Some Results for Multiparty Computation

Master Thesis

Aalborg University

September 5th 2018

Jaron Skovsted Gundersen

Contents

Opsummering af specialet

Dette speciale er en sammenfatning af nogle af de resultater jeg har opnået gennem mit Ph.d. forløb indtil videre. Derfor vil nogle af resultaterne være udeladt og andre vil blive præsenteret uden bevis, men artiklerne, med beviser og yderligere resultater, er vedhæftet som bilag. Jeg har valgt at præsentere resultaterne ud fra et multiparty computation synspunkt, hvorfor fokus meget ligger på hvilken indflydelse resultaterne har i denne sammenhæng. Multiparty computation er et begreb der dækker det, at flere personer ønsker at evaluere en funktion. Hver enkelt person har et eller flere inputs til funktionen, som de ønsker forbliver ukendte for de andre personer. Ved hjælp af en multiparty computation protokol er det netop muligt for personerne at opnå outputtet af funktionen uden at afsløre deres inputs.

Indtil videre har jeg en artikel udgivet og en anden i peer review. Begge artikler er jeg medforfatter på. Den første, som er i peer review, hedder "Improved Bounds on the Threshold in Ramp Secret Sharing". Denne artikel er i samarbejde med Ignacio Cascudo og Diego Ruano. Den anden, som hedder "Actively Secure OT-Extension from q-ary Linear Codes", er i samarbejde med Ignacio Cascudo og René Bødker Christensen. Jeg vil i dette speciale gennemgå nogle af de vigtigste resultater fra disse to artikler og forklare nogle af de indflydelser resultaterne har på multiparty computation. Derefter vil jeg også kort gennemgå mulige fremtidige projekter, herunder nogle resultater jeg allerede har arbejdet på vedrørende matrix-produkt koder, som er en speciel form for lineære koder.

Artiklen i peer review omhandler secret sharing, hvilket er en måde at dele en hemmelighed blandt nogle personer, sådan at de enkelte intet ved om hemmeligheden, men ved at forene de oplysninger de får, kan de genskabe hemmeligheden. Hvor jeg i artiklen præsenterer flere forskellige begrænsninger for lineær secret sharing, vil jeg i dette speciale have fokus på nogle få resultater fra artiklen og deres indflydelse på multiparty computation. Nogle af hovedresultaterne fra "Improved Bounds on the Threshold in Ramp Secret Sharing" er begrænsninger for privacy og reconstructions thresholds samt threshold gap. Begrænsningerne opnås ved at udtrykke et lineært secret sharing scheme ved

hjælp af to lineære koder, hvilket altid er muligt. Herefter udnyttes det, at disse thresholds kan udtrykkes ved hjælp af de relative generaliserede Hamming vægte for koderne, hvorefter grænser for de relative generaliserede Hamming vægte medfører grænser for thresholds'ne i lineær secret sharing.

Den anden artikel omhandler oblivious transfer, eller nærmere bestemt oblivious transfer extension. I den simpleste form for oblivious transfer bliver der overført én ud af to mulige beskeder fra en sender til en modtager. Modtageren vælger hvilken af de to beskeder han ønsker, mens senderen ikke finder ud af hvad modtageren vælger. Samtidigt lærer modtageren intet om den anden besked. Oblivious transfer extension er en måde, hvorpå man kan simulere mange oblivious transfers ud fra et mindre antal. Vi generaliserer en oblivious transfer extension protokol så man kan benytte lineære koder over et hvilket som helst legeme og ikke blot binære lineære koder. Vi giver eksempler, der illustrerer, at dette medfører, at man kan simulere det samme antal oblivious transfers ved at benytte et mindre antal end tidligere. Der er dog den ulempe, at man i de fleste tilfælde får en større kommunikations kompleksitet i de resterende dele af protokollen. Afvejningen mellem færre oblivious transfers og større kommunikations kompleksitet bliver også illustreret i nogle eksempler.

CHAPTER 1
Introduction

This thesis presents some results related to multiparty computation I have obtained through my Ph.d. study. It contains a summary of results from my articles [6, 8] together with my research plans for the remaining 2 years of my Ph.d. study in accordance to the rules for the qualification exam as part of my 4+4 Ph.d. plan at Aalborg University. Since the first part of this thesis is a summary of the two articles some of the proofs and results are not included. However, the two articles are given as an appendix. A few additional results relating some of the results to multiparty computation which were not included in the submissions are given with some more detail.

Up until now, I have one article published, namely [6], and another in peer review [8]. The paper in peer review, which is joint work with Ignacio Cascudo and Diego Ruano, is about some limitations in secret sharing, a concept which is very used in multiparty computation. The published paper is about oblivious transfer, which is used in many protocols for two-party computation, a special case of multiparty computation. This work is joint with Ignacio Cascudo and René Bødker Christensen.

Multiparty computation was first introduced in the two-party setting in [28] and afterwards generalized to any number of parties in [15]. The concept deals with the situation where some parties each holds an input to a function. The parties would like to learn the output of the function but are not willing to reveal their input to the other parties. A multiparty computation protocol is a solution to that problem. Such a protocol describes how the parties, by communicating with each other, can obtain the output without revealing their inputs if the parties follows the prescribed steps in the protocol.

In order to hide the inputs a multiparty computation protocol often uses some kind of secret sharing scheme, a concept introduced independently in [27] and [2]. Informally, a secret sharing scheme is a way to distribute a secret to some parties such that only some predetermined large subsets of parties are able to recover the secrets while small subsets of parties obtain no information about the secret.

Since many multiparty computation protocols depend on secret sharing schemes, limitations in secret sharing will imply limitations in multiparty computation as well. Such limitations for secret sharing exist already in the literature, see for instance [3, 4, 7]. In [8], we obtain new bounds for secret sharing and show that they improve on previously known bounds. In this thesis, I will also describe some of the implications these improved bounds have for multiparty computation through what is called (strongly) multiplicative secret sharing.

An oblivious transfer, first introduced in [24], is a special case of multiparty computation, where we only consider two parties. These are often called the sender and the receiver. In the simplest case of an oblivious transfer the sender has two inputs m_0 and m_1 and the receiver has a single bit b as input. The oblivious transfer allows the receiver to learn m_b and nothing else. Simultaneously, the sender does not learn anything about b. This concept can of course be generalized such that the sender has N inputs and the receiver learns $K < N$ of these inputs and nothing about the remaining.

Oblivious transfer is in fact a very useful tool for multiparty computation, and is often used in the case where we only consider two parties. Actually, it was shown in [20] that any function can be evaluated securely if one have access to the oblivious transfer functionality. However, the use of oblivious transfer in multiparty computation is not without challenges. Many of the well-known two-party protocols, see for instance [28], use a lot of oblivious transfers as a building block. But as it was shown in [18], oblivious transfer is very likely to require a public key cryptosystem and hence may be expensive to execute. This is why oblivious transfer extension is very interesting.

In an oblivious transfer extension we simulate a lot of oblivious transfers by a much smaller number and by using some cheaper cryptographic tools as well. This is the focus of [6], where we present a new oblivious transfer extension protocol. It generalizes the protocol from [22] by using q-ary linear codes instead of binary. In this thesis, I describe the protocol and point out some of the differences, advantages, and disadvantages of using another alphabet for the linear codes used in the protocol. I also present some specific examples to illustrate these differences between our protocol and the protocol from [22].

At last, I present some potential further works, which includes a section with results on squares of matrix-product codes. I also mention how the square of codes plays a role in the setting of multiparty computation. These results are not yet sent for publication.

Chapter 2
Preliminaries

As mentioned in the introduction, I will have a multiparty computation perspective through this thesis. However, I will not formally define what I mean by privacy and correctness for a multiparty computation protocol but settle for the intuitive explanation below. For precise definitions, and more about multiparty computation in general, I refer the reader to [12]. By correctness we mean that the parties should obtain the correct output of the function given the choice of inputs from the parties. This means that, depending on the assumptions on the adversary, we cannot ensure that the parties choose the right input. By privacy we mean that nothing about the inputs of the parties, beyond what is implied by the output of the function, must be revealed to the other parties. If for example two parties P_1, P_2 would like to compute $x + y$ where P_1 holds x and P_2 holds y. Then by learning $x + y$, P_2 would also know P_1's input, since by subtracting y from the output it obtains x and vice versa. However, this is not a breach of the privacy requirement since this is unavoidable from the fact that they both should learn $x + y$.

The reader is expected to be familiar with linear error-correcting codes, including the Hamming weight and distance. For a given linear code $\mathcal{C}$, I will use the notation $d(\mathcal{C})$ to denote the minimum distance. Additionally, I will use the notation $[n, k, d]_q$ to describe a linear code over $\mathbb{F}_q$ with length n, dimension k, and minimum distance at least d. Linear codes will be a fundamental building block throughout the different parts of this thesis.

Secret Sharing

As described in the introduction, secret sharing is a way to distribute a secret among some parties such that only predetermined large subsets of parties are able to reconstruct the secret. In this section I define this in a formal way by using Shannon's entropy function. This function takes a discrete stochastic variable X taking values in the set A and is given by

$$H_q(X) = -\sum_{x \in A} \Pr[x] \log_q(\Pr[x]),$$

where we say that the function has base q. With the Shannon's entropy function I define a secret sharing scheme.

2.1 Definition (Secret sharing scheme):
Let $S_0, S_1, \ldots, S_n$ be random variables taking values in the finite alphabets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n$ and let H_q be the Shannon entropy function with base q. A secret sharing scheme is given by the vector $(S_0, S_1, \ldots, S_n) \in \mathcal{S}_0 \times \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$, if it satisfies that

$$H_q(S_0) = \log_q |\mathcal{S}_0| \quad \textit{and} \quad H_q(S_0|S_1, S_2, \ldots, S_n) = 0.$$

We will consider S_0 as the secret and S_i for $i \in \{1, 2, \ldots, n\}$ as the share for the i'th party. Therefore, we will call a realization of $(S_1, S_2, \ldots, S_n)$ for a share vector. The Shannon entropy is a measure of uncertainty, and I remark that the requirement that $H_q(S_0) = \log_q |\mathcal{S}_0|$ means that the uncertainty about the secret is maximal. On the other hand, the second requirement means that given all the shares there is no uncertainty about the secret. In other words, the collection of all the shares shall uniquely determine the secret.

Instead of uncertainty one can also use Shannon's entropy to define how much information some subset of parties has. Denote by $\mathbf{S}_A = (S_i)_{i \in A}$ the collection of shares for the subset of parties corresponding to A. Then we define the mutual information as

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

We measure the information in q-bits. We note that if the uncertainty about S_0 given $\mathbf{S}_A$ is low then the information is high and vice versa. Additionally, it can be shown that the information will always be an integer between 0 and ℓ, both included.

The idea in multiparty computation is often that the parties use a secret sharing scheme to secretly distribute their inputs among the other parties. Then by doing calculations on the shares, maybe by interacting with each other, the parties will at the end hold a share in the same secret sharing scheme for the output of the function. If all the parties let the other know their share for the output, they can, by the last requirement in Definition 2.1, find the output from these shares. However, this vague explanation of a multiparty computation protocol indicates some demands or desires for secret sharing schemes. First of all, we need to be able to do the computations on the shares. Since we will assume that the secret and shares lies in finite fields, this means that we will be able to carry out sums and multiplications because every function can be written as a polynomial.

Additionally, we would like a high privacy meaning that if some parties are corrupted, they may not by joining their shares obtain information about the

other parties inputs. However, we would also like a low reconstruction, if for example some parties at some point stop participating in the protocol or is caught in cheating. Then we would like still to be able to reconstruct the output from the shares of the remaining parties.

The reconstruction and privacy for a secret sharing scheme is often measured by considering the reconstruction and privacy thresholds.

2.2 Definition (Privacy and reconstruction):
Let $A \subseteq \{1, 2, \ldots, n\}$ and denote by $\mathbf{S}_A$ the vector $(S_i)_{i \in A}$. The set A is said to be a privacy set if

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

and A is called a reconstruction set if

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

The set of all privacy sets is called the adversary structure, $\mathcal{A}$, and the set of all reconstruction sets is called the access structure, Γ.

We say that the secret sharing scheme has t-privacy if

$$\{A \subseteq \{1, 2, \ldots, n\} : |A| \leq t\} \subseteq \mathcal{A}.$$

The maximal t for which this holds is called the privacy threshold, t.

Similarly, we say that the secret sharing scheme has r-reconstruction if

$$\{A \subseteq \{1, 2, \ldots, n\} : |A| \geq r\} \subseteq \Gamma.$$

The minimal r for which this holds is called the reconstruction threshold, r. The difference between these thresholds $g = r - t$ is called the threshold gap.

One can also define partial privacy and reconstruction thresholds.

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

$$t_i = \max\{s \mid \forall A \subseteq \{1, 2, \ldots, n\}, |A| = s, I_q(S_0, \mathbf{S}_A) < i\}.$$

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

$$r_i = \min\{s \mid \forall A \subseteq \{1, 2, \ldots, n\}, |A| = s, I_q(S_0, \mathbf{S}_A) \geq i\}.$$

Here one should notice that $t_1 = t$ and $r_\ell = r$.

In order to compute a sum of two inputs, or a linear combination, we would like the secret sharing scheme to be linear. This means that a linear combination

of share vectors gives a share vector for the same linear combination of the corresponding secrets.

We can describe a linear secret sharing scheme using linear error-correcting codes. There are several definitions on linear secret sharing using linear codes, but in this thesis, I will use the definition with a nested code pair. Additionally, we restrict ourself to the case where the secret is an element in $\mathbb{F}_q^\ell$ and the shares are elements in $\mathbb{F}_q$.

2.4 Definition (Linear secret sharing scheme):
Let $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$ be two linear codes with dimension k_2 and k_1, respectively. Let $\ell = k_1 - k_2$ and let $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{k_2}, \mathbf{b}_{k_2+1}, \ldots, \mathbf{b}_{k_1}\}$ be a basis for $\mathcal{C}_1$ such that the first k_2 vectors is a basis for $\mathcal{C}_2$. Let $\mathcal{S}_0 = \mathbb{F}_q^\ell$ and let S_0 follow the uniform distribution on $\mathcal{S}_0$. For a realization of S_0, say $(s_1, s_2, \ldots, s_\ell)$, choose $a_1, a_2, \ldots, a_{k_2}$ uniform at random in $\mathbb{F}_q$ and define

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

Then the i'th entry of $\mathbf{c}$ is the realization of S_i, and $\mathbf{c}$ is therefore a share vector.

Clearly, this is a secret sharing scheme since the uniform distribution on the secret implies that the entropy of S_0 is maximal and since we are able to recover the secrets from $\mathbf{c}$ because the set of the $\mathbf{b}_i$'s is a basis for $\mathcal{C}_1$. Additionally, the scheme is also seen to be linear in the sense described above.

Note that the elements in $\mathcal{C}_1$ is the set of all the share vectors, and one should notice that $\mathcal{C}_2$ is the set of all the share vectors which are shares for the all zero secret.

From the nested code pair we can determine the privacy and reconstruction thresholds using the relatively generalized Hamming weights.

2.5 Definition (Relative generalized Hamming weights):
Given a nested code pair $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$ the i'th relative generalized Hamming weight is given by

$$M_i(\mathcal{C}_1, \mathcal{C}_2) = \min\{|\mathrm{supp}(D)| : D \subseteq \mathcal{C}_1, D \cap \mathcal{C}_2 = \{0\}, \dim(D) = i\},$$

where $\mathrm{supp}(D)$ is the set of indices in D which are not always zero.

In [14, 21] it is shown that the partial privacy and reconstruction thresholds t_i and r_i can be described by the relative generalized Hamming weights.

$$\begin{aligned} t_i &= M_i(\mathcal{C}_2^\perp, \mathcal{C}_1^\perp) - 1 \\ r_i &= n - M_{\ell-i+1}(\mathcal{C}_1, \mathcal{C}_2) + 1. \end{aligned} \tag{2.1}$$

It may be noted that the first relative generalized Hamming weight of $\mathcal{C}_2 \subsetneq \mathcal{C}_1$ is simply the minimum Hamming weight of the codewords in the set $\mathcal{C}_1 \setminus \mathcal{C}_2$.

In order to compute products of secrets we would like the secret sharing scheme to be (strongly) multiplicative. In the definition for multiplicative, I will use the notation $*$ to denote the component-wise product of two vectors. Additionally, for a set A I will denote by $\bar{A}$ the complement of A in the set $\{1, 2, \ldots, n\}$ and for a vector $\mathbf{c}$ and a set of indices A I use the notation $(\mathbf{c})_A$ to denote the vector of length $|A|$ with the entries in $\mathbf{c}$ corresponding to the indices in A.

2.6 Definition (Multiplicative and strongly multiplicative):
A linear secret sharing scheme is called multiplicative if there exists a linear reconstruction function $\varphi\colon \mathbb{F}_q^n \to \mathbb{F}_q^\ell$, such that for any two secrets $\mathbf{s}, \mathbf{t}$ with corresponding share vectors $\mathbf{c}, \mathbf{d}$, we have

$$\varphi(\mathbf{c} * \mathbf{d}) = \mathbf{s} * \mathbf{t}.$$

The scheme is called strongly multiplicative if there for any $A \in \mathcal{A}$ exists a linear reconstruction function $\varphi_A\colon \mathbb{F}_q^{|\bar{A}|} \to \mathbb{F}_q^\ell$, such that

$$\varphi_A((\mathbf{c} * \mathbf{d})_{\bar{A}}) = \mathbf{s} * \mathbf{t}.$$

The reason for considering strongly multiplicative is again if some of the parties stop cooperating or is excluded from the calculations since they were caught in cheating, then the remaining parties should still be able to obtain the output.

Oblivious Transfer

An oblivious transfer (OT) is a very useful cryptographic primitive. In general, for an $\binom{N}{K}$-OT where $K < N$, a sender inputs $m_1, m_2, \ldots, m_N$ and a receiver inputs $w_1, w_2, \ldots, w_K$, where $w_i \in \{1, 2, \ldots, N\}$ and $w_i \neq w_j$, when $i \neq j$. The receiver should learn m_{w_i} for $i = 1, 2, \ldots, K$, without the sender knowing which of the inputs the receiver obtained and without the receiver learning anything about the remaining inputs for the sender.

The $\binom{2}{1}$-OT functionality, where the sender has m_0 and m_1 as inputs and R has a single input $w \in \{0, 1\}$ is illustrated in Figure 2.1.

In [6] the focus is on OT-extension. An OT-extension is a way to simulate a large number of OT's, using a much smaller number of OT's called base OT's. In [6] we present a generalization of some previously known protocols which achieve this. Our generalization relies on using q-ary codes instead of binary codes. Since we will be working with OT-extension protocols I introduce a notation for the functionality of doing m $\binom{N}{1}$-OT's where the sender's input has string length κ.

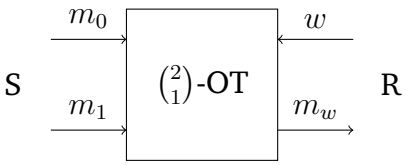

Figure 2.1. The $\binom{2}{1}$-OT

For this functionality I will use the notation $\mathcal{F}_{N\text{-OT}}^{\kappa,m}$. With this notation one can say that an OT-extension is implementing the functionality $\mathcal{F}_{N\text{-OT}}^{\kappa,m}$ using $\mathcal{F}_{N'\text{-OT}}^{\kappa',m'}$, where $m' < m$.

In the protocol from [6] we make use of q-ary linear codes and two functions, a pseudorandom generator PRG and some kind of hash function H. The reason for using the PRG is to reduce the complexity of the protocol since we only input a seed to the OT and afterwards extract this seed using PRG to obtain something which seems uniform in a larger space. This follows from the following definition.

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

The idea of using the hash function H is also that it outputs something computationally indistinguishable from the uniform distribution, and the sender can then use the outputs of this function to mask its inputs. In [6] we use some specific hash function with certain properties but I refer the reader to the article for the definition.

CHAPTER 3

Bounds on Thresholds and Limitations for Multiparty Computation

In this chapter, I will obtain bounds on the privacy and reconstruction thresholds along with the threshold gap by applying the generalized Griesmer bound for the first relative generalized Hamming weight. The bounds I present is from our article [8], where bounds on partial reconstruction and privacy thresholds where obtained by considering the i'th relative generalized Hamming weight as well. The bounds on the partial thresholds are presented without proofs in the second section of this chapter. I have in this thesis chosen to focus on the bounds on t, r and g for two reasons. The first since they are the most used and well-known in the secret sharing community and the second since this is what we need to study the limitations for multiparty computation.

I emphasize that the first two sections in this chapter presents some of the main results from my article [8] while the third section study some implications the results from the first sections have for multiparty computation. These implications are not discussed in [8].

Bounds on Thresholds

Bounds on the threshold gap have already been studied in the literature, both for linear but also for general secret sharing. In linear secret sharing the most well-known bound is given by $g \geq \ell$ which I refer to as the Singleton bound. This bound only consider the size of the secret compared to the size of the shares. Other bounds includes both the number of parties and the share size. Such bounds were given in [7]. Let

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

Then the bounds from [7] state that

$$\begin{aligned} g &\geq \mathrm{B}_{\mathrm{CCX}(1)}(n,q) && \text{if } 1 \leq t < r \leq n-1 \\ g &\geq \mathrm{B}_{\mathrm{CCX}(2)}(\mathrm{n},\mathrm{q},\ell) && \text{if } \ell \geq 2. \end{aligned} \tag{3.1}$$

I refer to these bounds as the first and second CCX bound. Note that the first bound do not include ℓ and therefore not depend on the relation between the secret and share size, while the second CCX bound include all these parameters.

One of the main results in [8] is a bound which, as the second CCX bound, includes all these parameters. The way we obtain this new bound is by considering the generalized Griesmer bound on the first relative generalized Hamming weight for the codes used in the construction.

Hence, I start by presenting the generalized Griesmer bound, a result from [29]. I will through this chapter use the notation k_1 and k_2 for the dimension of the codes $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

3.1 Proposition:
Let $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$ be a nested code pair and let $\ell = k_1 - k_2$. For $0 \leq i \leq \ell$, the i'th relative generalized Hamming weight satisfies

$$n \geq k_2 + M_i(\mathcal{C}_1, \mathcal{C}_2) + \sum_{j=1}^{\ell-i} \left\lceil \frac{q-1}{q^j(q^i-1)} M_i(\mathcal{C}_1, \mathcal{C}_2) \right\rceil. \tag{3.2}$$

This gives the following bounds, first shown in [8, Theorem 3.2].

3.2 Theorem:
Let a linear secret sharing scheme be given by a nested code pair $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$. Then for every $m \in \{0, 1, \ldots \ell - 1\}$ we have the following bounds

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

Proof:
First note that all the terms in the sum from (3.2) are positive, meaning that they are at least 1. Since we want bounds on t and r, we will because of (2.1) consider the first relative generalized Hamming weights. Hence, for an $m \in \{0, 1, \ldots \ell - 1\}$

we can write

$$n \geq k_2 + M_1(\mathcal{C}_1, \mathcal{C}_2) + M_1(\mathcal{C}_1, \mathcal{C}_2) \sum_{j=1}^{m} \frac{1}{q^j} + \ell - 1 - m \iff$$
$$M_1(\mathcal{C}_1, \mathcal{C}_2) \leq \frac{q^{m+1} - q^m}{q^{m+1} - 1}(n - k_1 + m + 1).$$

Now the results follows by combining this result with the expressions for t and r in (2.1). Here one should notice that $\dim(\mathcal{C}_2^{\perp}) = n - k_2$. The result on g simply comes from combining the results on t and r. ■

We showed in [8, Theorem 3.3] that there exists an m such that the bounds in Theorem 3.2 is at least as good as previously known bounds for the thresholds as long $\ell \geq 2$. Denoting by

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

[8, Theorem 3.3] states the following.

3.3 Theorem:
Let $\ell \geq 2$, then

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

and

$$\mathrm{B_{Gr}}^{(0)}(n, q, \ell) \geq \mathrm{B_{CCX(2)}}(n, q, \ell), \text{ when } \ell \geq n - 2(q - 1)$$
$$\mathrm{B_{Gr}}^{(1)}(n, q, \ell) \geq \mathrm{B_{CCX(2)}}(n, q, \ell), \text{ when } \ell \leq n - 2(q - 1).$$

It might also be noted that $\mathrm{B_{Gr}}^{(0)}(n, q, \ell) = \ell$ and therefore we have that the bound on g in Theorem 3.2 is at least as good as the Singleton bound in any cases. Additionally, we note that there exists a bound on t, r and g for every m, and in order to obtain the best bound one needs to choose the m carefully. However, as we see above choosing $m = 0$ or $m = 1$, which is possible when $\ell = 2$, is enough to obtain bounds which is at least as good as previously known bounds.

Bounds on Partial Thresholds

As mentioned before we also present bounds on the partial privacy and reconstruction thresholds in [8]. Because this is not the focus in this thesis I will in this section briefly present some of the main results regarding these thresholds and refer the reader to [8] for the proofs.

The first result is actually a generalization of Theorem 3.2, but I have chosen to state the theorem for t and r since this is my focus in this thesis. However, we proof similar bounds for t_i and r_i, where we show that

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

for all $m \in \{0, 1, \ldots, \ell - i\}$. Recalling that $t_1 = t$ and $r_\ell = r$ we obtain Theorem 3.2.

Additionally, [8, Theorem 4.2] states that if $t_j \geq j$ for some j then the threshold r_i satisfy

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

for $i \in \{j, j+1, \ldots, \ell\}$. Here one might note the connection to [7], where it is shown that $r \geq \frac{n}{q} + 1$ under the assumption that $t \geq 1$. So what we have done in [8] is loosening the assumption ($t \geq 1$ implies $t_j \geq j$ for every j) and obtaining a statement not only for r but for some of the partial reconstruction thresholds as well. In [7], they use the bound $r \geq \frac{n}{q} + 1$ to obtain the first CCX bound. We have followed more or less the same approach and generalized the first CCX bound. The generalization is shown in [8, Theorem 4.4]. Letting

$$a_i = t_i - t - i + 1$$
$$b_i = r - r_{\ell-i+1} - i + 1,$$

the theorem is as follows.

3.4 Theorem:
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 above. If $t_i \geq i$, then the threshold gap g satisfies

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

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

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

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

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

One should notice that since a_i and b_i are nonnegative (because t_i and r_i are strictly increasing with i) this bound is at least as good as the first CCX bound.

Implications for Multiparty Computation

In this section I consider some limitations Theorem 3.2 implies on multiparty computation. I emphasize that these considerations are not a part of [8].

Having lower bounds on g gives some limitations for (strongly) multiplicative secret sharing schemes and hence multiparty computation. Before going into the limitations I start by considering multiplicative secret sharing with respect to the codes. Recall, that we fix a basis for $\mathcal{C}_1$, such that the first k_2 elements is a basis for $\mathcal{C}_2$. This means that we can write $\mathcal{C}_1 = \mathcal{C}_2 \oplus L$ for some subspace L of dimension ℓ. Assume that we fix the last ℓ basis elements such that $\mathbf{b}_{k_2+i} = (\mathbf{e}_i, \mathbf{v}_i)$, where $\mathbf{e}_i$ is the i'th canonical basis vector for $\mathbb{F}_q^\ell$ and $\mathbf{v}_i$ is some vector in $\mathbb{F}_q^{n-\ell}$. This can always be obtained by doing linear combinations on the basis vectors in L or by reordering the parties. Hence, I will in the remaining of this thesis always assume that the basis has this form. Now I define

$$\begin{aligned} \hat{\mathcal{C}}_1 &= \operatorname{span}(\{\mathbf{b}_i * \mathbf{b}_j : 1 \leq i, j \leq k_1\}) \\ \hat{\mathcal{C}}_2 &= \operatorname{span}(\{\mathbf{b}_i * \mathbf{b}_j : 1 \leq i, j \leq k_1\} \setminus \{\mathbf{b}_i * \mathbf{b}_i : 1 \leq i \leq \ell\}) \\ \hat{L} &= \operatorname{span}(\{\mathbf{b}_i * \mathbf{b}_i : 1 \leq i \leq \ell\}). \end{aligned} \tag{3.3}$$

Note that the requirement on the basis before implies that $\dim(\hat{L}) = \ell$. If these codes define a linear secret sharing scheme, the scheme based on $\mathcal{C}_2 \subsetneq \mathcal{C}_1$ is a multiplicative secret sharing scheme. This follows from the following proposition, which to my knowledge is an unpublished result.

3.5 Proposition:
Let $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$ define a linear secret sharing scheme. The scheme is multiplicative if and only if $\hat{\mathcal{C}}_1 = \hat{\mathcal{C}}_2 \oplus \hat{L}$.

Proof:
Clearly, from (3.3), we have $\hat{\mathcal{C}}_1 = \hat{\mathcal{C}}_2 + \hat{L}$, so what might go wrong is that $\hat{\mathcal{C}}_2 \cap \hat{L} \neq \{\mathbf{0}\}$. Assume this is the case. Elements in $\hat{\mathcal{C}}_2$ must be shares for $\mathbf{0}$. However, the only share for $\mathbf{0}$ in $\hat{L}$ is the all zero vector. This means that there exists a $\mathbf{c} \neq \mathbf{0}$ in the intersection which is a share vector for both a nonzero and the all zero secret. Hence, we cannot define a reconstruction function.

On the other hand, assume that $\hat{\mathcal{C}}_1 = \hat{\mathcal{C}}_2 \oplus \hat{L}$. Let $\mathbf{c}, \mathbf{c}', \mathbf{d}, \mathbf{d}'$ be shares for $\mathbf{s}, \mathbf{s}', \mathbf{t}, \mathbf{t}'$ respectively. By the assumption, there are unique representations for $\mathbf{c} * \mathbf{c}'$ and $\mathbf{d} * \mathbf{d}'$ in the following way

$$\begin{aligned} \mathbf{c} * \mathbf{c}' &= s_1 s_1'(\mathbf{b}_1 * \mathbf{b}_1) + s_2 s_2'(\mathbf{b}_2 * \mathbf{b}_2) + \cdots + s_\ell s_\ell'(\mathbf{b}_\ell * \mathbf{b}_\ell) + \mathbf{v} \\ \mathbf{d} * \mathbf{d}' &= t_1 t_1'(\mathbf{b}_1 * \mathbf{b}_1) + t_2 t_2'(\mathbf{b}_2 * \mathbf{b}_2) + \cdots + t_\ell t_\ell'(\mathbf{b}_\ell * \mathbf{b}_\ell) + \mathbf{w}, \end{aligned}$$

for some $\mathbf{v}, \mathbf{w} \in \hat{\mathcal{C}}_2$. Now define the function $\varphi \colon \hat{\mathcal{C}}_1 \to \mathbb{F}_q^\ell$ given by

$$\varphi\left(a_1(\mathbf{b}_1 * \mathbf{b}_1) + a_2(\mathbf{b}_2 * \mathbf{b}_2) + \cdots + a_\ell'(\mathbf{b}_\ell * \mathbf{b}_\ell) + \mathbf{u})\right) = (a_1, a_2, \ldots, a_\ell),$$

where $\mathbf{u} \in \hat{\mathcal{C}}_2$. This is a well-defined map, since all elements in $\hat{\mathcal{C}}_1$ can be uniquely represented on the form above. Clearly, this function is a reconstruction function if it is linear. But this holds since for some $a, b \in \mathbb{F}_q$ we have

$$\begin{aligned}\varphi(a(\mathbf{c} * \mathbf{c}') + b(\mathbf{d} * \mathbf{d}')) &= (as_1s_1' + bt_1t_1', as_2s_2' + bt_2t_2', \ldots, as_\ell s_\ell' + bt_\ell t_\ell') \\ &= a(s_1s_1, s_2s_2', \ldots s_\ell s_\ell') + b(t_1t_1', t_2t_2', \ldots t_\ell t_\ell') \\ &= a\varphi(\mathbf{c} * \mathbf{c}') + b\varphi(\mathbf{d} * \mathbf{d}').\end{aligned}$$

■

I can now deduce the limitations from the following theorem, which is more or less a special case of [7, Theorem 5.4] but in the setting using nested codes as definition for linear secret sharing.

3.6 Theorem:
Let $\mathcal{C}_2 \subsetneq \mathcal{C}_1$ be a multiplicative secret sharing scheme with privacy parameter t and reconstruction parameter r, and let $\hat{r}$ be the reconstruction parameter for the scheme $\hat{\mathcal{C}}_2 \subsetneq \hat{\mathcal{C}}_1$ defined in (3.3)*. Then,*

$$r \leq \hat{r} - t.$$

Proof:
Take a set of parties A with cardinality $\hat{r} - t$. Let $\pi_A \colon \mathbb{F}_q^n \to \mathbb{F}_q^A$ be the projection map, meaning that for a vector $\mathbf{c} \in \mathbb{F}_q^n$, we have $\pi_A(\mathbf{c}) = (\mathbf{c})_A$. I will show that A is a reconstructing set by showing that $\ker \pi_A \cap \mathcal{C}_1 \subseteq \mathcal{C}_2$. This is equivalent to being a reconstructing set, since if a codeword $\mathbf{c} \in \mathcal{C}_1$ is such that $\pi_A(\mathbf{c}) = \mathbf{0}$ then the inclusion implies that it is a share vector for the all zero secret. By linearity, we obtain that if two codewords $\mathbf{c}, \mathbf{c}' \in \mathcal{C}_1$ are such that $\pi_A(\mathbf{c}) = \pi_A(\mathbf{c}')$, then $\mathbf{c} - \mathbf{c}' \in \mathcal{C}_2$ and hence they are share vectors for the same secret. This means that the parties in A are able to reconstruct to a unique secret from their shares.

Let $\mathbf{c} \in \ker \pi_A \cap \mathcal{C}_1$ and let $B \subseteq \bar{A}$ be a set with cardinality t, which always is possible since $|\bar{A}| = n - (\hat{r} - t) \geq t$ because $\hat{r} \leq n$. Since B is a privacy set, there exists a share vector $\mathbf{c}' \in \mathcal{C}_1$ for the all one secret such that $\pi_B(\mathbf{c}') = \mathbf{0}$. Now define $D = A \cup B$ and $\tilde{\mathbf{c}} = \mathbf{c}' * \mathbf{c}$. Since $\mathbf{c}'$ is a share vector for the all one secret $\tilde{\mathbf{c}}$ is a share vector for the same secret as $\mathbf{c}$ but in the scheme $\hat{\mathcal{C}}_2 \subsetneq \hat{\mathcal{C}}_1$. Notice that $|D| = |A| + |B| = \hat{r} - t + t = \hat{r}$ meaning that D is a reconstructing set in this scheme. But $\pi_D(\tilde{\mathbf{c}}) = \mathbf{0}$ by the choice of $\mathbf{c}$ and $\mathbf{c}'$. Hence, the secret of both $\tilde{\mathbf{c}}$ and $\mathbf{c}$ is the zero vector, meaning that $\mathbf{c} \in \mathcal{C}_2$. ■

For a linear secret sharing scheme to be multiplicative we require that $\hat{r} \leq n$ since we require that the parties are able to reconstruct products. Hence, from this theorem, we obtain that $r \leq n - t$ in order for the scheme to be multiplicative. This can be rewritten as $g \leq n - 2t$. Similarly, for a scheme to be strongly multiplicative,

we require that $\hat{r} \leq n - t$. This follows since the complementary of every privacy set, should be able to reconstruct products. Again this leads to $r \leq n - 2t$, which is equivalent to $g \leq n - 3t$.

Now consider the Singleton bound $g \geq \ell$ along with the two CCX bounds from (3.1). From these, we obtain the following bounds on t if we require that the scheme is multiplicative:

$$\begin{aligned} t &\leq \frac{n-\ell}{2} \\ t &\leq \frac{q-1}{2q-1}n - \frac{1}{2q-1} \\ t &\leq \frac{q}{2q+1}(n-\ell) + \frac{q-1}{2q+1}. \end{aligned}$$

Similarly, we obtain the following bounds on t, if we require that the scheme should be strongly multiplicative:

$$\begin{aligned} t &\leq \frac{n-\ell}{3} \\ t &\leq \frac{2}{3}\left(\frac{q-1}{2q-1}n - \frac{1}{2q-1}\right) \\ t &\leq \frac{2}{3}\left(\frac{q}{2q+1}(n-\ell) + \frac{q-1}{2q+1}\right). \end{aligned}$$

Now returning to the bound on g in Theorem 3.2 we obtain that

$$t \leq \frac{(q^{m+1}-q^m)(n-\ell+2m)}{2(q^{m+1}-1)} - \frac{q^m-1}{q^{m+1}-1},$$

if the scheme is multiplicative. Similarly, we obtain that

$$t \leq \frac{(q^{m+1}-q^m)(n-\ell+2m)}{3(q^{m+1}-1)} - \frac{2}{3}\left(\frac{q^m-1}{q^{m+1}-1}\right),$$

if the scheme is strongly multiplicative. To illustrate the difference between the bounds we consider an example below.

3.7 Example:
Let $q = 2$, $n = 100$ and $\ell = 10$. Then the Singleton bound on the threshold gap implies that $t \leq 45$ for the scheme to be multiplicative. Similarly, we obtain that $t \leq 33$ from the first CCX bound and $t \leq 36$ from the second CCX bound. If we consider the bound obtained using Theorem 3.2 and let $m = 4$, which gives the tightest bound in this example, we obtain that $t \leq 24$ if the scheme should be multiplicative.

If we instead considered strongly multiplicative secret sharing schemes, we would obtain the limitations $t \leq 30$ (Singleton), $t \leq 22$ (1st CCX), $t \leq 24$ (2nd CCX), and $t \leq 16$ (Theorem 3.2). ◀

We see in this example that we have a stronger limitation on t using the bound obtained from Theorem 3.2 than the other bounds. This implies that if we want a scheme to be (strongly) multiplicative, as we would like for the schemes we use in multiparty computation in order to be able to multiply inputs, we would require a lower privacy for the underlying secret sharing scheme.

In general, if $\ell \geq 2$, it would be the case that we obtain a stronger limitation on t from this new bound on the threshold gap instead of the other bounds since the bound on g in Theorem 3.2 improves the other bounds in these cases as we saw in Theorem 3.3.

Chapter 4
Oblivious Transfer Extension

Many multiparty computation protocols use oblivious transfers as a building block, especially two-party computation protocols. One of the most well-known protocol for two-party computation is based on garbled circuits, see for instance [28]. The idea of protocols using garbled circuits is that one of the parties is garbling (some kind of encrypting) the function the two parties would like to evaluate. Then it sends the garbled function along with its garbled inputs. The other party will receive its garbled inputs through some oblivious transfers and afterwards evaluate the garbled function on the garbled inputs. Afterwards, they can decrypt the garbled output.

If each party has a lot of inputs, they will need to perform many OT's, which means that the complexity of the protocol will be quite large. This is an example, where OT-extension could be very useful, since we could reduce the complexity of the protocol by simulating a large number of OT's by a much smaller one.

The Protocol

In [6] we present a new protocol for OT-extension, which generalizes the protocol from [22]. We elaborate on the connection with homomorphic commitments, also pointed out in [22], by presenting the protocol in a similar manner as in [9] with matrix notation through the most of the protocol. In the following, we present the generalization and prove the correctness of it. However, we would not in this thesis prove the security of the protocol but only give a sketch of the ideas. We refer to [6] for a complete proof of security.

Before initialising the protocol the sender and receiver should agree on a linear $[n, k, d]_q$ code. The choice of the code influences the protocol in various ways. First assume that we would like to implement the functionality $\mathcal{F}_{N\text{-OT}}^{\kappa,m}$. Recall that this functionality means m $\binom{N}{1}$-OT's with string length κ. Then, we require that $k = \log_q(N)$. Additionally, n would be the actually number of OT's we are performing and d would influence the security of the protocol. The protocol we present will then implement the functionality $\mathcal{F}_{N\text{-OT}}^{\kappa,m}$ having access to $\mathcal{F}_{2\text{-OT}}^{\kappa,n}$. We

notice that in the protocol the receiver has vectors as inputs and not numbers in $\{0, 1, \ldots, N-1\}$. However, the vectors are in $\mathbb{F}_q^k$ and hence there are $q^k = N$ possible inputs for the receiver. In this way, we simply identify the integers with a vector in $\mathbb{F}_q^k$, for instance by using the q-ary representation. In the protocol we will also use the notation $\Delta_{\mathbf{b}}$ to denote the diagonal matrix with the i'th entry of $\mathbf{b}$ on the (i,i)'th position. The full description of the protocol can be seen in Protocol 1 on page 21.

To argue that the protocol is correct and get a feeling of that the protocol is secure we consider

$$\mathbf{q}_i - \mathbf{w}G\Delta_{\mathbf{b}} = \mathbf{t}_i + \mathbf{c}_i\Delta_{\mathbf{b}} - \mathbf{w}G\Delta_{\mathbf{b}}, \tag{4.1}$$

where $\mathbf{c}_i$ is the i'th row of C. If the receiver has acted honestly during the protocol $\mathbf{c}_i = \mathbf{w}_i G$, and hence (4.1) reduces to $\mathbf{t}_i$ in the case where $\mathbf{w} = \mathbf{w}_i$. Because the receiver knows $\mathbf{t}_i$ it can compute $\mathsf{H}(\mathbf{t}_i)$ and obtain $\mathbf{v}_{\mathbf{w}_i,i}$. Again, if the receiver has followed the protocol we have that (4.1) for $\mathbf{w} \neq \mathbf{w}_i$ reduces to

$$\mathbf{t}_i + (\mathbf{w}_i - \mathbf{w})G\Delta_{\mathbf{b}}.$$

Since the receiver does not know $\mathbf{b}$ it has to guess the entries in $\mathbf{b}$ corresponding to the nonzero entries in the codeword $(\mathbf{w}_i - \mathbf{w})G$ before it will know the input to H. If the minimum distance of the code is high, this is a difficult task.

Additionally, we see that the consistency check will always pass if both parties follows the instructions. This follows from the fact that

$$MQ = MT_0 + MC\Delta_{\mathbf{b}} = \tilde{T} + \tilde{W}G\Delta_{\mathbf{b}},$$

where the last equality follows since $MC = MWG = \tilde{M}G$ if the receiver is honest.

The reason for adding the consistency check is to ensure that the receiver is actually choosing codewords in the matrix C. Otherwise, he will get caught during the check and the sender will abort with a very high probability. To see the proof of this I refer the reader to [6]. What might go wrong if the receiver do not choose codewords is that it might potentially learn something about $\mathbf{b}$, which might lead to breach of security. For more detail about this, see [19].

Comparison

In this section I compare Protocol 1 with the OT-extension protocol from [22] which uses binary codes.

First of all we see that Protocol 1 has more flexibility on N, since it was required that N was a power of two before. In this protocol, we can choose N to be any power of a prime. This might be beneficial for some applications where a specific

Protocol 1 (OT-Extension)

This protocol implements the functionality $\mathcal{F}_{N\text{-OT}}^{\kappa,m}$ having access to $\mathcal{F}_{2\text{-OT}}^{\kappa,n}$. 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 hash function $\mathsf{H}\colon \mathbb{F}_q^n \to \{0,1\}^\kappa$, see [6] for a definition. 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 $\mathbf{w} \in \mathbb{F}_q^k$.

1. **Initialization phase**
    1. S chooses uniformly at random $\mathbf{b} \in \{0,1\}^n$.
    2. 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 $i = 0, 1$, where $\mathsf{PRG}(N_i)$ is the matrix we obtain by using PRG on the columns in N_i.
    3. The parties call the functionality $\mathcal{F}_{2\text{-OT}}^{\kappa,n}$, where S acts as the receiver with input the i'th entry of $\mathbf{b}$ for the i'th OT. R acts as the sender with the i'th columns of N_0 and N_1 as input for the i'th OT. S receives $N = N_0 + (N_1 - N_0)\Delta_{\mathbf{b}}$, and by using PRG on the columns he had received, he can compute $T = T_0 + (T_1 - T_0)\Delta_{\mathbf{b}}$.
2. **Encoding phase**
    1. 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 $W = [W' \mid W'']^T$.
    2. R sets $C = WG$, and sends $U = C + T_0 - T_1$.
    3. S computes $Q = T + U\Delta_{\mathbf{b}}$. This implies that $Q = T_0 + C\Delta_{\mathbf{b}}$.
3. **Consistency check**
    1. S samples a uniformly random matrix $M' \in \mathbb{F}_q^{2s\times m}$ and sends this to R. They both define $M = [M' \mid I_{2s}]$.
    2. 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.
    3. S verifies that $MQ = \tilde{T} + \tilde{W}G\Delta_{\mathbf{b}}$. If this fails, S aborts the protocol.
4. **Output phase**
    1. 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 $\mathbf{v}_{\mathbf{w}_i,i} = \mathbf{y}_{\mathbf{w}_i,i} \oplus \mathsf{H}(\mathbf{t}_i)$.

N is wanted. We note that one always can choose a dimension such that 2^k is larger than the wanted N, so we do not obtain more possibilities in this case. However, it might be computationally inconvenient to choose an N almost double size of what is needed. As an example, if one would like N around 600, one needs to choose

a binary code of dimension 10 and get an $N = 1024$. On the other hand one can now choose a ternary code with dimension 6 to obtain $N = 729$, which is much closer to 600.

With respect to communication complexity there is a trade-off when we compare our protocol with the protocol from [22]. The main difference between the two protocols is that we use an $[n, k, d]_q$ code, while they use an $[n', k', d]_2$ code. In order to be fair in our comparison we would like to do 1-out-of-N OT's in both protocols. Hence, we limit our analysis to the case where $q = 2^r$ and $k' = kr$ for some integer r. Since we are considering codes over a larger alphabet and with $k \leq k'$ we can assume that $n \leq n'$, and typically, as we see later, these two conditions will often imply $n < n'$. This is also indicated by, for instance, the Singleton bound

$$n \geq k + d - 1.$$

We have that $k = log_q(N) < \log_2(N) = k'$, if $q > 2$. Hence, the requirement on n is bounded by $n \geq \log_q(N) + d - 1$, while $n' \geq \log_2(N) + d - 1$. This is of course only bounds on n and do not show that $n < n'$, but in the examples we show later, we see that this is often the case. As another illustration of that $n < n'$ in many cases, assume that $k' \geq 2$. A small argument will show that $n' \geq 3d/2$. However, by choosing $q = N$ we obtain that $k = 1$ implying that we can choose $n = d$ by using the repetition code. This is of course the extreme case and in the following we are more focussed on intermediate codes, meaning that $k > 1$ as well.

With the intuition of the connection of the codes and the notation fixed I can now turn the attention to the actually complexity comparison of the two protocols. The output phase for the two protocols has the same communication complexity, but while we obtain a decrease in the complexity in the initialization phase, due to the lowered number of OT's, we might see an increase of the complexity in the remaining phases.

In this thesis I will not go into details about the complexity but just mention that one often make use of OT-extension when m is very large. From this we argue in [6] that the dominant term for the communication complexity in the encoding phase and consistency check comes from sending U under the encoding phase. The dominant term here has a cost of sending $mn \log_2(q)$ bits. Comparing to the protocol from [22], where sending U has a dominant term of mn', we see that this is an increase in communication complexity of a factor $\log_2(q)n/n'$.

I have illustrated the trade-off for some specific codes in Table 4.1. Under the comparison column it is the value of $\log_2(q)n/n'$ I have presented under CC, in order to illustrate the increase in communication complexity for the parts of the protocol discussed before. Under n it is the fraction n'/n I have presented in order to illustrate the reduction in number of base OT's.

Code	N	n (Base OT's)	d	Comparison	
				n	CC
Repeated Golay code [22]	4096	384	128		
$\mathbb{F}_4$-code from table in [16]	4096	177	128	$\div\ 2.17$	$\times\ 0.92$
Punct. Walsh-Had. [22]	512	256	128		
Repeated simplex code over $\mathbb{F}_8$	512	146	128	$\div\ 1.75$	$\times\ 1.71$
$[511, 76, \geq 171]_2$-BCH [22]	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 [22]	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 4.1. 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.

In [22] they suggests three type of codes and list example of their parameters in a table. The three types of codes are punctured Walsh-Hadamard codes, repeated Golay codes and BCH codes.

To compare with the repeated Golay code, I have used the table from [16] and found a code with same distance and N over $\mathbb{F}_4$. It might be noticed here that the repeated Golay code is not necessarily the most optimal code, meaning that there might be a binary code having the same minimum distance and dimension but lower length.

To compare with the punctured Walsh-Hadamard code I take a repeated simplex code over $\mathbb{F}_8$. A simplex code has paramters $[\frac{q^k-1}{q-1}, k, q^{k-1}]_q$, so for $q = 8$ and $k = 3$ we obtain an $[73, 3, 64]_8$ code. By repeating it, we double the length and minimum distance while k remains unchanged. This is the parameters we have in Table 4.1.

For the BCH codes it was more difficult to find comparable codes. So, as we do in [8], we take some other BCH codes and ensured that both N and d is at least as high for the q-ary codes than the binary codes considered in [22]. Even though this is a disadvantage for our protocol we see in the table that in one of the examples we obtain a significantly decrease in the number of OT's while the communication cost for the remaining parts do not increase a lot.

CHAPTER 5
Further Work

In this chapter I describe some potential future work. As a first possible future outcome one could consider if some of the bounds from [8] could be generalized so they not only can be used in the linear case. This was for instance the case for the first CCX bound and the Singleton bound.

As another possible future outcome I am at the moment working on squares of matrix-product codes, which are relevant in the multiparty computation setting too. I will present some of the results I have obtained on this topic so far, and afterwards discuss some new research topics this gives rise to.

Squares of Matrix-product Codes

Consider the explanation of a multiparty computation protocol in Chapter 2 and the definition of (strongly) multiplicative secret sharing schemes, see Definition 2.6. From this, we see that when computing products of secrets the parties will compute the product of their shares. What the parties obtain is then a codeword in the squared code. More generally, we define the product of two linear codes $\mathcal{C}_1$ and $\mathcal{C}_2$ to be

$$\mathcal{C}_1 * \mathcal{C}_2 = \text{span}\{\mathbf{c}_1 * \mathbf{c}_2 \mid \mathbf{c}_i \in \mathcal{C}_i, i = 1, 2\},$$

where $*$ denotes the component-wise product of vectors. For a linear code $\mathcal{C}$, we denote by $\mathcal{C}^{*2} = \mathcal{C} * \mathcal{C}$ and call it the square of $\mathcal{C}$.

The parameters of the square are important for multiparty computation in several settings in addition to the explanation above. For example the MiniMac construction from [13] (a multiparty computation protocol) requires a high minimum distance on $\mathcal{C}^{*2}$ in order to be secure. Additionally, it is shown that one can construct a t-strongly multiplicative secret sharing scheme, with both secrets and shares in $\mathbb{F}_q$, from a linear code $\mathcal{C}$ if $d(\mathcal{C}^{*2}) \geq t + 2$ and $d(\mathcal{C}^{\perp}) \geq t + 2$.

Products of codes have interest in other areas as well. For example it has been used to attack cryptosystems [11] and it has been used in decoding algorithms for

linear error-correcting codes [23]. Therefore, general results of products of codes has also been studied in a few papers, see for instance [26].

This introduction on squares and products of codes motivates the need for describing the parameters of these. To be more specific, we often want a linear code, where both the code and the square have good parameters. This seems to be difficult since squaring is often a destructive operation for the dimension. In fact, it is shown in [10] that choosing a family of random linear codes either the codes or the squares will be asymptotically bad.

Therefore, it seems difficult to find codes where both the square and itself is good for large n. This is however possible, as it was shown in [25], where a family of binary linear codes is presented such that both the code and the square are asymptotically good. However, for applications it is also of interest to find codes of some specific lengths which has good squares. This is what is the purpose in the following, where we consider matrix-product codes.

Using matrix-product codes one can obtain longer codes from shorter ones. In the following, we give a short introduction to matrix-product codes and determine the parameters of the square of some matrix-product codes.

5.1 Definition:
Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s$ be linear codes over $\mathbb{F}_q$ and $A \in \mathbb{F}_q^{s\times m}$ be a matrix with full rank satisfying $s \leq m$. Then the matrix-product code is given by all $n \times m$ matrices in the set

$$\mathcal{C} = \{[\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_s]A \mid \mathbf{c}_i \in \mathcal{C}_i\},$$

and we write $\mathcal{C} = [\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s]A$.

When we talk about A in this section we always mean a matrix with the properties above. Additionally, we denote by A_i the matrix consisting of the first i rows of A and $\mathcal{C}_{A_i}$ the linear code spanned by the rows in A_i. We will use the notation $D_i = d(\mathcal{C}_{A_i})$. In the following proposition, we summarize some known facts about matrix-product codes. We will not proof the results but refer the reader to [1].

5.2 Proposition:
Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s$ be linear $[n, k_1, d_1]_q, [n, k_2, d_2]_q, \ldots, [n, k_s, d_s]_q$ codes with generator matrices $G_1, G_2, \ldots, G_s$ respectively. Then $\mathcal{C} = [\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s]A$ is an

$$[nm, k_1 + k_2 + \cdots + k_s, \geq \min\{d_1D_1, d_2D_2, \ldots, d_sD_s\}]_q$$

linear code and a generator matrix of $\mathcal{C}$ is given by

$$G = \begin{bmatrix} a_{11}G_1 & a_{12}G_1 & \cdots & a_{1m}G_1 \\ a_{21}G_2 & a_{22}G_2 & \cdots & a_{1m}G_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1}G_s & a_{s2}G_s & \cdots & a_{sm}G_s \end{bmatrix}.$$

It might also be noted that assuming that the codes are nested $\mathcal{C}_1 \supseteq \mathcal{C}_2 \supseteq \cdots \supseteq \mathcal{C}_s$, the bound on the minimum distance is fulfilled with equality, a result that is due to [17].

Now we consider some specific matrix-product codes. The first, also known as the $(u, u+v)$-construction, is given by setting $s = m = 2$ and letting

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \tag{5.1}$$

In this case we obtain the following result on the square.

5.3 Theorem:
Let A be as in (5.1)*. Given $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$, we obtain that the square of $\mathcal{C} = [\mathcal{C}_1, \mathcal{C}_2]A$ is given by $\mathcal{C}^{*2} = [\mathcal{C}_1^{*2}, (\mathcal{C}_1 + \mathcal{C}_2) * \mathcal{C}_2]A$, where $\mathcal{C}_1 + \mathcal{C}_2$ means the smallest linear code containing both $\mathcal{C}_1$ and $\mathcal{C}_2$.*

Proof:
Let G_1, G_2 be generator matrices for $\mathcal{C}_1, \mathcal{C}_2$ respectively. By Proposition 5.2 a generator matrix for $\mathcal{C}$ is given by

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

For two matrices A and B, we use the notation $A * B$ to denote the matrix which consists of all component-wise products of rows in A by rows in B. So if A is an $l \times n$ matrix and B is an $m \times n$ matrix, we have that $A * B$ is an $lm \times n$. This means that the rows in

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

spans $\mathcal{C}^{*2}$. By removing linearly dependent rows we see that we obtain a generator matrix for $[\mathcal{C}_1^{*2}, (\mathcal{C}_1 + \mathcal{C}_2) * \mathcal{C}_2]A$. ■

One should notice that if $\mathcal{C}_2 \subseteq \mathcal{C}_1$, the code $\mathcal{C}_1 + \mathcal{C}_2$ reduces to $\mathcal{C}_1$ and hence $\mathcal{C}^{*2} = [\mathcal{C}_1^{*2}, \mathcal{C}_1 * \mathcal{C}_2]A$. We state a similar result for another matrix-product code, and here we assume that the codes are nested in the theorem in order to express the square.

5.4 Theorem:
Let $\mathcal{C}_1 \supseteq \mathcal{C}_2 \supseteq \mathcal{C}_3$ be linear codes in $\mathbb{F}_q^n$. Furthermore, let

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & (p-1) & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

where $p \neq 2$ is the characteristic of $\mathbb{F}_q$. If $\mathcal{C} = [\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3]A$, then

$$\mathcal{C}^{*2} = [\mathcal{C}_1^{*2}, \mathcal{C}_1 * \mathcal{C}_2, \mathcal{C}_2^{*2} + \mathcal{C}_1 * \mathcal{C}_3]A.$$

Proof:
We use the same notation as in the previous proof and follow more or less the same procedure.

$$G * G = \begin{bmatrix} G_1 * G_1 & G_1 * G_1 & G_1 * G_1 \\ 0 & (p-1)G_1 * G_2 & G_1 * G_2 \\ 0 & G_2 * G_2 & G_2 * G_2 \\ 0 & 0 & G_1 * G_3 \\ 0 & 0 & G_2 * G_3 \\ 0 & 0 & G_3 * G_3 \end{bmatrix}.$$

Since we assumed that the codes are nested we can by doing row operations get rid of the last two "block rows". Additionally, we can replace the third "block row" by $\begin{bmatrix} 0 & 0 & G_2 * G_2 \end{bmatrix}$, and hence the result follows. ■

At last, we consider the case, where the matrix used in the construction is an $s \times q$ Vandermonde matrix.

5.5 Theorem:
Let $\mathcal{C}_0, \mathcal{C}_1, \cdots, \mathcal{C}_{s-1}$ be linear codes in $\mathbb{F}_q^n$ with generator matrices $G_0, G_1, \ldots, G_{s-1}$ respectively. Furthermore, let

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

where the α_i's are distinct elements in $\mathbb{F}_q$ and $s \leq q$ is some integer. Denote by $\mathcal{C} = [\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{s-1}]V(s)$. Then

$$\mathcal{C}^{*2} = [\sum_{i+j=0} \mathcal{C}_i * \mathcal{C}_j, \sum_{i+j=1} \mathcal{C}_i * \mathcal{C}_j, \ldots, \sum_{i+j=\tilde{s}-1} \mathcal{C}_i * \mathcal{C}_j]V(\tilde{s})$$

where $\tilde{s} = \min\{2s-1, q\}$ and $i+j$ is considered modulus q in the sums.

Proof:
Again the strategy is the same as before. The rows in $G * G$ is given by

$$\left[\alpha_1^{i+j} G_i * G_j \quad \alpha_2^{i+j} G_i * G_j \quad \cdots \quad \alpha_q^{i+j} G_i * G_j\right],$$

and the result follows. ■

These results show that if $\mathcal{C}_i$ comes from a family of codes where we can describe the products, we might be able to determine the square of the matrix-product code, which has length a multiple of the codes used in the construction. This could for instance be if $\mathcal{C}_i$ where cyclic codes or some kind of evaluation codes.

Problems from the Results on Matrix-Product Codes

The results about squares of matrix-product codes indicate that it could be interesting to study products of codes more in depth since, as we saw in the previous section, we can use products of different codes to describe the square of a matrix-product code, but also as it was mentioned in the introduction that product of codes is used in other areas as well. Therefore, determining the products of some different types of codes could be interesting.

The results on matrix-product codes also give rise to the question if one can, by using some specific codes in the construction, obtain codes, where both the code and the square are good. However, this give rise to a new question. Because what is "good", when we consider the square of a code with a specific length?

Not much research has been done on this topic since most research so far has considered the asymptotic setting. There are, however, some limitation bounds in [26] but there is still a huge gap between the constructions and bounds. One of the papers which contain constructions for some specific lengths is [5], where some specific cyclic codes have been studied. It could be interesting to compare the matrix-product codes with the parameters from this article.

Therefore, other research topics could also be construction of new bounds for squares of codes, both existence bounds, such as an Gilbert-Varshamov bound for squares, but also improved limitations bounds could be interesting. Such bounds could lead to a clarification of which codes are good when considering the parameters for both the code itself and the square.

Bibliography

[1] Tim Blackmore and Graham H. Norton. 'Matrix-Product Codes over Fq'. In: *Applicable Algebra in Engineering, Communication and Computing* 12.6 (Dec. 2001), pp. 477–500. DOI: 10.1007/PL00004226.

[2] G. R. Blakley. 'Safeguarding cryptographic keys'. In: *Managing Requirements Knowledge, International Workshop on* (1979), p. 313. DOI: 10.1109/AFIPS.1979.98.

[3] Carlo Blundo, Alfredo De Santis and Ugo Vaccaro. 'Efficient Sharing of Many Secrets'. In: *Proceedings of the 10th Annual Symposium on Theoretical Aspects of Computer Science*. STACS '93 (1993), pp. 692–703. URL: http://dl.acm.org/citation.cfm?id=646509.694823.

[4] Andrej Bogdanov, Siyao Guo and Ilan Komargodski. 'Threshold Secret Sharing Requires a Linear Size Alphabet'. In: *Theory of Cryptography* (2016). Ed. by Martin Hirt and Adam Smith, pp. 471–484. DOI: 10.1007/978-3-662-53644-5_18.

[5] Ignacio Cascudo. 'On squares of cyclic codes'. In: *CoRR* abs/1703.01267 (2017).

[6] Ignacio Cascudo, René Bødker Christensen and Jaron Skovsted Gundersen. 'Actively Secure OT-Extension from q-ary Linear Codes'. In: *Security and Cryptography for Networks*. Ed. by Dario Catalano and Roberto De Prisco. Cham: Springer International Publishing, 2018, pp. 333–348. DOI: 10.1007/978-3-319-98113-0_18.

[7] Ignacio Cascudo, Ronald Cramer and Chaoping Xing. 'Bounds on the Threshold Gap in Secret Sharing and its Applications'. In: *IEEE Transactions on Information Theory* 59.9 (Sept. 2013), pp. 5600–5612. ISSN: 0018-9448. DOI: 10.1109/TIT.2013.2264504.

[8] Ignacio Cascudo, Jaron Skovsted Gundersen and Diego Ruano. 'Improved Bounds on the Threshold Gap in Ramp Secret Sharing'. In: *IACR Cryptology ePrint Archive* (2018).

[9] Ignacio Cascudo et al. 'Rate-1, Linear Time and Additively Homomorphic UC Commitments'. In: *Advances in Cryptology – CRYPTO 2016*. Ed. by Matthew Robshaw and Jonathan Katz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 179–207. ISBN: 978-3-662-53015-3.

[10] Ignacio Cascudo et al. 'Squares of Random Linear Codes'. In: *IEEE Transactions on Information Theory* 61 (Mar. 2015), pp. 1159 –1173. DOI: 10.1109/TIT.2015.2393251.

[11] Alain Couvreur, Ayoub Otmani and Jean-Pierre Tillich. 'Polynomial Time Attack on Wild McEliece over Quadratic Extensions'. In: *Advances in Cryptology*. Springer Berlin Heidelberg, 2014, pp. 17–39. ISBN: 978-3-642-55220-5.

[12] Ronald Cramer, Ivan Bjerre Damgård and Jesper Buus Nielsen. *Secure multiparty computation and Secret Sharing*. Cambridge University Press, 2015. ISBN: 1-107-04305-0.

[13] Ivan Damgård and Sarah Zakarias. 'Constant-Overhead Secure Computation of Boolean Circuits using Preprocessing'. In: *Theory of Cryptography*. Ed. by Amit Sahai. Springer Berlin Heidelberg, 2013, pp. 621–641. ISBN: 978-3-642-36594-2.

[14] Olav Geil et al. 'Relative Generalized Hamming Weights of One-Point Algebraic Geometric Codes'. In: *IEEE Transactions on Information Theory* 60.10 (Oct. 2014), pp. 5938–5949. ISSN: 0018-9448. DOI: 10.1109/TIT.2014.2345375.

[15] Oded Goldreich, Silvio Micali and Avi Wigderson. 'How to Play ANY Mental Game'. In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. STOC '87. New York, New York, USA: ACM, 1987, pp. 218–229. DOI: 10.1145/28395.28420.

[16] Markus Grassl. *Bounds on the minimum distance of linear codes and quantum codes*. Online available at http://www.codetables.de. 2007.

[17] Fernando Hernando, Kristine Lally and Diego Ruano. 'Construction and decoding of matrix-product codes from nested codes'. In: *Applicable Algebra in Engineering, Communication and Computing* 20.5 (Oct. 2009), pp. 497–507. DOI: 10.1007/s00200-009-0113-5.

[18] Russell Impagliazzo and Steven Rudich. 'Limits on the Provable Consequences of One-way Permutations'. In: *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*. STOC '89. Seattle, Washington, USA: ACM, 1989, pp. 44–61. DOI: 10.1145/73007.73012.

[19] Yuval Ishai et al. 'Extending Oblivious Transfers Efficiently'. In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Springer Berlin Heidelberg, 2003, pp. 145–161. DOI: 10.1007/978-3-540-45146-4_9.

[20] Joe Kilian. 'Founding Crytpography on Oblivious Transfer'. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. STOC '88. ACM, 1988, pp. 20–31. DOI: 10.1145/62212.62215.

[21] Jun Kurihara, Tomohiko Uyematsu and Ryutaroh Matsumoto. 'Secret sharing schemes based on linear codes can be precisely characterized by the relative generalized Hamming weight'. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 95.11 (2012), pp. 2067–2075.

[22] Michele Orrù, Emmanuela Orsini and Peter Scholl. 'Actively Secure 1-out-of-N OT Extension with Application to Private Set Intersection'. In: *Topics in Cryptology – CT-RSA 2017*. Springer International Publishing, 2017, pp. 381–396. DOI: 10.1007/978-3-319-52153-4_22.

[23] Ruud Pellikaan. 'On decoding by error location and dependent sets of error positions'. In: *Discrete Mathematics* 106-107 (1992), pp. 369 –381. DOI: 10.1016/0012-365X(92)90567-Y.

[24] Michael O. Rabin. 'How to exchange secrets with oblivious transfer'. In: *Technical Report TR-81, Aiken Computation Lab, Harvard University* (1981).

[25] Hugues Randriambololona. 'Asymptotically Good Binary Linear Codes With Asymptotically Good Self-Intersection Spans'. In: *IEEE Transactions on Information Theory* 59 (May 2013), pp. 3038 –3045. DOI: 10.1109/TIT.2013.2237944.

[26] Hugues Randriambololona. 'On products and powers of linear codes under componentwise multiplication'. In: *Contemporary Math.* 637 (Apr. 2015).

[27] Adi Shamir. 'How to Share a Secret'. In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. DOI: 10.1145/359168.359176.

[28] Andrew Yao. 'Protocols for Secure Computations'. In: *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*. SFCS '82. IEEE Computer Society, 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.88.

[29] Zhuojun Zhuang et al. 'Some new bounds on relative generalized Hamming weight'. In: *2011 IEEE 13th International Conference on Communication Technology* (Sept. 2011), pp. 971–974. DOI: 10.1109/ICCT.2011.6158023.