_0, \mathbf{S}_A) \leq H_q(S_0)$. It equals 0 exactly when A is a privacy set and it equals $H_q(S_0)$ exactly when A is a reconstructing set. One should notice that for linear secret sharing schemes with $\mathcal{S}_0 = \mathbb{F}_q^\ell$ we have $H_q(S_0) = \ell$. Furthermore, it is shown in [21] that for such schemes the mutual information is given by

$$I_q(S_0, \mathbf{S}_A) = \dim \pi_A(C_1) - \dim \pi_A(C_2), \tag{A.8}$$

where π_A is the *projection* $\pi_A \colon \mathbb{F}_q^n \to \mathbb{F}_q^{|A|}$ given by $\pi_A(\mathbf{c}) = \mathbf{c}_A$. Hence, we conclude that, in linear secret sharing, the information about the secret held by some set of participants, when expressed in q-bits, is always an integer between 0 and ℓ. Furthermore, we have for a subset $A \subseteq \mathcal{I}^*$ and an element $i \in \mathcal{I}^* \setminus A$ that

$$I_q(S_0, \mathbf{S}_A) \leq I_q(S_0, \mathbf{S}_{A \cup \{i\}}) \leq I_q(S_0, \mathbf{S}_A) + 1.$$

The set of all privacy sets is called the *adversary structure* of the scheme and is denoted by $\mathcal{A}(\Sigma)$. Similarly, the set of all reconstructing sets is called the *access structure* and is denoted by $\Gamma(\Sigma)$. From these definitions we introduce some thresholds for the secret sharing schemes.

Definition 2.3 (Privacy and Reconstruction Threshold). *Let Σ be a secret sharing scheme with adversary structure $\mathcal{A}(\Sigma)$ and access structure $\Gamma(\Sigma)$. The privacy threshold t for the scheme Σ is given by the maximal s such that*

$$\{A \subseteq \mathcal{I}^* : |A| = s\} \subseteq \mathcal{A}(\Sigma).$$

Similarly, the reconstruction threshold r is given by the minimal s such that

$$\{A \subseteq \mathcal{I}^* : |A| = s\} \subseteq \Gamma(\Sigma).$$

Definition 2.4 (Threshold gap). *Let Σ be a secret sharing scheme, and let t and r be the privacy and reconstruction threshold, respectively. Then*

$$g = r - t$$

is the threshold gap.

By (A.7) it can be deduced that $0 \leq t < r \leq n$, and therefore the threshold gap g is always a positive integer. Secret sharing schemes with $r = t + 1$, and therefore $g = 1$, are called threshold secret sharing schemes. As mentioned in the introduction it is often desirable to have a small g, but this will have the disadvantage that the shares are large compared to the secret, which means one has to consider this trade-off.

In a secret sharing scheme with secrets larger than the shares, some subsets of participants will obtain partial information about the secret. This gives rise to defining the partial privacy and reconstruction thresholds in a similar manner that we defined t and r.

Definition 2.5 (Partial Privacy and Reconstruction Thresholds). *The i'th partial privacy threshold of a secret sharing scheme, t_i, is given by*

$$t_i = \max\{s \mid \forall A \subseteq \mathcal{I}^*, |A| = s, I_q(S_0, \mathbf{S}_A) < i\}.$$

Similarly, the i'th partial reconstruction threshold, r_i, is given by

$$r_i = \min\{s \mid \forall A \subseteq \mathcal{I}^*, |A| = s, I_q(S_0, \mathbf{S}_A) \geq i\}.$$

This means that t_i is the maximal number such that all sets of t_i participants do not obtain i q-bits of information. On the other hand, r_i is the minimal number such that all subsets of r_i participants can reconstruct i q-bits of information.

Since the information in q-bits is always a nonnegative integer and the maximum information is ℓ we have that $t = t_1$ and $r = r_\ell$.

We will denote the *dual* of a linear code C by $C^\perp$, the *minimum distance* by $d_{\min}(C)$, the *support* by

$$\operatorname{supp}(C) = \{i : \exists (c_1, c_2, \ldots, c_n) \in C, c_i \neq 0\},$$

and the *support weight* by $w_S(C) = |\operatorname{supp}(C)|$. With these definitions, the i'th *relative generalized Hamming weight* (RGHW) is defined as

$$M_i(C_1, C_2) = \min\{w_S(D) : D \subseteq C_1, D \cap C_2 = \{0\}, \dim(D) = i\}.$$

We notice that the first RGHW is simply the minimum Hamming weight of $C_1 \setminus C_2$, which implies that $d_{\min}(C_1) \leq M_1(C_1, C_2)$. For $C_2 = \{0\}$ we have $d_{\min}(C_1) = M_1(C_1, C_2)$.

In [17, 21] it is shown that the RGHWs characterize the partial privacy and reconstruction thresholds. They showed that

$$\begin{aligned} t_i &= M_i(C_2^\perp, C_1^\perp) - 1 \\ r_i &= n - M_{\ell-i+1}(C_1, C_2) + 1. \end{aligned} \tag{A.9}$$

Further, it is shown in [22] that $M_i(C_1, C_2)$ is strictly increasing with i, which implies that $t_i < t_{i+1}$ and $r_i < r_{i+1}$, for all $i = 1, 2 \ldots \ell - 1$.

In particular, (A.9) yields

$$\begin{aligned} t &= M_1(C_2^\perp, C_1^\perp) - 1 \\ r &= n - M_1(C_1, C_2) + 1 \\ g &= n - \Big(M_1(C_1, C_2) + M_1(C_2^\perp, C_1^\perp)\Big) + 2, \end{aligned} \tag{A.10}$$

which implies that

$$\begin{aligned} t &\geq d_{\min}(C_2^\perp) - 1 \\ r &\leq n - d_{\min}(C_1) + 1 \\ g &\leq n + 2 - \Big(d_{\min}(C_1) + d_{\min}(C_2^\perp)\Big). \end{aligned} \tag{A.11}$$

3 Bounds from the Generalized Griesmer Bound

In applications, we often want secret sharing schemes where the privacy and reconstruction thresholds are close to each other, which means that we want the threshold gap to be small. From this point of view, we could refer to the bounds in (A.11) as positive bounds.

However, as it was mentioned in the introduction, there are known restrictions for how small the shares of such schemes can be when one requires a small threshold gap. These restrictions come from two sources: the relative size of the secret with respect to the shares and the relation between the size of the shares and the total number of participants.

In this section we obtain new bounds for the threshold gap of linear secret sharing schemes that depend on the two aforementioned factors simultaneously and show how they improve previous bounds in all cases.

First we recall known bounds. As in the previous section, let $\mathbb{F}_q^\ell$ be the space of secrets and let each of the shares be an element of $\mathbb{F}_q$. Then, it is well-known that $g \geq \ell$. This is a consequence of the more general result, also valid for non-linear secret sharing schemes, that $g \geq H(S_0)/H(S_i)$ for every share S_i, as proved in [4]. Coming back to the linear case, it is interesting to see this bound in the light of partial privacy and reconstruction thresholds too: in the context of Wiretap channel type II, the results in [22] imply the following bounds on t_i and r_i:

$$\begin{aligned} t_i &\leq k_2 + i - 1 \\ r_i &\geq k_2 + i, \end{aligned} \tag{A.12}$$

which combined also yield $g \geq \ell$. This bound is of the first type mentioned above: it only depends on the relation between the size of the secret and the size of the shares, but does not take into account the number of participants. The bound is attainable by the Blakley-Meadows' secret sharing scheme, but this scheme requires $n \leq q$.

In [7] lower bounds on the threshold gap depending on the number of participants and its relation to the size of the shares were derived. If we denote by

$$\begin{aligned} \mathrm{B}_{\mathrm{CCX}(1)}(n,q) &= \frac{n+2}{2q-1}, \\ \mathrm{B}_{\mathrm{CCX}(2)}(n,q,\ell) &= \frac{n+2}{2q+1} + \frac{2q}{2q+1}(\ell-1), \end{aligned}$$

then the bounds in [7] state that

$$\begin{aligned} g &\geq \mathrm{B}_{\mathrm{CCX}(1)}(n,q), \qquad \text{if } 1 \leq t < r \leq n-1 \\ g &\geq \mathrm{B}_{\mathrm{CCX}(2)}(n,q,\ell), \quad \text{if } \ell \geq 2. \end{aligned} \tag{A.13}$$

Both bounds were proved in [7] for linear secret sharing schemes. However, the first one is also valid for non-linear secret sharing schemes, as shown in [6].

Note that both bounds exclude the case $\ell = 1$ and $t = 0$, and the case $\ell = 1$ and $r = n$. This is unavoidable, since in both cases there exist secret sharing schemes where n and q are unrestricted. Indeed in the first case the scheme consisting on simply distributing the secret to all participants fulfils $r = 1$, and hence $g = 1$. On the other hand, for the second case consider additive secret sharing schemes, where the secret is the sum of all the shares, implying that $t = n - 1$. Note that the second bound implies that the bound $g \geq \ell$ we mentioned above cannot be attained with equality for all n and q as long $\ell \geq 2$.

In the following, by considering RGHWs, we construct a new lower bound on the threshold gap for linear secret sharing schemes which, as in the case of $g \geq \mathrm{B}_{\mathrm{CCX}(2)}(n, q, \ell)$, also takes both the secret and the share size into account. Additionally, we will derive limitation bounds on t_i and r_i using the same approach. We will compare the bound on the threshold gap with the bounds in (A.13), showing improvement in most cases.

We first present the following bounds on the RGHWs from [31].

Proposition 3.1 (The generalized Griesmer bound on RGHW). *Let $C_2 \subsetneq C_1$ be linear codes. For $0 \leq i \leq k_1 - k_2 = \ell$, the i'th RGHW satisfies*

$$n \geq k_2 + M_i(C_1, C_2) + \sum_{j=1}^{\ell - i} \left\lceil \frac{q-1}{q^j(q^i - 1)} M_i(C_1, C_2) \right\rceil .$$

By using that $\lceil a \rceil \geq a$, for the first m terms in the sum, and $\lceil a \rceil \geq 1$, for the remaining terms, we write

$$\begin{aligned} &n \geq k_2 + M_i(C_1, C_2) + \frac{q-1}{q^i - 1} M_i(C_1, C_2) \sum_{j=1}^{m} \frac{1}{q^j} + \ell - i - m \Leftrightarrow \\ &n \geq k_1 - i - m + M_i(C_1, C_2) + \frac{q^m - 1}{q^{m+i} - q^m} M_i(C_1, C_2), \end{aligned}$$

which is equivalent to

$$M_i(C_1, C_2) \leq \frac{q^{m+i} - q^m}{q^{m+i} - 1} \left(n - k_1 + i + m\right). \tag{A.14}$$

Similar arguments show that

$$M_i(C_2^\perp, C_1^\perp) \leq \frac{q^{m+i} - q^m}{q^{m+i} - 1} \left(k_2 + i + m\right). \tag{A.15}$$

One should notice that different choices of m lead to different bounds on the RGHWs. It is not necessarily the highest possible m which gives the best bound, and hence we need to choose the parameter m carefully in order to make the bound as good as possible.

The expressions in (A.14) and (A.15) lead to the following bounds on the partial privacy and reconstruction thresholds together with the threshold gap as well.

Theorem 3.2. *Let $C_2 \subsetneq C_1$ define a linear secret sharing scheme. Then for $i \in \{1, 2, \ldots, \ell\}$,*

$$t_i \leq \frac{q^{m+i} - q^m}{q^{m+i} - 1}(k_2 + m + i) - 1,$$
$$r_{\ell-i+1} \geq \frac{q^m - 1}{q^{m+i} - 1}n + \frac{q^{m+i} - q^m}{q^{m+i} - 1}(k_1 - m - i) + 1,$$

for all $m \in \{0, 1, \ldots, \ell - i\}$. Now, let

$$\mathrm{B_{Gr}}^{(m)}(n, q, \ell) = \frac{q^m - 1}{q^{m+1} - 1}(n + 2) + \frac{q^{m+1} - q^m}{q^{m+1} - 1}(\ell - 2m).$$

Then the threshold gap satisfies

$$g \geq \mathrm{B_{Gr}}^{(m)}(n, q, \ell),$$

for all $m \in \{0, 1, \ldots, \ell - 1\}$.

Proof. For $r_{\ell-i+1}$ we combine (A.9) and (A.14) and obtain

$$r_{\ell-i+1} \geq n - \frac{q^{m+i} - q^m}{q^{m+i} - 1}(n - k_1 + i + m) + 1 \Leftrightarrow$$
$$r_{\ell-i+1} \geq \frac{q^m - 1}{q^{m+i} - 1}n + \frac{q^{m+i} - q^m}{q^{m+i} - 1}(k_1 - m - i) + 1.$$

Similarly the bound on t_i follows by combining (A.9) with (A.15).

In order to show the bound on g, we recall from (A.10) that

$$g = n + 2 - \left(M_1(C_1, C_2) + M_1(C_2^\perp, C_1^\perp)\right),$$

which by (A.14) and (A.15) yield

$$g \geq \frac{q^m - 1}{q^{m+1} - 1}(n + 2) + \frac{q^{m+1} - q^m}{q^{m+1} - 1}(\ell - 2m)$$

for all $m \in \{0, 1, \ldots, \ell - 1\}$. □

One should notice that $g \geq {\mathrm{B_{Gr}}}^{(0)}(n,q,\ell)$ leads to the well-known bound $g \geq \ell$. Hence, for secret sharing schemes having $\ell = 1$, this bound on the threshold gap do not improve the existing bounds. However, when $\ell \geq 2$ we will show that there exist choices of m such that ${\mathrm{B_{Gr}}}^{(m)}(n,q,\ell)$ is at least as good, and in almost all cases, better than the bounds $\mathrm{B_{CCX(1)}}(n,q)$ and $\mathrm{B_{CCX(2)}}(n,q,\ell)$ in (A.13). We only consider $m = 0$, which imply $g \geq \ell$ as explained above, and $m = 1$, which imply the bound

$$\begin{aligned} g \geq {\mathrm{B_{Gr}}}^{(1)}(n,q,\ell) &= \frac{q-1}{q^2-1}(n+2) + \frac{q^2-q}{q^2-1}(\ell-2) \\ &= \frac{n+2}{q+1} + \frac{q}{q+1}(\ell-2). \end{aligned}$$

One should notice that other choices of m could improve ${\mathrm{B_{Gr}}}^{(m)}(n,q,\ell)$, but in the following theorem we show that either $m = 0$ or $m = 1$ imply a bound which is at least as good as the known bounds.

Theorem 3.3. *Let $\ell \geq 2$, then*

$$ {\mathrm{B_{Gr}}}^{(1)}(n,q,\ell) \geq \mathrm{B_{CCX(1)}}(n,q), \tag{A.16} $$

and

$$ \begin{aligned} &{\mathrm{B_{Gr}}}^{(0)}(n,q,\ell) \geq \mathrm{B_{CCX(2)}}(n,q,\ell), \text{ when } \ell \geq n-2(q-1), \\ &{\mathrm{B_{Gr}}}^{(1)}(n,q,\ell) \geq \mathrm{B_{CCX(2)}}(n,q,\ell), \text{ when } \ell \leq n-2(q-1). \end{aligned} \tag{A.17} $$

Proof. In order to prove (A.16) we consider the difference

$$\begin{aligned} {\mathrm{B_{Gr}}}^{(1)}(n,q,\ell) - \mathrm{B_{CCX(1)}}(n,q) &= \frac{n+2}{q+1} + \frac{q}{q+1}(\ell-2) - \frac{n+2}{2q-1} \\ &= \frac{q-2}{(q+1)(2q-1)}(n+2) + \frac{q}{q+1}(\ell-2) \\ &\geq 0, \end{aligned} \tag{A.18}$$

where the inequality holds for all n and q, since $\ell \geq 2$ and $q \geq 2$.

To prove (A.17) we start by considering the difference

$$\begin{aligned} {\mathrm{B_{Gr}}}^{(0)}(n,q,\ell) - \mathrm{B_{CCX(2)}}(n,q,\ell) &= \ell - \left(\frac{n+2}{2q+1} + \frac{2q}{2q+1}(\ell-1) \right) \\ &= \frac{\ell - n + 2(q-1)}{2q+1}. \end{aligned}$$

This is greater than or equal to zero if

$$\ell \geq n - 2(q-1).$$

Similarly, the difference ${\mathrm{B}_{\mathrm{Gr}}}^{(1)}(n,q,\ell) - \mathrm{B}_{\mathrm{CCX}(2)}(n,q,\ell)$ is greater than or equal to zero if

$$\begin{aligned}
0 &\leq \frac{n+2}{q+1} + \frac{q}{q+1}(\ell-2) - \left(\frac{n+2}{2q+1} + \frac{2q}{2q+1}(\ell-1)\right) \Leftrightarrow \\
0 &\leq \frac{q}{(q+1)(2q+1)}(n-\ell+2) - \frac{2q^2}{(q+1)(2q+1)} \Leftrightarrow \\
\ell &\leq n - 2(q-1),
\end{aligned}$$

which proves (A.17). □

Remark
One should notice that the inequality in (A.18) *is strict if* $\ell > 2$ *or if* $\ell \geq 2$ *and* $q > 2$ *showing that the bound* ${\mathrm{B}_{\mathrm{Gr}}}^{(1)}(n,q,\ell)$ *is sharper in these cases. Similarly, if* $\ell \neq n - 2(q-1)$ *and* $\ell \geq 2$ *there exists a choice of* m *such that* ${\mathrm{B}_{\mathrm{Gr}}}^{(m)}(n,q,\ell) > \mathrm{B}_{\mathrm{CCX}(2)}(n,q,\ell)$.

In order to illustrate how much this new bound on the threshold gap improves the existing bounds we consider an example.

Example
Let $q = 2$, $n = 100$, *and* $\ell = 10$. *Then the well-known bound* $g \geq \ell$ *yields* $g \geq 10$. *The bound* $\mathrm{B}_{\mathrm{CCX}(1)}(100,2)$ *implies* $g \geq 34$. *Similarly, the bound* $\mathrm{B}_{\mathrm{CCX}(2)}(n,q,\ell)$ *implies* $g \geq 28$, *since we can round up because the threshold gap is an integer. However, for* $m = 4$, *which is the optimal value for* m *in this example, we have* $\left\lceil {\mathrm{B}_{\mathrm{Gr}}}^{(4)}(100,2,10) \right\rceil = 51$. *Hence, we conclude that a linear secret sharing scheme over* $\mathbb{F}_2$ *with* 100 *participants for sharing* 10*-bit long secrets has a threshold gap greater than or equal to* 51. ◀

We return to the bounds in Theorem 3.2 in Section 5, where the bounds are considered asymptotic. Before that, we will focus on the bound $\mathrm{B}_{\mathrm{CCX}(1)}(n,q)$.

4 Further Bounds on the Partial Reconstruction Thresholds

Now, we will consider the bound $g \geq \mathrm{B}_{\mathrm{CCX}(1)}(n,q)$ from [7] more in depth. This bound is obtained first by proving that $r \geq \frac{n}{q} + 1$ under the assumption that $t \geq 1$, later using shortening of secret sharing schemes to show $g \geq \frac{n-t+1}{q}$ (still assuming $t \geq 1$) and finally applying this bound to the scheme and its dual, which yields $g \geq \mathrm{B}_{\mathrm{CCX}(1)}(n,q)$ under the conditions $t \geq 1, r \leq n-1$.

In this section we consider the first step of that argument (the one showing $r \geq \frac{n}{q} + 1$ if $t \geq 1$) and explore its generalization to the partial reconstruction

and privacy thresholds when $\ell > 1$. First, we show that we can obtain the same bound on r but under a weaker assumption, $t_j \geq j$. Note that $t \geq 1$ implies $t_j \geq j$ for all j, since $t_j < t_{j+1}$ as mentioned in Section 2, but the converse is not necessarily true. Furthermore, we may extend the results to obtain bounds for the partial reconstruction thresholds as well. We will derive that

$$r_i \geq \frac{n}{q^{\ell-i+1}} + 1,$$

for $i \in \{j, j+1, \ldots, \ell\}$, if $t_j \geq j$. Notice that under the assumption $t \geq 1$ we obtain that $r_i \geq \frac{n}{q^{\ell-i+1}} + 1$, for all $1 \leq i \leq \ell$. Similarly, the result $r \geq \frac{n}{q} + 1$ holds even if we only assume that $t_\ell \geq \ell$. From these results on r_i we will also generalize the bound $g \geq \mathrm{B}_{\mathrm{CCX}(1)}(n, q)$ by using shortening of codes.

Before proving the new bound for partial reconstruction thresholds we shall consider Lemma 4.1 and introduce the following notation. For a subset $V \subseteq \mathbb{F}_q^n$, an element $a \in \mathbb{F}_q$, and an index $i \in \{1, \ldots, n\}$ define

$$(V)_{a,i} = \{\mathbf{v} \in V : \pi_i(\mathbf{v}) = a\}.$$

Note that if V is a linear code, where $(V)_{a,i} \neq \emptyset$ for some $a \neq 0$, then

$$|(V)_{a,i}| = |(V)_{b,i}| \tag{A.19}$$

for all $a, b \in \mathbb{F}_q$ by the linearity of V.

Lemma 4.1. *Let $C_2 \subsetneq C_1$ define a secret sharing scheme and assume that $t_j \geq j$ for some $j \in \{1, 2, \ldots, \ell\}$. Then there exists a set $W = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{\ell-j+1}\} \subseteq L$, such that the elements in W are linearly independent, and for all $m \in \{1, 2, \ldots, n\}$ and $k \in \{1, 2, \ldots, \ell - j + 1\}$, we either have that*

$$\pi_m(C_2) = \{0\} \text{ and } \pi_m(\mathbf{v}_k + C_2) = \{0\}$$

or

$$|(C_2)_{a,m}| = |(\mathbf{v}_k + C_2)_{a,m}| = q^{k_2 - 1}, \text{ for all } a \in \mathbb{F}_q.$$

Proof. Let $B = \{m : \pi_m(C_2) = \{0\}\}$ and notice that $\pi_B(C_1) = \pi_B(L \oplus C_2) = \pi_B(L)$. For any $A \subseteq B$ we have that $I_q(S_0, \mathbf{S}_A) = \dim \pi_A(C_1) = \dim \pi_A(L) \leq \ell$. Now consider the homomorphism $\pi_B \colon L \to \mathbb{F}_q^{|B|}$, and assume that $\dim \pi_B(L) \geq j$. Then one can puncture the code $\pi_B(L)$ at a set A with cardinality j, such that $\dim \pi_A(L) = j$. This contradicts the assumption that $t_j \geq j$. Hence, $\dim \pi_B(L) < j$, which means that the kernel of π_B has dimension at least $\ell - j + 1$. Let W consists of $\ell - j + 1$ linearly independent vectors in this kernel.

let $m \in \{0,1,\ldots,\ell-1\} \setminus B$ and a $\mathbf{v}_k \in W$. By (A.19), $|(C_2)_{a-\pi_m(\mathbf{v}_k),m}| = q^{k_2-1}$, for all $a \in \mathbb{F}_q$. This shows that $|(\mathbf{v}_k + C_2)_{a,m}| \geq q^{k_2-1}$, for all $a \in \mathbb{F}_q$. However, since C_2 and $\mathbf{v}_k + C_2$ can be considered as quotient classes in C_1/C_2, we have that $|C_2| = |\mathbf{v}_k + C_2| = q^{k_2}$, implying that $|(\mathbf{v}_k + C_2)_{a,m}| = q^{k_2-1}$ for all $a \in \mathbb{F}_q$. □

We can now prove the aforementioned generalizations on r_i.

Theorem 4.2. *Let $C_2 \subsetneq C_1$ define a secret sharing scheme. If $t_j \geq j$ the thresholds r_i satisfy*

$$r_i \geq \frac{n}{q^{\ell-i+1}} + 1,$$

for $i \in \{j, j+1, \ldots, \ell\}$.

Proof. By assumption $i \geq j$, implying that $\ell - i + 1 \leq \ell - j + 1$. Therefore, by Lemma 4.1 there exists $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{\ell-i+1} \in L$ linearly independent vectors satisfying for all $m \in \{1,2,\ldots,n\}$ and $k \in \{1,2\ldots,\ell-i+1\}$, that $\pi_m(C_2) = \{0\}$ and $\pi_m(\mathbf{v}_k + C_2) = \{0\}$ or

$$|(C_2)_{a,m}| = |(\mathbf{v}_k + C_2)_{a,m}| = q^{k_2-1},$$

for all $a \in \mathbb{F}_q$. We define the vector space

$$V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1}) = \langle \mathbf{v}_1 + \mathbf{r}_1, \mathbf{v}_2 + \mathbf{r}_2, \ldots, \mathbf{v}_{\ell-i+1} + \mathbf{r}_{\ell-i+1} \rangle,$$

for some vectors $\mathbf{r}_k$, and consider the sum

$$\sum_{\mathbf{r}_1 \in C_2} \sum_{\mathbf{r}_2 \in C_2} \cdots \sum_{\mathbf{r}_{\ell-i+1} \in C_2} w_S(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})).$$

Since $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{\ell-i+1}$ are linearly independent, $\mathbf{r}_k \in C_2$, and $\mathbf{v}_k \in L$, for all k, the set $V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})$ is an $\ell - i + 1$ dimensional vector space in C_1 having only $\mathbf{0}$ in common with C_2. Therefore, we conclude that

$$w_S(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})) \geq M_{\ell-i+1}(C_1, C_2)$$

and hence

$$\begin{aligned} &\sum_{\mathbf{r}_1 \in C_2} \sum_{\mathbf{r}_2 \in C_2} \cdots \sum_{\mathbf{r}_{\ell-i+1} \in C_2} w_S(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})) \\ &\geq q^{(\ell-i+1)k_2} M_{\ell-i+1}(C_1, C_2) \\ &= q^{(\ell-i+1)k_2}(n - r_i + 1), \end{aligned} \tag{A.20}$$

where the last equality follows from (A.9). Now notice that

$$w_S(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})) = \sum_{m=1}^{n} \dim \pi_m(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})),$$

which implies that

$$\sum_{\mathbf{r}_1 \in C_2} \sum_{\mathbf{r}_2 \in C_2} \cdots \sum_{\mathbf{r}_{\ell-i+1} \in C_2} w_S(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})) =$$
$$\sum_{\mathbf{r}_1 \in C_2} \sum_{\mathbf{r}_2 \in C_2} \cdots \sum_{\mathbf{r}_{\ell-i+1} \in C_2} \sum_{m=1}^{n} \dim \pi_m(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})) =$$
$$\sum_{m=1}^{n} \sum_{\mathbf{r}_1 \in C_2} \sum_{\mathbf{r}_2 \in C_2} \cdots \sum_{\mathbf{r}_{\ell-i+1} \in C_2} \dim \pi_m(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})).$$

In each term the dimension can either be zero or one. It is zero exactly when

$$\pi_m(\mathbf{r}_k) = -\pi_m(\mathbf{v}_k)$$

for all $k = 1, 2, \ldots, \ell - i + 1$. By the assumptions on $\mathbf{v}_k$, we have that $\pi_m(\mathbf{r}_k) = -\pi_m(\mathbf{v}_k)$ for at least $q^{k_2 - 1}$ of the elements $\mathbf{r}_k \in C_2$ for a specific m. Since this holds for all $k = 1, 2, \ldots, \ell - i + 1$, we have that $\pi_m(\mathbf{r}_k) = -\pi_m(\mathbf{v}_k)$, for all k, at least $q^{(\ell-i+1)(k_2-1)}$ times. Hence,

$$\begin{aligned}
&\sum_{m=1}^{n} \sum_{\mathbf{r}_1 \in C_2} \sum_{\mathbf{r}_2 \in C_2} \cdots \sum_{\mathbf{r}_{\ell-i+1} \in C_2} \dim \pi_m(V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{\ell-i+1})) \\
&\leq \sum_{m=1}^{n} q^{(\ell-i+1)k_2} - q^{(\ell-i+1)(k_2-1)} \\
&= n q^{(\ell-i+1)k_2} \left(1 - q^{-(\ell-i+1)}\right)
\end{aligned}$$

Combining this inequality with (A.20) we obtain that

$$q^{(\ell-i+1)k_2}(n - r_i + 1) \leq n q^{(\ell-i+1)k_2} \left(1 - q^{-(\ell-i+1)}\right) \Leftrightarrow$$
$$r_i \geq \frac{n}{q^{\ell-i+1}} + 1. \qquad \square$$

We first define the notion of shortening a secret sharing scheme and prove some results on the shortened schemes parameters before we prove the bounds on the threshold gap. Let $C_2 \subseteq C_1$ define a secret sharing scheme and let $A \subseteq \mathcal{I}^*$. Now define $\bar{A} = \mathcal{I}^* \setminus A$. Then the shortened secret sharing scheme is given by the code pair $C_2^A \subsetneq C_1^A$, where

$$C_i^A = \pi_{\bar{A}}(\ker \pi_A(C_i)).$$

Lemma 4.3. *Let $A \subseteq \mathcal{I}^*$ be a set of participants in the secret sharing scheme defined by $C_2 \subsetneq C_1$ such that $I_q(S_0, \mathbf{S}_A) = m$. Denote by ℓ^A the dimension of L^A, where L^A is a code such that $C_1^A = L^A \oplus C_2^A$. Additionally, denote by t_i^A and r_i^A the partial*

privacy and reconstruction thresholds of the shortened scheme $C_2^A \subsetneq C_1^A$, and let n^A be the length of the shortened codes. Then

$$\begin{aligned} n^A &= n - |A|, \\ \ell^A &= \ell - m \\ t_i^A &\geq t_{i+m} - |A|, \\ r_i^A &\leq r_{i+m} - |A|, \end{aligned}$$

for all $i \in \{1, 2, \ldots, \ell^A\}$.

Proof. The result on n^A follows from the definition of $\pi_{\bar{A}}$. For ℓ^A we use Forney's first duality lemma [14], stating that for a code C,

$$\dim C = \dim \pi_A(C) + \dim C^A.$$

This leads to

$$\begin{aligned} \ell^A &= \dim C_1^A - \dim C_2^A \\ &= k_1 - \dim \pi_A(C_1) - k_2 + \dim \pi_A(C_2) \\ &= \ell - m. \end{aligned}$$

Now let $B \subseteq \bar{A}$ and notice, that knowing $|B|$ shares in the scheme $C_2^A \subsetneq C_1^A$ corresponds to knowing $|B \cup A| = |B| + |A|$ shares in the scheme $C_2 \subsetneq C_1$. However, for $B = \emptyset$, we have $I_q(S_0, \mathbf{S}_\emptyset) = 0$ in the shortened scheme, while it gives $I_q(S_0, \mathbf{S}_A) = m$ in the original scheme. So the information held by B in the shortened scheme equals $I_q(S_0, \mathbf{S}_{A \cup B}) - m$ in the original scheme.

If $|B| + |A| \leq t_{i+m}$, the participants will know at most $i + m - 1$ q-bits in the scheme $C_2 \subsetneq C_1$. This corresponds to knowing at most $i - 1$ q-bits in the shortened scheme, and hence $t_i^A \geq t_{i+m} - |A|$, for $i \in \{1, 2, \ldots, \ell^A\}$.

Similarly for r_i^A, if $|B| + |A| \geq r_{i+m}$, the participants in B will know at least i q-bits in the shortened scheme, showing that $r_i^A \leq r_{i+m} - |A|$. □

We will use the notation a_i and b_i to describe the gaps between t_i and t, and r and $r_{\ell-i+1}$, respectively. Therefore, denote by

$$\begin{aligned} a_i &= t_i - t - i + 1 \\ b_i &= r - r_{\ell-i+1} - i + 1. \end{aligned} \tag{A.21}$$

Since $t = t_1$, $r = r_\ell$, we have that $a_1 = b_1 = 0$. Using that t_i and r_i are strictly increasing with i we have that $a_i \geq 0$ and $b_i \geq 0$.

Another way to interpret a_i and b_i is to consider the t_i's and r_i's as a staircase. Two consecutive t_i's differ by at least one unit. The values a_i measure how different the sequence of t_i behaves from the case where all these steps

$t'_i := t_i - t_{i-1}$ are exactly 1 (this happens in the Blakley-Meadows' scheme). Indeed $a_i - a_{i-1} = t_i - t_{i-1} - 1$. So if all steps t'_i are 1, then all a_i's are 0, and in general $a_i = \sum_{j=2}^{i}(t'_i - 1)$, the sum of "all deviations from 1" up to step i. An analogous relation holds with with r_i and b_i.

This also implies that a_i and b_i are non-decreasing with i, which is useful in the following theorem.

Theorem 4.4. *Let $C_2 \subsetneq C_1$ define a secret sharing scheme. Fix some $i \in \{1, 2, \ldots, \ell\}$ and let a_i and b_i be as in* (A.21)*. If $t_i \geq i$, then the threshold gap g satisfies*

$$g \geq \frac{n-t+1}{q} + \frac{q-1}{q}a_i. \tag{A.22}$$

If $r_{\ell-i+1} \leq n - i$, then the threshold gap g satisfies

$$g \geq \frac{r+1}{q} + \frac{q-1}{q}b_i. \tag{A.23}$$

If both $t_i \geq i$ and $r_{\ell-i+1} \leq n - i$, then the threshold gap g satisfies

$$g \geq \frac{n+2}{2q-1} + \frac{q-1}{2q-1}(a_i + b_i).$$

Proof. Choose A such that $|A| = t - 1 + a_i$. Hence, the shortened scheme given by $C_2^A \subsetneq C_1^A$ has parameters $n^A = n - t + 1 - a_i$, $r^A \leq r - t + 1 - a_i$, and $t_i^A \geq i$ by Lemma 4.3. By Theorem 4.2 and Lemma 4.3, the threshold gap now satisfies

$$\begin{aligned} g = r - t &\geq r^A + a_i - 1 \\ &\geq \frac{n^A}{q} + a_i \\ &= \frac{n-t+1-a_i}{q} + a_i \\ &= \frac{n-t+1}{q} + \frac{q-1}{q}a_i. \end{aligned}$$

By (A.9) and (A.10) one has that the dual scheme has thresholds $t_i^\perp = n - r_{\ell-i+1}$ and $r_{\ell-i+1}^\perp = n - t_i$. Therefore, the threshold gap of the dual scheme is the same as for the original and a_i of the dual equals b_i. We can use the bound in (A.22) on the dual scheme if it holds that $t_i^\perp \geq i$, but this is equivalent to the assumption $r_{\ell-i+1} \leq n - i$. Therefore, we obtain

$$g \geq \frac{n - t^\perp + 1}{q} + \frac{q-1}{q}b_i = \frac{r+1}{q} + \frac{q-1}{q}b_i.$$

The last bound is obtained by summing the bounds in (A.22) and (A.23).

$$2g \geq \frac{n-t+1+r+1}{q} + \frac{q-1}{q}(a_i+b_i) = \frac{n+g+2}{q} + \frac{q-1}{q}(a_i+b_i) \Leftrightarrow$$
$$g \geq \frac{n+2}{2q-1} + \frac{q-1}{2q-1}(a_i+b_i). \qquad \square$$

The bounds in [7], stating that

$$g \geq \frac{n-t+1}{q}, \quad g \geq \frac{r+1}{q}, \quad g \geq \mathrm{B}_{\mathrm{CCX}(1)}(n,q),$$

if $t \geq 1$ and $r \leq n-1$, are a particular case of this theorem.

In the following example we will consider a scheme attaining the bounds in Theorem 4.2. We will also note in which cases, for this particular example, the bounds from Theorem 4.4 are sharp. Similar examples of codes attaining the bound $g \geq \frac{n-t+1}{q}$ can be found in [27].

Example
Let $\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_{q^\ell}^T$ be all possible vectors in $\mathbb{F}_q^\ell$, and define the code C_1 from the $(\ell+1) \times q^\ell$ generator matrix

$$G = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_{q^\ell} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

where C_2 is generated by the last all-one row. Then clearly $I_q(S_0, S_j) = 1-1 = 0$ by (A.8) *for all $1 \leq j \leq n$, meaning that $t \geq 1$. In fact $t_i = i$ for all i in this example. This comes from the fact that the canonical basis vectors and the all zero vector lie in $\mathbb{F}_q^\ell$. The set of participants corresponding to these vectors is a set with cardinality $\ell+1$, which can reconstruct all ℓ q-bits. Therefore, $t_\ell \leq \ell$, and from this we conclude $t_i = i$. Hence, we will show that the bounds in Theorem 4.2 are sharp for this secret sharing scheme, that is*

$$r_i = \frac{n}{q^{\ell-i+1}} + 1 = \frac{q^\ell}{q^{\ell-i+1}} + 1 = q^{i-1} + 1.$$

We consider a set of participants A knowing $i-1$ q-bits, and derive that $|A| \leq q^{i-1}$, which means that $q^{i-1}+1$ participants will know at least i q-bits, and hence $r_i \leq q^{i-1}+1$. Combining this with Theorem 4.2 yields $r_i = q^{i-1}+1$.

Thus, assume that A knows $i-1$ q-bits and assume for contradiction that $|A| > q^{i-1}$. First notice that by (A.8)*, we have*

$$\dim \pi_A(C_1) = i - 1 + \dim \pi_A(C_2) = i$$

On the other hand, we can determine the dimension of $\pi_A(C_1)$ in another way by considering the generator matrix. Let $A = \{j_1, j_2, \ldots, j_k\}$, where $k > q^{i-1}$. Denote by

$$G_A = \begin{bmatrix} \mathbf{v}_{j_1} & \mathbf{v}_{j_2} & \cdots & \mathbf{v}_{j_k} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

and

$$G'_A = \begin{bmatrix} \mathbf{v}_{j_1} & \mathbf{v}_{j_2} & \cdots & \mathbf{v}_{j_k} \end{bmatrix}.$$

The rank of G_A equals $\dim \pi_A(C_1)$. Clearly, $\operatorname{rank}(G'_A) \leq \operatorname{rank}(G_A)$, but since $|A| > q^{i-1}$, we obtain $\operatorname{rank}(G'_A) = i$. This means that we have i linearly independent columns, and without loss of generality we denote these by $\mathbf{v}_{j_1}, \mathbf{v}_{j_2}, \ldots, \mathbf{v}_{j_i}$. Hence, all the columns in G'_A must be of the form

$$a_1\mathbf{v}_{j_1} + a_2\mathbf{v}_{j_2} + \cdots + a_i\mathbf{v}_{j_i},$$

for some $a_k \in \mathbb{F}_q$. However, since $\operatorname{rank}(G'_A) = \operatorname{rank}(G_A)$ one has that $\sum_{k=1}^{i} a_k = 1$. Therefore, $|A| \leq q^{i-1}$, contradicting the assumption on A.

From this we conclude that $r_i = q^{i-1} + 1$, showing that the bound in Theorem 4.2 is sharp for this example. The threshold gap in this example can also be determined; $g = r - t = q^{\ell-1} + 1 - 1 = q^{\ell-1}$. Now considering the bounds in Theorem 4.4 we show that some of these bounds are attained in this case as well. Since $t_i = i$, we have that $a_i = 0$ for all i in Theorem 4.4. Thus,

$$\frac{n - t + 1}{q} = \frac{q^\ell - 1 + 1}{q} = q^{\ell-1} = g,$$

which shows that the inequality in (A.22) *is sharp. We consider the inequality in* (A.23) *as well, and since b_i is non-decreasing we determine b_ℓ in order to make the bound as good as possible.*

$$b_\ell = r - r_1 - \ell + 1 = q^{\ell-1} + 1 - 2 - \ell + 1 = q^{\ell-1} - \ell.$$

Hence, the bound states

$$\begin{aligned} g \geq \frac{r+1}{q} + \frac{q-1}{q} b_\ell &= \frac{q^{\ell-1} + 2}{q} + \frac{q-1}{q}\left(q^{\ell-1} - \ell\right) \\ &= q^{\ell-1} - \ell + \frac{2+\ell}{q}. \end{aligned}$$

Note that there is no contradiction with $g = q^{\ell-1}$, since the bound does not hold for $\ell = 1$ and $q = 2$. When $\ell = 1$ we require, in order to use the bound, that $r \leq n - 1$, but $n = 2^1 = 2$ and $r = 2^{1-1} + 1 = 2$ in this case.

For this bound to be sharp $\ell = \frac{2+\ell}{q}$, which implies $\ell(q-1) = 2$. Therefore, this bound is attained in the case where $\ell = 2$ and $q = 2$. The same would then hold for the last bound in Theorem 4.4, since this bound is obtained by summing the two previous bounds. ◀

5 Asymptotic Comparisons

In this section we analyse the asymptotic behaviour of the bounds presented in Theorem 3.2 when the number of players n grows, and the size of the secret ℓ grows as a linear function of n.

We assume the setting considered in [16]; let $\{\Sigma_j\}_{j=1}^{\infty}$ denote an infinite family of $\mathbb{F}_q$-linear secret sharing schemes with increasing number of participants n_j and where Σ_j has secrets in $\mathbb{F}_q^{\ell_j}$, so that $\{\ell_j\}_{j=1}^{\infty}$ is a monotonely increasing sequence such that

$$\lim_{j\to\infty} \frac{\ell_j}{n_j} = L \text{ , for some } L \in \mathbb{R} \text{ with } 0 < L < 1.$$

To simplify, we assume that if we denote $k_1(j), k_2(j)$ the dimensions of the codes C_1 and C_2 in any nested code pair representation of $\{\Sigma_j\}$, then $\frac{k_1(j)}{n_j}$ converges to some $R_1 \in \mathbb{R}$ and $\frac{k_2(j)}{n_j}$ converges to some $R_2 \in \mathbb{R}$. Clearly, $L = R_1 - R_2$ since $\ell_j = k_1(j) - k_2(j)$.

Denote the privacy threshold and reconstruction threshold of Σ_j by $t(\Sigma_j)$ and $r(\Sigma_j)$ respectively. Furthermore, we define

$$\Omega^{(1)} = \liminf_{j\to\infty} \frac{t(\Sigma_j)}{n_j} \text{ and } \Omega^{(2)} = \limsup_{j\to\infty} \frac{r(\Sigma_j)}{n_j}.$$

Additionally, we denote the threshold gap of Σ_j by $g(\Sigma_j)$ and define

$$\Omega^{(3)} = \limsup_{j\to\infty} \frac{g(\Sigma_j)}{n_j}. \tag{A.24}$$

Note that $\Omega^{(3)}$ does not necessarily equal $\Omega^{(2)} - \Omega^{(1)}$. Indeed, in general we have

$$\begin{aligned} \Omega^{(3)} &= \limsup_{j\to\infty} \left(\frac{r(\Sigma_j)}{n_j} - \frac{t(\Sigma_j)}{n_j} \right) \\ &\leq \limsup_{j\to\infty} \frac{r(\Sigma_j)}{n_j} - \liminf_{j\to\infty} \frac{t(\Sigma_j)}{n_j} = \Omega^{(2)} - \Omega^{(1)}. \end{aligned} \tag{A.25}$$

but equality may not hold as the example illustrated in Figure A.1 shows. The lower dashed line illustrates $\Omega^{(1)}$ and the top dashed line $\Omega^{(2)}$. As we can see in the figure, the difference between $\Omega^{(2)}$ and $\Omega^{(1)}$ is larger than the actual threshold gap, which is the black vertical dashed line.

We now present the asymptotic version of the bound $g \geq {\mathrm{B_{Gr}}}^{(m)}(n, q, \ell)$ together with bounds on $\Omega^{(1)}$ and $\Omega^{(2)}$.

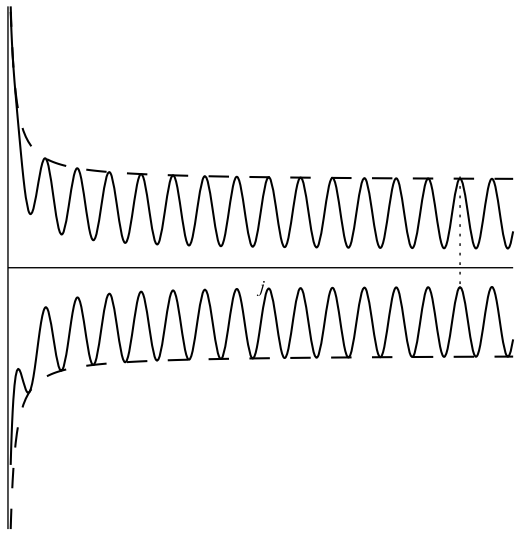

Fig. A.1: Illustration of $\Omega^{(3)}$. The solid lines illustrate $\frac{t(\Sigma_j)}{n_j}$ and $\frac{r(\Sigma_j)}{n_j}$, both as a function of j, and the black vertical dashed line illustrates $\frac{g(\Sigma_j)}{n_j}$ for a specific j.

Theorem 5.1. *Let $\{\Sigma_j\}$ be a family of secret sharing schemes over $\mathbb{F}_q$ as above. We have*

$$\begin{aligned}\Omega^{(1)} &\leq \frac{q-1}{q}R_2,\\ \Omega^{(2)} &\geq \frac{1}{q}+\frac{q-1}{q}R_1,\\ \Omega^{(3)} &\geq \frac{1}{q}+\frac{q-1}{q}L.\end{aligned} \tag{A.26}$$

Proof. We have by Theorem 3.2 that

$$\begin{aligned}\frac{t(\Sigma_j)}{n_j} &\leq \frac{q^{m_j+1}-q^{m_j}}{q^{m_j+1}-1}\left(\frac{k_2(j)}{n_j}+\frac{m_j}{n_j}+\frac{1}{n_j}\right)-\frac{1}{n_j},\\ \frac{r(\Sigma_j)}{n_j} &\geq \frac{q^{m_j}-1}{q^{m_j+1}-1}+\frac{q^{m_j+1}-q^{m_j}}{q^{m_j+1}-1}\left(\frac{k_1(j)}{n_j}-\frac{m_j}{n_j}-\frac{1}{n_j}\right)+\frac{1}{n_j},\\ \frac{g(\Sigma_j)}{n_j} &\geq \frac{q^{m_j}-1}{q^{m_j+1}-1}\left(1+\frac{2}{n_j}\right)+\frac{q^{m_j+1}-q^{m_j}}{q^{m_j+1}-1}\left(\frac{\ell_j}{n_j}-2\frac{m_j}{n_j}\right)\end{aligned} \tag{A.27}$$

where m_j is any choice of m for Σ_j in Theorem 3.2, i.e. $m_j \in \{0,\ldots,\ell_j-1\}$. In particular we can choose m_j as a function of n_j such that $m_j = o(n_j)$ but

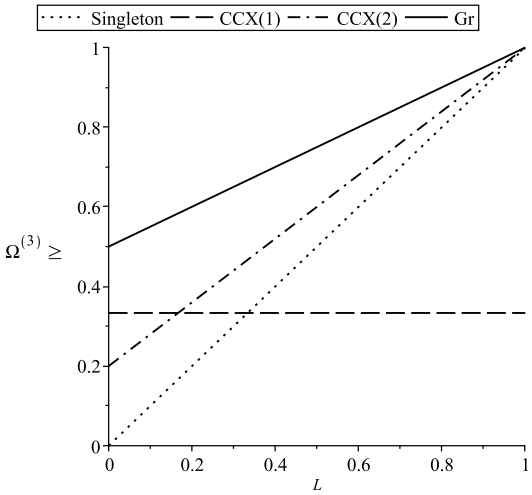

Fig. A.2: Comparison of asymptotic bounds on the threshold gap for $q = 2$.

still $\lim_{j\to\infty} m_j = \infty$, for example $m_j = \min\{\ell_j - 1, \lfloor \log n_j \rfloor\}$ (where $L > 0$ implies that for large enough j, we simply have $m_j = \lfloor \log n_j \rfloor$).

Letting j tend to infinity in (A.27) with such selection of m_j, we obtain the claimed result. □

It is not difficult to see that the bound

$$\Omega^{(3)} \geq \frac{1}{q} + \frac{q-1}{q} L$$

that we just derived is strictly tighter than the asymptotic versions of the bounds $g \geq \ell$, $\mathrm{B}_{\mathrm{CCX}(1)}(n, q)$, and $\mathrm{B}_{\mathrm{CCX}(2)}(n, q, \ell)$, which are respectively

$$\Omega^{(3)} \geq L,\ \Omega^{(3)} \geq \frac{1}{2q-1},\ \Omega^{(3)} \geq \frac{1}{2q+1} + \frac{2q}{2q+1} L,$$

for any q and any $0 < L < 1$. We show these four bounds on $\Omega^{(3)}$ in Figure A.2 for the case $q = 2$.

In the rest of this section, we collect known results on upper bounds for $\Omega^{(3)}$, and compare them with the lower bounds we have obtained.

We will consider algebraic geometric codes and random codes. As far as the authors know, secret sharing schemes from algebraic geometric codes yield the smallest values of $\Omega^{(3)}$ when the finite field $\mathbb{F}_q$ is sufficiently large, while random codes give smaller $\Omega^{(3)}$ for small q.

An algebraic geometric evaluation code is defined from an algebraic function field F, a divisor G of F (which determines a space of functions to be

evaluated) and a set of rational places in F (as evaluation points), the latter usually represented by a divisor D. We remit the reader to [29] for details. Secret sharing schemes defined from algebraic geometric codes where first considered in [8]. We here use the construction from [10], defined by a nested code pair where both codes are algebraic geometric codes defined using the same function field F and the set of all rational places as evaluation points, but different divisors G_1, G_2. Such secret sharing schemes then satisfy $t \geq k_2 - \mathcal{G}$ and $r \leq k_1 + \mathcal{G}$, where $\mathcal{G}$ is the genus of the function field, and k_1, k_2 are as always the dimensions of the two linear codes. Moreover, the length of these codes (and hence the number of shares n) is the number of rational places of the function field.

Consider now an optimal tower of function fields $\{F_j\}_{j=1}^{\infty}$, i.e. $\lim_{j\to\infty} \frac{N_j}{\mathcal{G}_j} = A(q)$ where $N_j, \mathcal{G}_j$ are respectively the number of rational places and genus in F_j and $A(q)$ is the so-called Ihara's constant and $\mathcal{G}_j$ is the genus. Applying the construction described above gives a family of secret sharing schemes such that

$$\Omega^{(1)} \geq R_2 - \frac{1}{A(q)},$$
$$\Omega^{(2)} \leq R_1 + \frac{1}{A(q)},$$

see [16]. By (A.25) this implies

$$\Omega^{(3)} \leq L + \frac{2}{A(q)}. \tag{A.28}$$

While Ihara's constant $A(q)$ has not been determined for every q, we sum up some known facts next. First $A(q) > 0$ for all q, and in fact $A(q) \geq c \log(q)$ for some constant c, see for instance [24]. On the other hand, $A(q) \leq \sqrt{q} - 1$, see [30]. If q is a perfect square, it was shown in [18] that $A(q) = \sqrt{q} - 1$. Furthermore, Garcia and Stichtenoth gave a explicit construction [15] of an optimal tower of function fields in this case.

Consequently, for small values of q, the bound in (A.28) is trivial since $A(q) \leq 2$. For large enough q, however, we have $A(q) > 2$ (for example for q square with $q \geq 16$).

We observe the following, in relation with the lower bounds: the difference between the upper bound (A.28) and the lower bound from Theorem 5.1 is

$$\frac{2}{A(q)} - \frac{1}{q}(1 - L).$$

Note that the term $\frac{1}{q}(1 - L)$ is precisely what the bound in Theorem 5.1 has gained with respect to the lower bound $\Omega^{(3)} \geq L$. However this factor is

overshadowed by the considerably larger factor $2/A(q)$. It is an interesting open question to bring these bounds together, by either proving stronger lower bounds or improving the known constructions.

For small finite fields, the bounds in (A.28) are trivial and the best upper bounds are achieved by inifinite families of secret sharing schemes based on random codes. We follow the results from [10]. The following result is a consequence of the fact that random codes are on the Gilbert-Varshamov bound.

Proposition 5.2. *Let C be a random variable with the uniform distribution taking values in the set of all $[n,k]$ linear codes over $\mathbb{F}_q$, and let $0 < d, d^\perp < (1-\frac{1}{q})n$ be integers. For a realization of $\mathsf{C} = C$ we then have*

$$P(d_{\min}(C) < d) \leq q^{k+n(H_q(\frac{d}{n})-1)}$$
$$P(d_{\min}(C^\perp) < d^\perp) \leq q^{nH_q(\frac{d^\perp}{n})-k},$$

where $H_q(x) = x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x)$ is the q-ary entropy function.

The q-ary entropy function H_q is strictly increasing and therefore injective in the interval $[0, 1-\frac{1}{q}]$ (we define $H_q(0) = 0$ as usual) and there its image is the interval $[0,1]$. We can therefore define the inverse $H_q^{-1} : [0,1] \to [0, 1-\frac{1}{q}]$. With this definition in mind, we can choose $\frac{d}{n} = H_q^{-1}(1-\frac{k}{n}-\varepsilon')$ and $\frac{d^\perp}{n} = H_q^{-1}(\frac{k}{n}-\varepsilon')$ for some $\varepsilon' > 0$ and both probabilities above become lower than or equal to $q^{-\varepsilon' n}$.

A linear code C_1 can be chosen uniformly at random from all $[n,k_1]$ linear codes by rejection sampling of the elements $\mathbf{b}_i$ in its basis. The subcode $C_2 \subsetneq C_1$ generated by the last k_2 basis elements is then also uniformly random among all $[n,k_2]$ linear codes. Combining Proposition 5.2 and the comment below with the inequalities in (A.11), we obtain

$$\begin{aligned} P\left(\frac{r}{n} < 1 - H_q^{-1}\left(1-\frac{k_1}{n}-\varepsilon'\right) + \frac{1}{n}\right) &\geq 1 - q^{-\varepsilon' n} \\ P\left(\frac{t}{n} > H_q^{-1}\left(\frac{k_2}{n}-\varepsilon'\right) - \frac{1}{n}\right) &\geq 1 - q^{-\varepsilon' n}, \end{aligned} \tag{A.29}$$

For any fixed $\varepsilon' > 0$ the probabilities in (A.29) are larger than 0, and hence by the probabilistic method we conclude that there exists an infinite family of secret sharing schemes with

$$\Omega^{(3)} \leq 1 - H_q^{-1}\left(1-R_1-\varepsilon'\right) - H_q^{-1}\left(R_2-\varepsilon'\right).$$

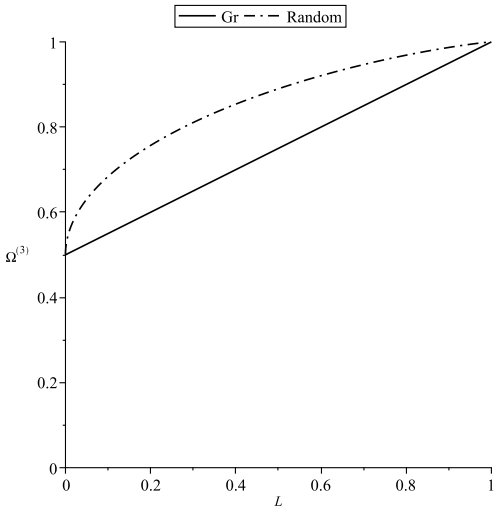

Fig. A.3: Comparison of asymptotic lower and upper bounds (when $R_1 = 1$) on the threshold gap for $q = 2$.

For a fixed $L = R_1 - R_2$, the smallest value of the right-hand side is attained by setting R_1 close to 1 (and hence R_2 close to $1 - L$) or, symmetrically, setting R_2 close to 0 (and $R_1 = L$). In that case, the inequality becomes

$$\Omega^{(3)} \leq 1 - H_q^{-1}(1 - L) + \varepsilon,$$

for any $\varepsilon > 0$.

In Figure A.3, we compare this upper bound, in the case $q = 2$, with our lower bound from equation (A.26).

At last, we make the following remark on the asymptotic behaviors of the partial privacy and reconstruction thresholds, which is one of the main focus in the work [16]. There the authors define

$$\Lambda^{(1)}(\delta_1) = \sup \left\{ \liminf_{j\to\infty} \frac{t_{m_1(j)}}{n_j} \middle| \right.$$
$$\left. \{m_1(j)\}_{j=1}^{\infty}, 1 \leq m_1(j) \leq \ell_j, \lim_{j\to\infty} \frac{m_1(j)}{n_j} = \delta_1 L \right\},$$
$$\Lambda^{(2)}(\delta_2) = \inf \left\{ \limsup_{j\to\infty} \frac{r_{\ell_j - m_2(j)+1}}{n_j} \middle| \right.$$
$$\left. \{m_2(j)\}_{j=1}^{\infty}, 1 \leq m_2(j) \leq \ell_j, \lim_{j\to\infty} \frac{m_2(j)}{n_j} = \delta_2 L \right\}.$$

That is, asymptotically, no fraction less than $\Lambda^{(1)}(\delta_1)$ of the participants holds more than a fraction of δ_1 of the secret. Similarly, $\Lambda^{(2)}(\delta_2)$ ensures that

asymptotically a fraction of $\Lambda^{(2)}(\delta_2)$ of the participants will be able to reconstruct a fraction of $1-\delta_2$ of the secret.

In [16] the gap between the limitations on $\Lambda^{(1)}(\delta_1)$ and $\Lambda^{(2)}(\delta_2)$ and what is possible to achieve is almost closed. The limitations considered there are derived from (A.12), i.e.,

$$\begin{aligned}\Lambda^{(1)}(\delta_1) &\leq R_2 + \delta_1 L,\\ \Lambda^{(2)}(\delta_2) &\geq R_1 - \delta_2 L.\end{aligned} \tag{A.30}$$

We can obtain the same bounds from Theorem 3.2 by setting $m = 0$. However, contrary to what happens in the proof of Theorem 5.1, choosing m as a small fraction of ℓ will not improve this bound in this case.

A Linear Secret Sharing

Proposition A.1. *A secret sharing scheme based on a nested code pair $C_2 \subseteq C_1$ is a linear secret sharing scheme.*

Proof. Clearly, $\mathcal{S}_i$ is a $\mathbb{F}_q$-linear subspace. So we need to show that $\mathbf{S}$ is uniformly distributed on some subspace $V \subseteq \mathcal{S}_0 \times \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. Indeed, this is the case for

$$V = \{(\mathbf{s}, \mathbf{c}) : \mathbf{s} \in \mathbb{F}_q^{\ell} \quad \text{and} \quad \mathbf{c} \in (s_1\mathbf{b}_1 + s_2\mathbf{b}_2 + \cdots + s_\ell \mathbf{b}_\ell) + C_2\}.$$

First of all it is a subspace, so we show that $\mathbf{S}$ is uniformly distributed on V.

$$\begin{aligned}P(\mathbf{S} = (\mathbf{s}, c_1, c_2, \ldots, c_n)) &= P(\mathbf{S}_{\mathcal{I}^*} = (c_1, c_2, \ldots, c_n) | S_0 = \mathbf{s}) P(S_0 = \mathbf{s})\\ &= \frac{1}{q^{k_2}} \frac{1}{q^{\ell}}\end{aligned}$$

for $\mathbf{S}$ in V, showing that this construction resulting in a linear secret sharing scheme. □

Proposition A.2. *All linear secret sharing schemes can be represented by a nested code pair $C_2 \subsetneq C_1$.*

Proof. Let a linear secret sharing scheme be given by $\mathbf{S}$. Let V be the subspace such that $\mathbf{S}$ is uniformly distributed on V, and define

$$\begin{aligned}C_2 &= \{\mathbf{c} : (\mathbf{0}, \mathbf{c}) \in V \text{ where } \mathbf{0} \in \mathcal{S}_0 \text{ and } \mathbf{c} \in \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \mathcal{S}_n\}\\ C_1 &= \{\mathbf{c} : (\mathbf{s}, \mathbf{c}) \in V \text{ where } \mathbf{s} \in \mathcal{S}_0 \text{ and } \mathbf{c} \in \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \mathcal{S}_n\}\end{aligned}$$

Clearly, $C_2 \subseteq C_1$ and both are linear subspaces and therefore linear codes. Denote by k_2 the dimension of C_2 and k_1 the dimension of C_1. Since both $\mathbf{S}$

and S_0 are uniformly distributed we also obtain that $\mathbf{S}|S_0 = \mathbf{0}$ is uniformly distributed on C_2 with probability function

$$p_{\mathbf{S}}(\mathbf{S}|S_0) = \frac{p_{\mathbf{S}}(\mathbf{S})}{p_{S_0}(S_0)} = \frac{\frac{1}{q^{k_1}}}{\frac{1}{q^{\ell}}} = \frac{1}{q^{k_1-\ell}}.$$

Hence, $k_2 = k_1 - \ell$. Because all the shares uniquely determine the secret in a secret sharing scheme, there is a one-to-one correspondence between C_1 and V, showing that for any possible outcome of $\mathbf{S}$ there is a corresponding element in C_1. Therefore, we can represent the scheme using the nested codes $C_2 \subsetneq C_1$. □

References

[1] A. Beimel, M. Burmester, Y. Desmedt, and E. Kushilevitz, "Computing functions of a shared secret," *SIAM J. Discrete Math.*, vol. 13, no. 3, pp. 324–345, 2000. [Online]. Available: http://dx.doi.org/10.1137/S0895480195288819

[2] G. R. Blakley, "Safeguarding cryptographic keys," *Managing Requirements Knowledge, International Workshop on*, vol. 00, p. 313, 1979.

[3] G. R. Blakley and C. Meadows, "Security of ramp schemes," *Advances in Cryptology: Proceedings of CRYPTO 84*, pp. 242–268, 1985.

[4] C. Blundo, A. D. Santis, and U. Vaccaro, "Efficient sharing of many secrets," *Proceedings of the 10th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 692–703, 1993. [Online]. Available: http://dl.acm.org/citation.cfm?id=646509.694823

[5] C. Blundo, A. D. Santis, and U. Vaccaro, "On secret sharing schemes," *Inf. Process. Lett.*, vol. 65, no. 1, pp. 25–32, 1998. [Online]. Available: https://doi.org/10.1016/S0020-0190(97)00194-4

[6] A. Bogdanov, S. Guo, and I. Komargodski, "Threshold secret sharing requires a linear size alphabet," *Theory of Cryptography*, pp. 471–484, 2016.

[7] I. Cascudo, R. Cramer, and C. Xing, "Bounds on the threshold gap in secret sharing and its applications," *IEEE Transactions on Information Theory*, vol. 59, no. 9, pp. 5600–5612, September 2013.

[8] H. Chen and R. Cramer, "Algebraic geometric secret sharing schemes and secure multi-party computations over small fields," *Advances in Cryptology - CRYPTO 2006*, pp. 521–536, August 2006.

[9] H. Chen, R. Cramer, R. de Haan, and I. C. Pueyo, "Strongly multiplicative ramp schemes from high degree rational points on curves," *Advances in Cryptology – EUROCRYPT 2008*, pp. 451–470, 2008.

[10] H. Chen, R. Cramer, S. Goldwasser, R. de Haan, and V. Vaikuntanathan, "Secure computation from random error correcting codes," *Advances in Cryptology - EUROCRYPT 2007*, pp. 291–310, 2007.

[11] R. Cramer, I. B. Damgård, N. Döttling, S. Fehr, and G. Spini, "Linear secret sharing schemes from error correcting codes and universal hash functions," *Advances in cryptology—EUROCRYPT 2015. Part II*, vol. 9057, pp. 313–336, 2015. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-46803-6_11

[12] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[13] Y. Dodis, A. Sahai, and A. Smith, "On perfect and adaptive security in exposure-resilient cryptography," in *Advances in Cryptology — EUROCRYPT 2001*, B. Pfitzmann, Ed. Springer Berlin Heidelberg, 2001, pp. 301–324.

[14] G. D. Forney, "Dimension/length profiles and trellis complexity of linear block codes," *IEEE International Symposium on Information Theory*, vol. 40, November 1994. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=340452

[15] A. Garcia and H. Stichtenoth, "On the asymptotic behaviour of some towers of function fields over finite fields," *Journal of Number Theory*, vol. 61, no. 2, pp. 248–273, 1996.

[16] O. Geil, S. Martin, U. Martínez-Peñas, R. Matsumoto, and D. Ruano, "On asymptotically good ramp secret sharing schemes," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E100.A, no. 12, pp. 2699–2708, 2017.

[17] O. Geil, S. Martin, R. Matsumoto, D. Ruano, and Y. Luo, "Relative generalized hamming weights of one-point algebraic geometric codes," *IEEE Transactions on Information Theory*, vol. 60, no. 10, pp. 5938–5949, October 2014.

[18] Y. Ihara, "Some remarks on the number of rational points of algebraic curves over finite fields," *J. Fac. Sci. Univ. Tokyo Sect. IA Math.*, vol. 28, no. 3, pp. 721–724 (1982), 1981.

[19] W.-A. Jackson and K. M. Martin, "A combinatorial interpretation of ramp schemes," *Australasian J. Combinatorics*, vol. 14, pp. 51–60, 1996.

[20] J. Kilian and N. Nisan, "Unpublished result," 1991.

[21] J. Kurihara, T. Uyematsu, and R. Matsumoto, "Secret sharing schemes based on linear codes can be precisely characterized by the relative generalized hamming weight," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 11, pp. 2067–2075, 2012.

[22] Y. Luo, C. Mitrpant, A. J. H. Vinck, and K. Chen, "Some new characters on the wire-tap channel of type ii," *IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 1222–1229, March 2005.

[23] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. North-Holland Publishing Company, 1977, vol. 16.

[24] H. Niederreiter and C. Xing, *Rational Points on Curves over Finite Fields: Theory and Applications*. New York, NY, USA: Cambridge University Press, 2001.

[25] W. Ogata and K. Kurosawa, "Some basic properties of general nonperfect secret sharing schemes," *j-jucs*, vol. 4, no. 8, pp. 690–704, August 1998. [Online]. Available: http://www.jucs.org/jucs_4_8/some_basic_properties_of

[26] W. Ogata, K. Kurosawa, and S. Tsujii, "Nonperfect secret sharing schemes," in *Advances in Cryptology — AUSCRYPT '92*, J. Seberry and Y. Zheng, Eds. Springer Berlin Heidelberg, 1992, pp. 56–66.

[27] M. B. Paterson and D. R. Stinson, "A simple combinatorial treatment of constructions and threshold gaps of ramp schemes," *Cryptography Commun.*, vol. 5, no. 4, pp. 229–240, December 2013. [Online]. Available: http://dx.doi.org/10.1007/s12095-013-0082-1

[28] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, November 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176

[29] H. Stichtenoth, *Algebraic Function Fields and Codes*, 2nd ed., ser. Graduate Texts in Mathematics. Springer, 2009.

[30] S. G. Vlèduts and V. G. Drinfeld, "The number of points of an algebraic curve," *Funktsional. Anal. i Prilozhen.*, vol. 17, no. 1, pp. 68–69, 1983.

[31] Z. Zhuang, Y. Luo, A. J. H. Vinck, and B. Dai, "Some new bounds on relative generalized hamming weight," *2011 IEEE 13th International Conference on Communication Technology*, pp. 971–974, September 2011.

Paper B

Actively Secure OT-Extension from q-ary Linear Codes

Ignacio Cascudo, René Bødker Christensen, and Jaron Skovsted Gundersen

The paper was presented at the 11^{th} *Conference on Security and Cryptography for Networks Conference 2018* and is published in *Lecture Notes in Computer Science* Vol. 11035, pp. 333–348, 2018.

The layout has been revised.

Abstract

We consider recent constructions of 1*-out-of-N OT-extension from Kolesnikov and Kumaresan (CRYPTO 2013) and from Orrú et al. (CT-RSA 2017), based on binary error-correcting codes. We generalize their constructions such that q-ary codes can be used for any prime power q. This allows to reduce the number of base* 1*-out-of-*2 *OT's that are needed to instantiate the construction for any value of N, at the cost of increasing the complexity of the remaining part of the protocol. We analyze these trade-offs in some concrete cases.*

1 Introduction

A K-out-of-N oblivious transfer, or $\binom{N}{K}$-OT, is a cryptographic primitive that allows a sender to input N messages and a receiver to learn exactly K of these with neither the receiver revealing which messages he has chosen to learn nor the sender revealing the other $N - K$ input messages. This is a fundamental cryptographic primitive in the area of secure multiparty computation, and in fact [9] showed that any protocol for secure multiparty computation can be implemented if the OT functionality is available. However, the results in [6] indicate that OT is very likely to require a public key cryptosystem, and therefore implementing OT is relatively expensive. Unfortunately, well-known protocols such as Yao's garbled circuits [13] and the GMW-compiler [5] rely on using a large number of independent instances of OT. It is therefore of interest to reduce the number of OT's used in a protocol in an attempt to reduce the overall cost. This can be done using what is called OT-extensions, where a large number of OT's are simulated by a much smaller number of base OT's together with the use of cheaper symmetric crypto primitives, such as pseudorandom generators.

Beaver showed in [1] that OT-extension is indeed possible, but it was not before 2003 that an efficient $\binom{2}{1}$-OT-extension protocol was presented by Ishai et al. in [7]. In addition, while this protocol had security against passive adversaries, subsequent has work showed that active security can be achieved at a small additional cost [8].

In [10], Kolesnikov and Kumaresan noticed that Ishai et al. were in essence relying on the fact that the receiver encodes its input as a codeword in a repetition code, and therefore one can generalize their idea by using other codes, such as the Walsh-Hadamard code, which not only obtains efficiency improvements for $\binom{2}{1}$-OT-extension, but also allows to generalize the protocol into passively secure $\binom{N}{1}$-OT-extension. In such an extension protocol the base OT's are $\binom{2}{1}$-OT's, but the output consist of a number of $\binom{N}{1}$-OT's. In more recent work, Orrù et al. [12] transformed the protocol

by [10] into an actively secure $\binom{N}{1}$-OT-extension protocol by adding a "consistency check" which is basically a zero-knowledge proof that the receiver is indeed using codewords of the designated code to encode his selections. As shown in [12], 1-out-of-N oblivious transfer has a direct application to the problem of private set inclusion and, via this connection, to the problem of private set intersection. In fact this application requires only a randomized version of $\binom{N}{1}$-OT, where the sender does not have input messages, but these are generated by the functionality and can be accessed on demand by the sender. The structure of the aforementioned OT extension protocols is especially well suited for this application, since such a randomized functionality is essentially implemented by the same protocol without the last step, where the sender would send its masked inputs to the receiver.

The aforementioned papers on $\binom{N}{1}$-OT-extension relied on the use of binary linear codes, and the concrete parameters of the resulting construction, the number of OT's and the value of N, are given respectively by the length and size of the binary linear code being used. Furthermore, the construction requires that the minimum distance of the code is at least the desired security parameter. Well-known bounds on linear codes, such as the Plotkin, Griesmer or Hamming bounds [11], provide lower bounds for the length of a code with certain size and minimum distance, and therefore these imply lower bounds on the number of base OT's for the OT-extension protocol. In fact, even if we omit the requirement on the minimum distance, we can see that at least $\log_2 N$ base OT's are needed for those extension protocols.

In this paper, we discuss the use of q-ary linear codes, where q can be any power of a prime, as a way of reducing the number of required base OT's in the 1-out-of-N OT-extension constructions mentioned above. We show that one can easily modify the protocol in [12] to work with q-ary codes, rather than just binary. Given that all parameters of the code still have the same significance for the construction and, in particular, N is still the size (the number of codewords) of the code, we obtain a reduction in the number of base OT's required: indeed, for given fixed values N and d, the minimal length among all q-ary linear codes of size N and minimum distance d becomes smaller as q increases. In particular one can show cases where the lower bound of $\log_2 N$ base OT's can be improved even if we have relatively large minimum distance.

This improvement, however, comes at a cost: since we need to communicate elements of a larger field, the communication complexity of the OT-extension protocol (not counting the complexity of the base OT's) increases. This increase is compensated to some extent by the fact that this communication complexity also depends on the number of base OT's.

The concrete tradeoffs obtained by the use of q-ary codes depend of course on N and the security level. We show several examples comparing explicit results listed in [12] and the q-ary alternative achieving the same (or similar)

N and security level. For example, for the largest value of N considered in [12] we show that by using a linear code over the finite field of 8 elements, we need less than half of the base OT's, while the communication complexity increases only by 33%.

When q is a power of two, we can show an improvement on the complexity of the consistency check that we use in the case of a general q. Namely, the consistency check in [12] works by asking the receiver, who has previously used the base OT's to commit to both the codewords encoding his selections and some additional random codewords, to open sums of random subsets of these codewords. The natural way of generalizing this to a general prime power q is to ask the receiver to open random linear combinations over $\mathbb{F}_q$ of the codewords. However, in case q is a power of two, we show that it is enough to open random linear combinations over $\mathbb{F}_2$, i.e., sums, just as in [12] (naturally, this extends to the case where q is a power of p, where it would be enough to open combinations over $\mathbb{F}_p$). The advantage of this generalization is of course that the verifier needs to send less information to describe the linear combinations that it requests to open, and in addition less computation is required from the committer to open these combinations.

We give a presentation of the protocol and its security proof that is inspired by a recent work on homomorphic universally composable secure commitments [2]. As noted in [12], there is a strong similarity between the OT-extension protocol constructions in the aforementioned works and several protocol constructions in a line of work on homomorphic UC commitments [2–4]. In the first part of the OT-extension protocol in [10], the base OT's are used for the receiver to eventually create an additive 1-out-of-2 sharing of each coordinate in the codewords encoding his selection, so that the sender learns exactly one share of each. This is essentially the same as the committing phase of the passively secure homomorphic UC commitment proposed in [3] (one can say that the receiver from the OT-extension protocol has actually committed to his inputs at that point). In order to achieve active security, a consistency check was added in [4], which is basically the same as the one introduced in [12] in the context of OT-extension. Finally, [2] generalized this consistency check by proving that rather than requesting the opening of uniformly random linear combinations of codewords, these combinations can be determined by a hash function randomly selected from an almost universal family of hash functions. This leads to asymptotical complexity gains, both in terms of communication and computation (since one can use linear time encodable almost universal hash functions which can in addition be described by short seeds), but in our case it also allows us to give a unified proof of security in both the case where the linear combinations for the consistency check are taken over $\mathbb{F}_q$ and when they are taken over the subfield.

The work is structured as follows. After the preliminaries in Section 2,

we present our OT-extension protocol and prove its security in Section 3. In Section 4, we show that the communication cost can be reduced by performing the consistency checks over a subfield, and finally Section 5 contains a comparison with previous protocols.

2 Preliminaries

This section contains the basic definitions needed to present and analyse the protocol for OT-extension.

2.1 Notation

For a bitstring $\mathbf{b} \in \{0,1\}^n$, we will use the notation $\Delta_{\mathbf{b}}$ to denote the diagonal matrix in $\mathbb{F}_q^{n\times n}$ with entries from the vector $\mathbf{b}$, i.e. the (i,i)-entry of $\Delta_{\mathbf{b}}$ is b_i. Note that for vectors $\mathbf{b}, \mathbf{c} \in \mathbb{F}_q^n$, the product $\mathbf{c}\Delta_{\mathbf{b}}$ equals the componentwise product of $\mathbf{b}$ and $\mathbf{c}$.

2.2 Linear Codes

Since our protocol depends heavily on linear codes, we recall here the basics of this concept. First, a (not necessarily linear) code of length n over an alphabet Q is a subset $\mathcal{C} \subseteq Q^n$. An $\mathbb{F}_q$-linear code $\mathcal{C}$ is an $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^n$. The dimension k of this subspace is called the dimension of the code, and therefore $\mathcal{C}$ is isomorphic to $\mathbb{F}_q^k$. A linear map $\mathbb{F}_q^k \to \mathcal{C}$ can be described by a matrix $G \in \mathbb{F}_q^{k\times n}$, which is called a generator matrix for $\mathcal{C}$. Note that G acts on the right, so $\mathbf{w} \in \mathbb{F}_q^k$ is mapped to $\mathbf{w}G \in \mathcal{C}$ by the aforementioned linear map.

For $\mathbf{x} \in \mathbb{F}_q^n$ we define the support of $\mathbf{x}$ to be the set indices where $\mathbf{x}$ is nonzero, and we denote this set by $\mathrm{supp}(\mathbf{x})$. Using this definition we can turn $\mathbb{F}_q^n$ into a metric space. This is done by introducing the Hamming weight and distance. The Hamming weight of $\mathbf{x}$ is defined as $w_H(\mathbf{x}) = |\mathrm{supp}(\mathbf{x})|$, and this induces the Hamming distance $d_H(\mathbf{x},\mathbf{y}) = w_H(\mathbf{x}-\mathbf{y})$, where $\mathbf{y} \in \mathbb{F}_q^n$ as well. The minimum distance d of a linear code $\mathcal{C}$ is defined to be

$$d = \min\{d_H(\mathbf{c},\mathbf{c}') \mid \mathbf{c},\mathbf{c}' \in \mathcal{C}, \mathbf{c} \neq \mathbf{c}'\},$$

and by the linearity of the code it can be shown that in fact

$$d = \min\{w_H(\mathbf{c}) \mid \mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}\}.$$

Since n, k, and d are fixed for a given linear code $\mathcal{C}$ over $\mathbb{F}_q$, we often refer to it as an $[n,k,d]_q$-code.

It may be shown that if $\mathbf{x} \in \mathbb{F}_q^n$ is given by $\mathbf{c} + \mathbf{e}$ for some codeword $\mathbf{c} \in \mathcal{C}$ and an error vector $\mathbf{e}$ with $w_H(\mathbf{e}) < d$, it is possible to recover $\mathbf{c}$ from $\mathbf{x}$ and $\text{supp}(\mathbf{e})$. This process is called erasure decoding.

Another way to see erasure decoding is by considering punctured codes. For a set of indices $E \subseteq \{1, 2, \ldots, n\}$ we denote the projection of $\mathbf{x} \in \mathbb{F}_q^n$ onto the indices not in E by $\pi_E(\mathbf{x})$. For a code $\mathcal{C}$ and a set of indices E, we call $\pi_E(\mathcal{C})$ a punctured code. Now consider the case where $|E| < d$, which implies the existence of a bijection between $\mathcal{C}$ and $\pi_E(\mathcal{C})$. This is the fact exploited in erasure decoding, where E is the set of indices where the errors occur.

As in [2], we will use interleaved codes. If $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a linear code, $\mathcal{C}^{\odot s}$ denotes the set of $s \times n$-matrices with entries in $\mathbb{F}_q$ whose rows are codewords of $\mathcal{C}$. We can also see such an $s \times n$-matrix as a vector of length n with entries in the alphabet $\mathbb{F}_q^s$. Then we can see $\mathcal{C}^{\odot s}$ as a non-linear[1] code of length n over the alphabet $\mathbb{F}_q^s$.

Since the alphabet $\mathbb{F}_q^s$ contains a zero element (the all zero vector), we can define the notions of Hamming weight and Hamming distance in the space $(\mathbb{F}_q^s)^n$. We can then speak about the minimum distance of $\mathcal{C}^{\odot s}$ and even though $\mathcal{C}^{\odot s}$ is not a linear code, it is easy to see that the minimum distance of $\mathcal{C}^{\odot s}$ coincides with its minimum nonzero weight, and also with the minimum distance of $\mathcal{C}$.

2.3 Cryptographic Definitions

Consider a sender S and a receiver R participating in a cryptographic protocol. The sender holds $\mathbf{v}_{j,i} \in \{0,1\}^\kappa$ for $j = 1, 2, \ldots, N$ and $i = 1, 2, \ldots, m$. For each i the receiver holds a choice integer $w_i \in [1, N]$. We let $\mathcal{F}_{N\text{-}\mathsf{OT}}^{\kappa,m}$ denote the ideal functionality that, on inputs $\mathbf{v}_{j,i}$ from S and w_i from R, outputs $\mathbf{v}_{w_i,i}$ for $i = 1, 2, \ldots, m$ to the receiver R. For ease of notation, we will let the sender input N matrices of size $\kappa \times m$ with entries in $\{0,1\}$, and the receiver a vector of length m, with entries in $[1, N]$. Hence, for the i'th OT the sender's inputs are the i'th column of each matrix, and the receiver's input is the i'th entry of the vector.

The protocol presented in Section 3 relies on two functions with certain security assumptions, the foundations of which we define in the following. For the first function let $\mathcal{X}$ be a probability distribution. The min-entropy of $\mathcal{X}$ is given by

$$H_\infty(\mathcal{X}) = -\log(\max_x \Pr[X = x]),$$

where X is any random variable following the distribution $\mathcal{X}$. If $H_\infty(\mathcal{X}) = t$ we say that $\mathcal{X}$ is t-min-entropy. This is used in the following definition.

[1] The code is linear over $\mathbb{F}_q$, but not the alphabet $\mathbb{F}_q^s$.

Definition 2.1 (t-min-entropy strongly $\mathcal{C}$-correlation robustness). *Consider a linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$, and let $\mathcal{X}$ be a distribution on $\{0,1\}^n$ with min-entropy t. Fix $\{\mathbf{t}_i \in \mathbb{F}_q^n \mid i = 1, 2, \ldots, m\}$ from some probability distribution and let κ be a positive integer. An efficiently computable function $\mathsf{H}\colon \mathbb{F}_q^n \to \{0,1\}^\kappa$ is said to be t-min-entropy strongly $\mathcal{C}$-correlation robust if*

$$\{\mathsf{H}(\mathbf{t}_i + \mathbf{c}\Delta_{\mathbf{b}}) \mid i = 1, 2, \ldots, m, \mathbf{c} \in \mathcal{C}\}$$

is computationally indistinguishable from the uniform distribution on $\{0,1\}^{\kappa m|\mathcal{C}|}$ when $\mathbf{b}$ is sampled according to the distribution $\mathcal{X}$.

The second type of function we need is a pseudorandom generator.

Definition 2.2. *A pseudorandom generator is a function $\mathsf{PRG}\colon \{0,1\}^\kappa \to \mathbb{F}_q^m$ such that the output of PRG is computationally indistinguishable from the uniform distribution on $\mathbb{F}_q^m$.*

If $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ is a $\kappa \times n$-matrix with entries in $\{0,1\}$ for some integer n, we use the notation $\mathsf{PRG}(A) = [\mathsf{PRG}(\mathbf{a}_1), \mathsf{PRG}(\mathbf{a}_2), \ldots, \mathsf{PRG}(\mathbf{a}_n)]$ where we see $\mathsf{PRG}(\mathbf{a}_i)$ as columns of an $m \times n$ matrix.

In addition to the usual concept of advantage, one can also consider the conditional advantage as it is done in [12]. Let A be an event such that there exist x_0 and x_1 in the sample space of the two random variables X_0 and X_1, respectively, where $\Pr[X_i = x_i \mid A] > 0$ for $i = 0, 1$. Then we define the conditional advantage of a distinguisher $\mathcal{D}$ given A as

$$\mathrm{Adv}(\mathcal{D}|A) = \Big|\Pr[\mathcal{D}(X_0) = 0|A] - \Pr[\mathcal{D}(X_1) = 0|A]\Big|.$$

We end this section by presenting the following lemma, which allows us to bound the advantage by considering disjoint cases. The proof follows by the law of total probability and the triangle inequality.

Lemma 2.3. *Let $A_1, A_2, \ldots, A_n$ be events as above. Additionally, assume that the events are disjoint. If $\sum_{i=1}^n \Pr[A_i] = 1$, then*

$$\mathrm{Adv}(\mathcal{D}) \leq \sum_{i=1}^{n} \mathrm{Adv}(\mathcal{D} \mid A_i) \Pr[A_i]$$

for any distinguisher $\mathcal{D}$.

3 Actively Secure OT-Extension

In this section we describe and analyse a generalization of the protocol described in [12] which uses OT-extensions to implement the functionality $\mathcal{F}_{N\text{-}\mathsf{OT}}^{\kappa,m}$ by using only $n \leq m$ base OT's, which are 1-out-of-2. Our OT-extension protocol is also using 1-out-of-2 base OT's, but works with q-ary linear codes instead of binary. Our main result is summarized in the following theorem.

Theorem 3.1. *Given security parameters κ and s, let $\mathcal{C}$ be an $[n,k,d]_q$ linear code with $k = \log_q(N)$ and $d \geq \max\{\kappa, s\}$. Additionally, let $\mathsf{PRG}\colon \{0,1\}^\kappa \to \mathbb{F}_q^{m+2s}$ be a pseudorandom generator and let $\mathsf{H}\colon \mathbb{F}_q^n \to \{0,1\}^\kappa$ be a t-min-entropy strongly $\mathcal{C}$-correlation robust function for all $t \in \{n-d+1, n-d+2, \ldots, n\}$. If we have access to $\mathcal{C}$, the functions PRG and H, and the functionality $\mathcal{F}_{2\text{-}\mathsf{OT}}^{\kappa,n}$, then the protocol in Figure B.1 on page 66 implements the functionality $\mathcal{F}_{N\text{-}\mathsf{OT}}^{\kappa,m}$.*

The protocol is computationally secure against an actively corrupt adversary.

3.1 The Protocol

We start by noticing that in our protocol R has inputs $\mathbf{w}_i \in \mathbb{F}_q^k$ rather than choice integers $w_i \in [1, N]$. However, the number of elements in $\mathbb{F}_q^k$ is $q^k = N$, and hence $\mathbf{w}_i$ can for instance be the q-ary representation of w_i. In this way we have a bijection between selection integers and input vectors.

Our protocol is, like the protocol in [12], very similar to the original protocol in [7]. The idea in this protocol is that we first do OT's with the roles of the participants interchanged such that the sender learns some randomness chosen by the receiver. Afterwards, R encodes his choice vectors using the linear code $\mathcal{C}$ and hides the value with a one-time pad. He sends these to S, who will combine this information with the outputs of the OT functionality to obtain a set of vectors, only m of which R can compute; namely the ones corresponding to his input vectors. When S applies a t-min-entropy strongly $\mathcal{C}$-correlation robust function H to the set of vectors, he can use the outputs as one-time pads of his input strings. Like in [12] the protocol contains a consistency check to ensure that R acts honestly, or otherwise he will get caught with overwhelming probability. The full protocol is presented in Figure B.1 on page 66.

In order to argue that the protocol is correct, we see that for each i, the sender S computes and sends the values $\mathbf{y}_{\mathbf{w},i}$ for all $\mathbf{w} \in \mathbb{F}_q^k$. Since $k = \log_q(N)$, this yields N strings for each $i \in \{1, 2, \ldots, m\}$. The receiver R obtains one of these because

$$\mathsf{H}(\mathbf{q}_i - \mathbf{w}_i G \Delta_{\mathbf{b}}) = \mathsf{H}(\mathbf{q}_i - \mathbf{c}_i \Delta_{\mathbf{b}}) = \mathsf{H}(\mathbf{t}_i).$$

Furthermore, if both S and R act honestly, the consistency checks in phase 3 will always pass. This follows from the observation that

$$\tilde{T} + \tilde{W} G \Delta_{\mathbf{b}} = M(T_0 + C\Delta_{\mathbf{b}}) = MQ.$$

Hence, we note that if only passive security is needed in Protocol 1, we can omit phase 3 and set $s = 0$. The aforementioned steps are included to ensure

[1] In Section 4, we show if the protocol relies on a code over $\mathbb{F}_{p^r}$, it is enough to choose $M' \in \mathbb{F}_p^{2s \times m}$.

Protocol 1: OT-Extension

1. **Initialization phase**

   (a) S chooses uniformly at random $\mathbf{b} \in \{0,1\}^n$.

   (b) R generates uniformly at random two seed matrices N_0, $N_1 \in \{0,1\}^{\kappa \times n}$ and defines the matrices $T_i = \mathsf{PRG}(N_i) \in \mathbb{F}_q^{(m+2s)\times n}$ for $i = 0, 1$.

   (c) The participants call the functionality $\mathcal{F}_{\text{2-OT}}^{\kappa,n}$, where S acts as the receiver with input $\mathbf{b}$, and R acts as the sender with inputs (N_0, N_1). S receives $N = N_0 + (N_1 - N_0)\Delta_{\mathbf{b}}$, and by using PRG, he can compute $T = T_0 + (T_1 - T_0)\Delta_{\mathbf{b}}$.

2. **Encoding phase**

   (a) Let $W' \in \mathbb{F}_q^{k\times m}$ be the matrix which has $\mathbf{w}_i$ as its columns. R generates a uniformly random matrix $W'' \in \mathbb{F}_q^{k\times 2s}$, and defines the $(m+2s)\times k$-matrix $W = [W' \mid W'']^T$.

   (b) R sets $C = WG$, and sends $U = C + T_0 - T_1$.

   (c) S computes $Q = T + U\Delta_{\mathbf{b}}$. This implies that $Q = T_0 + C\Delta_{\mathbf{b}}$.

3. **Consistency check**

   (a) S samples a uniformly random matrix $M' \in \mathbb{F}_q^{2s\times m}$ and sends this to R.[2] They both define $M = [M' \mid I_{2s}]$.

   (b) R computes the $2s \times n$-matrix $\tilde{T} = MT_0$ and the $2s \times k$-matrix $\tilde{W} = MW$ and sends these matrices to S.

   (c) S verifies that $MQ = \tilde{T} + \tilde{W}G\Delta_{\mathbf{b}}$. If this fails, S aborts the protocol.

4. **Output phase**

   (a) Denote by $\mathbf{q}_i$ and $\mathbf{t}_i$, the i'th rows of Q and T_0, respectively. For $i = 1, 2, \ldots, m$ and for all $\mathbf{w} \in \mathbb{F}_q^k$, S computes $\mathbf{y}_{\mathbf{w},i} = \mathbf{v}_{\mathbf{w},i} \oplus \mathsf{H}(\mathbf{q}_i - \mathbf{w}G\Delta_{\mathbf{b}})$ and sends these to R. For $i = 1, 2, \ldots, m$, R can recover $\mathbf{v}_{\mathbf{w}_i,i} = \mathbf{y}_{\mathbf{w}_i,i} \oplus \mathsf{H}(\mathbf{t}_i)$.

Fig. B.1: This protocol implements the functionality $\mathcal{F}_{N\text{-OT}}^{\kappa,m}$ having access to $\mathcal{F}_{\text{2-OT}}^{\kappa,n}$. The security of the protocol is controlled by the security parameters κ and s. The sender S and the receiver R have agreed on a linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with generator matrix G of dimension $k = \log_q(N)$ and minimum distance $d \geq \max\{\kappa, s\}$. The protocol uses a pseudorandom generator $\mathsf{PRG}\colon \{0,1\}^\kappa \to \mathbb{F}_q^{m+2s}$ and a function $\mathsf{H}\colon \mathbb{F}_q^n \to \{0,1\}^\kappa$, which is t-min-entropy strongly $\mathcal{C}$-correlation robust for all $t \in \{n-d+1, n-d+2, \ldots, n\}$. R has m inputs $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m \in \mathbb{F}_q^k$, which act as selection integers. S has inputs $\mathbf{v}_{\mathbf{w},i} \in \{0,1\}^\kappa$, indexed by $i \in \{1, 2, \ldots, m\}$ and $\mathbf{w} \in \mathbb{F}_q^k$.

that the receiver uses codewords in the matrix C. What a malicious receiver might gain by choosing rows which are not codewords is explained in [7, Sec. 4].

3.2 Proofs of Security

In this section we give formal proofs for security. The proof of security against a malicious sender works more or less the same as the proof in [12] but in a different notation. For completeness, we have included this proof. However, we present the proof against a malicious receiver in another way, where the structure, some strategies, and some arguments differ from the original proof.

Theorem 3.2. *The Protocol in Figure B.1 is computationally secure against an actively corrupt sender.*

Proof. To show this theorem we give a simulator, which simulates the view of the sender during the protocol. The view of S is $\text{View}_S = \{N, U, \tilde{T}, \tilde{W}\}$. The simulator Sim_S works as follows.

1. Sim_S receives $\mathbf{b}$ from S and defines a uniformly random matrix N, sets $T = \mathsf{PRG}(N)$, and passes N back to S.

2. Then Sim_S samples U uniformly at random and sends this to S. Additionally, it computes Q as S should.

3. In phase 3 the simulator receives M' from S, and constructs M. The matrix $\tilde{W}$ is sampled uniformly at random in $\mathbb{F}_q^{2s\times k}$, and using this, Sim_S sets $\tilde{T} = MQ - \tilde{W}G\Delta_{\mathbf{b}}$. It sends $\tilde{T}$ and $\tilde{W}$ to S.

4. Sim_S receives $\mathbf{y}_{\mathbf{w},i}$ from S and since Sim_S already knows Q and $\mathbf{b}$, it can recover $\mathbf{v}_{\mathbf{w},i} = \mathbf{y}_{\mathbf{w},i} \oplus \mathsf{H}(\mathbf{q}_i - \mathbf{w}G\Delta_{\mathbf{b}})$ and pass these to the ideal functionality $\mathcal{F}_{N\text{-}\mathsf{OT}}^{\kappa,m}$.

We now argue that the simulator produces values indistinguishable from View_S. The matrix N is distributed identically in the real and ideal world. Since both T_0 and T_1 are outputs of a pseudorandom generator, the matrix $T_0 - T_1$, and therefore also U, is computationally indistinguishable from a uniformly random matrix. In the real world, $\tilde{W} = M'(W')^T + (W'')^T$ is uniform since W'' is chosen uniformly. The simulator Sim_S constructs $\tilde{T}$ such that the consistency check will pass. This will always be the case in the real world, and hence S cannot distinguish between the real and ideal world. Additionally, we note that step 4 ensures that the receiver obtains the same output in both worlds. This shows security against an actively corrupt sender.

We now shift our attention to an actively corrupt receiver. This proof is not as straight forward as for the sender. The idea is to reduce the problem of breaking the security of the protocol to the problem of breaking the assumptions on H. Before delving into the proof itself, we will introduce some lemmata and notations that will aid in the proof. The focus of these will be the probability that certain events happen during the protocol. These events are based on situations that determine the simulator's ability or inability to simulate the real world. Essentially, they are the event that R passes the consistency check, which we denote by PC; the event that R has introduced errors in too many positions, denoted by LS; and the event that the error positions from the consistency check line up with the errors in C, which we call ES. These will be defined more precisely below.

Inspired by the notation in the protocol, we define

$$\tilde{C} = MC. \tag{B.1}$$

A corrupt receiver may deviate from the protocol and may send an erroneous $\tilde{W}$, which we denote by $\tilde{W}_*$. Let

$$\bar{C} = \tilde{C} - \tilde{W}_* G$$

and let $E = \mathrm{supp}(\bar{C})$, where $\bar{C}$ is interpreted in $\mathcal{C}^{\odot 2s}$. When writing $\tilde{C}$, $\bar{C}$, and E later in this section these are the definitions we are implicitly referring to.

Lemma 3.3. *Let $\mathcal{C}$, C, and M be as in Protocol 1. Further, let* LS *be the event that $|E| \geq s$, and let* ES *be the event that for every $C' \in \mathcal{C}^{\odot 2s}$ there exists a $\hat{C} \in \mathcal{C}^{\odot m+2s}$ such that $\mathrm{supp}(\tilde{C} - C') = \mathrm{supp}(C - \hat{C})$. Then the probability that neither* ES *nor* LS *happen is at most q^{-s}.*

Proof. The matrix M' in Protocol 1 is chosen uniformly at random, and hence M can be interpreted as a member of a universal family of linear hashes. Thus, this lemma is a special case of [2, Theorem 1] when letting $m' = m + 2s$, $s' = s$, and $t' = 0$ where the primes denote the parameters in [2]. Additionally, note that our event LS happens if MC has distance at least s from $\mathcal{C}^{\odot 2s}$.

We will now bound the probability that an adversary is able to pass the consistency check, even if C contains errors.

Lemma 3.4. *Let* PC *denote the event that the consistency check passes. Then*

$$\Pr[\mathsf{PC}] \leq 2^{-|E|}.$$

Proof. In order to compute $\Pr[\mathsf{PC}]$, we consider $\bar{C}$ and $\bar{T} = \tilde{T} - \tilde{T}_*$, where the $*$ indicates that the matrix may not be constructed as described in the

protocol. The event PC happens if $MQ = \tilde{T}_* + \tilde{W}_* G\Delta_{\mathbf{b}}$. However, from the definition of Q, $MQ = \tilde{T} + \tilde{C}\Delta_{\mathbf{b}}$, implying that PC happens if and only if

$$\tilde{T} + \tilde{C}\Delta_{\mathbf{b}} = \tilde{T}_* + \tilde{W}_* G\Delta_{\mathbf{b}} \quad \Longleftrightarrow \quad \bar{T} = -\bar{C}\Delta_{\mathbf{b}}.$$

Now consider $\bar{T}$ and $\bar{C}$ in $(\mathbb{F}_q^n)^{\odot 2s}$, meaning that the entries $\bar{C}_j$ and $\bar{T}_j$ are elements in $\mathbb{F}_q^{2s}$. If the adversary chooses $\bar{C}_j = 0$ for some $j \in \{1, 2, \ldots, n\}$, it must choose $\bar{T}_j = 0$ as well since the check would fail otherwise. If it chooses $\bar{C}_j \neq 0$, it has two options. Either bet that $b_j = 0$ and set $\bar{T}_j = 0$ or bet that $b_j = 1$ and set $\bar{T}_j = -\bar{C}_j$. This means that for each entry $j \in E$ the adversary has probability $\frac{1}{2}$ of guessing the correct value of b_j. For every entry $j \notin E$, each possible b_j gives a consistent value since $\bar{C}_j = \bar{T}_j = 0$. By this and the independence of the entries in $\mathbf{b}$, it follows that the probability of the check passing is bounded by $\Pr[\mathsf{PC}] \leq 2^{-|E|}$.

This immediately gives the following corollary.

Corollary 3.5. *If* LS *denotes the same event as in Lemma 3.3, then*

$$\Pr[\mathsf{PC} \mid \mathsf{LS}] \leq 2^{-s}.$$

We now have the required results to prove the security of Protocol 1 against an actively corrupt receiver. The events PC, LS, and ES from the previous lemmata and corollaries will also be used in the proof of the following theorem.

Theorem 3.6. *The Protocol in Figure B.1 is computationally secure against an actively corrupt receiver.*

Proof. As in the proof of Theorem 3.2, we construct a simulator Sim_R simulating the view of the receiver, which is $\mathrm{View}_R = \{M', \mathbf{y}_{\mathbf{w},i}\}$. The simulator works as follows.

1. Sim_R receives N_0 and N_1 from R.
2. The simulator receives U from R and combines these with $T_0 = \mathsf{PRG}(N_0)$ and $T_1 = \mathsf{PRG}(N_1)$ to reconstruct the matrix C. Additionally, it samples uniformly at random an internal value $\mathbf{b}$. Using this $\mathbf{b}$, the simulator Sim_R computes $Q = T_0 + C\Delta_{\mathbf{b}}$.
3. Sim_R samples a random M' like the sender would have done in the protocol and sends this to R. In return, it receives $\tilde{T}_*$ and $\tilde{W}_*$, where the $*$ indicates that the vectors may not be computed according to the protocol. The simulator runs the consistency check and aborts if it fails.

4. Otherwise, it erasure decodes each row of C by letting E be the erasures to obtain W'. If the decoding fails, it aborts. If the decoding succeeds, the simulator gives W' as inputs to the ideal functionality $\mathcal{F}^{\kappa,m}_{N\text{-}\mathsf{OT}}$, which returns the values $\mathbf{v}_{\mathbf{w}_i,i}$ to Sim_R. It can now compute $\mathbf{y}_{\mathbf{w}_i,i} = \mathbf{v}_{\mathbf{w}_i,i} \oplus \mathsf{H}(\mathbf{q}_i - \mathbf{w}_i G\Delta_{\mathbf{b}})$, and chooses $\mathbf{y}_{\mathbf{w},i}$ uniformly at random in $\mathbb{F}_q^\kappa$ for all $\mathbf{w} \neq \mathbf{w}_i$.

The matrix M' is uniformly distributed both in the real and ideal world. Hence, we only need to show that the output $\mathbf{y}_{\mathbf{w},i}$ produced by the simulator is indistinguishable from the output of the protocol.

Let $\mathcal{Z}$ be a distinguisher for distinguishing between a real world execution of the protocol and an ideal execution using the simulator. By Lemma 2.3 its advantage is bounded by

$$\begin{aligned}\mathrm{Adv}(\mathcal{Z}) \leq &\mathrm{Adv}(\mathcal{Z} \mid \overline{\mathsf{PC}}) + \mathrm{Adv}(\mathcal{Z} \mid \mathsf{PC}, \mathsf{LS}) \Pr[\mathsf{PC} \mid \mathsf{LS}] \\ &+ \mathrm{Adv}(\mathcal{Z} \mid \mathsf{PC}, \overline{\mathsf{LS}}, \overline{\mathsf{ES}}) \Pr[\overline{\mathsf{LS}}, \overline{\mathsf{ES}}] + \mathrm{Adv}(\mathcal{Z} \mid \mathsf{PC}, \overline{\mathsf{LS}}, \mathsf{ES}) \Pr[\mathsf{PC}],\end{aligned} \tag{B.2}$$

where we have omitted some probability factors since they are all at most 1. Notice that $\mathbf{y}_{\mathbf{w}_i,i}$ is constructed identically in both worlds. The remaining $\mathbf{y}_{\mathbf{w},i}$ are uniformly distributed in the ideal world, but constructed as

$$\mathbf{y}_{\mathbf{w},i} = \mathbf{v}_{\mathbf{w},i} \oplus \mathsf{H}(\mathbf{q}_i - \mathbf{w} G\Delta_{\mathbf{b}}) \tag{B.3}$$

in the real world. Also notice that, if the consistency check fails, the simulator aborts before constructing the $\mathbf{y}_{\mathbf{w},i}$. This is the same as in the real world, and the only information R has received before this is M', which is identically distributed in both worlds. Hence, the simulator is perfect in this case. This implies that the first term on the right-hand side in (B.2) is zero.

Since the consistency check by the simulator is identical to the consistency check done by S, it follows that the probability for the consistency check to pass even if R might have sent inconsistent values is the same in both worlds. This means that $\Pr[\mathsf{PC} \mid \mathsf{LS}] \leq 2^{-s}$ by Corollary 3.5. In a similar fashion, Lemma 3.3 implies that the penultimate term in (B.2) can be bounded above by q^{-s}. In summary, (B.2) can be rewritten as

$$\mathrm{Adv}(\mathcal{Z}) \leq 2^{-s} + q^{-s} + \mathrm{Adv}(\mathcal{Z} \mid \mathsf{PC}, \overline{\mathsf{LS}}, \mathsf{ES}) 2^{-|E|}. \tag{B.4}$$

To show that this is negligible in κ and s, assume the opposite; that is, $\mathcal{Z}$ has non-negligible advantage. We then construct a distinguisher $\mathcal{D}$ breaking the security assumptions on H.

The distinguisher $\mathcal{D}$ simulates the protocol with minor changes in order to produce its input to the challenger. After receiving the challenge it uses the output of $\mathcal{Z}$ to respond. There exist inputs and random choices for R and

S, which maximize the advantage of $\mathcal{Z}$, and we can assume that $\mathcal{D}$ has fixed these in its simulation. This also means that PC, $\overline{\mathsf{LS}}$ and ES happen in the simulation since otherwise, $\mathrm{Adv}(\mathcal{Z})$ is negligible.

Because ES happens, puncturing C in the positions in E gives a codeword in $\pi_E(\mathcal{C}^{\odot m+2s})$. Further, the event $\overline{\mathsf{LS}}$ ensures that this corresponds to a unique codeword in $\mathcal{C}^{\odot m+2s}$. Hence, $\mathcal{D}$ is able to erasure decode and for $i = 1, 2, \ldots, m+2s$ obtain $\mathbf{c}_i = \mathbf{w}_i G + \mathbf{e}_i$, where $\mathbf{c}_i$ is the i'th row of C, $w_H(\mathbf{e}_i) < d$, and $\mathrm{supp}(\mathbf{e}_i) \subseteq E$.

The following arguments use that no matter which $\mathbf{b}$ the challenger chooses, the distinguisher $\mathcal{D}$ knows $\mathbf{e}_i \Delta_{\mathbf{b}}$. This follows from the fact that PC has happened and therefore b_j for $j \in E$ is known to the adversary, which is simulated by $\mathcal{D}$. Hence, the distinguisher is able to construct $\mathbf{t}'_i = \mathbf{t}_i + \mathbf{e}_i \Delta_{\mathbf{b}}$, where the $\mathbf{b}$ is the vector eventually chosen by the challenger, and $\mathbf{t}_i$ the i'th row of T_0. Letting $t = n - |E|$, define the probability distribution $\mathcal{X}$ to be the uniform distribution on $\mathbb{F}_2^n$ under the condition that the indices in E are fixed to the corresponding entry of $\mathbf{b}$. By uniformity this distribution has min-entropy t. The distinguisher passes $\mathcal{X}$ and the $\mathbf{t}'_i$ to the challenger. It receives back $\mathbf{x}_{\mathbf{w},i}$ for all $i = 1, 2, \ldots, n$ and $\mathbf{w} \in \mathbb{F}_q^k$ and needs to distinguish them between being uniformly random and being constructed as

$$\mathbf{x}_{\mathbf{w},i} = \mathsf{H}(\mathbf{t}'_i + \mathbf{w} G \Delta_{\mathbf{b}}), \tag{B.5}$$

As in the protocol, let $Q = T_0 + C\Delta_{\mathbf{b}}$, where $\mathbf{b}$ is again the vector chosen by the challenger. Therefore, if $\mathbf{x}_{\mathbf{w},i}$ is constructed as in (B.5), we have that

$$\begin{aligned}
\mathbf{x}_{\mathbf{w},i} &= \mathsf{H}(\mathbf{t}_i + \mathbf{e}_i \Delta_{\mathbf{b}} + \mathbf{w} G \Delta_{\mathbf{b}}) \\
&= \mathsf{H}(\mathbf{q}_i - \mathbf{c}_i \Delta_{\mathbf{b}} + \mathbf{e}_i \Delta_{\mathbf{b}} + \mathbf{w} G \Delta_{\mathbf{b}}) \\
&= \mathsf{H}(\mathbf{q}_i - (\mathbf{w}_i - \mathbf{w}) G \Delta_{\mathbf{b}}).
\end{aligned}$$

The distinguisher will now construct and input to $\mathcal{Z}$ the following

$$\begin{aligned}
\mathbf{y}_{\mathbf{w}_i,i} &= \mathbf{v}_{\mathbf{w}_i,i} \oplus \mathsf{H}(\mathbf{t}'_i), \\
\mathbf{y}_{\mathbf{w},i} &= \mathbf{v}_{\mathbf{w},i} \oplus \mathbf{x}_{\mathbf{w}_i - \mathbf{w},i}, \quad \text{for } \mathbf{w} \neq \mathbf{w}_i.
\end{aligned}$$

Since $\mathbf{t}'_i = \mathbf{t}_i + \mathbf{e}_i \Delta_{\mathbf{b}} = \mathbf{q}_i - \mathbf{w}_i G \Delta_{\mathbf{b}}$, we have that $\mathbf{y}_{\mathbf{w}_i,i}$ is identical to the value computed in both the real and ideal worlds.

For the remaining $\mathbf{w}$ we notice that if the challenger has chosen $\mathbf{x}_{\mathbf{w},i}$ uniformly at random, then the values $\mathbf{y}_{\mathbf{w},i}$ are uniformly distributed as well. This is the same as the simulator will produce in the ideal world. On the other hand, if $\mathbf{x}_{\mathbf{w},i} = \mathsf{H}(\mathbf{t}'_i + \mathbf{w} G \Delta_{\mathbf{b}})$, then we have $\mathbf{y}_{\mathbf{w},i} = \mathbf{v}_{\mathbf{w},i} \oplus \mathsf{H}(\mathbf{q}_i - \mathbf{w} \Delta_{\mathbf{b}})$. This is exactly the same as produced during the protocol in the real world. Hence, $\mathcal{D}$ can feed the values $\mathbf{y}_{\mathbf{w},i}$ to $\mathcal{Z}$, which can distinguish between the real and ideal world, and depending on the answer from $\mathcal{Z}$, $\mathcal{D}$ can distinguish whether the $\mathbf{x}_{\mathbf{w},i}$ are uniformly distributed or are constructed as

$\mathsf{H}(\mathbf{t}'_i + \mathbf{w}G\Delta_{\mathbf{b}})$. Hence, the advantage of $\mathcal{D}$ is the same as that of $\mathcal{Z}$ under the restriction that PC, $\overline{\mathsf{LS}}$, and ES happen. This means that

$$\mathrm{Adv}(\mathcal{D}) = \mathrm{Adv}(\mathcal{Z}|\mathsf{PC}, \overline{\mathsf{LS}}, \mathsf{ES}) \geq 2^{|E|}\left(\mathrm{Adv}(\mathcal{Z}) - 2^{-s} - q^{-s}\right), \tag{B.6}$$

where the inequality comes from (B.4). This contradicts that H is t-min-entropy strongly $\mathcal{C}$-correlation robust, and therefore $\mathcal{Z}$ must have negligible advantage in the security parameters κ and s.

4 Consistency Check in a Subfield

Assume that $q = 2^r$ and that $r \mid s$. By restricting the matrix M' in Protocol 1 to have entries in $\mathbb{F}_2$, the set of possible matrices M form a 2^{-2s}-almost universal family of hashes. The probability in Lemma 3.3 can then be replaced by 2^{-s} by setting $m' = m + 2s$, $s' = \frac{s}{r}$, and $t' = 2s(1 - 1/r)$. This modification will show itself in (B.4), but here only the term q^{-s} is replaced by 2^{-s}, and hence the advantage will still be negligible in κ and s. However, choosing M' in a subfield reduces the communication complexity, since the number of bits needed to transmit M' is lowered by a factor of r. Furthermore, the computation of $\tilde{T}$ and $\tilde{W}$ can be done using only sums in $\mathbb{F}_q$, instead of multiplication and sums.

This method of reducing the communication complexity can be done to an intermediate subfield, which will give a probability bound between q^{-s} and 2^{-s}. In a similar way, this procedure could also be applied to fields of other characteristics.

5 Comparison

We compare the parameters of our modified construction with those that can be achieved by the actively secure OT-extension construction from [12]. We will show that the ability to use larger finite fields in our modified construction induces a tradeoff between the number of base OT's that are needed for a given N and given security parameters (and hence also the complexity of the set-up phase), and the complexity of the encoding and consistency check phases of the extension protocol.

We have shown that given an $[n, k, d]_q$-code, with $d \geq \max\{\kappa, s\}$, one can build an OT-extension protocol that implements the functionality $\mathcal{F}_{N\text{-}\mathsf{OT}}^{\kappa,m}$ using the functionality $\mathcal{F}_{2\text{-}\mathsf{OT}}^{\kappa,n}$, where $N = q^k$. The parameters achieved in [12] are the same as we obtain in the case $q = 2$.

We will limit our analysis to the case where $q = 2^r$, and $r \mid s$. We fix the security parameters s and κ, and fix N to be a power of q, $N = q^k$. Note then

that $N = 2^{k \cdot \log_2 q}$. Let n' and n be the smallest integers for which there exist an $[n', k \log_2 q, \geq d]_2$-linear code and an $[n, k, \geq d]_q$-linear code, respectively. As we discuss later, we can always assume that $n \leq n'$, and in most cases it is in fact strictly smaller. Therefore, by using q-ary codes one obtains a reduction on the number of base OT's from n' to n, and therefore a more efficient initialization phase. Note for example that the binary construction always requires at least a minimum of $\log_2 N$ base OT's, while using q-ary codes allows to weaken this lower bound to $n \geq \log_q N$.

On the other hand, however, this comes at the cost of an increase in the communication complexity of what we have called the encoding and consistency check phases of the protocol since we need to send a masking of codewords over a larger field. We compare these two phases separately since the consistency check is only needed for an actively secure version of the protocol and it has a smaller cost than the encoding phase anyway. In the encoding phase, [12] communicates a total of $(m + s)n'$ bits, while our construction communicates $(m + 2s)n \log_2 q$ bits. However, typically $m \gg s$, and therefore we only compare the terms mn' and $mn \log_2 q$. Hence, the communication complexity of this phase gets multiplied by a factor $\log_2 q \cdot n/n'$. During the consistency check phase, which is less communication intensive, [12] communicates a total of $sm + sn' + sk \log_2 q$ bits while our construction communicates $2sm + 2sn \log_2 q + 2sk \log_2 q$ bits when using the method from Section 4.

We now discuss in more detail the rates between n and n' that we can obtain for different values of q. In order to do that, having fixed d and k, let n' and n denote the minimum values for which $[n', k \log_2 q, \geq d]_2$-linear codes and $[n, k, \geq d]_q$-linear codes exist. Let k' denote $k \log_2 q$. It is easy to see that $n \leq n'$ by considering a generator matrix for the binary code of length n' and considering the code spanned over $\mathbb{F}_q$ by that same matrix. In many situations, however, n is in fact considerably smaller than n'. The extreme case is when $q = N$, and therefore $k = 1$, in which case one can take the repetition code over $\mathbb{F}_q$ and set $n = d$. It is difficult to give a general tight bound on the relation between n and n', although at least we can argue that $n \leq n' - k' + k$: indeed, given an $[n', k', \geq d]_2$-code $\mathcal{C}_2$ then one can obtain an $[n', k', \geq d]_q$-code $\mathcal{C}_q$ by simply considering the linear code spanned over the field $\mathbb{F}_q$ by the generator matrix of $\mathcal{C}_2$ and then shorten[3] $\mathcal{C}_q$ at $k' - k$ positions, after which we obtain an $[n, \geq k, \geq d]_q$-code $\mathcal{C}$, with $n = n' - k' + k$. This bound is however by no means tight in general. We now consider concrete examples of codes, that will be summarized in Table B.1.

[3]Shortening a code at positions $i_1, \ldots, i_t$ means first taking the subcode consisting of all codewords with $0's$ at all those positions and then erasing those coordinates.

Code	N	n (Base OT's)	d	Comparison n	Comparison CC
Walsh-Had. [10]	256	256	128		
Juxt. simplex code over $\mathbb{F}_4$	256	170	128	$\div$ 1.51	$\times$ 1.33
Punct. Walsh-Had. [12]	512	256	128		
Juxt. simplex code over $\mathbb{F}_8$	512	146	128	$\div$ 1.75	$\times$ 1.71
$[511, 76, \geq 171]_2$-BCH [12]	2^{76}	511	≥ 171		
$[455, 48, \geq 174]_4$-BCH over $\mathbb{F}_4$	2^{96}	455	≥ 174	$\div$ 1.12	$\times$ 1.78
$[1023, 443, \geq 128]_2$-BCH [12]	2^{443}	1023	≥ 128		
$[455, 154, \geq 128]_8$-BCH over $\mathbb{F}_8$	2^{462}	455	≥ 128	$\div$ 2.25	$\times$ 1.33

Table B.1: Comparison of using binary and q-ary codes for OT-extension. In the last two columns we consider the decrease in the number of base OT's and increase in the dominant term of the communication complexity in the encoding phase when we consider a q-ary construction.

Small values of N

For relatively small values of N ($N < 1000$), [10] suggests the use of Walsh-Hadamard codes, with parameters $[2^{k'}, k', 2^{k'-1}]_2$, while [12] improves on this by using punctured Walsh-Hadamard codes instead.
Punctured Walsh-Hadamard codes (also known as first order Reed-Muller codes) are $[2^{k'-1}, k', 2^{k'-2}]_2$-linear codes. These are the shortest possible binary linear codes for those values of N and d, as they attain the Griesmer bound. In terms of N, the parameters can be written as $[N/2, \log_2 N, N/4]_2$.

The natural generalization of these codes to $\mathbb{F}_q$ are first order q-ary Reed Muller codes, which have parameters $[q^{k-1}, k, q^{k-1} - q^{k-2}]_q$. Moreover, there is a q-ary generalization of Walsh-Hadamard codes, known as simplex codes, which have parameters $[\frac{q^k-1}{q-1}, k, q^{k-1}]_q$.

For example for $q = 4$, the parameters of the simplex code can be written in terms of N as $[(N-1)/3, \log_4 N, N/4]_4$, and hence, for the same values of d and N, the number of base OT's is reduced by a factor $3/2$ since $n/n' < 2/3$. On the other hand, the communication complexity of the encoding phase increases by a factor $2n/n' < 4/3$ compared to using binary punctured Walsh-Hadamard codes. We note, however, that this comparison is only valid if N is a power of 4.

Because of the fact that N needs to be a power of q, in the comparison table below it will be convenient to use the juxtaposition of two copies of the same code. This means that given an $[n, k, d]_q$ code $\mathcal{C}'$, we can obtain a $[2n, k, 2d]_q$ code by sending each symbol in a codeword twice. With respect to the examples listed in [12], we see that by choosing an adequate finite field and using juxtapositions of simplex codes, the number of OT's gets divided by a factor slightly over 1.5, while the communication complexity increases

by a somewhat smaller factor.

Larger values of N

For larger values of N, [12] suggests using binary BCH codes. We use q-ary BCH codes instead. It is difficult to find BCH codes that match exactly the parameters (N, d) from [12] so in our comparison we have always used larger values of both N and d. This is actually not too advantageous for our construction since the codes in [12] were selected so that their length is of the form $2^m - 1$ (what is called primitive binary BCH codes, which usually yields the constructions with best parameters) and that results in a range of parameters where it is not adequate to choose primitive q-ary BCH codes. Nevertheless, in the case where the large value $N' = 2^{443}$ is considered in [12], we can reduce the number of base OT's needed to less than half, while the communication complexity only increases by $4/3$, and in addition to that we achieve a larger value $N = 2^{462}$. Observe that, for this value of N, with a binary code the number of base OT's would be restricted by the naïve bound $n' \geq \log_2 N = 462$ in any case (i.e. even if $d = 1$), while using a code over $\mathbb{F}_8$ we only need to use 455.

Acknowledgements

The authors wish to thank Claudio Orlandi for providing helpful suggestions during the early stages of this work, and Peter Scholl for his valuable comments.

References

[1] D. Beaver, "Correlated pseudorandomness and the complexity of private computations," in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. ACM, 1996, pp. 479–488.

[2] I. Cascudo, I. B. Damgård, B. David, N. Döttling, and J. B. Nielsen, "Rate-1, linear time and additively homomorphic uc commitments," in *Advances in Cryptology – CRYPTO 2016*, M. Robshaw and J. Katz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 179–207.

[3] I. Cascudo, I. B. Damgård, B. David, I. Giacomelli, J. B. Nielsen, and R. Trifiletti, "Additively homomorphic uc commitments with optimal amortized overhead," in *Public-Key Cryptography – PKC 2015*, J. Katz, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 495–515.

[4] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, and R. Trifiletti, "On the complexity of additively homomorphic uc commitments," in *Theory of Cryptography*,

E. Kushilevitz and T. Malkin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 542–565.

[5] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87. ACM, 1987, pp. 218–229.

[6] R. Impagliazzo and S. Rudich, "Limits on the provable consequences of one-way permutations," in *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '89. ACM, 1989, pp. 44–61.

[7] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, *Extending Oblivious Transfers Efficiently*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 145–161.

[8] M. Keller, E. Orsini, and P. Scholl, "Actively secure ot extension with optimal overhead," in *Advances in Cryptology – CRYPTO 2015*, R. Gennaro and M. Robshaw, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 724–741.

[9] J. Kilian, "Founding crytpography on oblivious transfer," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. ACM, 1988, pp. 20–31.

[10] V. Kolesnikov and R. Kumaresan, *Improved OT Extension for Transferring Short Secrets*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 54–70.

[11] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*, 1st ed. North Holland, 1983.

[12] M. Orrù, E. Orsini, and P. Scholl, *Actively Secure 1-out-of-N OT Extension with Application to Private Set Intersection*. Cham: Springer International Publishing, 2017, pp. 381–396.

[13] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, ser. SFCS '82. Washington, DC, USA: IEEE Computer Society, 1982, pp. 160–164.

Paper C

Squares of Matrix-product Codes

Ignacio Cascudo, Jaron Skovsted Gundersen, and Diego Ruano

The paper has been published in the
Finite Fields and Their Applications Vol. 62, p. 101606, 2020.

The layout has been revised.

Abstract

The component-wise or Schur product $C * C'$ of two linear error-correcting codes C and C' over certain finite field is the linear code spanned by all component-wise products of a codeword in C with a codeword in C'. When $C = C'$, we call the product the square of C and denote it C^{*2}. Motivated by several applications of squares of linear codes in the area of cryptography, in this paper we study squares of so-called matrix-product codes, a general construction that allows to obtain new longer codes from several "constituent" codes. We show that in many cases we can relate the square of a matrix-product code to the squares and products of their constituent codes, which allow us to give bounds or even determine its minimum distance. We consider the well-known $(u, u + v)$-construction, or Plotkin sum (which is a special case of a matrix-product code) and determine which parameters we can obtain when the constituent codes are certain cyclic codes. In addition, we use the same techniques to study the squares of other matrix-product codes, for example when the defining matrix is Vandermonde (where the minimum distance is in a certain sense maximal with respect to matrix-product codes).

1 Introduction

Component-wise or Schur products of linear error-correcting codes have been studied for different purposes during the last decades, from efficient decoding to applications in several different areas within cryptography. Given two linear (over some finite field $\mathbb{F}$) codes C_1, C_2 of the same length we define the component-wise product of the codes $C_1 * C_2$ to be the span over $\mathbb{F}$ of all component-wise products $\mathbf{c}_1 * \mathbf{c}_2$, where $\mathbf{c}_i \in C_i$.

Many of these applications involve the study of the main parameters of interest in the theory of error-correcting codes, namely dimension and minimum distance, where in this case we want to analyze the behaviour of some of these parameters both on the factor codes and their product simultaneously. Recall that the dimension $\dim C$ of a linear code C is its dimension as a vector space over the finite field, and the minimum distance $d(C)$ of C is the smallest Hamming distance between two distinct codewords of the code. Additionally, some applications involve the analysis of the minimum distance of the duals of the factor codes (recall that the dual $C^\perp$ of C is the linear code containing all vectors that are orthogonal to all codewords of C under the standard inner product over $\mathbb{F}$).

One of the first applications where component-wise products of codes became relevant concerned error decoding via the notion of error-locating pairs in the works of Pellikaan and of Duursma and Kötter [13, 22]. An error-locating pair for a code C is a pair C_1, C_2 where $C_1 * C_2 \subseteq C^\perp$, and the number of errors the pair is able to correct depends on the dimensions and minimum

distances of the codes and their duals. More precisely, it is required that $\dim(C_1) > t$ and $d(C_2^\perp) > t$ if we should be able to locate t errors.

Later on, the use of component-wise products found several applications in the area of cryptography. For example, some attacks to variants of the McEliece cryptosystem (which relies on the assumption that it is hard to decode a general linear code) use the fact that the dimension of the product $C * C$ tends to be much larger when C is a random code than when C has certain algebraic structure, which can be used to identify algebraic patterns in certain subcodes of the code defining the cryptosystem, see for instance [7, 8, 19, 23, 27].

A different cryptographic problem where products of codes are useful is private information retrieval, where a client can retrieve data from a set of servers allocating a coded database in such a way that the servers do not learn what data the client has accessed. In [15] a private information retrieval protocol based on error-correcting codes was shown, where it is desirable to use two linear codes C_1 and C_2 such that $\dim(C_1)$, $d(C_2^\perp)$, and $d(C_1 * C_2)$ are simultaneously high.

In this work, however, we are more interested in the application of products of codes to the area of secure multiparty computation. The goal of secure multiparty computation is to design protocols which can be used in the situation where a number of parties, each holding some private input, want to jointly compute the output of a function on those inputs, without at any point needing that any party reveals his/her input to anybody. A central component in secure computation protocols is secure multiplication, which different protocols realize in different ways. Several of these protocols require to use an error-correcting code C whose square $C^{*2} = C * C$ has large minimum distance while there are additional conditions on C which vary across the different protocols.

For example a well known class of secure computation protocols [1, 6, 9] relies on the concept of strongly multiplicative secret sharing scheme formalized in [9]. Such secret sharing schemes can be constructed from linear codes C where the amount of colluding cheating parties that the protocol can tolerate is $\min(d(C^\perp), d(C^{*2})) + 2$, where $C^\perp$ is the dual code to C. These two minimum distances are therefore desired to be simultaneously high. For more information about secret sharing and multiparty computation, see for instance [10].

Other more recent protocols have the less stringent requirement that $d(C^{*2})$ and $\dim C$ are simultaneously large. This is the case of the MiniMac [12] protocol, a secure computation protocol to evaluate boolean circuits, and its successor Tinytables [11]. In those protocols, the cheating parties have certain probability of being able to disrupt the computation, but this probability is bounded by $2^{-d(C^{*2})}$, meaning that a high distance on the square will give a

higher security. On the other hand, a large relative dimension, or rate, of C will reduce the communication cost, so it is desirable to optimize both parameters. A very similar phenomenon occurs in recent work about commitment schemes, which are a building block of many multiparty computation protocols; in fact, when these schemes have a number of additional homomorphic properties and in addition can be composed securely, we can base the entire secure computation protocol on them [14]. Efficient commitment schemes with such properties were constructed in [5] based on binary linear codes, where multiplicative homomorphic properties require again to have a relatively large $d(C^{*2})$ (see [5, section 4]) and the rate of the code is also desired to be large to reduce the communication overhead.

These applications show the importance of finding linear codes where the minimum distance of the square $d(C^{*2})$ is large relative to the length of the codes and where some other parameter (in some cases $\dim(C)$, in others $d(C^{\perp})$) is also relatively large. Moreover, it is especially interesting for the applications that the codes are binary, or at least be defined over small fields.

Powers of codes, and more generally products, have been studied in several works such as [3, 4, 20, 24–26] from different perspectives. In [25] an analog of the Singleton bound for $\dim(C)$ and $d(C^{*2})$ was established, and in [20] it is shown that Reed Solomon codes are essentially the only codes which attain this bound unless some of the parameters are very restricted. However, Reed Solomon codes come with the drawback that the field size must be larger than or equal to the length of the codes. Therefore, finding asymptotically good codes over a fixed small field has also been studied, where in this case asymptotically good means that both $\dim(C)$ and $d(C^{*2})$ grows linearly with the length of the code C. In [24] the existence of such a family over the binary field was shown, based on recent results on algebraic function fields. However, it seems like most families of codes do not have this property: in fact, despite the well known fact that random linear codes will, with high probability, be over the Gilbert Varshamov bound, and hence are asymptotically good in the classical sense, this is not the case when we impose the additional restriction that $d(C^{*2})$ is linear in the length, as it is shown in [4]. The main result in [4] implies that for a family of random linear codes either the code or the square will be asymptotically bad.

The asymptotical construction from [24], despite being very interesting from the theoretical point of view, has the drawbacks that the asymptotics kick in relatively late and moreover, the construction relies on algebraic geometry, which makes it computationally expensive to construct such codes. Motivated by the aforementioned applications to cryptography, [3] focuses on codes with fixed lengths (but still considerably larger than the size of the field), and constructs cyclic codes with relatively large dimension and minimum distance of their squares. In particular, the parameters of some of these codes are explicitly computed in the binary case.

This provides a limited constellation of parameters that we know that are achievable for the tuple consisting of length of C, $\dim(C)$ and $d(C^{*2})$. It is then interesting to study what other parameters can be attained, and a natural way to do so is to study how the square operation behaves under known procedures in coding theory that allow to construct new codes from existing codes.

One such construction is matrix-product codes, where several codes can be combined into a new longer code. Matrix-product codes, formalized in [2], is a generalization of some previously known code constructions, such as for example the $(u, u+v)$-construction, also known as Plotkin sum. Matrix-product codes have been studied in several works, including [2, 16, 17, 21].

1.1 Results and Outline

In this work, we study squares of matrix-product codes. We show that in several cases, the square of a matrix-product code can be also written as a matrix-product code. This allows us to determine new achievable parameters for the squares of codes.

More concretely, we start by introducing matrix-product codes and products of codes in Section 2. Afterwards, we determine the product of two codes when both codes are constructed using the $(u, u+v)$-construction in Section 3. In Section 4, we restrict ourselves to squares of codes and exemplify what parameters we can achieve using cyclic codes in the $(u, u+v)$-construction in order to compare the parameters with the codes from [3].

At last, in Section 5, we consider other constructions of matrix-product codes. In particular, we consider the case where the defining matrix is Vandermonde, which is especially relevant because such matrix-product codes achieve the best possible minimum distance that one can hope for with this matrix-product strategy. We show that the squares of these codes are again matrix-product codes, and if the constituent codes of the original matrix-product code are denoted C_i, then the ones for the square are all of the form $\sum_{i+j=l} C_i * C_j$ for some l. This is especially helpful for determining the parameters if the C_i's are for example algebraic geometric codes. We remark that this property also holds for the other constructions we study in this paper, but only when the C_i's are nested. Finally, we also study the squares of a matrix-product construction from [2] where we can apply the same proof techniques as we have in the other constructions.

2 Preliminaries

Let $\mathbb{F}_q$ be the finite field with q elements. A linear code C is a subspace of $\mathbb{F}_q^n$. When C has dimension k, we will call it an $[n,k]_q$ code. A generator

matrix for a code C is a $k \times n$ matrix consisting of k basis vectors for C as the k rows. The Hamming weight of $\mathbf{x} \in \mathbb{F}_q^n$, denoted $w(\mathbf{x})$, is the number of nonzero entries in $\mathbf{x}$ and the Hamming distance between $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ is given by $d(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y})$. By the linearity of C the minimum Hamming distance taken over all pairs of distinct elements in C is the same as the minimum Hamming weight taking over all non-zero elements in C, and therefore we define the minimum distance of C to be $d(C) = \min_{\mathbf{x} \in C \setminus \{0\}} \{w(\mathbf{x})\}$. If it is known that $d(C) = d$ (respectively if we know that $d(C) \geq d$) then we call C an $[n, k, d]_q$ code (resp. $[n, k, \geq d]_q$). We denote by $C^\perp$ the dual code to C, i.e., the vector space given by all elements $\mathbf{y} \in \mathbb{F}_q^n$ such that for every $\mathbf{x} \in C$, $\mathbf{x}$ and $\mathbf{y}$ are orthogonal with respect to the standard inner product in $\mathbb{F}_q^n$. If C is an $[n, k]_q$ code then $C^\perp$ is an $[n, n-k]_q$ code.

We recall the definition and basic properties of matrix-product codes (following [2]) and squares of codes.

Definition 2.1 (Matrix-product code). *Let $C_1, \ldots, C_s \subseteq \mathbb{F}_q^n$ be linear codes and let $A \in \mathbb{F}_q^{s \times l}$ be a matrix with rank s (implying $s \leq l$). Then we define the matrix-product code $C = [C_1, \ldots, C_s]A$, as the set of all matrix products $[\mathbf{c}_1, \ldots, \mathbf{c}_s]A$, where $\mathbf{c}_i = (c_{1i}, \ldots, c_{ni})^T \in C_i$.*

We call $A = \left[a_{ij}\right]_{i=1,\ldots,s, j=1,\ldots,l}$ the defining matrix and the C_i's the constituent codes. We can consider a codeword $\mathbf{c}$, in a matrix-product code, as a matrix of the form

$$\mathbf{c} = \begin{bmatrix} c_{11}a_{11} + c_{12}a_{21} + \cdots + c_{1s}a_{s1} & \cdots & c_{11}a_{1l} + c_{12}a_{2l} + \cdots + c_{1s}a_{sl} \\ \vdots & \ddots & \vdots \\ c_{n1}a_{11} + c_{n2}a_{21} + \cdots + c_{ns}a_{s1} & \cdots & c_{n1}a_{1l} + c_{n2}a_{2l} + \cdots + c_{ns}a_{sl} \end{bmatrix}, \tag{C.1}$$

using the same notation for the $\mathbf{c}_i$'s as in the definition. Reading the entries in this matrix in a column-major order, we can also consider $\mathbf{c}$ as a vector of the form

$$\mathbf{c} = \left(\sum_{i=1}^{s} a_{i1}\mathbf{c}_i, \ldots, \sum_{i=1}^{s} a_{il}\mathbf{c}_i \right) \in \mathbb{F}_q^{nl}. \tag{C.2}$$

We sum up some known facts about matrix-product codes in the following proposition.

Proposition 2.2. *Let $C_1, \ldots, C_s \subseteq \mathbb{F}_q^n$ be linear $[n, k_1], \ldots, [n, k_s]$ codes with generator matrices $G_1, \ldots, G_s$, respectively. Furthermore, let $A \in \mathbb{F}_q^{s \times l}$ be a matrix with rank s and let $C = [C_1, \ldots, C_s]A$. Then C is an $[nl, k_1 + \cdots + k_s]$ linear code and*

a generator matrix of C is given by

$$G = \begin{bmatrix} a_{11}G_1 & \cdots & a_{1l}G_1 \\ \vdots & \ddots & \vdots \\ a_{s1}G_s & \cdots & a_{sl}G_s \end{bmatrix}.$$

We now turn our attention to the minimum distance of C. Denote by A_i the matrix consisting of the first i rows of A and let C_{A_i} be the linear code spanned by the rows in A_i. From [21], we have the following result on the minimum distance.

Proposition 2.3. *We are making the same assumptions as in Proposition 2.2, and write $D_i = d(C_{A_i})$ and $d_i = d(C_i)$. Then the minimum distance of the matrix-product code C satisfies*

$$d(C) \geq \min\{D_1 d_1, D_2 d_2, \ldots, D_s d_s\}. \tag{C.3}$$

The following corollary is from [16].

Corollary 2.4. *If we additionally assume that $C_1 \supseteq \cdots \supseteq C_s$, equality occurs in the bound in* (C.3).

The dual of a matrix-product code is also a matrix-product code, if we make some assumptions on the matrix A, as it was noted in [2].

Proposition 2.5. *Let $C = [C_1, C_2, \ldots, C_s]A$ be a matrix product code. If A is an invertible square matrix then*

$$C^{\perp} = [C_1^{\perp}, C_2^{\perp}, \ldots, C_s^{\perp}](A^{-1})^T$$

Additionally, if J is the $s \times s$ matrix given by

$$J = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & \iddots & \vdots & \vdots \\ 1 & \cdots & 0 & 0 \end{bmatrix}$$

the dual can be described as

$$C^{\perp} = [C_s^{\perp}, C_{s-1}^{\perp}, \ldots, C_1^{\perp}](J(A^{-1})^T).$$

Notice that with regard to Proposition 2.3 the last expression is often more useful since $d(C_i^{\perp})$ will often decrease when i increases.

Now, we turn our attention to products and squares of codes. We denote by $*$ the component-wise product of two vectors. That is, if $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$, then $\mathbf{x} * \mathbf{y} = (x_1 y_1, \ldots, x_n y_n)$. With this definition in mind, we define the product of two linear codes.

Definition 2.6 (Component-wise (Schur) products and squares of codes). *Given two linear codes $C, C' \subseteq \mathbb{F}_q^n$ we define their component-wise product, denoted by $C * C'$, as*

$$C * C' = \langle \{ \mathbf{c} * \mathbf{c}' \mid \mathbf{c} \in C, \mathbf{c}' \in C' \} \rangle.$$

The square of a code C is $C^{*2} = C * C$.

First note that the length of the product is the same as the length of the original codes. Regarding the other parameters (dimension and minimum distance) we enumerate some known results only in the case of the squares C^{*2} since this will be our primary focus.

If

$$G = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix}$$

is a generator matrix for C, then $\{\mathbf{g}_i * \mathbf{g}_j \mid 1 \leq i \leq j \leq n\}$ is a generating set for C^{*2}. However, it might not be a basis since some of the vectors might be linearly dependent. If additionally, a submatrix consisting of k columns of G is the identity, the set $\{\mathbf{g}_i * \mathbf{g}_i\}$ consists of k linearly independent vectors. Since there is always a generator matrix satisfying this, this implies $k \leq \dim(C^{*2}) \leq \frac{k(k+1)}{2}$, where $k = \dim(C)$, and $d(C^{*2}) \leq d(C)$.

In most cases, however, $d(C^{*2})$ is much smaller than $d(C)$. For example, the Singleton bound for squares [25] states that $d(C^{*2}) \leq \max\{1, n-2k+2\}$ (which is much restrictive than the Singleton bound for C, which states that $d(C) \leq n-k+1$). Additionally, the codes for which $d(C^{*2}) = n-2k+2$ have been characterized in [20], where it was shown that essentially only Reed-Solomon codes, certain direct sums of self-dual codes, and some degenerate codes have this property. Furthermore, it is shown in [4] that taking a random code with dimension k the dimension of C^{*2} will with high probability be $\min\{n, \frac{k(k+1)}{2}\}$. Therefore, often $\dim(C^{*2}) \gg \dim(C)$ and hence typically $d(C^{*2}) \ll d(C)$.

3 The $(u, u+v)$-Construction

In this section, we will consider one of the most well-known matrix-product codes, namely the $(u, u+v)$-construction. We obtain this construction when we let

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \in \mathbb{F}_q^{2\times 2} \tag{C.4}$$

be the defining matrix.
Note that if $C = [C_1, C_2]\, A$ then $d(C) \geq \min\{2d(C_1), d(C_2)\}$ and $d(C^\perp) \geq \min\{2d(C_2^\perp), d(C_1^\perp)\}$. This can easily be deduced from Propositions 2.3 and 2.5 by constructing $J(A^{-1})^T$.

In the following theorem, we will determine the product of two codes C and C' when both codes come from the $(u, u+v)$-construction. We will use the notation $C + C'$ to denote the smallest linear code containing both C and C'.

Theorem 3.1. *Let $C_1, C_2, C_1', C_2' \subseteq \mathbb{F}_q^n$ be linear codes. Furthermore, let A be as in* (C.4) *and denote by $C = [C_1, C_2]A$ and by $C' = [C_1', C_2']A$. Then*

$$C * C' = [C_1 * C_1', C_1 * C_2' + C_2 * C_1' + C_2 * C_2']A.$$

Proof. Let G_1, G_2, G_1', G_2' be generator matrices for C_1, C_2, C_1', C_2' respectively. By Proposition 2.2, we have that

$$G = \begin{bmatrix} G_1 & G_1 \\ 0 & G_2 \end{bmatrix}, \quad G' = \begin{bmatrix} G_1' & G_1' \\ 0 & G_2' \end{bmatrix}$$

are generator matrices for C and C' respectively. A generator matrix for $C * C'$ can be obtained by making the componentwise products of all the rows in G with all the rows in G' and afterwards removing all linearly dependent rows. We denote by $G * G'$ the matrix consisting of all componentwise products of rows in G with rows in G'. Then

$$G * G' = \begin{bmatrix} G_1 * G_1' & G_1 * G_1' \\ 0 & G_1 * G_2' \\ 0 & G_2 * G_1' \\ 0 & G_2 * G_2' \end{bmatrix}.$$

The set of rows in $G_i * G_j'$ is a generating set for $C_i * C_j'$. Hence, by removing linearly dependent rows we obtain a generator matrix of the form

$$\tilde{G} = \begin{bmatrix} \tilde{G}_1 & \tilde{G}_1 \\ 0 & \tilde{G}_2 \end{bmatrix},$$

where $\tilde{G}_1$ is a generator matrix for $C_1 * C_1'$, and $\tilde{G}_2$ for $C_1 * C_2' + C_2 * C_1' + C_2 * C_2'$. By using Proposition 2.2 once again, we see that $\tilde{G}$ is a generator matrix for the code $[C_1 * C_1', C_1 * C_2' + C_2 * C_1' + C_2 * C_2']A$ proving the theorem. □

The following corollary consider the square of a code from the $(u, u+v)$-construction, and in the remaining of the paper the focus will be on squares.

Corollary 3.2. *Let $C_1, C_2 \subseteq \mathbb{F}_q^n$ be linear codes. Furthermore, let A be as in* (C.4) *and denote by $C = [C_1, C_2]A$. Then*

$$C^{*2} = [C_1^{*2}, (C_1 + C_2) * C_2]A,$$

and we have that

$$d(C^{*2}) \geq \min\{2d(C_1^{*2}), d((C_1 + C_2) * C_2)\}. \tag{C.5}$$

Additionally, if $C_2 \subseteq C_1$ we obtain

$$C^{*2} = [C_1^{*2}, C_1 * C_2]A,$$

and we have that

$$d(C^{*2}) = \min\{2d(C_1^{*2}), d(C_1 * C_2)\}. \tag{C.6}$$

Proof. The results follows by setting $C_i' = C_i$ in Theorem 3.1 implying $C' = C$, and we obtain that

$$C^{*2} = [C_1 * C_1, C_1 * C_2 + C_2 * C_1 + C_2 * C_2]A = [C_1^{*2}, (C_1 + C_2) * C_2]A.$$

If $C_2 \subseteq C_1$ we have $C_1 + C_2 = C_1$. The bound in (C.5) follows directly from Proposition 2.3, and (C.6) follows by Corollary 2.4. □

4 Constructions from Binary Cyclic Codes

In this section, we exemplify what parameters we can achieve for C and C^{*2} when we use the $(u, u + v)$-construction together with cyclic codes as constituent codes. We start by presenting some basics of cyclic codes.

Cyclic codes are linear codes which are invariant under cyclic shifts. That is, if $\mathbf{c} = (c_0, c_1, \ldots, c_{n-2}, c_{n-1})$ is a codeword then $(c_{n-1}, c_0, c_1, \ldots, c_{n-2})$ is as well. We will assume that $\gcd(n, q) = 1$. A cyclic code of length n over $\mathbb{F}_q$ is isomorphic to an ideal in $R = \mathbb{F}_q[x]/\langle x^n - 1\rangle$ generated by a polynomial g, where $g | x^n - 1$. The isomorphism is given by

$$c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} \mapsto (c_0, c_1, \ldots, c_{n-1}),$$

and we notice that a cyclic shift is represented by multiplying by x. The cyclic code generated by g has dimension $n - \deg g$. To bound the minimum distance of the code, we introduce the q-cyclotomic cosets modulo n.

Definition 4.1 (q-cyclotomic coset modulo n). *Let $a \in \mathbb{Z}/n\mathbb{Z}$. Then the q-cyclotomic coset modulo n of a is given by*

$$[a] = \{aq^j \bmod n \mid j \geq 0\}.$$

Now let $\beta^n = 1$ and $\beta^k \neq 1$ for $1 \leq k \leq n$, meaning that β is a primitive n-th root of unity in an algebraic closure of $\mathbb{F}_q$. Since $g | x^n - 1$ every root of g must be of the form β^j for some $j \in \{0, 1, \ldots, n-1\} = \mathbb{Z}/n\mathbb{Z}$. This leads to the following definition which turns out to be useful in describing the parameters of a cyclic code.

Definition 4.2 (Defining and generating set). *Denote by* $J = \{j \in \mathbb{Z}/n\mathbb{Z} \mid g(\beta^j) = 0\}$ *and* $I = \{j \in \mathbb{Z}/n\mathbb{Z} \mid g(\beta^j) \neq 0\}$. *Then we call* J *the defining set and* I *the generating set of the cyclic code* C *generated by* g.

We remark that $g = \prod_{j\in J}(x - \beta^j) = (x^n - 1)/\prod_{i\in I}(x - \beta^i)$, implying that $|I|$ is the dimension of the cyclic code generated by g. We note that I and J must be a union of q-cyclotomic cosets modulo n. Now we define the amplitude of I as

$$\mathrm{Amp}(I) = \min\{i \in \mathbb{Z} \mid \exists c \in \mathbb{Z}/n\mathbb{Z} \text{ such that } I \subseteq \{c, c+1, \ldots, c+i-1\}\}.$$

As a consequence of the BCH-bound, see for example [3], we have that the minimum distance of the code generated by g is greater than or equal to $n - \mathrm{Amp}(I) + 1$.

Hence, we see that both the dimension and minimum distance depend on I, and since C is uniquely determined by I, we will use the notation $C(I)$ to describe the cyclic code generated by $g = (x^n - 1)/(\prod_{i\in I}(x - \beta^i))$. To summarize, we have that $C(I)$ is a cyclic linear code with parameters

$$[n, |I|, \geq n - \mathrm{Amp}(I) + 1]_q. \tag{C.7}$$

The dual of a cyclic code is also a cyclic code. In fact, $C(I)^\perp = C(-J)$, where $-J = \{-j \bmod n \mid j \in J\}$. Clearly $\mathrm{Amp}(-J) = \mathrm{Amp}(J)$ which shows that $C(I)^\perp$ is a cyclic linear code with parameters

$$[n, |J|, \geq n - \mathrm{Amp}(J) + 1]_q. \tag{C.8}$$

Furthermore, we remark that $\mathrm{Amp}(J) = n - \max\{s \mid \{i, i+1, \ldots, i+s-1\} \subseteq I \text{ for some } i\}$, i.e n minus the size of the largest set of consecutive elements in I. We conclude that the minimum distance of $C(I)^\perp$ is strictly larger than the size of any set of consecutive elements in I.

We consider cyclic codes for the $(u, u+v)$-construction, and therefore we will need the following proposition.

Proposition 4.3. *Let* I_1 *and* I_2 *be unions of* q*-cyclotomic cosets, and let* $C(I_1)$ *and* $C(I_2)$ *be the corresponding cyclic codes. Then*

$$C(I_1) + C(I_2) = C(I_1 \cup I_2)$$
$$C(I_1) * C(I_2) = C(I_1 + I_2),$$

where $I_1 + I_2 = \{a + b \bmod n \mid a \in I_1, b \in I_2\}$.

We obtain this result by describing the cyclic codes as a subfield subcode of an evaluation code and generalizing Theorem 3.3 in [3]. The proof of this proposition is very similar to the one in [3] and can be found in Appendix A. The proposition implies the following corollary.

Corollary 4.4. *Let I_1 and I_2 be unions of q-cyclotomic cosets, and let $C(I_1)$ and $C(I_2)$ be the corresponding cyclic codes. Then $C(I_1) * C(I_2)$ is an*

$$[n, |I_1 + I_2|, \geq n - \mathrm{Amp}(I_1 + I_2) + 1]_q$$

cyclic code.

Now, let $C(I_1)$ and $C(I_2)$ be two cyclic codes over $\mathbb{F}_q$ of length n, and let

$$C = [C(I_1), C(I_2)] \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \tag{C.9}$$

Then C is a

$$[2n, |I_1| + |I_2|, \geq \min\{2(n - \mathrm{Amp}(I_1) + 1), n - \mathrm{Amp}(I_2) + 1\}]_q$$

linear code. This is in fact a quasi-cyclic code of index 2, see for instance [16, 18]. By combining Corollary 3.2 with Proposition 4.3, we obtain that

$$C^{*2} = [C(I_1 + I_1), C(I_2 + (I_1 \cup I_2))] \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \tag{C.10}$$

And from Propositions 2.2 and 2.3, and Corollary 4.4, we obtain that

$$\dim(C^{*2}) = |I_1 + I_1| + |I_2 + (I_1 \cup I_2)|,$$

and

$$d(C^{*2}) \geq \min\{2(n - \mathrm{Amp}(I_1 + I_1) + 1), n - \mathrm{Amp}(I_2 + (I_1 \cup I_2)) + 1)\}. \tag{C.11}$$

Therefore, it is of interest to find I_1 and I_2 such that the cardinalities of these sets are relatively large, implying a large dimension of C, while at the same time $\mathrm{Amp}(I_1 + I_1)$ and $\mathrm{Amp}(I_2 + (I_1 \cup I_2))$ are relatively small, implying a large minimum distance on the square.

To exemplify what parameters we can obtain we will use some specific cyclic codes from [3] based on the notion of s-restricted weights of cyclotomic cosets introduced in the same article. Let $n = q^r - 1$ for some r and for a number $t \in \{0, 1, \ldots, n-1\}$ let $(t_{r-1}, t_{r-2}, \ldots, t_0)$ be its q-ary representation, i.e. $t = \sum_{i=0}^{r-1} t_i q^i$, where $t_i \in \{0, 1, \ldots, q-1\}$. Then for an $s \leq r$ the s-restricted weight is defined as

$$w_q^{(s)}(t) = \max_{i \in \{0,1,\ldots,r-1\}} \sum_{j=0}^{s-1} t_{i+j}.$$

We will not go into details about these s-restricted weights but we refer the reader to [3] for more information. However, we remark that [3] proves that

this weight notion satisfies $w_q^{(s)}(v) \leq w_q^{(s)}(t) + w_q^{(s)}(u)$ if $v = t + u$, and that $w_q^{(s)}(t) = w_q^{(s)}(u)$ whenever t and u are in the same cyclotomic coset. The latter implies that we can talk about the s-restricted weight of a cyclotomic coset.

Let $W_{r,s,m}$ denote the union of all cyclotomic cosets modulo $q^r - 1$ with s-restricted weights lower than or equal to m. I.e.

$$W_{r,s,m} = \{t \in \{0, 1, \ldots, q^r - 2\} \mid w_q^{(s)}(t) \leq m\}.$$

We remark that the minimum distance of $C(W_{r,s,m})$ can be deduced using that an upper bound for $\mathrm{Amp}(W_{r,s,m})$ is $\max W_{r,s,m} + 1$ and the maximum element of $W_{r,s,m}$ can be easily deduced [3]. Furthermore, the dimension of $C(W_{r,s,m})$ is simply the cardinality of $W_{r,s,m}$, which either can be counted for the specific choices of r, s, and m, or can be expressed as a recurrent sequence in r (for a fixed selection of adequately small s and m) using an argument involving counting closed walks of length r in certain graph, see [3]. Finally, it is not hard to realize that the largest set of consecutive elements in $W_{r,s,m}$ is $\{0, 1, \ldots, 2^{m+1} - 2\}$ and thus, by the remarks about the dual distance after equation (C.8), we have that $d(C(W_{r,s,m})^\perp) \geq 2^{m+1}$.

We define the code

$$C = [C(W_{r,s,m_1}), C(W_{r,s,m_2})] \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},$$

where we let $m_1 \geq m_2$. Note that $d(C^\perp) \geq \min\{2 \cdot 2^{m_2+1}, 2^{m_1+1}\}$. From (C.10) we conclude that

$$C^{*2} = [C(W_{r,s,m_1} + W_{r,s,m_1}), C(W_{r,s,m_1} + W_{r,s,m_2})] \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{C.12}$$

since $W_{r,s,m_1} \cup W_{r,s,m_2} = W_{r,s,m_1}$. It is noted in [3] that $W_{r,s,m_i} + W_{r,s,m_j} = W_{r,s,m_i+m_j}$ does not hold in general, but the inclusion $W_{r,s,m_i} + W_{r,s,m_j} \subseteq W_{r,s,m_i+m_j}$ holds. However, we are able to determine the exact dimension for C^{*2} in (C.12) by computing $W_{r,s,m_1} + W_{r,s,m_i}$ for $i = 1, 2$. Additionally, when computed these, we can bound the minimum distance directly from (C.11). This is what we do in Table C.1 for the following choices. We present the parameters for C and C^{*2} when setting $q = 2$, $s = 5$, $m_1 = 2$, and $m_2 = 1$. In this case we have $d(C^\perp) \geq 8$ for each r.

We make a comparison to the cyclic codes from [3]. They present codes constructed using the 3-restricted weight with $m = 1$ (Table 1 in [3]) and using the 5-restricted weight with $m = 2$ (Table 2 in [3]). Let any one of our new codes from Table C.1 have parameters

$$(n, \dim(C), d(C^{*2})) = (n, k, d^*).$$

r	n	$\dim(C)$	$d(C) \geq$	$\dim(C^{*2})$	$d(C^{*2}) \geq$
5	62	22	14	57	2
6	126	29	30	99	6
7	254	37	62	163	14
8	510	54	126	348	18
9	1022	86	238	650	38
10	2046	142	462	1319	66
11	4094	233	926	2543	134

Table C.1: Parameters for C and C^{*2} using the 5-restricted weight

First we compare to Table 1 from [3], where there always exists a code C' with length $n+1$, $\dim(C') < k$, and $d((C')^{*2}) > d^*$. Hence, our new codes have larger dimension but lower minimum distance for the square compared to these codes, for comparable lengths. On the other hand, in Table 2 from [3] there is a code C' with length $n+1$ and $\dim(C') \geq k$ (i.e. the dimensions of the codes from [3] are larger than those in our table). However, the minimum distances of the squares for the codes in [3] satisfy

$$d((C')^{*2}) = \begin{cases} d^*+1 & \text{for } r = 5,6,8,10,11 \\ d^*-5 & \text{for } r = 7,9 \end{cases}.$$

Thus, even though the dimension of our codes are lower than the ones from Table 2 in [3], for $r = 7$ and $r = 9$ we obtain that $d^* > d((C')^{*2})$.

Therefore, our results on matrix-product codes allow us to obtain codes with a different trade-off between $\dim C$ and $d(C^{*2})$ than those from [3], where we can obtain a larger distance of the square at the expense of reducing the dimension with respect to one of the tables there, and viceversa with respect to the other.

5 Other Matrix-Product Codes

In this section, we consider squares of some other families of matrix-product codes. We start by determining the square of C when C is a matrix-product code where the defining matrix A is Vandermonde.

Theorem 5.1. *Let $C_0, C_1, \cdots, C_{s-1}$ be linear codes in $\mathbb{F}_q^n$. Furthermore, let*

$$V_q(s) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1^1 & \alpha_2^1 & \cdots & \alpha_{q-1}^1 \\ \vdots & \vdots & & \vdots \\ \alpha_1^{s-1} & \alpha_2^{s-1} & \cdots & \alpha_{q-1}^{s-1} \end{bmatrix},$$

where the α_i's are distinct nonzero elements in $\mathbb{F}_q$ and $s \leq q-1$ is some positive integer. Denote by $C = [C_0, C_1, \ldots, C_{s-1}]V_q(s)$. Then

$$C^{*2} = [\sum_{i+j=0} C_i * C_j, \sum_{i+j=1} C_i * C_j, \ldots, \sum_{i+j=\tilde{s}-1} C_i * C_j]V_q(\tilde{s})$$

where $\tilde{s} = \min\{2s-1, q-1\}$ and the sums $i+j$ are modulo $q-1$.

Proof. Let $G_0, G_1, \ldots, G_{s-1}$ be generator matrices of $C_0, C_1, \ldots, C_{s-1}$ respectively and let G be a generator matrix for C. Using the same notation as in the proof of Theorem 3.1, $G * G$ contains all rows of the form

$$(\alpha_1^{i+j} G_i * G_j, \ldots, \alpha_q^{i+j} G_i * G_j)$$

for $i, j = 0, 1, \ldots, s-1$. Note that if $i+j \geq q-1$ then $\alpha^{i+j} = \alpha^{i+j-q+1}$ and hence we can consider $i+j$ modulo $q-1$. Thus if $l \leq q-1$ and $i+j \equiv l \pmod{q-1}$ we could write the coefficients in front of $G_i * G_j$ as α_k^l for $k = 1, 2, \ldots, n$. Removing linearly dependent rows this results in a generator matrix for a matrix-product code of the form

$$C^{*2} = [\sum_{i+j=0} C_i * C_j, \sum_{i+j=1} C_i * C_j, \ldots, \sum_{i+j=\tilde{s}-1} C_i * C_j]V_q(\tilde{s}), \tag{C.13}$$

where again $i+j$ is considered modulo $q-1$. □

As we will show below, the fact that we obtain codes of the form $\sum_{i+j=l} C_i * C_j$ is especially helpful for determining the parameters of C^{*2} in some cases. We remark that the same phenomenon occurs in the case of the $(u, u+v)$ construction but only if the codes C_i are nested.

Note also that $C(V_q(s)_i)$ (the linear code spanned by the first i rows of $V_q(s)$) is a Reed Solomon code[1] of length $q-1$ and dimension i and hence we have that $d(C(V_q(s)_i)) = q-i$, for $i = 1, 2, \ldots, s$. Applying Proposition 2.3 with $D_i = d(C(V_q(s)_{i+1})) = q-i-1$ and $d_i = d(C_i)$ for $i = 0, 1, \ldots, s-1$ we obtain that C is a

$$\left[(q-1)n, k_0 + k_1 + \cdots + k_{s-1}, \geq \min_{i \in \{0,1,\cdots,s-1\}} \{(q-i-1)d_i\}\right]_q$$

linear code, and C^{*2} has minimum distance greater than or equal to

$$\min_{l \in \{0,1,\cdots,\tilde{s}-1\}} \Big\{(q-l-1)d\big(\sum_{i+j=l} C_i * C_j\big)\Big\}. \tag{C.14}$$

[1] A Reed-Solomon code is an MDS code meaning that it achieves the highest possible minimum distance for a given length and dimension. Thus the D_i's are maximal and hence we obtain the best possible bound for the minimum distance we can hope for using the matrix-product construction.

Even though the expression in (C.13) may at first sight seem hard to work with, this is not the case if the C_i's come from some specific families of codes. For example, Proposition 4.3 tells us that $\sum_{i+j=l} C_i * C_j$ will again be a cyclic code if the C_i's are cyclic and we will be able to determine its generating set from the generating sets of the C_i's.

Additionally, one could consider the case where the C_i's are Reed-Solomon codes or more generally algebraic geometric codes. Let $D = P_1 + P_2 + \cdots + P_n$ be a formal sum of rational places in a function field over $\mathbb{F}_q$ and let $G_i = r_{i,1}Q_1 + r_{i,2}Q_2 + \cdots + r_{i,m}Q_m$ where all the Q_i's and P_j's are different. An algebraic geometric code $C_i = C_{\mathcal{L}}(D, G_i)$ is the evaluation of the elements in the Riemann-Roch space $\mathcal{L}(G_i)$ in the places from D. It is then known that $C_i * C_j \subseteq C_{\mathcal{L}}(D, G_i + G_j)$ and $C_i + C_j \subseteq C_{\mathcal{L}}(D, H)$, where $H = \sum_{k=1}^{m} \max\{r_{i,k}, r_{j,k}\}Q_k$. Hence, we can find a lower bound for $d(C^{*2})$ from (C.14) using the fact that from the above observations we can find algebraic geometric codes containing $\sum_{i+j=l} C_i * C_j$ where we can control the minimum distance. Furthermore, if $\deg G_i \geq 2g + 1$ and $\deg G_j \geq 2g$, where g is the genus of the function field, we obtain that $C_{\mathcal{L}}(D, G_i) * C_{\mathcal{L}}(D, G_j) = C_{\mathcal{L}}(D, G_i + G_j)$, see for instance [7]. Similarly, if we only consider one-point codes, meaning that $G_i = r_i Q$, we obtain that $C_i \subseteq C_j$ if $r_j \geq r_i$ and hence $C_i + C_j = C_j$. We exemplify some specific constructions with algebraic geometric codes, more specific one-point Hermitian codes, in the following example.

Example
We will not go into details about the Hermitian function field and codes, but we do mention that the Hermitian function field is defined over $\mathbb{F}_{q^2}$, it has genus $g = \frac{q(q-1)}{2}$, and it has $q^3 + 1$ rational places, where one of these places is the place at infinity. Denote the place at infinity by Q and the remaining rational places by P_i, for $i = 1, 2, \ldots, q^3$, and let $D = \sum_{i=1}^{q^3} P_i$. Then a Hermitian code is given by the algebraic geometric code $C_r = C_{\mathcal{L}}(D, rQ)$. This is a $[q^3, r - g + 1, q^3 - r]_{q^2}$ code as long as $2g \leq r \leq q^3 - q^2 - 1$, see for instance [28]. Denote by

$$C(r, s) = [C_{r+s-1}, C_{r+s-2}, \ldots, C_r]V_{q^2}(s),$$

where $2 \leq s \leq \frac{q^2}{2}$ and $2g + 1 \leq r$. Furthermore, assume that $r + s \leq \frac{q^3 - q^2 + 1}{2}$. With such a construction we have that

$$(C(r, s))^{*2} = [C_{2r+2s-2}, C_{2r+2s-3}, \ldots, C_{2r}]V_{q^2}(2s - 1) = C(2r, 2s - 1) \quad \text{(C.15)}$$

from the observations about algebraic geometric codes above the example. Note that $2r + 2s - 2 \leq q^3 - q^2 - 1$ implying that all the Hermitian codes in (C.15) *satisfy*

that their r is lower than $q^3 - q^2 - 1$. Hence,

$$\begin{aligned} d((C(r,s))^{*2}) &= \min_{i=0,1\ldots,2s-2}\{(q^2-i-1)(q^3-2r-2s+2+i)\} \\ &= (q^2-2s+1)(q^3-2r), \end{aligned}$$

where the last equality follows from the following observations:

$$\begin{aligned} &(q^2-i-1)(q^3-2r-2s+2+i) \\ &= (q^2-2s+1+(2s-2-i))(q^3-2r-(2s-2-i)) \\ &= (q^2-2s+1)(q^3-2r)+(2s-2-i)(q^3-2r-q^2+2s-2-2s+2+i) \\ &= (q^2-2s+1)(q^3-2r)+(2s-2-i)(q^3-q^2-2r+i). \end{aligned}$$

From the restrictions on r the last factor is positive. Hence, the minimum is attained when $i = 2s-2$. Similar arguments show that $d(C(r,s)) = (q^2-s)(q^3-r)$. Summing up, we have that $C(r,s)$ has parameters

$$\left[q^5-q^3, s\left(r-g+\frac{s+1}{2}\right), (q^2-s)(q^3-r)\right]_{q^2}$$

and its square is the code $C(2r, 2s-1)$ with parameters

$$\left[q^5-q^3, (2s-1)\,(2r-g+s)\,, (q^2-2s+1)(q^3-2r)\right]_{q^2}.$$

The parameters of this construction for $q = 4$ can be found in Table C.2. ◀

Finally, we remark that the strategy used in the proof of Theorems 3.1 and 5.1 is more or less identical. Thus, such strategy may also be used for other matrix-product codes. For example, we can consider the matrix-product codes with defining matrix

$$MS_p = \left[\binom{p-i}{j-1} \bmod p\right]_{i,j=1,2,\ldots,p},$$

where p is a prime number. We remark that $\binom{n}{k} = 0$ if $k > n$ and that these matrix-product codes were also considered in [2]. We show that, as for Theorems 3.1 and 5.1, we can express the square of such matrix-product codes.

Theorem 5.3. *Let p be a prime number and let $C_1 \supseteq C_2 \supseteq \cdots \supseteq C_p$ be linear codes in $\mathbb{F}_q^n$, where $\mathbb{F}_q$ has characteristic p. Denote $C = [C_1, C_2, \ldots, C_p]MS_p$ and*

$$MS_p^* = \left[\binom{p-i}{j-1}\binom{p-1}{j-1} \bmod p\right]_{i,j=1,2,\ldots,p}.$$

Then

$$C^{*2} = [\sum_{i+j=2} C_i * C_j, \sum_{i+j=3} C_i * C_j, \ldots, \sum_{i+j=p+1} C_i * C_j]MS_p^*.$$

Field size	r	s	n	k	d	k^*	d^*
16	13	2	960	17	714	66	494
16	16	2	960	23	672	84	416
16	19	2	960	29	630	102	338
16	22	2	960	35	588	120	260
16	13	4	960	38	612	168	342
16	16	4	960	50	576	210	288
16	19	4	960	62	540	252	234
16	13	6	960	63	510	286	190
16	16	6	960	81	480	352	160
16	13	7	960	77	459	351	114
16	16	7	960	98	432	429	96
16	13	8	960	92	408	420	38
16	16	8	960	116	384	510	32

Table C.2: Parameters for $C(s,r)$ and $(C(s,r))^{*2}$ with the construction from Theorem 5.1 and Hermitian codes. We use the notation $k = \dim(C(s,r))$, $k^* = \dim((C(s,r))^{*2})$, $d = d(C(s,r))$, and $d^* = d((C(s,r))^{*2})$.

Proof. Let G_i be a generator matrix for C_i and let G be a generator matrix for C. As in the previous proofs we construct $G * G$, the matrix whose rows span C^{*2}. The rows in $G * G$ are given by

$$\left[\binom{p-m}{j-1}\binom{p-n}{j-1} G_m * G_n\right]_{j=1,2,\ldots,p} \tag{C.16}$$

for $1 \leq m \leq n \leq p$. We remark that the codes are nested and hence we might assume that rows in $G_m * G_n$ are also rows in $G_i * G_n$ when $m \geq i$.

We will show that by doing row operations on $G * G$ we can obtain the rows

$$\left[\binom{p-1}{j-1}\binom{p-m-n+1}{j-1} G_m * G_n\right]_{j=1,2,\ldots,p}. \tag{C.17}$$

Notice that the coefficients only depends on $m+n$, so proving that we can obtain (C.17) by row operations proves the theorem.

In order to obtain this, we note that we can replace the rows corresponding to $G_m * G_n$ by any linear combination of the form

$$\sum_{i=1}^{m} a_i \left[\binom{p-i}{j-1}\binom{p-n}{j-1} G_m * G_n\right]_{j=1,2,\ldots,p}. \tag{C.18}$$

Here we have used that, as remarked in the beginning of the proof, rows in $G_m * G_n$ are included in $G_i * G_n$ when $i \leq m$ and therefore the rows in the linear combination given by (C.18) are included in those from (C.16).

Thus, in order to end the proof, we need to show that there exist coefficients $a_i \in \mathbb{F}_q$ such that

$$\sum_{i=1}^{m} a_i \binom{p-i}{j-1}\binom{p-n}{j-1} = \binom{p-1}{j-1}\binom{p-m-n+1}{j-1}, \text{ for } j = 1, 2, \ldots, p. \tag{C.19}$$

We observe that if $\binom{p-n}{j-1} = 0$ then $\binom{p-n-m+1}{j-1} = 0$. Hence, we only need to prove that (C.19) holds for $j - 1 \leq p - n$.

Now note that

$$\binom{p-i}{j-1} = \frac{(p-j)(p-j-1)\cdots(p-i-j+2)}{(p-1)(p-2)\cdots(p-i+1)}\binom{p-1}{j-1}, \quad \text{and}$$
$$\binom{p-m-n+1}{j-1} = \frac{(p-n-j+1)(p-n-j)\cdots(p-n-j-m+3)}{(p-n)(p-n-1)\cdots(p-n-m+2)}\binom{p-n}{j-1}$$

Plugging this into (C.19) and dividing by $\binom{p-1}{j-1}\binom{p-n}{j-1}$ we see that

$$\sum_{i=1}^{m} a_i \prod_{k=0}^{i-2} \frac{p-j-k}{p-1-k} = \prod_{k=0}^{m-2} \frac{p-n-j+1-k}{p-n-k}.$$

The right hand side can be considered as a degree $m-1$ polynomial in the variable j. The left hand side is a sum of a_i times a polynomial of degree $i-1$ in the variable j. Hence, there is a value a_m such that the coefficients of j^{m-1} on both sides coincide. If we set the value of a_m, we can then determine the value of a_{m-1} such that the coefficients of j^{m-2} coincide, and so on. In this way we can inductively choose the a_i's such that (C.19) holds. □

We remark that the condition that the codes are nested is essential in the proof since otherwise the linear combination in (C.18) is not valid.

In summary, we have shown in this section that the squares of some of the most well-known matrix-product codes are again matrix-product codes of a form simple enough that we can relate their minimum distance to that of the squares and products of the constituent codes (in some cases we need those constituent codes to be nested). Furthermore, the strategies used in the proofs from the different constructions are similar, which also suggests that the same techniques may be used for showing similar results in the case of other families of matrix-product codes.

A Products and Sums of Cyclic Codes

It is possible to describe a cyclic code as a subfield subcode of an evaluation code. For a set $M \subseteq \{1, \ldots, n-1\}$ and the extension field $\mathfrak{F} = \mathbb{F}_q(\odot)$ denote

by

$$\mathcal{P}(M) = \left\{ \sum_{i \in M} a_i X^i \mid a_i \in \mathfrak{F} \right\}.$$

Furthermore, let $\mathrm{ev}_\beta(f) = (f(1), f(\beta), \ldots, f(\beta^{n-1}))$ and define

$$\mathcal{B}(M) = \{\mathrm{ev}_\beta(f) \mid f \in \mathcal{P}(M)\} \subseteq \mathfrak{F}^n.$$

This is a linear code over $\mathfrak{F}$. We can obtain a linear code over $\mathbb{F}_q$ by taking the subfield subcode $\mathcal{B}(M) \cap \mathbb{F}_q^n$. Letting $M = -I = \{-i \bmod n \mid i \in I\}$, we obtain the cyclic code with generating set I, i.e.

$$C(I) = \mathcal{B}(-I) \cap \mathbb{F}_q^n,$$

see for instance Lemma 2.22 in [3]. We generalize Theorem 3.3 in [3], stating that $C(I)^{*2} = C(I + I)$, to $C(I_1) * C(I_2) = C(I_1 + I_2)$ below. The proofs are almost identical to the ones in [3].

Lemma A.1. *Let I_1 and I_2 be cyclotomic cosets. Then*

$$\mathcal{B}(-I_1) * \mathcal{B}(-I_2) = \mathcal{B}(-(I_1 + I_2)),$$

where $I_1 + I_2 = \{a + b \bmod n \mid a \in I_1, b \in I_2\}$.

Proof. We start by proving the inclusion $\mathcal{B}(-I_1) * \mathcal{B}(-I_2) \subseteq \mathcal{B}(-(I_1 + I_2))$. Let $\mathbf{v} \in \mathcal{B}(-I_1) * \mathcal{B}(-I_2)$. Then for $f_1 \in \mathcal{P}(-I_1)$ and $f_2 \in \mathcal{P}(-I_2)$, we have

$$\mathbf{v} = \mathrm{ev}_\beta(f_1) * \mathrm{ev}_\beta(f_2) = \mathrm{ev}_\beta(f_1 f_2).$$

Since $f_1 f_2 \in \mathcal{P}(-(I_1 + I_2))$ the inclusion follows.

For the other inclusion, let $\mathbf{w} \in \mathcal{B}(-(I_1 + I_2))$. Then for

$$f = \sum_{i \in -(I_1 + I_2)} a_i X^i \in \mathcal{P}(-(I_1 + I_2)),$$

where $a_i \in \mathfrak{F}$, we have

$$\mathbf{w} = \mathrm{ev}_\beta(f) = \sum_{i \in -(I_1 + I_2)} a_i \mathrm{ev}_\beta(X^i).$$

Since $i \in -(I_1 + I_2)$, there exists $j \in -I_1$ and $k \in -I_2$, such that $X^i = X^j X^k$, and therefore $\mathrm{ev}_\beta(X^i) = \mathrm{ev}_\beta(X^j) * \mathrm{ev}_\beta(X^k)$. Hence, $\mathbf{w}$ is an $\mathfrak{F}$-linear combination of elements in $\mathcal{B}(-I_1) * \mathcal{B}(-I_2)$ showing that $\mathbf{w} \in \mathcal{B}(-I_1) * \mathcal{B}(-I_2)$. □

With this lemma, we are able to proof Proposition 4.3.

Proposition 4.3
Let I_1 and I_2 be unions of q-cyclotomic cosets, and let $C(I_1)$ and $C(I_2)$ be the corresponding cyclic codes. Then

$$C(I_1) + C(I_2) = C(I_1 \cup I_2),$$
$$C(I_1) * C(I_2) = C(I_1 + I_2).$$

where $I_1 + I_2 = \{a + b \bmod n \mid a \in I_1, b \in I_2\}$. ◀

Proof. Let $\tilde{C}$ be the smallest linear code containing $C(I_1)$ and $C(I_2)$. Every codeword $\mathbf{c} \in \tilde{C}$ must be of the form $a_1\mathbf{c}_2 + a_2\mathbf{c}_2$, where $a_1, a_2 \in \mathbb{F}_q$ and $\mathbf{c}_i \in C(I_i)$. Now, let T be the function that do a cyclic shift. Then

$$T(\mathbf{c}) = T(a_1\mathbf{c}_1 + a_2\mathbf{c}_2) = a_1T(\mathbf{c}_1) + a_2T(\mathbf{c}_2),$$

and since $T(\mathbf{c}_i) \in C(I_i)$, we also have $\mathbf{c} \in \tilde{C}$. Therefore, $\tilde{C}$ is cyclic. For $C(I_i)$ to be included in the cyclic code $\tilde{C}$ the generating polynomial g for $\tilde{C}$ must divide g_i, the generating polynomial for $C(I_i)$. The smallest code, which means the polynomial with highest degree, satisfying this is $g = \gcd(g_1, g_2)$. This implies that $\tilde{C} = C(I_1 \cup I_2)$.

To show the second equality, we start by noticing that $\mathcal{B}(-I_i) = C(I_i)_{\mathfrak{F}}$, i.e. extending the code $C(I_i)$ to scalars over $\mathfrak{F}$ gives back $\mathcal{B}(-I_i)$. This is shown in [3]. Additionally, we use Lemma 2.23(iii) in [26] implying that $C(I_1)_{\mathfrak{F}} * C(I_2)_{\mathfrak{F}} = (C(I_1) * C(I_2))_{\mathfrak{F}}$. Putting these two statements together we obtain

$$\begin{aligned} C(I_1) * C(I_2) &= (C(I_1) * C(I_2))_{\mathfrak{F}} \cap \mathbb{F}_q^n = (C(I_1)_{\mathfrak{F}} * C(I_2)_{\mathfrak{F}}) \cap \mathbb{F}_q^n \\ &= (\mathcal{B}(-I_1) * \mathcal{B}(-I_2)) \cap \mathbb{F}_q^n = \mathcal{B}(-(I_1 + I_2)) \cap \mathbb{F}_q^n \\ &= C(I_1 + I_2), \end{aligned}$$

where we have used Lemma A.1 in the second to last step. □

References

[1] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. ACM, 1988, pp. 1–10.

[2] T. Blackmore and G. H. Norton, "Matrix-product codes over fq," *Applicable Algebra in Engineering, Communication and Computing*, vol. 12, no. 6, pp. 477–500, 12 2001.

[3] I. Cascudo, "On squares of cyclic codes," *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 1034–1047, 02 2019.

[4] I. Cascudo, R. Cramer, D. Mirandola, and G. Zémor, "Squares of random linear codes," *IEEE Transactions on Information Theory*, vol. 61, pp. 1159 – 1173, 03 2015.

[5] I. Cascudo, I. B. Damgård, B. M. David, N. Döttling, R. Dowsley, and I. Giacomelli, "Efficient uc commitment extension with homomorphism for free (and applications)," *IACR Cryptology ePrint Archive*, vol. 2018, p. 983, 2018.

[6] D. Chaum, C. Crépeau, and I. B. Damgård, "Multiparty unconditionally secure protocols," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. ACM, 1988, pp. 11–19.

[7] A. Couvreur, I. Márquez-Corbella, and R. Pellikaan, "Cryptanalysis of mceliece cryptosystem based on algebraic geometry codes and their subcodes," *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 5404–5418, Aug 2017.

[8] A. Couvreur, A. Otmani, and J.-P. Tillich, "Polynomial time attack on wild mceliece over quadratic extensions," *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 404–427, 01 2017.

[9] R. Cramer, I. B. Damgård, and U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," in *Advances in Cryptology — EUROCRYPT 2000*. Springer Berlin Heidelberg, 2000, pp. 316–334.

[10] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[11] I. B. Damgård, J. B. Nielsen, M. Nielsen, and S. Ranellucci, "The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited," in *Advances in Cryptology – CRYPTO 2017*. Cham: Springer International Publishing, 2017, pp. 167–187.

[12] I. B. Damgård and S. Zakarias, "Constant-overhead secure computation of boolean circuits using preprocessing," in *Theory of Cryptography*. Berlin, Heidelberg: Springer, 2013, pp. 621–641.

[13] I. M. Duursma and R. Kötter, "Error-locating pairs for cyclic codes," *IEEE Transactions on Information Theory*, vol. 40, pp. 1108 – 1121, 08 1994.

[14] T. K. Frederiksen, B. Pinkas, and A. Yanai, "Committed mpc," in *Public-Key Cryptography – PKC 2018*. Springer International Publishing, 2018, pp. 587–619.

[15] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, and D. A. Karpuk, "Private information retrieval from coded databases with colluding servers," *SIAM Journal on Applied Algebra and Geometry*, vol. 1, no. 1, pp. 647–664, 2017.

[16] F. Hernando, K. Lally, and D. Ruano, "Construction and decoding of matrix-product codes from nested codes," *Applicable Algebra in Engineering, Communication and Computing*, vol. 20, no. 5, pp. 497–507, 10 2009.

[17] F. Hernando and D. Ruano, "New linear codes from matrix-product codes with polynomial units," *Advances in Mathematics of Communications*, vol. 4, p. 363, 2009.

[18] K. Lally and P. Fitzpatrick, "Algebraic structure of quasicyclic codes," *Discrete Applied Mathematics*, vol. 111, no. 1, pp. 157 – 175, 2001, coding and Cryptology.

[19] L. Minder and A. Shokrollahi, "Cryptanalysis of the sidelnikov cryptosystem," in *Advances in Cryptology - EUROCRYPT 2007*. Springer Berlin Heidelberg, 2007, pp. 347–360.

[20] D. Mirandola and G. Zémor, "Critical pairs for the product singleton bound," *IEEE Transactions on Information Theory*, vol. 61, pp. 4928 – 4937, 06 2015.

[21] F. Özbudak and H. Stichtenoth, "Note on niederreiter-xing's propagation rule for linear codes," *Applicable Algebra in Engineering, Communication and Computing*, vol. 13, no. 1, pp. 53–56, 04 2002.

[22] R. Pellikaan, "On decoding by error location and dependent sets of error positions," *Discrete Mathematics*, vol. 106-107, pp. 369 – 381, 1992.

[23] R. Pellikaan and I. Márquez-Corbella, "Error-correcting pairs for a public-key cryptosystem," *Journal of Physics: Conference Series*, vol. 855, 06 2017.

[24] H. Randriambololona, "Asymptotically good binary linear codes with asymptotically good self-intersection spans," *IEEE Transactions on Information Theory*, vol. 59, pp. 3038 – 3045, 05 2013.

[25] H. Randriambololona, "An upper bound of singleton type for componentwise products of linear codes," *IEEE Transactions on Information Theory*, vol. 59, pp. 7936 – 7939, 09 2013.

[26] H. Randriambololona, "On products and powers of linear codes under componentwise multiplication," *Contemporary Math.*, vol. 637, 04 2015.

[27] C. Wieschebrink, "Cryptanalysis of the niederreiter public key scheme based on grs subcodes," in *Post-Quantum Cryptography*. Springer Berlin Heidelberg, 2010, pp. 61–72.

[28] K. Yang and P. V. Kumar, "On the true minimum distance of hermitian codes," in *Coding Theory and Algebraic Geometry*. Springer Berlin Heidelberg, 1992, pp. 99–107.

Paper D

A Secret-Sharing Based MPC Protocol for Boolean Circuits with Good Amortized Complexity

Ignacio Cascudo and Jaron Skovsted Gundersen

The paper will be presented at the 18^{th} *Theory of Cryptography Conference 2020* and will be published in
Lecture Notes in Computer Science.

The layout has been revised.

Abstract

We present a new secure multiparty computation protocol in the preprocessing model that allows for the evaluation of a number of instances of a boolean circuit in parallel, with a small online communication complexity per instance of 10 *bits per party and multiplication gate. Our protocol is secure against an active dishonest majority, and can also be transformed, via existing techniques, into a protocol for the evaluation of a single "well-formed" boolean circuit with the same complexity per multiplication gate at the cost of some overhead that depends on the topology of the circuit.*

Our protocol uses an approach introduced recently in the setting of honest majority and information-theoretical security which, using an algebraic notion called reverse multiplication friendly embeddings, essentially transforms a batch of evaluations of an arithmetic circuit over a small field into one evaluation of another arithmetic circuit over a larger field. To obtain security against a dishonest majority we combine this approach with the well-known SPDZ protocol that operates over a large field. Structurally our protocol is most similar to MiniMAC, a protocol which bases its security on the use of error-correcting codes, but our protocol has a communication complexity which is half of that of MiniMAC when the best available binary codes are used. With respect to certain variant of MiniMAC that utilizes codes over larger fields, our communication complexity is slightly worse; however, that variant of MiniMAC needs a much larger preprocessing than ours. We also show that our protocol also has smaller amortized communication complexity than Committed MPC, a protocol for general fields based on homomorphic commitments, if we use the best available constructions for those commitments. Finally, we construct a preprocessing phase from oblivious transfer based on ideas from MASCOT and Committed MPC.

1 Introduction

The area of secure multiparty computation (MPC) studies how to design protocols that allow for a number of parties to jointly perform computations on private inputs in such a way that each party learns a private output, but nothing else than that. In the last decade efficient MPC protocols have been developed that can be used in practical applications.

In this work we focus on secret-sharing based MPC protocols, which are among the most used in practice. In secret-sharing based MPC, the target computation is represented as an arithmetic circuit consisting of sum and multiplication gates over some algebraic ring; each party initially shares her input among the set of parties, and the protocol proceeds gate by gate, where at every gate a sharing of the output of the gate is created; in this manner eventually parties obtain shares of the output of the computation, which can then be reconstructed.

A common practice is to use the preprocessing model, where the computation is divided in two stages: a preprocessing phase, that is completely independent from the inputs and whose purpose is to distribute some correlated randomness among the parties; and an online phase, where the actual computation is performed with the help of the preprocessing data. This approach allows for pushing much of the complexity of the protocol into the preprocessing phase and having very efficient online computations in return.

Some secret sharing based MPC protocols obtain security against any static adversary which actively corrupts all but one of the parties in the computation, assuming that the adversary is computationally bounded. Since in the active setting corrupted parties can arbitrarily deviate from the protocol, some kind of mechanism is needed to detect such malicious behaviour, and one possibility is the use of information-theoretic MACs to authenticate the secret shared data, which is used in protocols such as BeDOZa [3] and SPDZ [14].

In SPDZ this works as follows: the computation to be performed is given by an arithmetic circuit over a large finite field $\mathbb{F}$. There is a global key $\alpha \in \mathbb{F}$ which is secret shared among the parties. Then for every value $x \in \mathbb{F}$ in the computation, parties obtain not only additive shares for that value, but also for the product $\alpha \cdot x$ which acts as a MAC for x. The idea is that if a set of corrupt parties change their shares and pretend that this value is $x + e$, for some nonzero error e, then they would also need to guess the correction value $\alpha \cdot e$ for the MAC, which amounts to guessing α since $\mathbb{F}$ is a field. In turn this happens with probability $1/|\mathbb{F}|$ which is small when the field is large.

The problem is that over small fields the cheating success probability $1/|\mathbb{F}|$ is large. While one can take a large enough extension field $\mathbb{L}$ of $\mathbb{F}$ (e.g. if $\mathbb{F} = \mathbb{F}_2$, then $\mathbb{L}$ could be the field of 2^s elements) and embed the whole computation into $\mathbb{L}$, this looks wasteful as communication is blown up by a factor of s.

An alternative was proposed in MiniMAC [15]. MiniMAC uses a batch authentication idea: if we are willing to simultaneously compute k instances of the same arithmetic circuit over a small field at once, we can bundle these computations together and see them as a computation of an arithmetic circuit over the ring $\mathbb{F}^k$, where the sum and multiplication operations are considered coordinatewise. Note the same authentication technique as in SPDZ does not directly work over this ring (if $|\mathbb{F}|$ is small): if we define the MAC of a data vector $\mathbf{x}$ in $\mathbb{F}^k$ to be $\boldsymbol{\alpha} * \mathbf{x}$ where the key $\boldsymbol{\alpha}$ is now also a vector in $\mathbb{F}^k$ and $*$ is the coordinatewise product, the adversary can introduce an error in a single coordinate with probability $1/|\mathbb{F}|$. Instead, MiniMAC first encodes every vector $\mathbf{x}$ as a larger vector $C(\mathbf{x})$ by means of a linear error-correcting code C with large minimum distance d, and then defines the MAC as $\boldsymbol{\alpha} * C(\mathbf{x})$. Now introducing an error requires to change at least d coordinates of $C(\mathbf{x})$ and the MAC can be fooled with probability only $1/|\mathbb{F}|^d$. However, when processing

multiplication gates, the minimum distance d^* of the so-called Schur square code C^* also needs to be large. These requirements on the minimum distance of these two codes have an effect on the communication overhead of the protocol, because the larger d and d^* are, the worse the relation between the length of messages and the length of the encoding.

This same article shows how to adapt this technique for computing a *single* boolean "well-formed" circuit while retaining the efficiency advantages of the batch simultaneous computation of k circuits. The idea is that if the target boolean circuit is structured into layers of addition and multiplication gates, where each layer has a large number of gates and its inputs are outputs of previous layers, then we can organize them into blocks of k gates of the same type, which can be computed using the above method. We then need an additional step that directs each block of outputs of a layer into the right block of inputs of next layers; this uses some additional preprocessed random sharings, and some openings, which slightly increases the communication complexity of the protocol.

In this paper, we explore an alternative to the error-correcting codes approach from MiniMAC, using an idea recently introduced in the honest majority, information-theoretically secure setting [8]. The point is that we can embed the ring $\mathbb{F}_q^k$ in some extension field of $\mathbb{F}_q$ in such a way that we can make the operations of both algebraic structures, and in particular the products (in one case the coordinatewise product, in the other the product in the extension field), "somewhat compatible": i.e., we map $\mathbb{F}_q^k$ into a slightly larger field $\mathbb{F}_{q^m}$ with some dedicated linear "embedding" map ϕ, that satisfies that for any two vectors $\mathbf{x}, \mathbf{y}$ in $\mathbb{F}_q^k$ the field product $\phi(\mathbf{x}) \cdot \phi(\mathbf{y})$ contains all information about $\mathbf{x} * \mathbf{y}$, in fact there exists a "recovery" linear map ψ such that $\mathbf{x} * \mathbf{y} = \psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$. The pair (ϕ, ψ) is called a (k, m)-reverse multiplication friendly embedding (RMFE) and was introduced in [5, 8]. With such tool, [8] embeds k evaluations of a circuit over $\mathbb{F}_q$ (i.e. an evaluation of an arithmetic circuit over $\mathbb{F}_q^k$ with coordinatewise operations) into one evaluation of a related circuit over $\mathbb{F}_{q^m}$, which is securely computed via an information-theoretically secure MPC protocol for arithmetic circuits over that larger field (more precisely the Beerliova-Hirt protocol [2]). The use of that MPC protocol over $\mathbb{F}_{q^m}$ is not black-box, however, as there are a number of modifications that need to be done at multiplication and input gates, for which certain additional correlated information has to be created in the preprocessing phase. Note that the reason for introducing this technique was that Beerliova-Hirt uses Shamir secret sharing schemes and hyperinvertible matrices, two tools that are only available over large finite fields (larger than the number of parties in the protocol).

1.1 Our Contributions

In this paper we construct a new secure computation protocol in the dishonest majority setting that allows to compute several instances of a boolean circuit at an amortized cost.[1] We do this by combining the embedding techniques from [8] with the SPDZ methodology. As opposed to [8], where one of the points of the embedding was precisely to use Shamir secret sharing, in our construction vectors $\mathbf{x} \in \mathbb{F}_2^k$ are still additively shared in $\mathbb{F}_2^k$, and it is only the MACs which are constructed and shared in the field $\mathbb{F}_{2^m}$: the MAC of $\mathbf{x}$ will be $\alpha \cdot \phi(\mathbf{x})$ where ϕ is the embedding map from the RMFE. Only when processing a multiplication gate, authenticated sharings where the data are shared as elements in $\mathbb{F}_{2^m}$ are temporarily used. MACs are checked in a batched fashion at the output gate, at which point the protocol aborts if discrepancies are found.

By this method we obtain a very efficient online phase where processing multiplication gates need each party to communicate around 10 bits[2] per evaluation of the circuit, for statistical security parameters like $s = 64, 128$ (meaning the adversary can successfully cheat with probability at most 2^{-s}, for which in our protocols we need to set $m \geq s$).

Our protocol can also be adapted to evaluating a single instance of a boolean circuit by quite directly adapting the ideas in MiniMAC that we mentioned above, based on organizing the circuit in layers, partitioning the layers in blocks of gates and adding some preprocessing that allows to map each block into the appropriate one in the next layer. The reason is that the maps used between layers of gates are $\mathbb{F}_2$-linear, and essentially all we need to use is the $\mathbb{F}_2$-linearity of the map ϕ from the RMFE. The actual complexity added by this transformation is quite dependent on the topology of the circuit. Under some general assumptions one can expect to add 2 bits of communication per gate.

Our online phase follows a similar pattern to MiniMAC in the sense that, up to the output phase, every partial opening of a value in $\mathbb{F}_2^k$ takes place when a partial opening of a C-encoding occurs in MiniMAC. Respectively, we need to open values in $\mathbb{F}_{2^m}$ whenever MiniMAC opens C^*-encodings. At every multiplication gate, both protocols need to apply "re-encoding functions" to convert encodings back to the base authentication scheme, which requires a preprocessed pair of authenticated sharings of random correlated elements.

However, the encoding via RMFE we are using is more compact than the one in MiniMAC; the comparison boils down to comparing the "expansion

[1] Our ideas can be extended to arithmetic circuits over other small fields.

[2] Here we assume that broadcasting messages of M bits requires to send M bits to every other player, which one can achieve with small overhead that vanishes for large messages [14, full version]

factor" m/k of RMFEs with the ratio k^*/k between the dimensions of C^* and C for the best binary codes with good distances of C^* [7]. We cut the communication cost of multiplication gates by about half with respect to MiniMAC where those binary codes are used. We achieve even better savings in the case of the output gates since in this case MiniMAC needs to communicate full vectors of the same length as the code, while the input and addition gates have the same cost.

We also compare the results with a modified version of MiniMAC proposed by Damgård, Lauritsen and Toft [13], that allows to save communication cost of multiplication gates, by essentially using MiniMAC over the field of 256 elements, at the cost of a much larger amount of preprocessing that essentially provides authenticated sharings of bit decompositions of the $\mathbb{F}_{256}$-coordinates of the elements in a triple, so that parties can compute bitwise operations. This version achieves a communication complexity that is around 80% of that of our protocol, due to the fact that this construction can make use of Reed-Solomon codes. However, it requires to have created authenticated sharings of 19 elements, while ours need 5 and as far as we know there is no explicit preprocessing protocol that has been proposed for this version of MiniMAC.

Finally we compare the results with Committed MPC [16], a secret-sharing based protocol which uses (UC-secure) homomorphic commitments for authentication, rather than information-theoretical MACs. In particular, this protocol can also be used for boolean circuits, given that efficient constructions of homomorphic commitments [9, 10, 17] over $\mathbb{F}_2$ have been proposed. These constructions of homomorphic commitments also use error-correcting codes. We find that, again, the smaller expansion m/k of RMFE compared to the relations between the parameters for binary error-correcting codes provides an improvement in the communication complexity of a factor ~ 3 for security parameters $s = 64, 128$.

We also provide a preprocessing phase producing all authenticated sharings of random correlated data that we need. The preprocessing follows the steps of MASCOT [19] (see also [18]) based on OT extension, with some modifications due to the slightly different authentication mechanisms we have and the different format of our preprocessing. All these modifications are easily to carry out based on the fact that ϕ and ψ are linear maps over $\mathbb{F}_2$. Nevertheless, using the "triple sacrificing steps" from MASCOT that assure that preprocessed triples are not malformed presents problems in our case for technical reasons. Instead, we use the techniques from Committed MPC [16] in that part of the triple generation.

1.2 Related Work

The use of information-theoretical MACs in secret-sharing based multiparty computation dates back to BeDOZa (Bendlin et al., [3]), where such MACs where established between every pair of players. Later SPDZ (Damgård et al., [14]) introduced the strategy consisting of a global MAC for every element of which every party has a share, and whose key is likewise shared among parties. Tiny OT (Nielsen et al., [21]), a 2-party protocol for binary circuits, introduced the idea of using OT extension in the preprocessing phase. Larraia et al., [20] extended these ideas to a multi-party protocol by using the SPDZ global shared MAC approach. MiniMAC (Damgård and Zakarias, [15]), as explained above, used error-correcting codes in order to authenticate vectors of bits, allowing for efficient parallel computation of several evaluations of the same binary circuits on possibly different inputs. Damgård et al., [13] proposed several improvements for the implementation of MiniMAC, among them the use of an error correcting code over an extension field, trading smaller communication complexity for a larger amount of preprocessing. Frederiksen et al., [18] gave new protocols for the construction of preprocessed multiplication triples in fields of characteristic two, based on OT extension, and in particular provided the first preprocessing phase for MiniMAC. MASCOT (Keller et al., [19]) built on some of these ideas to create preprocessing protocols for SPDZ based on OT extension. Committed MPC (Frederiksen et al., [16]) is a secret-sharing based secure computation protocol that relies on UC-secure homomorphic commitments instead of homomorphic MACs for authentication, but other than that, it follows a similar pattern to the protocols above. Efficient constructions of UC-secure homomorphic commitments from OT have been proposed by Frederiksen et al., [17] and Cascudo et al., [10] based on error correcting codes. Later, in [9] a modified construction from extractable commitments, still using error-correcting codes, was proposed that presents an important advantage for its use in Committed MPC, namely the commitment schemes are multi-verifier.

The notion of reverse multiplication friendly embedding was first explicitly defined and studied in the context of secure computation by Cascudo et al. in [8] and independently by Block et al. in [5]. The former work is in the context of information-theoretically secure protocols, while the latter studied 2-party protocols over small fields where the assumed resource is OLE over an extension field. This is partially based on a previous work also by Block et al., [4] where (asymptotically less efficient) constructions of RMFEs were implicitly used.

2 Preliminaries

Let $\mathbb{F}_q$ denote a finite fields with q elements. Vectors are denoted with bold letters as $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and componentwise products of two vectors are denoted by $\mathbf{x} * \mathbf{y} = (x_1 \cdot y_1, x_2 \cdot y_2, \ldots, x_n \cdot y_n)$. Fixing an irreducible polynomial f of degree m in $\mathbb{F}_q[X]$, elements in the field $\mathbb{F}_{q^m}$ with q^m elements can be represented as polynomials in $\mathbb{F}_q[X]$ with degree $m-1$, i.e $\alpha = \alpha_0 + \alpha_1 \cdot X + \cdots + \alpha_{m-1} \cdot X^{m-1} \in \mathbb{F}_{q^m}$, where $\alpha_i \in \mathbb{F}_q$. The sums and products of elements are defined modulo f.

In our protocols we will assume a network of n parties who communicate by secure point-to-point channels, and an static adversary who can actively corrupt up to $n-1$ of these parties. Our proofs will be in the universal composable security model [6].

We recall the notion of reverse multiplication friendly embeddings from [8].

Definition 2.1. *Let $k, m \in \mathbb{Z}^+$. A pair of $\mathbb{F}_q$-linear maps (ϕ, ψ), where $\phi \colon \mathbb{F}_q^k \to \mathbb{F}_{q^m}$ and $\psi \colon \mathbb{F}_{q^m} \to \mathbb{F}_q^k$ is called a $(k, m)_q$-reverse multiplication friendly embedding (RMFE) if for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^k$*

$$\mathbf{x} * \mathbf{y} = \psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$$

In other words, this tool allows to multiply coordinatewise two vectors over $\mathbb{F}_q$ by first embedding them in a larger field with ϕ, multiplying the resulting images and mapping the result back to a vector over $\mathbb{F}_q$ with the other map ψ.

Several results about the existence of such pairs can be found in [8], both in the asymptotic and concrete settings. For our results we will only need the following construction, which can be obtained via simple interpolation techniques:

Theorem 2.2 ([8]). *For all $r \leq 33$, there exists a $(3r, 10r-5)_2$ -RMFE.*

However, we remark that for implementations, it might be more useful to consider the following constructions of RMFEs which can also be deduced from the general framework in [8] (also based on polynomial interpolation). They have worse rate k/m than those in Theorem 2.2, but they have the advantage that their image can be in a field of degree a power of two, e.g. $\mathbb{F}_{q^m} = \mathbb{F}_{2^{64}}$ or $\mathbb{F}_{2^{128}}$.

Theorem 2.3. *For any $r \leq 16$, there exists a $(2r, 8r)_2$-RMFE.*[3]

[3]Specifically the result is obtained by noticing that the proof of Lemma 4 in [8] can also be used to show the existence of $(k, 2k)_q$-RMFE for any $q \leq k+1$, and then composing $(2,4)_2$ and $(r, 2r)_{16}$-RMFEs in the manner of Lemma 5 in the same paper.

For our numerical comparisons we will mainly consider the constructions with better rate in Theorem 2.2 and point out that, should one want to use Theorem 2.3 instead, then some small overhead in communication is introduced.

It is important to understand some properties and limitations of the RMFEs. Because ϕ and ψ are $\mathbb{F}_q$-linear then

$$\phi(\mathbf{x}+\mathbf{y}) = \phi(\mathbf{x}) + \phi(\mathbf{y}), \quad \psi(x+y) = \psi(x) + \psi(y)$$

holds for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^k$ and $x, y \in \mathbb{F}_{q^m}$. However, for example

$$\phi(\mathbf{x} * \mathbf{y}) \neq \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$$

in general. Likewise we will need to take into account that the composition $\phi \circ \psi : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}$ is a linear map over $\mathbb{F}_q$ but *not* over $\mathbb{F}_{q^m}$. Therefore

$$(\phi \circ \psi)(x+y) = (\phi \circ \psi)(x) + (\phi \circ \psi)(y) \text{ for all } x, y \in \mathbb{F}_{q^m}, \text{ but}$$
$$(\phi \circ \psi)(\alpha \cdot x) \neq \alpha \cdot (\phi \circ \psi)(x)$$

for $\alpha, x \in \mathbb{F}_{q^m}$ in general (it does hold when $\alpha \in \mathbb{F}_q$, but this is not too relevant).

These limitations on the algebra of ϕ and ψ posed certain obstacles in the information-theoretical setting [8], since processing multiplication gates required to compute gates given by the map $\phi \circ \psi$, and this cannot be treated as a simple linear gate over $\mathbb{F}_{q^m}$. The additivity of $\phi \circ \psi$ combined with certain involved preprocessing techniques saved the day there. For completion (and comparison to our paper) we sum up some of the main details of [8] in the full version of this paper [11]. In our case, we will again encounter problems caused by these limitations as we explain next, but can solve them in a different way.

3 The Online Phase

In this section we present our protocol for computing simultaneously k instances of a boolean circuit in parallel, which we can see as computing one instance of an arithmetic circuit over the ring $\mathbb{F}_2^k$ of length k boolean vectors with coordinatewise sum and product.

Our strategy is to have mixed authenticated sharings: inputs and the rest of values in the computation $\mathbf{x}$ are additively shared as vectors over $\mathbb{F}_2^k$ (we refer to this as data shares), but their MACs are elements $\alpha \cdot \phi(\mathbf{x})$ in the larger field $\mathbb{F}_{2^m}$, where $\alpha \in \mathbb{F}_{2^m}$ is (as in SPDZ) a global key that is additively shared among the parties from the beginning (with $\alpha^{(i)}$ denoting the share for party

P_i), and parties hold additive shares of $\alpha \cdot \phi(\mathbf{x})$ also in the field $\mathbb{F}_{2^m}$ (the MAC shares). We will denote the authentication of $\mathbf{x}$ by $\langle \mathbf{x} \rangle$. That is

$$\langle \mathbf{x} \rangle = \Big((\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}), (m^{(1)}(\mathbf{x}), m^{(2)}(\mathbf{x}), \ldots, m^{(n)}(\mathbf{x})) \Big)$$

where each party P_i holds an additive share $\mathbf{x}^{(i)} \in \mathbb{F}_2^k$ and a MAC share $m^{(i)}(\mathbf{x}) \in \mathbb{F}_{2^m}$, such that $\sum_{i=1}^n m^{(i)}(\mathbf{x}) = \alpha \cdot \sum_{i=1}^n \phi(\mathbf{x}^{(i)}) = \alpha \cdot \phi(\mathbf{x})$.

The additivity of ϕ guarantees that additions can still be computed locally, and we can define $\langle \mathbf{x} \rangle + \langle \mathbf{y} \rangle = \langle \mathbf{x} + \mathbf{y} \rangle$ where every party just adds up their shares for both values. Moreover, given a public vector $\mathbf{a}$ and $\langle \mathbf{x} \rangle$, parties can also locally compute an authenticated sharing of $\mathbf{a} + \mathbf{x}$ as

$$\mathbf{a} + \langle \mathbf{x} \rangle = \Big((\mathbf{x}^{(1)} + \mathbf{a}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}), \\ (\alpha^{(1)} \cdot \phi(\mathbf{a}) + m^{(1)}(\mathbf{x}), \ldots, \alpha^{(n)} \cdot \phi(\mathbf{a}) + m^{(n)}(\mathbf{x})) \Big)$$

This allows to easily process addition with constants. Moreover, this also allows us to explain how inputs are shared in the first place. In the preprocessing phase parties have created for each input gate an authenticated random values $\langle \mathbf{r} \rangle$ where $\mathbf{r}$ is known to the party that will provide the input $\mathbf{x}$ at that gate. This party can just broadcast the difference $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{r}$, and then parties simply add $\boldsymbol{\epsilon} + \langle \mathbf{r} \rangle = \langle \mathbf{x} \rangle$ by the rule above.

As in SPDZ, parties in our protocol do not need to open any MAC until the output gate. At the output gate, the parties check MACs on random linear combinations of all values partially opened during the protocol, ensuring that parties have not cheated except with probability at most 2^{-m} (we need that $m \geq s$ if s is the statistical security parameter); then, they open the result of the computation and also check that the MAC of the result is correct.

A harder question, as usual, is how to process multiplication gates; given $\langle \mathbf{x} \rangle$, $\langle \mathbf{y} \rangle$ parties need to compute $\langle \mathbf{x} * \mathbf{y} \rangle$ which implies not only obtaining an additive sharing of $\mathbf{x} * \mathbf{y}$ but also of its MAC $\alpha \cdot \phi(\mathbf{x} * \mathbf{y})$. If we try to apply directly the well-known Beaver's technique [1] we encounter the following problem. Suppose we have obtained a random triple $\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{a} * \mathbf{b} \rangle$ from the preprocessing phase and, proceeding as usual, parties partially open the values $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{a}$, $\boldsymbol{\delta} = \mathbf{y} - \mathbf{b}$ (a partially opening is an opening of the shares but not the MAC shares). From here, computing data shares for $\mathbf{x} * \mathbf{y}$ is easy; however, the obstacle lies in computing shares of $\alpha \cdot \phi(\mathbf{x} * \mathbf{y})$. Indeed

$$\alpha \cdot \phi(\mathbf{x} * \mathbf{y}) = \alpha \cdot \phi(\mathbf{a} * \mathbf{b}) + \alpha \cdot \phi(\mathbf{a} * \boldsymbol{\delta}) + \alpha \cdot \phi(\boldsymbol{\epsilon} * \mathbf{b}) + \alpha \cdot \phi(\boldsymbol{\epsilon} * \boldsymbol{\delta}),$$

and the two terms in the middle present a problem: for example for $\alpha \cdot \phi(\mathbf{a} * \boldsymbol{\delta})$ we have by the properties of the RMFE

$$\alpha \cdot \phi(\mathbf{a} * \boldsymbol{\delta}) = \alpha \cdot \phi(\psi(\phi(\mathbf{a}) \cdot \phi(\boldsymbol{\delta}))) = \alpha \cdot (\phi \circ \psi)(\phi(\mathbf{a}) \cdot \phi(\boldsymbol{\delta}))$$

However, $\phi \circ \psi$ is only $\mathbb{F}_2$-linear, and not $\mathbb{F}_{2^m}$-linear, so we cannot just "take α inside the argument" and use the additive sharing of $\alpha \cdot \phi(\mathbf{a})$ given in $\langle \mathbf{a} \rangle$ to compute a sharing of the expression above. Instead, we use a two-step process to compute multiplication gates, for which we need to introduce regular SPDZ sharings on elements $x \in \mathbb{F}_{2^m}$. I.e. both x and its MAC $\alpha \cdot x$ are additively shared in $\mathbb{F}_{2^m}$. We denote these by $[x]$, that is

$$[x] = \Big((x^{(1)}, x^{(2)}, \ldots, x^{(n)}), (m^{(1)}(x), m^{(2)}(x), \ldots, m^{(n)}(x)) \Big),$$

where P_i will hold $x^{(i)}$ and $m^{(i)}(x) \in \mathbb{F}_{2^m}$ with $\sum_{i=1}^{n} m^{(i)}(x) = \alpha \cdot \sum_{i=1}^{n} x^{(i)}$.

To carry out the multiplication we need to preprocess a triple $(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{c} \rangle)$ where $\mathbf{c} = \mathbf{a} * \mathbf{b}$, and a pair of the form $(\langle \psi(r) \rangle, [r])$ where r is a random element in $\mathbb{F}_{2^m}$. In the first step of the multiplication we compute and partially open

$$[\sigma] = [\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) - \phi(\mathbf{a}) \cdot \phi(\mathbf{b}) - r]. \tag{D.1}$$

This can be computed from the $\boldsymbol{\epsilon}$ and δ described above (details will be given later). In the second step, we create $\langle \mathbf{x} * \mathbf{y} \rangle$ from (D.1) by using the properties of the RMFE; namely, $\mathbf{x} * \mathbf{y} = \psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$ and $\mathbf{a} * \mathbf{b} = \psi(\phi(\mathbf{a}) \cdot \phi(\mathbf{b}))$, so applying ψ on σ in (D.1) yields $\mathbf{x} * \mathbf{y} - \mathbf{a} * \mathbf{b} - \psi(r)$ because of the additivity of ψ. Adding $\langle \mathbf{a} * \mathbf{b} \rangle + \langle \psi(r) \rangle$ (the yet unused preprocessed elements) gives $\langle \mathbf{x} * \mathbf{y} \rangle$.

We still need to explain how to construct $[\sigma]$. For this we introduce some algebraic operations on the two types of authenticated sharings and public values. First given a public vector $\mathbf{a}$ and a shared vector $\mathbf{x}$ we define:

$$\mathbf{a} * \langle \mathbf{x} \rangle = \Big((\phi(\mathbf{a}) \cdot \phi(\mathbf{x}^{(1)}), \ldots, \phi(\mathbf{a}) \cdot \phi(\mathbf{x}^{(n)})), (\phi(\mathbf{a}) \cdot m^{(1)}(\mathbf{x}), \ldots, \phi(\mathbf{a}) \cdot m^{(n)}(\mathbf{x})) \Big)$$

Note that the data shares are shares of $\phi(\mathbf{a}) \cdot \phi(\mathbf{x})$, which is an element of $\mathbb{F}_{2^m}$, and the MAC shares also correspond to additive shares of $\alpha \cdot \phi(\mathbf{a}) \cdot \phi(\mathbf{x})$. However, the data shares are not distributed uniformly in $\mathbb{F}_{2^m}$ because ϕ is not surjective, so one cannot say this equals $[\phi(\mathbf{a}) \cdot \phi(\mathbf{x})]$. Nevertheless, given another $[z]$, with $z \in \mathbb{F}_{2^m}$, it is true that $\mathbf{a} * \langle \mathbf{x} \rangle + [z] = [\phi(\mathbf{a}) \cdot \phi(\mathbf{x}) + z]$ where the sum on the left is defined by just local addition of the data and MAC shares. We also define

$$\begin{aligned} \langle \mathbf{x} \rangle + [y] = \Big(& (\phi(\mathbf{x}^{(1)}) + y^{(1)}, \ldots, \phi(\mathbf{x}^{(n)}) + y^{(n)}), \\ & (m^{(1)}(\mathbf{x}) + m^{(1)}(y), \ldots, m^{(n)}(\mathbf{x}) + m^{(n)}(y)) \Big) = [\phi(\mathbf{x}) + y] \end{aligned}$$

Now, given $\langle \mathbf{x} \rangle, \langle \mathbf{y} \rangle$ and a triple $\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{a} * \mathbf{b} \rangle$, parties can open $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{a}$, $\boldsymbol{\delta} = \mathbf{y} - \mathbf{b}$ and construct

$$\begin{aligned} \boldsymbol{\epsilon} * \langle \mathbf{y} \rangle + \boldsymbol{\delta} * \langle \mathbf{x} \rangle - \phi(\boldsymbol{\epsilon}) \cdot \phi(\boldsymbol{\delta}) - [r] &= [\phi(\boldsymbol{\epsilon}) \cdot \phi(\mathbf{y}) + \phi(\boldsymbol{\delta}) \cdot \phi(\mathbf{x}) - \phi(\boldsymbol{\epsilon}) \cdot \phi(\boldsymbol{\delta}) - r] \\ &= [\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) - \phi(\mathbf{a}) \cdot \phi(\mathbf{b}) - r], \end{aligned}$$

where the latter equality can be seen by developing the expressions for $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}$, and using the additivity of ϕ. The obtained sharing is the $[\sigma]$ we needed above . Summing up, the whole multiplication gate costs 2 openings of sharings of vectors in $\mathbb{F}_2^k$ and one opening of a share of an element in $\mathbb{F}_{2^m}$. Every multiplication gate requires fresh preprocessed correlated authenticated sharings $(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{a} * \mathbf{b} \rangle)$ and $(\langle \psi(r) \rangle, [r])$ for random $\mathbf{a}, \mathbf{b}, r$.

We present formally the online protocol we just explained, the functionality it implements, and the functionalities needed from preprocessing. The functionality constructing the required preprocessed randomness is given in Figure D.2, and relies on the authentication functionality in Figure D.1. The latter augments the one in MASCOT [19] allowing to also authenticate vectors and to compute linear combinations involving the two different types of authenticated values and which can be realized by means of the $[\cdot]$- and $\langle \cdot \rangle$-sharings.

The functionality for our MPC protocol is in Figure D.3 and the protocol implementing the online phase is in Figure D.4.

Theorem 3.1. Π_{Online} *securely implements* $\mathcal{F}_{\text{MPC}}$ *in the* $\mathcal{F}_{\text{Prep}}$*-hybrid model.*

Proof. The correctness follows from the explanation above. For more details we refer to the full version, but we also note that the online phase from this protocol is similar to the online phases of protocols such as [14–16, 19], except that in every multiplication we additionally need to use the pair $(\langle \psi(r) \rangle, [r])$ in order to transform a $[\cdot]$-sharing into $\langle \mathbf{x} * \mathbf{y} \rangle$. However, since r is uniformly random in the field $\mathbb{F}_{2^m}$, the opened value σ masks any information on $\mathbf{x}$, $\mathbf{y}$.□

3.1 Comparison with MiniMAC and Committed MPC

We compare the communication complexity of our online phase with that of MiniMAC [15] and Committed MPC [16], two secret-sharing based MPC protocols which are well-suited for simultaneously evaluating k instances of the same boolean circuit. We will count broadcasting a message of M bits as communicating $M(n-1)$ bits (M bits to each other party). This can be achieved using point-to-point channels as described in the full version of [14].

Functionality $\mathcal{F}_{\text{Auth}}$

The functionality maintains two dictionaries Val and ValField, to keep track of authenticated values. We remark that we can store elements from $\mathbb{F}_2^k$ in Val and elements from $\mathbb{F}_{2^m}$ in ValField. Entries in the dictionaries cannot be changed.

1. **Input:** On input
$$(\text{Input}, (\text{id}_1, \text{id}_2, \ldots \text{id}_s), (\text{id}'_1, \text{id}'_2, \ldots \text{id}'_t), (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s), (x_1, x_2, \ldots, x_t), P_i)$$
from P_i and $(\text{Input}, (\text{id}_1, \text{id}_2, \ldots \text{id}_s), (\text{id}'_1, \text{id}'_2, \ldots \text{id}'_t), P_i)$ from all other parties, set $\text{Val}[\text{id}_j] = \mathbf{x}_j$ for $j = 1,2,\ldots,s$ and $\text{ValField}[\text{id}'_j] = x_j$ for $j = 1,2,\ldots,t$.
2. **Add:** On input $(\text{Add}, \bar{\text{id}}, \text{id}, a))$ from all parties. If a is an id store $\text{Val}[\bar{\text{id}}] = \text{Val}[\text{id}] + \text{Val}[a]$. If a is a vector in $\mathbb{F}_2^k$ store $\text{Val}[\bar{\text{id}}] = \text{Val}[\text{id}] + a$.
3. **LinComb:** On input
$$(\text{LinComb}, \bar{\text{id}}, (\text{id}_1, \text{id}_2, \ldots \text{id}_s), (\text{id}'_1, \text{id}'_2, \ldots \text{id}'_t), a_1, a_2, \ldots, a_{s+t}, a)$$
from all parties, where a_j is in $\mathbb{F}_{2^m}$ or $\mathbb{F}_2^k$ and $t \geq 1$. Define $\tilde{a}_j$ to be a_j if $a_j \in \mathbb{F}_{2^m}$, and $\phi(a_j)$ if $a_j \in \mathbb{F}_2^k$, and store $\text{ValField}[\bar{\text{id}}] = \sum_{j=1}^{s} \tilde{a}_j \cdot \phi(\text{Val}[\text{id}_j]) + \sum_{j=1}^{t} \tilde{a}_{s+j} \cdot \text{ValField}[\text{id}'_j] + \tilde{a}$.
4. **Open:** On input $(\text{Open}, \text{Dict}, \text{id}, S)$ from all parties, where S is a non-empty subset of parties. If $\text{Dict} = \text{Val}$ and $\text{Val}[\text{id}] \neq \perp$ wait for an $\mathbf{x}$ from the adversary and send $\mathbf{x}$ to the honest parties in S. If $\text{Dict} = \text{ValField}$ and $\text{ValField}[\text{id}] \neq \perp$ wait for an x from the adversary and send x to the parties in S.
5. **Check:** On input
$$(\text{Check}, (\text{id}_1, \text{id}_2, \ldots, \text{id}_s), (\text{id}'_1, \text{id}'_2, \ldots, \text{id}'_t), (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s), (x_1, x_2, \ldots, x_t))$$
from every party wait for an input from the adversary. If they input OK, $\text{Val}[\text{id}_j] = \mathbf{x}_j$ for $j = 1,2,\ldots,s$ and $\text{ValField}[\text{id}'_j] = x_j$ for $j = 1,2,\ldots,t$ return OK to all parties. Otherwise abort.

Notation: We will use the notation $\langle \mathbf{x} \rangle$ to refer to a value $\mathbf{x} \in \mathbb{F}_2^k$ stored in Val, and the notation $[x]$ to refer to a value $x \in \mathbb{F}_{2^m}$ stored in ValField.

Fig. D.1: Functionality – Authentication

Functionality $\mathcal{F}_{\text{Prep}}$

This functionality has the same features as $\mathcal{F}_{\text{Auth}}$ along with the following commands.

1. **InputPair:** On input $(\text{InputPair}, \text{id}, P_i)$ from all parties let P_i choose $\mathbf{r} \in \mathbb{F}_2^k$ at random and call $\mathcal{F}_{\text{Auth}}$ with input $(\text{Input}, \text{id}, \mathbf{r}, P_i)$ to obtain $\langle \mathbf{r} \rangle$. Output $\langle \mathbf{r} \rangle$ to all parties and $\mathbf{r}$ to P_i.
2. **ReEncodePair:** On input $(\text{ReEncodePair}, \text{id}_1, \text{id}_2)$ sample a random field element $r \in \mathbb{F}_{2^m}$ and set $\text{Val}[\text{id}_1] = \psi(r)$ and $\text{ValField}[\text{id}_2] = r$.
3. **Triple:** On input $(\text{Triple}, \text{id}_a, \text{id}_b, \text{id}_c)$ from all parties, sample two random vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^k$ and set $(\text{Val}[\text{id}_a], \text{Val}[\text{id}_b], \text{Val}[\text{id}_c]) = (\mathbf{a}, \mathbf{b}, \mathbf{a} * \mathbf{b})$.

Fig. D.2: Functionality – Preprocessing

Communication complexity of our protocol.

Partially opening a $\langle \cdot \rangle$-authenticated secret involves $2k(n-1)$ bits of communication, since we have one selected party receive the share of each other

Functionality $\mathcal{F}_{\text{MPC}}$

1. **Initialize:** On input Init from all players setup an empty dictionary Val.
2. **Input:** On input (Input,id,$\mathbf{x}$,P_i) from P_i and (Input,id,P_i) from all other parties where $\mathbf{x} \in \mathbb{F}_2^k$ and Val[id] $= \perp$ set Val[id] $= \mathbf{x}$.
3. **Add:** On input (Add,id_1,id_2,id_3) from all parties where Val[id_1] $\neq \perp$ and Val[id_2] $\neq \perp$, set Val[id_3] $=$ Val[id_1] $+$ Val[id_2].
4. **Multiply:** On input (Mult,id_1,id_2,id_3) from all parties where Val[id_1] $\neq \perp$ and Val[id_2] $\neq \perp$, set Val[id_3] $=$ Val[id_1] $*$ Val[id_2].
5. **Output:** On input (Output,id) from all parties when Val[id] $\neq \perp$ retrieve $z =$ Val[id] and send z to the adversary. Wait for an input from the adversary, if the adversary inputs OK send z to the honest parties. Otherwise abort.

Fig. D.3: Functionality – MPC

Protocol Π_{Online}

1. **Initialize:** The parties call the preprocessing functionality $\mathcal{F}_{\text{Prep}}$ to obtain input pairs $(\mathbf{r}, \langle \mathbf{r} \rangle)$ for each party, re-encode pairs $(\langle \psi(r) \rangle, [r])$, and multiplication triples $(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{c} \rangle)$.
2. **Input:** For an input gate for which P_i has input $\mathbf{x} \in \mathbb{F}_2^k$ the parties do the following
   (a) P_i takes a pair $(\mathbf{r}, \langle \mathbf{r} \rangle)$ and broadcasts $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{r}$.
   (b) The parties compute $\langle \mathbf{x} \rangle = \boldsymbol{\epsilon} + \langle \mathbf{r} \rangle$.
3. **Add:** To compute componentwise addition of $\langle \mathbf{x} \rangle$ and $\langle \mathbf{y} \rangle$ the parties locally compute $\langle \mathbf{x} + \mathbf{y} \rangle = \langle \mathbf{x} \rangle + \langle \mathbf{y} \rangle$.
4. **Multiply:** To compute a componentwise multiplication of $\langle \mathbf{x} \rangle$ and $\langle \mathbf{y} \rangle$, take the next available multiplication triple $(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{c} \rangle)$ and pair $(\langle \psi(r) \rangle, [r])$.
   (a) Set $\langle \boldsymbol{\epsilon} \rangle = \langle \mathbf{x} \rangle - \langle \mathbf{a} \rangle$ and $\langle \boldsymbol{\delta} \rangle = \langle \mathbf{y} \rangle - \langle \mathbf{b} \rangle$ and partially open $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}$.
   (b) Compute $[\sigma] = \boldsymbol{\epsilon} * \langle \mathbf{y} \rangle + \boldsymbol{\delta} * \langle \mathbf{x} \rangle - \phi(\boldsymbol{\epsilon}) \cdot \phi(\boldsymbol{\delta}) - [r] = [\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) - \phi(\mathbf{a}) \cdot \phi(\mathbf{b}) - r]$ and partially open σ.
   (c) Compute $\psi(\sigma) + \langle \mathbf{c} \rangle + \langle \psi(r) \rangle = \langle \mathbf{x} * \mathbf{y} \rangle$ and output this value.
5. **Output:** This stage is entered when the players have an unopened sharing $\langle \mathbf{z} \rangle$ which they want to output. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s$ be all opened $\langle \cdot \rangle$-sharings, i.e. $\mathbf{x}_j \in \mathbb{F}_2^k$ and let $x_1, x_2, \ldots, x_t$ be all opened $[\cdot]$-sharings, i.e. $x_j \in \mathbb{F}_{2^m}$. The parties do the following:
   (a) Call $\mathcal{F}_{\text{Auth.Check}}$ with inputs $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s)$ and $(x_1, x_2, \ldots, x_t)$.
   (b) If the check passes, partially open $\mathbf{z}$.
   (c) Call $\mathcal{F}_{\text{Auth.Check}}$ with input $\mathbf{z}$
   (d) If the check passes, output $\mathbf{z}$ to all parties.

Fig. D.4: Online phase

party and broadcast the reconstructed value. Likewise, partially opening a $[\cdot]$-authenticated value communicates $2m(n-1)$ bits. In our online phase,

every input gate requires $k(n-1)$ bits of communication. Multiplication gates require the partial opening of two $\langle\cdot\rangle$-authenticated values and one $[\cdot]$-authenticated value, hence $(4k+2m)(n-1)$ bits of communication. An output gate requires to do a MAC-check on (a linear combination of) previously partially opened values, then partially opening the output, and finally doing a MAC check on the output. A MAC check require every party to communicate a MAC share in $\mathbb{F}_{2^m}$, for a total of mn bits communicated. Hence output gates require $2k(n-1)+2mn$ bits of communication.

MiniMAC.

MiniMAC uses a linear error correcting code C with parameters $[\ell,k,d]$ (i.e., it allows for encoding of messages from $\mathbb{F}_2^k$ into $\mathbb{F}_2^\ell$ and has minimum distance d). Parties have additive shares of encodings $C(\mathbf{x})$, where the shares are also codewords, and shares of the MAC $\boldsymbol{\alpha} * C(\mathbf{x})$, which can be arbitrary vectors in $\mathbb{F}_2^\ell$. In addition, at multiplication gates C^*-encodings of information are needed, where C^* is the code $C^* = \text{span}\{\mathbf{x} * \mathbf{y} \mid \mathbf{x},\mathbf{y} \in C\}$, the smallest linear code containing the coordinatewise product of every pair of codewords in C^*, with parameters $[\ell,k^*,d^*]$. We always have $d \geq d^*$, and the cheating success probability of the adversary in the protocol is 2^{-d^*}, so we need $d^* \geq s$ for the statistical parameter s. The online phase of MiniMAC has a very similar communication pattern to ours: a multiplication requires to open two elements encoded with C (coming from the use of Beaver's technique) and one encoded with C^*. Since shares of C-(resp C^*-)encodings are codewords in C (resp C^*), and describing such codewords require k bits (resp. k^* bits)[4] the total communication complexity is $(4k+2k^*)(n-1)$, so the difference with our protocol depends on the difference between the achievable parameters for their k^* and our m, compared below. Input gates require $k(n-1)$ bits, as in our case, and for output gates, since MAC shares are arbitrary vectors in $\mathbb{F}_2^\ell$, a total of $2k(n-1)+2\ell n$ bits are sent. See full version for more details on this.

Committed MPC.

Committed MPC [16] is a secret-sharing based MPC protocol that relies on UC-secure additively homomorphic commitments for authentication, rather than on MACs. Efficient commitments of this type have been proposed in works such as [9, 10, 17] where the main ingredient[5] is again a linear error correcting code C with parameters $[\ell,k,d]$. In committed MPC, for every

[4] We observe that this is more lenient than the description of MiniMAC in [13, 15] where it is implied that ℓ bits need to be sent in order to do these openings.

[5] The constructions rely also on OT (in the first two cases) and extractable commitments (in the third) but these primitives are only used in a preprocessing phase.

$\mathbf{x} \in \mathbb{F}_2^k$, each party P_i holds an additive share $\mathbf{x}_i \in \mathbb{F}_2^k$ to which she commits towards every other party P_j (in the multi-receiver commitment from [9], this can be accomplished by only one commitment). During most of the online phase there are only partial openings of values and only at output gates the commitments are checked. Multiplication is done through Beaver's technique. In this case only two values $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}$ are partially opened. In exchange, parties need to communicate in order to compute commitments to $\boldsymbol{\delta} * \mathbf{a}$ (resp. $\boldsymbol{\epsilon} * \mathbf{b}$) given $\boldsymbol{\delta}$, and commitments to $\mathbf{a}$ (resp. $\boldsymbol{\epsilon}$ and commitments to $\mathbf{b}$) at least with current constructions for UC-secure homomorphic commitments. [16, full version, fig. 16] provides a protocol where each of these products with known constant vectors requires to communicate one full vector of length ℓ and two vectors of k^* components (again ℓ is the length of C and k^* is the dimension of C^*). In total the communication complexity of a multiplication is $(4k + 2k^* + \ell)(n-1)$ bits. Output gates require to open all the commitments to the shares of the output. Since opening commitments in [9, 10, 17] requires to send two vectors of length ℓ to every other party, which has a total complexity of $2\ell(n-1)n$. Input gates have the same cost as the other two protocols.

Concrete parameters.

Summing up we compare the communication costs of multiplication and output gates in Table D.1 since these are the gates where the communication differs.

	MiniMAC	Committed MPC	Our protocol
Multiply	$(4k + 2k^*)(n-1)$	$(4k + 2k^* + \ell)(n-1)$	$(4k + 2m)(n-1)$
Output	$2 \cdot \ell \cdot n + 2k(n-1)$	$2 \cdot \ell \cdot (n-1)n$	$2 \cdot m \cdot n + 2k(n-1)$

Table D.1: Total number of bits communicated in the different gates in the online phases, when computing k instances of a boolean circuit in parallel. Communication per party is obtained dividing by n.

The key quantities are the relation between m/k (in our case) and k^*/k and ℓ/k in the other two protocols. While the possible parameters ℓ,k,d of linear codes have been studied exhaustively in the theory of error-correcting codes, relations between those parameters and k^*, d^* are much less studied, at least in the case of binary codes. As far as we know, the only concrete non-asymptotic results are given in [7, 12]. In particular, the parameters in Table D.2 are achievable.

On the other hand, the parameters for our protocol depend on parameters achievable by RMFEs. By Theorem 2.2 for all $1 \leq r \leq 33$, there exists a RMFE with $k = 3r$ and $m = 10r - 5$. Some specific values are shown in Table D.3.

This leads to the communication complexities *per computed instance of the*

ℓ	k	$d \geq$	k^*	$d^* \geq$	k^*/k	ℓ/k
2047	210	463	1695	67	8.07	9.75
4095	338	927	3293	135	9.74	12.11

Table D.2: Parameters for C and C^{*2} from [7].

k	m	m/k
21	65	3.10
42	135	3.21

Table D.3: Parameters for RMFE from [8].

boolean circuit for security parameters $s = 64$ and $s = 128$ given in Table D.4. For larger security parameter, the comparison becomes more favourable to our technique, since the "expansion factor" m/k degrades less than the one for known constructions of squares of error correcting codes.

Sec. par.	Phase	MiniMAC	Committed MPC	Our protocol
$s = 64$	Multiply	$20.14 \cdot (n-1)$	$29.89 \cdot (n-1)$	$10.2 \cdot (n-1)$
	Output	$19.5 \cdot n + 2(n-1)$	$19.5 \cdot (n-1)n$	$6.2 \cdot n + 2(n-1)$
$s = 128$	Multiply	$23.48 \cdot (n-1)$	$35.58 \cdot (n-1)$	$10.42 \cdot (n-1)$
	Output	$24.22 \cdot n + 2(n-1)$	$24.22 \cdot (n-1)n$	$6.42 \cdot n + 2(n-1)$

Table D.4: Total number of bits sent per instance at multiplication and output gates

If instead we want to use Theorem 2.3, so that we can define the MACs over a field of degree a power of two, then the last column would have complexities $12 \cdot (n-1)$ and $8 \cdot n + 2(n-1)$ in both the cases $s = 64$ and $s = 128$.

Comparison with an online communication-efficient version of MiniMAC.

In [13], a version of MiniMAC is proposed which uses linear codes over the extension field $\mathbb{F}_{256}$. The larger field enables to use a Reed-Solomon code, for which $k^* = 2k - 1$. However, because this only gives coordinatewise operations in $\mathbb{F}_{256}^k$, the protocol needs to be modified in order to allow for bitwise operations instead. The modified version requires the opening of two C^*-encodings at every multiplication gate and a more complicated and much larger preprocessing, where in addition to creating certain type of multiplication triple, the preprocessing phase needs to provide authenticated sharings of 16 other vectors created from the bit decompositions of the coordinates of the two "factor" vectors in the triple. As far as we know, no preprocessing phase that creates these authenticated elements has been proposed.

The amortized communication complexity of that protocol is of $8(n-1)$ bits per multiplication gate, per instance of the circuit, which is slightly less than 80% of ours. On the other hand, we estimate that the complexity of the preprocessing would be at least 4 times as that of our protocol and possibly larger, based on the number of preprocessed elements and their correlation.

Computation and storage.

In terms of storage, each authenticated share of a k-bit vector is $m + k$ bits, which is slightly over 4 bits per data bit. MiniMAC and Committed MPC require a larger storage of $\ell + k$ bits because the MAC shares/commitments are in $\mathbb{F}_2^\ell$. In [13] shares are also 4 bits per data bit because of using RS codes, but the amount of preprocessed data is much larger. In terms of computation, while our protocol does slightly better for additions (again because of the shorter shares, and since the addition in $\mathbb{F}_{2^m}$ is as in $\mathbb{F}_2^m$), and the same happens with additions required by multiplication gates, computing the terms $\epsilon * \langle \mathbf{y} \rangle$, $\delta * \langle \mathbf{x} \rangle$, $\phi(\epsilon) \cdot \phi(\delta)$ requires in total 5 multiplications in $\mathbb{F}_{2^m}$ which, being field multiplications, are more expensive than the coordinatewise ones required by MiniMAC, even if some of them are in a larger space $\mathbb{F}_2^\ell$.

4 From Batch Computations to Single Circuit Computations

We explain now how to adapt our protocol, which was presented as a protocol for the simultaneous secure evaluation of k instances of the same boolean circuit, into a protocol that computes a single evaluation of a boolean circuit with little overhead, as long as the circuit is sufficiently "well-formed". This is a quite straightforward adaptation of the ideas presented in [15]. The technique can be used in general for any boolean circuit but it works better when the circuit satisfies a number of features, which we can loosely sum up as follows:

- The circuit is organized in layers, each layer consisting of the same type of gate (either additive or multiplicative). We number the layers in increasing order from the input layer (layer 0) to the output layer.
- For most layers, the number of gates u is either a multiple of k or large enough so that the overhead caused by the need to add u' dummy gates to obtain a multiple of k and compute the gates in batches of k is negligible.
- For most pairs of layers i and j, where $i < j$, the number of output bits from layer i that are used as inputs in layer j is either 0 or sufficiently large so that we do not incur in much overhead by adding dummy outputs or inputs (again to achive blocks of size exactly k).

The idea from [15] is that given a layer of u gates, where we can assume $u = t \cdot k$ we organize the inputs of the layers in t blocks of k gates, and we will compute each block by using the corresponding subroutine in our protocol.

For that we need to have authenticated shared blocks of inputs $\langle \mathbf{x} \rangle$, $\langle \mathbf{y} \rangle$ where the i-th coordinates x_i, y_i are the inputs of the i-th gate in the block. This assumes gates are of fan-in 2. For the case of addition gates, we can also support of course arbitrary fan-in gates, but then we want to have the same fan-in in every gate in the same block, again to avoid overheads where we need to introduce dummy 0 inputs. In any case at the end of the computation of this layer we obtain t authenticated sharings $\langle \mathbf{z} \rangle$.

The question is how to now transition to another layer j. Let us assume that layer j takes inputs from l blocks $\langle \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{x}_l \rangle$ of k bits each coming from some previous layer. Of course the issue is that we are not guaranteed that we can use these as input blocks for the layer j. We will likely need to reorganize the bits in blocks, we may need to use some of the bits more than once, and we may not need to use some of the bits of some output blocks. At first sight this reorganization may look challenging, because note that the bits of each $\mathbf{x}_i$ can be "quite intertwined" in the MAC $\alpha \cdot \phi(\mathbf{x}_i)$.

However in all generality, we can define l' functions $F_1, \ldots, F_{l'} : \mathbb{F}_2^{kl} \to \mathbb{F}_2^k$ such that if we write $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l)$ the concatenation of the output blocks, then $F_1(\mathbf{X}), \ldots, F_{l'}(\mathbf{X})$ are the input blocks we need. These maps are $\mathbb{F}_2$-linear; in fact, each of the coordinates of each F_i are either a projection to one coordinate of the input or the 0-map. We assume that all these reorganizing functions can be obtained from the description of the function and therefore they are known and agreed upon by all parties.

Calling $F = (F_1, F_2, \ldots, F_{l'})$, suppose we can obtain by preprocessing

$$(\langle \mathbf{r}_1 \rangle, \langle \mathbf{r}_2 \rangle, \ldots, \langle \mathbf{r}_l \rangle), (\langle F_1(\mathbf{R}) \rangle, \langle F_2(\mathbf{R}) \rangle, \ldots, \langle F_{l'}(\mathbf{R}) \rangle),$$

where $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_l)$ is again the concatenation in $\mathbb{F}_2^{kl}$. To ease the notation we will write $(\langle \mathbf{R} \rangle, \langle F(\mathbf{R}) \rangle)$ and call this a reorganizing pair.

Then, reorganizing is done in the following way. The parties compute $\langle \mathbf{x}_j \rangle - \langle \mathbf{r}_j \rangle$ and open these values for $j = 1, 2, \ldots, l$. Afterwards, they compute

$$F_j(\mathbf{x}_1 - \mathbf{r}_1, \ldots, \mathbf{x}_l - \mathbf{r}_l) + \langle F_j(\mathbf{r}_1, \ldots, \mathbf{r}_l) \rangle = \langle F_j(\mathbf{x}_1, \ldots, \mathbf{x}_l) \rangle$$

which holds by the linearity of F_j.

We can add this property to our setup above by including the supplements in Figure D.5 to $\mathcal{F}_{\text{Prep}}$, $\mathcal{F}_{\text{MPC}}$, and Π_{Online}. Apart from this we also need to point out that at the input layer, a party may need to add dummy inputs so that her input consists of a number of blocks of k bits.

Of course, it looks as though we have moved the problem to the preprocessing phase, as we still need to construct the reorganizing random pairs $(\langle \mathbf{R} \rangle, \langle F(\mathbf{R}) \rangle)$. But this will be easy because of the $\mathbb{F}_2$-linearity of the maps ϕ and F.

The communication complexity of each reorganizing round is that of opening l vectors in $\mathbb{F}_2^k$, therefore $2lk(n-1)$ bits of communication. Therefore, the efficiency of this technique clearly depends much on the topology of

Functionality $\mathcal{F}_{\text{Prep}}$ (supplement)

4. **ReOrgPair:** On input $(\text{ReOrgPair}, F, (\text{id}_1, \text{id}_2, \ldots, \text{id}_l), (\text{id}'_1, \text{id}'_2, \ldots, \text{id}'_{l'}))$ where $F = (F_1, F_2, \ldots, F_{l'})$, sample l random vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_l$ and set $\text{Val}[\text{id}_j], = \mathbf{r}_j$ for $j = 1,2,\ldots,l$ and $\text{Val}[\text{id}'_j], = F_j(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_l)$ for $j = 1,2,\ldots,l'$.

Functionality $\mathcal{F}_{\text{MPC}}$ (supplement)

6. **Reorganize:** On input $(\text{ReOrg}, F, (\text{id}_1, \text{id}_2, \ldots, \text{id}_l), (\text{id}'_1, \text{id}'_2, \ldots, \text{id}'_{l'})$ compute $F(\text{Val}[\text{id}_1], \text{Val}[\text{id}_2], \ldots, \text{Val}[\text{id}_l]) = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{l'})$. Set $\text{Val}[\text{id}'_j] = \mathbf{z}_j$ for $j = 1,2,\ldots,l'$.

Protocol Π_{Online} (supplement)

6. **Reorganize:** To reorganize between the layers, take a corresponding reorganizing pair $(\langle \mathbf{R} \rangle, \langle F(\mathbf{R}) \rangle)$.
   (a) Compute $\langle \boldsymbol{\epsilon}_j \rangle = \langle \mathbf{x}_j \rangle - \langle \mathbf{r}_j \rangle$ and open $\boldsymbol{\epsilon}_j$ for $j = 1,2,\ldots,l$.
   (b) Compute $F_j(\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_l) + \langle F_j(\mathbf{R}) \rangle = \langle F_j(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l) \rangle$ for $j = 1,2,\ldots,l'$ and input these to the next layer.

Fig. D.5: Reorganizing supplement

the circuit. For example if all the output bits of a given layer are used in the next layer and only there, then we can say that this technique adds roughly 2 bits of communication per party per gate.

5 Preprocessing

In this section, we present how to obtain the preprocessed correlated information we need in our online protocols. The implementation of authentication and construction of multiplication triples is adapted in a relatively straightforward way from MASCOT. This is because MASCOT is based on bit-OT extension, and working bit-by-bit is well suited for our situation because of the maps ϕ, ψ being $\mathbb{F}_2$-linear. For the preprocessing of multiplication triples we do need to introduce some auxiliary protocols with respect to MASCOT: one is the preprocessing of reencoding pairs $(\langle \psi(r) \rangle, [r])$ that we anyway need for the online protocol; another one creates $[r]$ for a random r in the kernel of ψ, which we need in order to avoid some information leakage in the sacrifice step. Both types of preprocessing can be easily constructed based on the $\mathbb{F}_2$-linearity of ψ. Finally, we use the sacrifice step in Committed MPC, rather than the one in MASCOT, because of some technical issues regarding the fact that the image of ϕ is not the entire $\mathbb{F}_{2^m}$ which creates problems when opening certain sharings.

Aside from the aforementioned multiplication triples $(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{c} \rangle)$ where

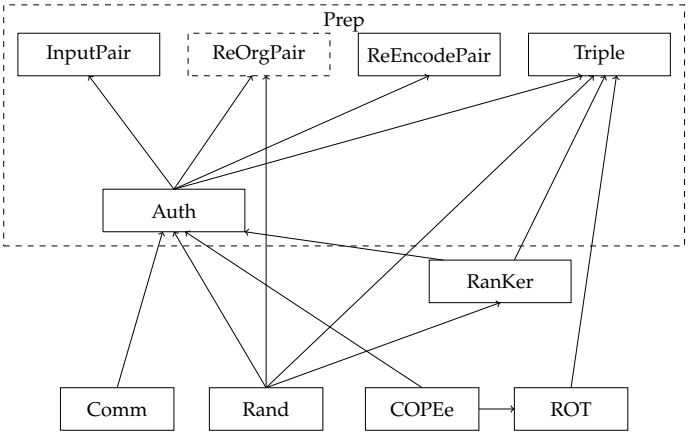

Fig. D.6: Overview of dependency of the protocols needed for the preprocessing.

$\mathbf{c} = \mathbf{a} * \mathbf{b}$, for the online phase we also need to generate input pairs $(\mathbf{r}, \langle \mathbf{r} \rangle)$, reencoding pairs of the form $(\langle \psi(r) \rangle, [r])$, and (in case we want to use the techniques in Section 4) layer reorganizing pairs $(\langle \mathbf{R} \rangle, \langle F(\mathbf{R}) \rangle)$.

To obtain an overview of the way the functionalities presented in this section are dependent on each consider Figure D.6. We use the following basic ideal functionalities: parties can generate uniform random elements in a finite set using the functionality $\mathcal{F}_{\text{Rand}}$ (for the sake of notational simplicity we omit referring to $\mathcal{F}_{\text{Rand}}$ in protocols). Moreover, parties have access to a commitment functionality $\mathcal{F}_{\text{Comm}}$, see Figure D.7. We will also make use of

---

Functionality $\mathcal{F}_{\text{Rand}}$

1. Upon receiving (Rand, S) from all parties, where S is a finite set, choose a uniform random number $r \in S$ and send it to all parties.

Functionality $\mathcal{F}_{\text{Comm}}$

1. Upon receiving (Comm, x, P_i) from P_i and (Comm, P_i) from all other parties the functionality stores x. When receiving an opening command from all parties, the functionality sends x to all parties.

Fig. D.7: Functionalities – Randomness generation and Commitment

a functionality $\mathcal{F}_{\text{ROT}}^{n,k}$ that implements n 1-out-of-2 oblivious transfers of k-bit strings (Figure D.8).

We adapt the correlated oblivious product evaluation functionality $\mathcal{F}_{\text{COPEe}}$ defined in MASCOT [19]. We recall how this functionality works: we again see the field $\mathbb{F}_{2^m}$ as $\mathbb{F}_2[X]/(f)$ for some irreducible polynomial $f \in \mathbb{F}_2[X]$. Then $\{1, X, X^2, \ldots, X^{m-1}\}$ is a basis for $\mathbb{F}_{2^m}$ as a $\mathbb{F}_2$-vector space. The function-

Functionality $\mathcal{F}_{\text{ROT}}^{n,k}$

1. Upon receiving (ROT, P_i, P_j) from party P_i and $(\text{ROT}, P_i, P_j, \mathbf{b})$ from party P_j, where $\mathbf{b} \in \{0,1\}^n$, the functionality chooses $\mathbf{r}_0^l, \mathbf{r}_1^l \in \{0,1\}^k$ uniformly at random and sends these to P_i, while it sends $\mathbf{r}_{b_l}^l$ to P_j for $l = 1,2,\ldots,n$.

Fig. D.8: Functionality – Random OT

ality as described in [19] takes an input $\alpha \in \mathbb{F}_{2^m}$ from one of the parties P_B in the initialization phase; then there is an arbitrary number of extend phases where on input $x \in \mathbb{F}_{2^m}$ from P_A, the functionality creates additive sharings of $\alpha \cdot x$ for the two parties. However, if P_A is corrupted it may instead decide to input a vector of elements $(x_0, x_1, \ldots, x_{m-1}) \in (\mathbb{F}_{2^m})^m$, and in that case the functionality outputs a sharing of $\sum_{i=0}^{m-1} x_i \cdot \alpha_i \cdot X^i$ (where α_i are the coordinates of α in the above basis). The honest case would correspond to all x_i being equal to x. This functionality from MASCOT corresponds to the steps Initialize and ExtendField in our version Figure D.10. We augment this by adding the step ExtendVector, where party P_A can input a vector $\mathbf{x} \in \mathbb{F}_2^k$ and the functionality outputs an additive sharing of $\alpha \cdot \phi(\mathbf{x}) \in \mathbb{F}_{2^m}$. If party P_A is corrupted it may instead input $(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{m-1}) \in (\mathbb{F}_2^k)^m$. In that case the functionality outputs an additive sharing of $\sum_{i=0}^{m-1} \phi(\mathbf{x}_i) \cdot \alpha_i \cdot X^i$, and note that this is more restrictive for the corrupted adversary than ExtendField since the values $\phi(\mathbf{x}_i)$ are not free in $\mathbb{F}_{2^m}$ but confined to the image of ϕ. We define the functionality $\mathcal{F}_{\text{COPEe}}$ in Figure D.9 and present a protocol implementing the functionality in Figure D.10.

Proposition 5.1. Π_{COPEe} *securely implements* $\mathcal{F}_{\text{COPEe}}$ *in the* $\mathcal{F}_{OT}^{m,\lambda}$*-hybrid model.*

Proof. The commands Initialize and ExtendField are as in [19] (the latter being called Extend there). The proof for our ExtendVector command is analogous to the one for the ExtendField except, as explained, because the ideal functionality restricts the choice by a corrupt P_A of the element that is secret shared. We briefly show the simulation of ExtendVector together with Initialize.

If P_B is corrupted, the simulator receives $(\alpha_0, \ldots, \alpha_{m-1})$ from the adversary, and simulates the initialization phase by sampling the seeds at random, and sending the corresponding one to the adversary. It simulates the ExtendVector phase by choosing $\mathbf{u}_i$ uniformly at random in the corresponding domain, computes q as an honest P_B would do and inputs this to the functionality. Indistinguishability holds by the pseudorandomness of F, as shown in [19].

If P_A is corrupted then the simulator receives the seeds from the adversary in the Initialize phase, and from there it computes all the $\mathbf{t}_b^i$ in the ExtendVector phase. Then when the adversary sends $\mathbf{u}_i$, the simulators extract $\mathbf{x}_i = \mathbf{u}_i - \mathbf{t}_0^i + \mathbf{t}_1^i$ and inputs $t = -\sum_{i=0}^{m-1} \phi(\mathbf{t}_0^i) \cdot X^i$ and $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$

Functionality $\mathcal{F}_{\text{COPEe}}$

This functionality runs with two parties P_A and P_B and an adversary $\mathcal{A}$. The Initialize phase is run once first. The ExtendVector and ExtendField may be run an arbitrary number of times, in arbitrary order.

1. **Initialize:** On input $\alpha \in \mathbb{F}_{2^m}$ from P_B the functionality stores this value. We identify α by the vector $(\alpha_0,\alpha_1,\ldots,\alpha_{m-1}) \in \mathbb{F}_2^m$, s.t. $\alpha = \sum_{i=0}^{m-1} \alpha_i \cdot X^i$.
2. **ExtendVector:** P_A inputs a vector $\mathbf{x} \in \mathbb{F}_2^k$.
   (a) If P_A is corrupt receive $t \in \mathbb{F}_{2^m}$ and $(\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_{m-1}) \in (\mathbb{F}_2^k)^m$ from $\mathcal{A}$, where the numbers indicate that $\mathbf{x}_i$ might be different from $\mathbf{x}$. Then compute q such that $q + t = \sum_{i=0}^{m-1} \phi(\mathbf{x}_i) \cdot \alpha_i \cdot X^i$.
   (b) If both parties are honest sample $t \in \mathbb{F}_{2^m}$ at random and compute q such that $q + t = \alpha \cdot \phi(\mathbf{x})$.
   (c) If only P_B is corrupt then receive $q \in \mathbb{F}_{2^m}$ from $\mathcal{A}$ and compute t such that $q + t = \alpha \cdot \phi(\mathbf{x})$.
   (d) Output t to P_A and q to P_B.
3. **ExtendField:** P_A inputs a field element $x \in \mathbb{F}_{2^m}$.
   (a) If P_A is corrupt receive $t \in \mathbb{F}_{2^m}$ and $(x_0,x_1,\ldots,x_{m-1}) \in (\mathbb{F}_{2^m})^m$ from $\mathcal{A}$, where the numbers indicate that x_i might be different from x. Then compute q such that $q + t = \sum_{i=0}^{m-1} x_i \cdot \alpha_i \cdot X^i$.
   (b) If both parties are honest sample $t \in \mathbb{F}_{2^m}$ at random and compute q such that $q + t = \alpha \cdot x$.
   (c) If only P_B is corrupt then receive $q \in \mathbb{F}_{2^m}$ from $\mathcal{A}$ and compute t s.t. $q + t = \alpha \cdot x$.
   (d) Output t to P_A and q to P_B.

Fig. D.9: Functionality – Correlated oblivious product evaluation with errors.

to $\mathcal{F}_{\text{COPEe}}$. In this case all outputs are computed as in the real world and indistinguishability follows. □

5.1 Authentication

In protocol Π_{Auth} (Figures D.11, D.12, and D.13), we use $\mathcal{F}_{\text{COPEe}}$ to implement $\mathcal{F}_{\text{Auth}}$.

In the initialize phase each pair of parties (P_i,P_j) call the initialize phase from $\mathcal{F}_{\text{COPEe}}$ where P_i inputs a MAC key. Afterwards P_j can create authenticated sharings to the desired values, both of boolean vectors and of elements in the larger field: namely P_j constructs additive random sharings of the individual values and uses the appropriate extend phase of $\mathcal{F}_{\text{COPEe}}$ to obtain additive sharings of the MACs. At last, a random linear combination of the values chosen by P_j is checked. Here privacy is achieved by letting P_j include a dummy input x_{t+1} to mask the other inputs.

Protocol Π_{COPEe}

The protocol is a two party protocol with parties P_A and P_B that uses PRFs $F\colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \mathbb{F}_2^k$ and $F_{\text{Field}}\colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \mathbb{F}_{2^m}$, has access to the ideal functionality $\mathcal{F}_{\text{ROT}}^{m,\lambda}$, and maintains a global counter $j := 0$. The Initialize phase is run once first, and then the ExtendVector and ExtendField may be run an arbitrary number of times, in arbitrary order.

1. **Initialize:** On input $\alpha \in \mathbb{F}_{2^m}$ from P_B:
   (a) The parties engage in $\mathcal{F}_{\text{ROT}}^{m,\lambda}$ where P_B inputs $(\alpha_0,\alpha_1,\ldots,\alpha_{m-1}) \in \mathbb{F}_2^m$ s.t. $\alpha = \sum_{i=0}^{m-1} \alpha_i \cdot X^i \in \mathbb{F}_{2^m}$. P_A receives $\left\{(\mathbf{k}_0^i,\mathbf{k}_1^i)\right\}_{i=0}^{m-1}$ and P_B receives $\mathbf{k}_{\alpha_i}^i$ for $i = 0,1,\ldots,m-1$.
2. **ExtendVector:** On input $\mathbf{x} \in \mathbb{F}_2^k$ from P_A:
   (a) For $i = 0,1,\ldots,m-1$:
      i. Define $\mathbf{t}_0^i = F(\mathbf{k}_0^i,j) \in \mathbb{F}_2^k$, $\mathbf{t}_1^i = F(\mathbf{k}_1^i,j) \in \mathbb{F}_2^k$ so P_A knows $(\mathbf{t}_0^i,\mathbf{t}_1^i)$ and P_B knows $\mathbf{t}_{\alpha_i}^i$.
      ii. P_A sends $\mathbf{u}_i = \mathbf{t}_0^i - \mathbf{t}_1^i + \mathbf{x}$ to P_B.
      iii. P_B computes $\mathbf{q}_i = \alpha_i \cdot \mathbf{u}_i + \mathbf{t}_{\alpha_i}^i = \mathbf{t}_0^i + \alpha_i \cdot \mathbf{x} \in \mathbb{F}_2^k$.
   (b) $j := j + 1$
   (c) P_B outputs $q = \sum_{i=0}^{m-1} \phi(\mathbf{q}_i) \cdot X^i$ and P_A outputs $t = -\sum_{i=0}^{m-1} \phi(\mathbf{t}_0^i) \cdot X^i$
3. **ExtendField:** On input $x \in \mathbb{F}_{2^m}$ from P_A:
   (a) For $i = 0,1,\ldots,m-1$:
      i. Define $t_0^i = F_{\text{Field}}(\mathbf{k}_0^i,j) \in \mathbb{F}_{2^m}, t_1^i = F_{\text{Field}}(\mathbf{k}_1^i,j) \in \mathbb{F}_{2^m}$, so P_A knows (t_0^i,t_1^i) and P_B knows $t_{\alpha_i}^i$.
      ii. P_A sends $u_i = t_0^i - t_1^i + x$ to P_B.
      iii. P_B computes $q_i = \alpha_i \cdot u_i + t_{\alpha_i}^i$.
   (b) $j := j + 1$.
   (c) P_B outputs $q = \sum_{i=0}^{m-1} q_i \cdot X^i$ and P_A outputs $t = -\sum_{i=0}^{m-1} t_0^i \cdot X^i$.

Fig. D.10: Correlated oblivious product evaluation with errors.

Proposition 5.2. Π_{Auth} *securely implements* $\mathcal{F}_{\text{Auth}}$ *in the* $(\mathcal{F}_{\text{COPEe}},\mathcal{F}_{\text{Rand}},\mathcal{F}_{\text{Comm}})$*-hybrid model*

Proof. Since the proof is similar to the proof of security for $\Pi_{[[\cdot]]}$ in [19], we point out the differences and argue why it does not have an impact on the security.

First of all note that our functionality, in contrary to $\Pi_{[[\cdot]]}$, has an Add command and a LinComb command. This is because we reserve the LinComb command for linear combinations which output $[\cdot]$-sharings, while Add outputs a $\langle\cdot\rangle$-sharing. In any case, the Add and LinComb command consist of local computations so it is trivial to argue their security. The Initialize command only invokes the Initialize command from the ideal functionality $\mathcal{F}_{\text{COPEe}}$, which is exactly the same as in [19]. Since the Open command lets

Protocol Π_{Auth} – Part 1

This protocol additively shares and authenticates elements in $\mathbb{F}_2^k$ or $\mathbb{F}_{2^m}$, and allows linear operations and openings to be carried out on these shares. Note that the **Initialize** procedure only needs to be called once, to set up the MAC key. We assume access to the ideal functionalities $\mathcal{F}_{\text{Rand}}$, $\mathcal{F}_{\text{Comm}}$, and $\mathcal{F}_{\text{COPEe}}$.

1. **Initialize:** Each party P_i samples a MAC key share $\alpha^{(i)} \in \mathbb{F}_{2^m}$. Each pair of parties (P_i, P_j) for $i \neq j$ calls $\mathcal{F}_{\text{COPEe.Initialize}}$ where P_i inputs $\alpha^{(i)}$.
2. **Input:** On input $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s \in \mathbb{F}_2^k$ and $x_1, x_2, \ldots, x_t \in \mathbb{F}_{2^m}$ from P_j the parties do the following:
   (a) P_j samples random element $x_{t+1} \in \mathbb{F}_{2^m}$.
   (b) For $h = 1,2,\ldots,s$, P_j generates additive sharing $\sum_{i=1}^n \mathbf{x}_h^{(i)} = \mathbf{x}_h$ and sends $\mathbf{x}_h^{(i)}$ to P_i. Similarly, for $l = 1,2,\ldots,t+1$, P_j generates additive sharing $\sum_{i=1}^n x_l^{(i)} = x_l$ and sends $x_l^{(i)}$ to P_i.
   (c) For every $i \neq j$, P_i and P_j call $\mathcal{F}_{\text{COPEe.ExtendVector}}$ s times where P_j inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s$ and $\mathcal{F}_{\text{COPEe.ExtendField}}$ $t+1$ times with inputs $x_1, x_2, \ldots, x_{t+1}$.
   (d) P_i receives $q_h^{(i,j)} \in \mathbb{F}_{2^m}$ and P_j receives $t_h^{(j,i)} \in \mathbb{F}_{2^m}$ such that
   $$q_h^{(i,j)} + t_h^{(j,i)} = \alpha^{(i)} \cdot \phi(\mathbf{x}_h), \qquad \text{for } h = 1,2,\ldots,s$$
   $$q_{l+s}^{(i,j)} + t_{l+s}^{(j,i)} = \alpha^{(i)} \cdot x_l, \qquad \text{for } l = 1,2,\ldots,t+1$$
   (e) Each P_i, $i \neq j$ defines the MAC shares $m^{(i)}(\mathbf{x}_h) = q_h^{(i,j)}$ for $h = 1,2,\ldots,s$ and $m^{(i)}(x_l) = q_{l+s}^{(i,j)}$ for $l = 1,2,\ldots,t+1$. P_j computes MAC share
   $$m^{(j)}(\mathbf{x}_h) = \alpha^{(j)} \cdot \phi(\mathbf{x}_h) + \sum_{i \neq j} t_h^{(j,i)} \qquad \text{for } h = 1,2,\ldots,s$$
   $$m^{(j)}(x_l) = \alpha^{(j)} \cdot x_l + \sum_{i \neq j} t_{l+s}^{(j,i)} \qquad \text{for } l = 1,2,\ldots,t+1$$
   This implies that we have $\langle \mathbf{x}_h \rangle$ for $h = 1,2,\ldots,s$ and $[x_l]$ for $l = 1,2,\ldots,t+1$
   (f) The parties call $\mathcal{F}_{\text{Rand}}(\mathbb{F}_{2^m}^{s+t+1})$ to obtain $(r_1, \ldots, r_{s+t+1})$.
   (g) Compute $[y] = \sum_{h=1}^s r_h \cdot \langle \mathbf{x}_h \rangle + \sum_{l=1}^{t+1} r_{s+l} \cdot [x_l]$ by calling $\Pi_{\text{Auth.LinComb}}$ and open y by calling $\Pi_{\text{Auth.Open}}$.
   (h) Call $\Pi_{\text{Auth.Check}}$ on y. If the check succeeds output $\langle \mathbf{x}_h \rangle$ for $h = 1,2,\ldots,s$, and $[x_l]$ for $l = 1,2,\ldots,t$.

Fig. D.11: Authenticated shares – Part 1.

the adversary choose what to open to there is not much to discuss here either.

Therefore, what we need to discuss is the Input and Check commands. The idea is that if the check in the input phase is passed and the adversary opens to incorrect values later on, then the probability to pass a check later on will be negligible. In comparison to [19], we have both values in $\mathbb{F}_{2^m}$ and vectors in $\mathbb{F}_2^k$, but we can still use the same arguments there, because the

Protocol Π_{Auth} – Part 2

3. **Add:** On input $(\text{Add}, \bar{\text{id}}, \text{id}, a)$ the parties do the following. If a is an index of Val they retrieve shares and MAC shares $\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, m^{(i)}(\mathbf{x}), m^{(i)}(\mathbf{y})$ where $\mathbf{x}$ corresponds to id and $\mathbf{y}$ corresponds to the index a in Val. P_i computes

$$\mathbf{x}^{(i)} + \mathbf{y}^{(i)} \quad \text{and} \quad m^{(i)}(\mathbf{x}) + m^{(i)}(\mathbf{y})$$

and stores these under $\bar{\text{id}}$. If a is a vector, i.e. $a = \mathbf{a}$, they retrieve the share and MAC share $\mathbf{x}^{(i)}, m^{(i)}(\mathbf{x})$ where $\mathbf{x}$ corresponds to id in Val. P_i computes

$$\mathbf{x}^{(i)} + \begin{cases} \mathbf{a} & \text{if } i = 1 \\ 0 & \text{if } i \neq 1 \end{cases} \quad \text{and} \quad m^{(i)}(\mathbf{x}) + \alpha^{(i)} \cdot \phi(\mathbf{a}).$$

and stores these under $\text{Val}[\bar{\text{id}}]$.

4. **LinComb:** On input $(\text{LinComb}, \bar{\text{id}}, (\text{id}_1, \text{id}_2, \ldots \text{id}_s), (\text{id}'_1, \text{id}'_2, \ldots \text{id}'_t), c_1, c_2, \ldots, c_{s+t}, c)$ where $t \geq 1$, the P_i retrieves its shares and MAC shares $\left\{\mathbf{x}_j^{(i)}, m^{(i)}(\mathbf{x}_j)\right\}_{j=1,2,\ldots,s}$ corresponding to id_j in Val and $\left\{x_j^{(i)}, m^{(i)}(x_j)\right\}_{j=1,2,\ldots,t}$ corresponding to id'_j in ValField. P_i computes

$$y^{(i)} = \sum_{j=1}^{s} c_j \cdot \phi(\mathbf{x}_j^{(i)}) + \sum_{j=1}^{t} c_{s+j} \cdot x_j^{(i)} + \begin{cases} c & \text{if } i = 1 \\ 0 & \text{if } i \neq 1 \end{cases}$$

$$m^{(i)}(y) = \sum_{j=1}^{s} c_j \cdot m^{(i)}(\mathbf{x}_j) + \sum_{j=1}^{t} c_{s+j} \cdot m^{(i)}(x_j) + c \cdot \alpha^{(i)}$$

and stores these under $\bar{\text{id}}$ in ValField.

Fig. D.12: Authenticated shares – Part 2.

check in the Input phase and all further checks are in $\mathbb{F}_{2^m}$ and therefore the simulation and indistinguishability is following by the exact same arguments as in [19]. □

5.2 Input, Reencoding, and Reorganizing Pairs

The two functionalities $\mathcal{F}_{\text{COPEe}}$ and $\mathcal{F}_{\text{Auth}}$ are the building blocks for the preprocessing. They are very similar in shape to the MASCOT functionalities but with some few corrections to include that sharings can be of vectors instead of field elements in $\mathbb{F}_{2^m}$. With these building blocks we can produce the randomness needed for the online phase. First of all, we produce input pairs with protocol $\Pi_{\text{InputPair}}$ in Figure D.14. Proposition 5.3 is straightforward.

Proposition 5.3. $\Pi_{\text{InputPair}}$ *securely implements* $\mathcal{F}_{\text{Prep.InputPair}}$ *in the* $\mathcal{F}_{\text{Auth}}$*-hybrid model.*

We also need to construct pairs to re-encode $[\cdot]$-sharings to $\langle\cdot\rangle$-sharings after a multiplication. A protocol $\Pi_{\text{ReEncodePair}}$ for producing the pairs $(\langle\psi(r)\rangle, [r])$ for random $r \in \mathbb{F}_{2^m}$ is shown in Figure D.15.

Protocol Π_{Auth} – Part 3

5. **Open:** On input $(\text{Open},\text{Dict},\text{id},S)$ party P_i retrieves the share corresponding to the dictionary and index, sends the share to P_j (the party with lowest index in S) who sums the shares and sends the sum back to the other parties in S.
6. **Check:**
   (a) On input
   $$(\text{Check},(\text{id}_1,\text{id}_2,\ldots,\text{id}_s),(\text{id}'_1,\text{id}'_2,\ldots,\text{id}'_t),(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_s),(x_1,x_2,\ldots,x_t))$$
   parties sample a random vector $(r_1,r_2,\ldots,r_{s+t}) \in \mathbb{F}_{2^m}^{s+t}$. P_i retrieves its MAC shares $m^{(i)}(\mathbf{x}_j)$ for $j = 1,2,\ldots,s$ corresponding to id_j in Val and $m^{(i)}(x_j)$ for $j = 1,2,\ldots,t$ corresponding to id'_j in ValField. Define
   $$y = \sum_{j=1}^{s} r_j \cdot \phi(\mathbf{x}_j) + \sum_{j=1}^{t} r_{s+j} \cdot x_j$$
   and let P_i compute
   $$m^{(i)}(y) = \sum_{j=1}^{s} r_j \cdot m^{(i)}(\mathbf{x}_j) + \sum_{j=1}^{t} r_{s+j} \cdot m^{(i)}(x_j)$$
   (b) P_i calls $\mathcal{F}_{\text{Comm}}$ to commit to $\sigma^{(i)} = m^{(i)}(y) - \alpha^{(i)} \cdot y$ and afterwards open the commitment.
   (c) The parties check if $\sigma^{(1)} + \sigma^{(2)} + \cdots + \sigma^{(n)} = 0$ and abort otherwise.

Fig. D.13: Authenticated shares – Part 3.

Protocol $\Pi_{\text{InputPair}}$

The protocol generates $(\mathbf{r},\langle\mathbf{r}\rangle)$ where $\mathbf{r} \in \mathbb{F}_2^k$ is chosen randomly by P_i, the party calling the protocol.

1. **Construct:**
   (a) P_i chooses $\mathbf{r} \in \mathbb{F}_2^k$ uniformly at random.
   (b) P_i calls $\mathcal{F}_{\text{Auth.Input}}$ to obtain $\langle\mathbf{r}\rangle$ and output this authenticated share.

Fig. D.14: Creating input pairs.

Proposition 5.4. $\Pi_{\text{ReEncodePair}}$ *securely implements* $\mathcal{F}_{\text{Prep.ReEncodePair}}$ *in the* $(\mathcal{F}_{\text{Auth}},\mathcal{F}_{\text{Rand}})$*-hybrid model with statistical security parameter* s*.*

Proof. First notice that at least one of the parties is honest and hence $r_j = \sum_{i=1}^{n} r_j^{(i)}$ is random because one of the terms is. Suppose that at the end of the Combine phase parties have created $(\langle\mathbf{s}_j\rangle,[r_j])$, where possibly $\mathbf{s}_j \neq \psi(r_j)$.

Let $\boldsymbol{\epsilon}_j = \mathbf{s}_j - \psi(r_j)$ for all j. By $\mathbb{F}_2$-linearity of ψ, $\mathbf{b}_i - \psi(b_i) = \sum_{j=1}^{t+s} a_{ij}\boldsymbol{\epsilon}_j$. Hence if all $\boldsymbol{\epsilon}_j = \mathbf{0}$, the check passes for all i. While if there is some $\boldsymbol{\epsilon}_j \neq \mathbf{0}$,

$j = 1,\ldots,t$, then for every i the probability that $\sum_{j=1}^{t+s} a_{ij}\epsilon_j = \mathbf{0}$ is at most $1/2$.

Since the checks are independent we obtain that if some $\epsilon_j \neq \mathbf{0}$, $j = 1,\ldots,t$ then the protocol will abort except with probability at most 2^{-s}. Note also that $b_i = r_{t+i} + \sum_{j=1}^{t} a_{ij} r_j$, so opening the b_i reveals no information about the output values $r_1,\ldots,r_t$. □

Protocol $\Pi_{\text{ReEncodePair}}$

The protocol generates $(\langle\psi(r_j)\rangle,[r_j])$ for $j = 1,2,\ldots,t$, where r_j is random in $\mathbb{F}_{2^m}$ and unknown to all parties. We assume access to the functionalities $\mathcal{F}_{\text{Rand}}$ and $\mathcal{F}_{\text{Auth}}$.

1. **Construct:**
   (a) P_i chooses $r_j^{(i)}$ for $j = 1,2,\ldots,t+s$ uniformly at random in $\mathbb{F}_{2^m}$.
   (b) P_i calls $\mathcal{F}_{\text{Auth.Input}}$ to obtain $[r_j^{(i)}]$ and $\langle\psi(r_j^{(i)})\rangle$.
   (c) Compute $[r_j] = \sum_{i=1}^{n}[r_j^{(i)}]$ and $\langle\psi(r_j)\rangle = \sum_{i=1}^{n}\langle\psi(r_j^{(i)})\rangle$ for $j = 1,2,\ldots,t+s$.
2. **Sacrifice:**
   (a) Call $\mathcal{F}_{\text{Rand}}(\mathbb{F}_2^t)$ to obtain $\mathbf{a}_i'$ for $i = 1,2,\ldots,s$ and define $\mathbf{a}_i = (\mathbf{a}_i', \mathbf{e}_i) \in \mathbb{F}_2^{t+s}$ where $\mathbf{e}_i$ is the i'th canonical basis vector of length s.
   (b) Compute $[b_i] = \sum_{j=1}^{t+s} a_{ij}[r_j]$ and $\langle\mathbf{b}_i\rangle = \sum_{j=1}^{t+s} a_{ij}\langle\psi(r_j)\rangle$, where a_{ij} is the j'th entry of $\mathbf{a}_i$, and partially open b_i and $\mathbf{b}_i$.
   (c) If $\psi(b_i) \neq \mathbf{b}_i$ for some $i \in \{1,2,\ldots,s\}$ then abort.
   (d) Call $\mathcal{F}_{\text{Auth.Check}}$ on the opened values $\mathbf{b}_i$ and b_i.
3. **Output:** Output $(\langle\psi(r_j)\rangle,[r_j])$ for $j = 1,2,\ldots,t$.

Fig. D.15: Re-encode pairs.

Finally, a protocol for producing reorganizing pairs is given in Figure D.16.

Proposition 5.5. $\Pi_{\text{ReOrgPair}}$ *securely implements* $\mathcal{F}_{\text{Prep.ReOrgPair}}$ *in the* $(\mathcal{F}_{\text{Auth}},\mathcal{F}_{\text{Rand}})$*-hybrid model with statistical security parameter* s*.*

The proof of this proposition is similar to that of Proposition 5.4.

5.3 Multiplication Triples

Our protocol Π_{Triple} for constructing triples is given in Figure D.18. We note that $\mathbf{c} = \mathbf{a} * \mathbf{b} = \sum_{i,j} \mathbf{a}^{(i)} * \mathbf{b}^{(j)}$ and hence sharings of $\mathbf{c}$ can be obtained by adding sharings of the summands, where each of the summands only require two parties P_i and P_j to interact. Again, the construction step is much like the construction step from the protocol Π_{Triple} in [19]. where we have modified the protocol such that it produces triples $(\langle\mathbf{a}\rangle,\langle\mathbf{b}\rangle,\langle\mathbf{c}\rangle)$ instead of $([a],[b],[c])$.

Protocol $\Pi_{\text{ReOrgPair}}$

The protocol generates $(\langle \mathbf{R}_h \rangle, \langle F(\mathbf{R}_h) \rangle)$ where $\mathbf{R}_h = (\mathbf{r}_{h,1}, \ldots, \mathbf{r}_{h,l})$ and $F = (F_1, \ldots, F_{l'})$ is a linear function $F \colon \mathbb{F}_2^{kl} \to \mathbb{F}_2^{kl'}$ for $h = 1,2,\ldots,t$. Furthermore, $\mathbf{r}_{h,j}$ is random in $\mathbb{F}_2^k$ and unknown to all parties. We assume access to the functionalities $\mathcal{F}_{\text{Rand}}$ and $\mathcal{F}_{\text{Auth}}$.

1. **Construct:**
   (a) P_i chooses $\mathbf{r}_{h,j}^{(i)}$ for $j = 1,2,\ldots,l$ and $h = 1,2,\ldots t+s$ uniformly at random in $\mathbb{F}_2^k$.
   (b) P_i calls $\mathcal{F}_{\text{Auth.Input}}$ to obtain $\langle \mathbf{r}_{h,j}^{(i)} \rangle$ and $\langle F_{j'}(\mathbf{r}_{h,1}^{(i)}, \ldots, \mathbf{r}_{h,l}^{(i)}) \rangle$ for $j = 1,2,\ldots,l$, $j' = 1,2,\ldots,l'$ and $h = 1,2,\ldots,t+s$.
   (c) The parties compute $\langle \mathbf{r}_{h,j} \rangle = \sum_{i=1}^{n} \langle \mathbf{r}_{h,j}^{(i)} \rangle$ and $\langle F_{j'}(\mathbf{r}_{h,1}, \ldots \mathbf{r}_{h,l}) \rangle = \sum_{i=1}^{n} \langle F_{j'}(\mathbf{r}_{h,1}^{(i)}, \ldots \mathbf{r}_{h,l}^{(i)}) \rangle$. Thus we have $(\langle \mathbf{R}_h \rangle, \langle F(\mathbf{R}_h) \rangle)$ for $h = 1,2,\ldots,t+s$.
2. **Sacrifice:**
   (a) Call $\mathcal{F}_{\text{Rand}}(\mathbb{F}_2^t)$ to obtain $\mathbf{a}'_i$ for $i = 1,2,\ldots,s$ and define $\mathbf{a}_i = (\mathbf{a}'_i, \mathbf{e}_i) \in \mathbb{F}_2^{t+s}$ where $\mathbf{e}_i$ is the i'th canonical basis vector of length s.
   (b) Compute $\langle \mathbf{B}_i \rangle = \sum_{h=1}^{t+s} a_{ih} \langle \mathbf{R}_h \rangle$ and $\langle \mathbf{D}_i \rangle = \sum_{h=1}^{t+s} a_{ih} \langle F(\mathbf{R}_h) \rangle$, where a_{ih} is the h'th entry of $\mathbf{a}_i$, and partially open $\mathbf{B}_i$ and $\mathbf{D}_i$.
   (c) If $F(\mathbf{B}_i) \neq \mathbf{D}_i$ for some $i \in \{1,2,\ldots,s\}$ then abort.
   (d) Call $\mathcal{F}_{\text{Auth.Check}}$ on the opened values $\mathbf{B}_i$ and $\mathbf{D}_i$.
3. **Output:** Output $(\langle \mathbf{R}_h \rangle, \langle F(\mathbf{R}_h) \rangle)$ for $h = 1,2,\ldots,t$.

Fig. D.16: Re-organize pairs.

However, after authentication, we use techniques from Committed MPC [16] to check correctness and avoid leakage on the produced triples. Indeed using the combine and sacrifice steps in MASCOT presents some problems in our case: in the sacrificing step in MASCOT parties take two triples $([a],[b],[c])$ and $([\hat{a}],[b],[\hat{c}])$ and start by opening a random combination $s \cdot [a] - [\hat{a}]$ to some value ρ, so that they can later verify that $s \cdot [c] - [\hat{c}] - \rho \cdot [b]$ opens to 0. Since the second triple will be disregarded, and $s \cdot a - \hat{a}$ completely masks a since $\hat{a}$ is uniformly random, no information is revealed about a. In our case we would have triples $(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{c} \rangle)$ and $(\langle \hat{\mathbf{a}} \rangle, \langle \mathbf{b} \rangle, \langle \hat{\mathbf{c}} \rangle)$ and sample a random $s \in \mathbb{F}_{2^m}$, it would not be the case that $\phi(\hat{\mathbf{a}})$ would act as a proper one-time pad for $s \cdot \phi(\mathbf{a})$[6]. A similar problem would arise for adapting the combine step in [19].

Therefore, we proceed as in [16]: in the protocol Π_{Triple} we start by constructing additive sharings of $N = \tau_1 + \tau_1 \cdot \tau_2^2 \cdot T$ triples. Then some of these triples are opened and it is checked that they are correct. This guarantees that most of the remaining triples are correct. The remaining triples are then organized in buckets and for each bucket all but one of the triples are sacri-

[6]Sampling $\mathbf{s} \in \mathbb{F}_2^k$ instead would not solve the problem since $\mathbf{s} * \langle \mathbf{a} \rangle - \langle \hat{\mathbf{a}} \rangle$ is not a proper $[\cdot]$-sharing as described in Section 3.

Protocol $\Pi_{\text{TripleConstruct}}$

The protocol produces N multiplication triples.

1. **Construction:**
   (a) P_i samples $\mathbf{a}_l^{(i)}, \mathbf{b}_l^{(i)} \in \mathbb{F}_2^k$ for $l = 1,2,\ldots,N$. Denote by $a_{h,l}^{(i)}$, $b_{h,l}^{(i)}$ the h'th entry of $\mathbf{a}_l^{(i)}$, $\mathbf{b}_l^{(i)}$, respectively.
   (b) For $l = 1,2,\ldots,N$ every ordered pair (P_i, P_j) does the following:
      i. The pair call $\mathcal{F}_{\text{ROT}}^{k,1}$ where P_i inputs $a_{h,l}^{(i)}$ for the h'th instance.
      ii. P_j receives $t_{0,h,l}^{(j,i)}, t_{1,h,l}^{(j,i)} \in \mathbb{F}_2$ and P_i receives $t_{a_{h,l}^{(i)},h,l}^{(j,i)}$ for $h = 1,2,\ldots,k$. Denote by $\mathbf{t}_l^{(j,i)}$ the vector having $t_{0,h,l}^{(j,i)}$ as entries and $\mathbf{t}_{1,l}^{(j,i)}$ the vector having $t_{1,h,l}^{(j,i)}$ as entries for $h = 1,2,\ldots,k$. Similarly, denote by $\mathbf{t}_{\mathbf{a}_l^{(i)}}^{(j,i)}$ the vector having $t_{a_{h,l}^{(i)},h,l}^{(j,i)}$ as entries.
      iii. P_j sends $\mathbf{u}_l^{(j,i)} = \mathbf{t}_l^{(j,i)} - \mathbf{t}_{1,l}^{(j,i)} + \mathbf{b}_l^{(j)}$.
      iv. P_i sets $\mathbf{q}_l^{(j,i)} = \mathbf{a}_l^{(i)} * \mathbf{u}_l^{(j,i)} + \mathbf{t}_{a_l^{(i)}}^{(j,i)} = \mathbf{t}_l^{(j,i)} + \mathbf{a}_l^{(i)} \cdot \mathbf{b}_l^{(j)}$.
      v. P_i sets $\mathbf{c}_{i,j,l}^{(i)} = \mathbf{q}_l^{(j,i)}$ and P_j sets $\mathbf{c}_{i,j,l}^{(j)} = -\mathbf{t}_l^{(j,i)}$
   (c) Each party P_i computes $\mathbf{c}_l^{(i)} = \mathbf{a}_l^{(i)} * \mathbf{b}_l^{(i)} + \sum_{j \neq i} \mathbf{c}_{i,j,l}^{(i)} + \mathbf{c}_{j,i,l}^{(i)}$

   Now we have $\mathbf{c}_l = \sum_{i=1}^n \mathbf{c}_l^{(i)} = \sum_{i=1}^n \mathbf{a}_l^{(i)} * \sum_{i=1}^n \mathbf{b}_l^{(i)} = \mathbf{a}_l * \mathbf{b}_l$ *for* $l = 1,2,\ldots,N$
2. **Authenticate:**
   (a) P_i calls $\mathcal{F}_{\text{Auth.Input}}$ to obtain $\langle \mathbf{a}_l^{(i)} \rangle$, $\langle \mathbf{b}_l^{(i)} \rangle$, and $\langle \mathbf{c}_l^{(i)} \rangle$.
   (b) Parties compute $\langle \mathbf{a}_l \rangle = \sum_{i=1}^n \langle \mathbf{a}_l^{(i)} \rangle$ and similarly to obtain $\langle \mathbf{b}_l \rangle$ and $\langle \mathbf{c}_l \rangle$.

Fig. D.17: Construction of multiplication triples.

fied in order to guarantee that the remaining triple is correct with very high probability. In order to be able to open proper sharings in the sacrifice step we need to add authenticated sharings of an element in the kernel of ψ. We present a functionality serving that purpose in Figure D.19 and a protocol implementing it in Figure D.20.

Proposition 5.6. Π_{RanKer} *securely implements* $\mathcal{F}_{\text{RanKer}}$ *in the* $(\mathcal{F}_{\text{Auth}}, \mathcal{F}_{\text{Rand}})$*-hybrid model with statistical security parameter* s*.*

The proof of this proposition is similar to that of Proposition 5.4. The correctness follows from the additivity of ψ.

The sacrifice step opens the door for a selective failure attack, where the adversary can guess some information about the remaining triples from the fact that it has not aborted, so a final combining step is used to remove this leakage.

Protocol Π_{Triple}

The protocol generates T multiplication triples $(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \langle \mathbf{c} \rangle)$ where $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^k$ are random vectors and $\mathbf{c} = \mathbf{a} * \mathbf{b}$. The integers τ_1, τ_2 are bucket sizes and are for security reasons. Let $N = \tau_1 + \tau_1 \cdot \tau_2^2 \cdot T$. We assume access to the functionalities $\mathcal{F}_{\text{Auth}}$, $\mathcal{F}_{\text{ROT}}^{m,k}$, $\mathcal{F}_{\text{Rand}}$, and $\mathcal{F}_{\text{RanKer}}$, and we call $\Pi_{\text{TripleConstruct}}$ as a subprotocol.

1. **Construction:** Call $\Pi_{\text{TripleConstruct}}$ to produce N multiplication triples.
2. **Cut-and-choose:**
   (a) Call $\mathcal{F}_{\text{Rand}}$ to obtain $(l_1, l_2, \ldots, l_{\tau_1})$, where $l_i \neq l_j$ when $i \neq j$.
   (b) Open $\langle \mathbf{a}_{l_j} \rangle$, $\langle \mathbf{b}_{l_j} \rangle$, and $\langle \mathbf{c}_{l_j} \rangle$ for $j = 1, 2, \ldots, \tau_1$. Abort if $\mathbf{c}_{l_j} \neq \mathbf{a}_{l_j} * \mathbf{b}_{l_j}$ for some j.
3. **Sacrifice:**
   (a) Use $\mathcal{F}_{\text{Rand}}$ to randomly divide the remaining $N - \tau_1$ triples into $\tau_2^2 \cdot T$ buckets with τ_1 triples in each.
   (b) In each bucket we denote the triples by $(\langle \mathbf{a}_l \rangle, \langle \mathbf{b}_l \rangle, \langle \mathbf{c}_l \rangle)$ for $l = 1, \ldots, \tau_1$ and call $\mathcal{F}_{\text{RanKer}}$ to obtain $[r_l]$, $l = 2, \ldots, \tau_1$ for each bucket.
      i. Compute $\langle \boldsymbol{\epsilon}_l \rangle = \langle \mathbf{a}_l \rangle - \langle \mathbf{a}_1 \rangle$ and $\langle \boldsymbol{\delta}_l \rangle = \langle \mathbf{b}_l \rangle - \langle \mathbf{b}_1 \rangle$ and open $\boldsymbol{\epsilon}_l$ and $\boldsymbol{\delta}_l$ for $l = 2, \ldots, \tau_1$.
      ii. Compute $[\sigma_l] = \mathbf{1} * \langle \mathbf{c}_l \rangle - \mathbf{1} * \langle \mathbf{c}_1 \rangle - \boldsymbol{\epsilon}_l * \langle \mathbf{b}_1 \rangle - \boldsymbol{\delta}_l * \langle \mathbf{a}_1 \rangle - \phi(\boldsymbol{\epsilon}_l) \cdot \phi(\boldsymbol{\delta}_l) + [r_l]$ and open σ_l for $l = 2, \ldots, \tau_2$. Abort if $\psi(\sigma_l) \neq \mathbf{0}$. Otherwise, call $(\langle \mathbf{a}_1 \rangle, \langle \mathbf{b}_1 \rangle, \langle \mathbf{c}_1 \rangle)$ a correct triple.
4. **Combine:**
   (a) Combine on $\mathbf{a}$: Use $\mathcal{F}_{\text{Rand}}$ to randomly divide the remaining $\tau_2^2 \cdot T$ non-malformed triples into $\tau_2 \cdot T$ buckets with τ_2 in each. Denote the triples in each bucket by $(\langle \mathbf{a}_l \rangle, \langle \mathbf{b}_l \rangle, \langle \mathbf{c}_l \rangle)$ for $l = 1, \ldots, \tau_2$ and call $\mathcal{F}_{\text{ReEncodePair}}$ to obtain one pair for each bucket. Combine the triples in each bucket as follows:
      i. Compute $\langle \mathbf{a}' \rangle = \sum_{l=1}^{\tau_2} \langle \mathbf{a}_l \rangle$ and $\langle \mathbf{b}' \rangle = \langle \mathbf{b}_1 \rangle$
      ii. For $l = 2, 3, \ldots \tau_2$: Compute $\langle \boldsymbol{\epsilon}_l \rangle = \langle \mathbf{b}_1 \rangle - \langle \mathbf{b}_l \rangle$ and open $\boldsymbol{\epsilon}_l$
      iii. Compute $[\sigma'] = \mathbf{1} * \langle \mathbf{c}_1 \rangle + \sum_{l=2}^{\tau_2} \boldsymbol{\epsilon}_l * \langle \mathbf{a}_l \rangle + \mathbf{1} * \langle \mathbf{c}_l \rangle - [r]$, where $[r]$ is from the reencoding pair.
      iv. Open σ' and set $\langle \mathbf{c}' \rangle = \psi(\sigma') + \langle \psi(r) \rangle = \langle \mathbf{a}' * \mathbf{b}' \rangle$ and call $(\langle \mathbf{a}' \rangle, \langle \mathbf{b}' \rangle, \langle \mathbf{c}' \rangle)$ a good triple.
   (b) Combine on $\mathbf{b}$: Use $\mathcal{F}_{\text{Rand}}$ to randomly divide the remaining $\tau_2 \cdot T$ non-malformed triples into T buckets with τ_2 in each. Denote the triples in each bucket by $(\langle \mathbf{a}_l \rangle, \langle \mathbf{b}_l \rangle, \langle \mathbf{c}_l \rangle)$ for $l = 1, \ldots, \tau_2$ and call $\mathcal{F}_{\text{ReEncodePair}}$ to obtain one pair for each bucket. Combine the triples in each bucket as follows:
      i. Compute $\langle \mathbf{b}' \rangle = \sum_{l=1}^{\tau_2} \langle \mathbf{b}_l \rangle$ and $\langle \mathbf{a}' \rangle = \langle \mathbf{a}_1 \rangle$
      ii. For $l = 2, 3, \ldots \tau_2$: Compute $\langle \boldsymbol{\epsilon}_l \rangle = \langle \mathbf{a}_1 \rangle - \langle \mathbf{a}_l \rangle$ and open $\boldsymbol{\epsilon}_l$
      iii. Compute $[\sigma'] = \mathbf{1} * \langle \mathbf{c}_1 \rangle + \sum_{l=2}^{\tau_2} \boldsymbol{\epsilon}_l * \langle \mathbf{b}_l \rangle + \mathbf{1} * \langle \mathbf{c}_l \rangle - [r]$, where $[r]$ is from the reencoding pair.
      iv. Open σ' and set $\langle \mathbf{c}' \rangle = \psi(\sigma') + \langle \psi(r) \rangle = \langle \mathbf{a}' * \mathbf{b}' \rangle$ and call $(\langle \mathbf{a}' \rangle, \langle \mathbf{b}' \rangle, \langle \mathbf{c}' \rangle)$ a good triple.
   (c) Call $\mathcal{F}_{\text{Auth.Check}}$ on all opened values so far. If the check succeeds output the T good triples.

Fig. D.18: Multiplication triples.

Functionality $\mathcal{F}_{\text{RanKer}}$

This functionality is an extension to $\mathcal{F}_{\text{Prep}}$

1. **RanKer** On input (RanKer,id) sample a random field element $r \in \ker(\psi)$ and set ValField[id] $= r$.

Fig. D.19: Functionality – Authenticated random element in $\ker(\psi)$.

Protocol Π_{RanKer}

The protocol generates $[r_j]$ for $j = 1,2,\ldots,t$, where $[r_j]$ is random in $\ker(\psi)$ and unknown to all parties. We assume access to the functionality $\mathcal{F}_{\text{Rand}}$.

1. **Construct:**
   (a) P_i chooses $r_j^{(i)}$ for $j = 1,2,\ldots,t+s$ uniformly at random in $\ker(\psi)$.
   (b) P_i calls $\mathcal{F}_{\text{Auth.Input}}$ to obtain $[r_j^{(i)}]$.
   (c) Compute $[r_j] = \sum_{i=1}^{n} [r_j^{(i)}]$ for $j = 1,2,\ldots,t+s$.
2. **Sacrifice:**
   (a) Call $\mathcal{F}_{\text{Rand}}(\mathbb{F}_2^t)$ to obtain $\mathbf{a}'_i$ for $i = 1,2,\ldots,s$ and define $\mathbf{a}_i = (\mathbf{a}'_i, \mathbf{e}_i) \in \mathbb{F}_2^{t+s}$ where $\mathbf{e}_i$ is the i'th canonical basis vector of length s.
   (b) Compute $[b_i] = \sum_{j=1}^{t+s} a_{ij}[r_j]$, where a_{ij} is the j'th entry of $\mathbf{a}_i$, and partially open b_i.
   (c) If $b_i \notin \ker(\psi)$ for some $i \in \{1,2,\ldots,s\}$ then abort.
   (d) Call $\mathcal{F}_{\text{Auth.Check}}$ on the opened values b_i.
3. **Output:** Output $[r_j]$ for $j = 1,2,\ldots,t$.

Fig. D.20: Authenticated random element in $\ker(\psi)$.

Proposition 5.7. *Π_{Triple} securely implements $\mathcal{F}_{\text{Prep.Triple}}$ in the $(\mathcal{F}_{\text{Auth}}, \mathcal{F}_{\text{ROT}}^{m,k}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{RanKer}})$-hybrid model.*

The proof uses similar arguments as in [16] and can be found in the full version.

Proposition 5.8. *$\Pi_{\text{InputPair}}$, $\Pi_{\text{ReEncodePair}}$, and Π_{Triple} securely implements $\mathcal{F}_{\text{Prep}}$ in the $(\mathcal{F}_{\text{Auth}}, \mathcal{F}_{\text{ROT}}^{m,k}, \mathcal{F}_{\text{Rand}})$-hybrid model.*

Proof. This follows directly from Propositions 5.3, 5.4, and 5.7. □

Complexity of Preprocessing

We briefly describe the communication complexity for producing the randomness needed for the online phase. Starting by considering the construction of an input pair the only communication we have to consider here is a

single call to $\mathcal{F}_{\text{Auth.Input}}$. The main cost of authentication is the call to Π_{COPEe} where the parties needs to send $mk(n-1)$ bits for each vector authenticated. In the case where a field element is authenticated instead they need to send $m^2(n-1)$ bits. Furthermore, the party who is authenticating needs to send the shares of the vector authenticating but this has only a cost of $k(n-1)$ bits. At last, the check is carried out but we assume that the parties authenticate several vectors/values in a batch and hence this cost is amortized away.

For the re-encoding pairs we assume that t is much larger than s. This means that in order to obtain a single pair the parties need to authenticate n field elements and n vectors. Once again we assume that the check is amortized away, so this gives a total cost of sending $(m^2 + mk)n(n-1)$ bits.

The same assumption, that t is much larger than s, is made for the reorganizing pairs and the random elements in the kernel of ψ. This means that the amortized cost of producing a reorganizing pair is $(l + l')n$ vector-authentications and to obtain $[r]$ for $r \in \ker(\psi)$ costs n authentication amortized.

Regarding the communication for obtaining a single multiplication triple we ignore the vectors sent in the construction since the authentication is much more expensive. Besides authentication we make $\tau_1\tau_2^2 n(n-1)$ calls to $\mathcal{F}_{\text{ROT}}^{k,1}$. We authenticate $3\tau_1\tau_2^2 n$ vectors in the construction. Furthermore, we need $(\tau_2 - 1)\tau_2^2$ elements from $\mathcal{F}_{\text{RanKer}}$ and 2 reencoding pairs for the construction of the triple. The cost of the remaining steps is not close to be as costly, so we ignore these.

In [16] it is suggested to use $\tau_1 = \tau_2 = 3$. The cost of preparing a multiplication gate using these parameters is that of producing 3 reencoding pairs (2 for the preprocessing and 1 for the online phase), 18 authenticated elements in the kernel of ψ and the multiplication triple which yields 27 calls to $\mathcal{F}_{\text{ROT}}^{k,1}$ and $3 \cdot 27$ authentication of vectors. Thus using $m = 3.1k$ from Table D.3 in order to obtain security $s \geq 64$ and ignoring the calls to $\mathcal{F}_{\text{ROT}}^{k,1}$ the communication becomes

$$\begin{aligned} &3 \cdot (3.1^2 + 3.1)k^2 n(n-1) + 18 \cdot 3.1^2 k^2 n(n-1) + 3 \cdot 27 \cdot 3.1 \cdot k^2 n(n-1) \text{ bits} \\ &= 462.21 \cdot k^2 n(n-1) \text{ bits.} \end{aligned}$$

Similarly, in order to obtain $s \geq 128$ we use $m = 3.21k$ from Table D.3 and the communication becomes $486.03 \cdot k^2 n(n-1)$ bits.

References

[1] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology — CRYPTO '91*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 420–432.

[2] Z. Beerliová-Trubíniová and M. Hirt, "Perfectly-secure MPC with linear communication complexity," in *Theory of Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 213–230.

[3] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias, "Semi-homomorphic encryption and multiparty computation," in *Advances in Cryptology – EUROCRYPT 2011*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 169–188.

[4] A. R. Block, H. K. Maji, and H. H. Nguyen, "Secure computation based on leaky correlations: High resilience setting," in *Advances in Cryptology – CRYPTO 2017*. Cham: Springer International Publishing, 2017, pp. 3–32.

[5] A. R. Block, H. K. Maji, and H. H. Nguyen, "Secure computation with constant communication overhead using multiplication embeddings," in *Progress in Cryptology – INDOCRYPT 2018*. Cham: Springer International Publishing, 2018, pp. 375–398.

[6] R. Canetti, "Universally composable security: a new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 10 2001, pp. 136–145.

[7] I. Cascudo, "On squares of cyclic codes," *IEEE Transactions on Information Theory*, vol. 65, no. 2, pp. 1034–1047, 02 2019.

[8] I. Cascudo, R. Cramer, C. Xing, and C. Yuan, "Amortized complexity of information-theoretically secure MPC revisited," in *Advances in Cryptology – CRYPTO 2018*. Cham: Springer International Publishing, 2018, pp. 395–426.

[9] I. Cascudo, I. Damgård, B. David, N. Döttling, R. Dowsley, and I. Giacomelli, "Efficient uc commitment extension with homomorphism for free (and applications)," in *Advances in Cryptology – ASIACRYPT 2019*. Cham: Springer International Publishing, 2019, pp. 606–635.

[10] I. Cascudo, I. Damgård, B. David, N. Döttling, and J. B. Nielsen, "Rate-1, linear time and additively homomorphic uc commitments," in *Advances in Cryptology – CRYPTO 2016*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 179–207.

[11] I. Cascudo and J. S. Gundersen, "A Secret-Sharing Based MPC Protocol for Boolean Circuits with Good Amortized Complexity (full version)," Cryptology ePrint Archive, Report 2020/162, 2020, https://eprint.iacr.org/2020/162.pdf.

[12] I. Cascudo, J. S. Gundersen, and D. Ruano, "Squares of matrix-product codes," *Finite Fields and Their Applications*, vol. 62, p. 101606, 2020.

[13] I. Damgård, R. Lauritsen, and T. Toft, "An empirical study and some improvements of the minimac protocol for secure computation," in *Security and Cryptography for Networks*. Cham: Springer International Publishing, 2014, pp. 398–415.

[14] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology – CRYPTO 2012*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662.

[15] I. B. Damgård and S. Zakarias, "Constant-overhead secure computation of boolean circuits using preprocessing," in *Theory of Cryptography*. Berlin, Heidelberg: Springer, 2013, pp. 621–641.

[16] T. K. Frederiksen, B. Pinkas, and A. Yanai, "Committed mpc," in *Public-Key Cryptography – PKC 2018*. Springer International Publishing, 2018, pp. 587–619.

[17] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, and R. Trifiletti, "On the complexity of additively homomorphic uc commitments," in *Theory of Cryptography*, E. Kushilevitz and T. Malkin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 542–565.

[18] T. K. Frederiksen, M. Keller, E. Orsini, and P. Scholl, "A unified approach to MPC with preprocessing using OT," in *Proceedings, Part I, of the 21st International Conference on Advances in Cryptology – ASIACRYPT 2015 - Volume 9452*. Berlin, Heidelberg: Springer-Verlag, 2015, p. 711–735.

[19] M. Keller, E. Orsini, and P. Scholl, "MASCOT: Faster malicious arithmetic secure computation with oblivious transfer," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 830–842.

[20] E. Larraia, E. Orsini, and N. P. Smart, "Dishonest majority multi-party computation for binary circuits," in *Advances in Cryptology – CRYPTO 2014*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 495–512.

[21] J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra, "A new approach to practical active-secure two-party computation," in *Advances in Cryptology – CRYPTO 2012*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 681–700.

ISSN (online): 2446-1636
ISBN (online): 978-87-7210-826-1

AALBORG UNIVERSITY PRESS