Quantum Codes and Multiparty Computation

A Coding Theoretic Perspective

Qualification Exam • Master's Thesis

6th June 2018

René Bødker Christensen

AALBORG UNIVERSITY
STUDENT REPORT

Department of Mathematical Sciences

Skjernvej 4A, 9220 Aalborg Øst

Telephone 99408800

http://math.aau.dk

Title:

Quantum Codes and Multiparty Computation: A Coding Theoretic Perspective

Project period:

Qualification exam

Spring 2018

Student:

René Bødker Christensen

Supervisor:

Ignacio Cascudo Pueyo

Number of copies: 2

Page number: 31 (38 incl. appendices)

Finished: 6th June 2018

Synopsis:

This work has been written to satisfy the requirements for the qualification exam, which is part of the 4+4 PhD programme. It contains an in-depth summary of two research papers submitted to peer-reviewed journals or conferences during the first two years of study. In addition, it covers a manuscript that is still work in progress. The topics are reliable message transmission, OT-extension, and constructions of nested codes. The first two topics are relevant in the area of secure multiparty computation. The third can be used to construct both secret sharing schemes and asymmetric quantum codes.
The unifying theme in the work is the application of coding theoretic principles.

Summary in Danish

Dette speciale er skrevet som en del af kvalifikationseksamenen i forbindelse med 4+4-PhD-forløbet. Det indeholder derfor en dybdegående redegørelse for forskningsresultaterne frembragt i løbet af de første to år af uddannelsen. Specialet præsenterer tre artikler, hvoraf to er indsendt til videnskabelige tidsskrifter eller konferencer med peer-review, og én forventes indsendt få måneder efter aflevering.

Den første artikel, 'On one-round reliable message transmission', omhandler det problem, der kaldes *reliable message transmission*. Her er en afsender og en modtager forbundet af et antal kanaler, hvoraf nogle er kontrolleret af en modpart. Målet er at sikre, at en besked sendt af afsenderen når pålideligt frem til modtageren, selv hvis modparten ændrer informationen sendt over de kanaler, modparten kontrollerer. Baseret på arbejde af Bishop m.fl. og Patra m.fl. præsenteres to protokoller, der opnår den optimale transmissionsrate for passende størrelser af meddelelser. Forbedringerne i forhold til de tidligere protokoller er, at man enten kan sende større beskeder, eller bruge et legeme med færre elementer. Derudover illustreres forskellene mellem de to protokoller gennem konkrete eksempler.

'Actively Secure OT-Extension from q-ary Linear Codes' er skrevet i samarbejde med Ignacio Cascudo og Jaron S. Gundersen. Emnet er *oblivious transfer* (OT), som er en grundsten i mange kryptografiske protokoller. Mere specifikt handler artiklen om, hvordan et lille antal OT's kan udvides til et langt større antal OT's, hvilket har en lavere omkostning samlet set. I artiklen generaliseres en eksisterende protokol, så det er muligt at benytte koder over $\mathbb{F}_q$ frem for binære koder. Det vises, hvordan denne ændring fører til en reduktion i antallet OT's, der er nødvendige, for at kunne udvide til et bestemt, ønsket antal. Prisen for denne reduktion er, at protokollen samlet set kræver at flere bits sendes. En række eksempler illustrerer dog, at denne stigning kan være acceptabel i forhold til forbedringen i antallet af OT's.

Slutteligt behandles det endnu ikke afsluttede manuskript 'On nested code pairs from the Hermitian curve', der bruger indlejrede kodepar til at konstruere både asymmetriske kvantekoder og såkaldte *secret sharing schemes*. Dette manuskript er skrevet i samarbejde med Olav Geil, og de forbedrede kodepar stammer fra det Hermitiske funktionslegeme. De er konstrueret sådan, at deres relative afstande er lig en givet designafstand. Først præsenteres grænser for dimensionerne af de enkelte koder givet deres designafstande, og hernæst præsenteres formler for, hvornår to koder af denne type er indeholdt i hinanden. Desuden præsenteres en

anden konstruktion fra [CG18], hvor almindelige etpunkts-Hermitkoder benyttes. Ved at udnytte forbedrede grænser for afstandene, viser det sig, at kodeparrenes relative afstand overgår én af de ikke-relative afstande. Dette kan være ønskeligt i forbindelse med kvantekodning. De kvantekoder, der fremkommer af begge de nye typer af indlejrede kodepar, sammenlignes også med allerede kendte konstruktioner af asymmetriske kvantekoder.

Specialet afsluttes med en kort optegnelse over planerne for de resterende to år af PhD-forløbet. Dette indebærer både fremtidige forskningsprojekter og forventninger omkring det kommende udlandsophold.

Contents

Chapter 1 Preliminaries

This chapter contains a minimal introduction to some of the theory that serves as foundation for my research. The presentation contains no proofs and is merely meant to establish the notation and terminology used in the later chapters.

1.1 Linear codes

A q-ary linear code $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^n$. For $\mathbf{x} \in \mathbb{F}_q^n$, we define its Hamming weight as $\mathrm{w}_H(\mathbf{x}) = |\operatorname{supp} \mathbf{x}|$, where $\operatorname{supp} \mathbf{x}$ denotes the set of non-zero indices of $\mathbf{x}$. This induces the Hamming distance defined as

$$d_H(\mathbf{x}, \mathbf{y}) = \mathrm{w}_H(\mathbf{x} - \mathbf{y}), \quad \mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n.$$

One of the most important properties of a linear code $\mathcal{C}$ is its minimal distance, $d(\mathcal{C})$. As the name suggests, this is the minimal distance between any two distinct codewords in $\mathcal{C}$. The linearity of the code, however, implies that the minimal distance is also given by

$$d(\mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}} \mathrm{w}_H(\mathbf{c}).$$

If $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a subspace of dimension k and minimal distance d, it is referred to as an $[n, k, d]_q$ code.

The use of a linear code can counteract two types of data corruption: errors and erasures. Errors occur when the symbol in an unknown position of a codeword is altered. Erasures occur if a position is known to be altered. It may be shown [HP03; Thm. 1.11.6] that a word containing e erasures and t errors can be decoded to the original codeword if $e + 2t < d(\mathcal{C})$. In particular, when no erasures occur, decoding to the original codeword can only be guaranteed if $t < d(\mathcal{C})/2$. Otherwise, there may be several candidate corrections, or we may even correct to a different codeword.

A well-known class of codes is the Reed-Solomon codes, which are based on polynomial evaluation. For these, [Sud97] showed that it is possible to correct beyond half the minimal distance by relaxing the requirement that the decoding algorithm returns a single codeword. Instead, it returns a list of potential codewords, and for this reason the method is called list-decoding. This procedure, however, only works for Reed-Solomon codes up to a certain rate. Later, [GS98] improved the decoding algorithm, and this allows list-decoding of Reed-Solomon codes of higher rates. When using these algorithms, there is a trade-off between the parameters of the code and the maximal size of the list of candidate codewords. In particular, we shall be interested in lists of constant size in Section 2.1, and this restricts the possible code parameters.

Algebraic geometry codes

In Section 2.3, we consider a class of codes constructed from an algebraic function field. A field extension F/K is said to be an algebraic function field if F is a finite algebraic extension of $K(x)$ for some $x \in F$ that is transcendental over K. To such a function field, we associate a set of so-called places, which are in some sense generalizations of evaluation points. That is, for each element $z \in F$, we can speak of $z(P)$, the evaluation of z in the place P. We may also consider zeros and poles of z, and the orders of these.

When working with algebraic function fields, we use the concept of divisors, which are formal sums of places with integer coefficients. To each divisor D, we associate its Riemann-Roch space, $\mathcal{L}(D) \subsetneq F$. The general definition of such spaces may be found in [Sti09], but in this work we shall only need Riemann-Roch spaces of the form $\mathcal{L}(\lambda Q)$ for a single place Q. This space contains the elements in F, that have a pole of order at most λ in Q, but no other poles.

For the construction of algebraic geometry codes, we use function fields of the form $F/\mathbb{F}_q$. Additionally, we only consider the so-called rational places since evaluation in these places is guaranteed to give an element of $\mathbb{F}_q$. Let $P_1, P_2, \ldots, P_n, Q$ be distinct rational places of F, and define the divisor $D = P_1 + P_2 + \ldots + P_n$. For each $\lambda \in \mathbb{N}$ we define the algebraic geometry code

$$\mathcal{C}_{\mathcal{L}}(D, \lambda Q) = \left\{ \big(z(P_1), z(P_2), \ldots, z(P_n)\big) \mid z \in \mathcal{L}(\lambda Q) \right\}.$$

One may show that this is indeed a linear code over $\mathbb{F}_q$. In addition, its dual is another algebraic geometry code $\mathcal{C}_\Omega(\mathcal{C}, \lambda Q)$. For the details, see [Sti09; Chap. 2].

Nested code pairs

If $\mathcal{C}_1, \mathcal{C}_2$ are q-ary codes such that $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$, we call them a pair of nested codes, or equivalently a nested code pair. For such a pair, we define its relative distance as

$$d(\mathcal{C}_1, \mathcal{C}_2) = \min_{\mathbf{c} \in \mathcal{C}_1 \setminus \mathcal{C}_2} \mathrm{w}_H(\mathbf{c}).$$

Given a nested code pair $\mathcal{C}_2 \subsetneq \mathcal{C}_1$, the duals $\mathcal{C}_1^\perp \subsetneq \mathcal{C}_2^\perp$ form a nested code pair as well. Subsequently, the relative distance $d(\mathcal{C}_2^\perp, \mathcal{C}_1^\perp)$ can also be considered. From the definition, it follows immediately that the relative distances can be bounded by

$$d(\mathcal{C}_1, \mathcal{C}_2) \geq d(\mathcal{C}_1) \quad \text{and} \quad d(\mathcal{C}_2^\perp, \mathcal{C}_1^\perp) \geq d(\mathcal{C}_2^\perp), \tag{1.1}$$

but these are not always tight. In some cases, one of the relative distances exceeds the corresponding non-relative distance.

As will become evident from the following sections, the pairs of nested codes and their relative distances provide critical information when dealing with certain types of problems.

1.2 Secret sharing

Secret sharing is the concept of distributing information about a secret among a number of participants in such a way that only prescribed sets of participants can reconstruct the secret. The information given to each participant is called a share, and a method describing how to construct such shares is called a secret sharing scheme. When using a secret sharing scheme, it is necessary to determine the ability or inability of a set of participants to learn information about the shared secret. If a set of participants can reconstruct the secret, it is called a reconstructing set. On the other hand, if their pool of shares reveals nothing about the secret, the set is called a privacy set. As such, two important properties of a secret sharing scheme are the reconstructing number, r, and the privacy number, t. The former is defined as the minimal integer r such that any r participants form a reconstructing set, and the latter is the greatest integer t such that any t participants form a privacy set.

Nested code pairs lend themselves to a particularly nice construction of secret sharing schemes. Consider $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$, and denote by k_i the dimension of $\mathcal{C}_i$. Additionally, let $\ell = k_1 - k_2$ denote the codimension. By fixing bases $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{k_2}\}$ of $\mathcal{C}_2$ and $\{\mathbf{b}_1, \dots, \mathbf{b}_{k_2}, \mathbf{b}_{k_2+1}, \dots, \mathbf{b}_{k_1}\}$ of $\mathcal{C}_1$, we can share the secret $(s_1, s_2, \dots, s_\ell)$ among n participants by constructing the codeword

$$\mathbf{c} = \sum_{i=1}^{k_2} a_i \mathbf{b}_i + \sum_{i=1}^{\ell} s_i \mathbf{b}_{k_2+i},$$

where $a_i \in \mathbb{F}_q$ are chosen uniformly at random. The share of the i'th participant is then c_i. When using this construction, [KUM12] showed that the reconstruction number is given by $r = n - d(\mathcal{C}_1, \mathcal{C}_2) + 1$, and the privacy number by $t = d(\mathcal{C}_2^\perp, \mathcal{C}_1^\perp) - 1$.

1.3 Secure multiparty computation

The name secure multiparty computation covers a wide variety of problems where a number of participants wish to compute the output of some function without revealing their individual inputs. In a "yes"/"no"-vote for instance, the participants are required to learn the total number of "yes"-votes, but the individual votes must remain private. In other words, each participant has an input $x_i \in \{0, 1\}$ corresponding to either "yes" or "no", and they wish to compute the value of the function $f(x_1, x_2, \dots, x_n) = \sum_{i=1}^n x_i$.

In an ideal world, participants could agree on some third party that they all trust. They could then send their inputs to this trusted third party, who could perform the computation and reveal only the result to the participants. In reality, however, it may not be possible to agree on a third party that is deemed trustworthy by all participants. Additionally, this system has a single point of failure. If the third party is in some way compromised, this will reveal every single input.

Protocols for secure multiparty computation aim to offer the same functionality as sending the inputs to a trusted third party, without actually doing so.[†] Instead the participants perform a number of steps consisting of both local computations and communication between participants. This allows them to reach the result themselves. Intuitively, we say that a protocol is secure if it reveals no more than the third party would have done. Proving such a claim formally is done by providing a simulator, which will – loosely speaking – simulate the output of the protocol using only the output of an ideal third party. No simulators are presented in this work, but more details may for instance be found in [CDN15].

Models for multiparty computation contain the concept of an adversary, which will control a number of the participants and attempt to gain information about the inputs of the remaining participants. Depending on the type of adversary, it may also attempt to skew the results in some way. An adversary is called *passive* if the participants under its control follow the steps of the protocol, but all the information seen during the protocol is pooled in an attempt to extract knowledge about the unknown inputs. A more powerful and nefarious adversary is *active*, meaning that it may deviate from the protocol specification if this helps in extracting information. Based on this, we call a protocol passively or actively secure depending on the type of adversary it offers protection against.

1.4 Quantum codes

With the advent of quantum computing, interest in quantum error-correcting codes has increased. Where classical information is susceptible only to bits being flipped, information encoded in a quantum system has to withstand both bit flips and phase shifts, but also combinations hereof; see for instance [CRSS97; CRSS98; NC10]. Mathematically, a q-ary linear asymmetric quantum error-correcting code is a k-dimensional subspace of $\mathbb{C}^{q^k}$, and the possible errors can be represented using the unitary operators X and Z. These represent a bit flip and a phase shift, respectively. If a q-ary linear quantum error-correcting code is capable of correcting $\lfloor (d_z - 1)/2 \rfloor$ phase shift errors and $\lfloor (d_x - 1)/2 \rfloor$ bit flip errors, the code is said to be an $[[n, k, d_z/d_x]]_q$ code. Traditionally, the two types of errors were treated equally, and for this reason only the smallest of the distances d_z, d_x was associated with the code. As shown in [IM07], however, most quantum systems experience phase errors much more frequently than bit flips. Hence, it is beneficial to seek asymmetric quantum codes with $d_z > d_x$ since this allows better correction of the more common type of errors.

[†]Some protocols may still rely on a trusted third party for some steps.

In this setting, nested code pairs play a vital role as well. The CSS-construction named after Calderbank, Shor and Steane transform such a code pair into an asymmetric quantum code, whose distances depend on the relative distances of the codes and their duals.

1.4.1 Proposition (SKR09; Lem. 3.1):
Let $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$ be a nested pair of codes, and let $\ell = \dim \mathcal{C}_1 - \dim \mathcal{C}_2$. Then there exists an $[[n, \ell, d_z/d_x]]_q$ asymmetric quantum code with $d_z = d(\mathcal{C}_1, \mathcal{C}_2)$ and $d_x = d(\mathcal{C}_2^\perp, \mathcal{C}_1^\perp)$.

If the bounds in (1.1) are equalities, the associated quantum code is called *pure*. Otherwise, it is *impure*, and it can in fact be desirable for a quantum code to be impure [AKS06].

Contributions CHAPTER 2

This chapter contains an overview of the research I have done during the first two years of the PhD-study. This has resulted in two papers, both of which have been submitted to a peer-reviewed scientific journals and a peer-reviewed conference, respectively. These are 'On one-round reliable message transmission' [Chr17], and 'Actively Secure OT-Extension from q-ary Linear Codes' [CCG18]. The latter is written in collaboration with Ignacio Cascudo and Jaron S. Gundersen. Additionally, the chapter contains results from an as of yet unpublished manuscript called 'On nested code pairs from the Hermitian curve' [CG18]. This is written in collaboration with Olav Geil, and we expect to submit it to a peer-reviewed journal in the near future. The two papers and the manuscript are treated in three separate sections, each of which is concluded by discussing the expected implications of the presented results.

Rather than repeating the proofs and detailed analysis from each paper, the aim of the following sections is to recap the overall results, and in some cases to elaborate on some of the details that were left out of the submitted versions. Naturally, this exposition will contain material which is largely a facsimile from each of the three papers. In some instances, the layout and notation have been altered to ensure a unified treatment throughout this chapter, but in any case a reference to the original source is clearly specified.

Although the three topics may seem quite different, the results within all rely heavily on techniques from coding theory. Sections 2.1 treats a problem that is not strictly within the area of secure multiparty computation, but its results may be used as part of a multiparty computation protocol. The topic of Section 2.2, extension of oblivious transfer, is one of the classical problems from the area of secure multiparty computation. Finally, the contents of Section 2.3 are in some sense purely coding theoretic, but the presented results may be applied to both secret sharing and asymmetric quantum codes.

2.1 Reliable message transmission

Initially inspired by the concept of *robust distributed storage* presented in [BPRW16], I started work on reliable message transmission. This is a model where a sender and a receiver are connected via a number of channels, and some subset hereof is controlled by an adversary. The objective is to allow the sender to transmit some message to the receiver within a certain number of rounds. This type of problem was first introduced by [DDWY93] under the name *secure message transmission*. This initial setting requires the transmission to be perfectly private[†] and perfectly reliable in the sense that no information is leaked to the adversary, and that the receiver will always recover the correct message. Continuing in the footsteps of [FW00], where these conditions were relaxed to allow some positive probability of failure, reliable message transmission does not provide privacy, but only reliability. That is, we will fix a desired upper bound δ on the probability that the receiver recovers a wrong message or no message at all.

In [FW00] it was shown that an adversary controlling t out of $n = 2t + 1$ channels is the maximal corruption that we can hope to counteract by a reliable message transmission protocol. In this setting, a straightforward, naïve solution is to simply broadcast the message across each channel. Since the majority of the channels are honest – i.e. not controlled by the adversary – the receiver can recover the correct message by a majority vote. This leads to perfect reliability, but the number of bits transmitted is n times the message length. Another simple solution could be to use a Reed-Solomon code of length n, and send each symbol of the codeword across the corresponding channel. But in order to have any hope of correcting errors introduced by the adversary, it must be the case that $t < d/2 = (n-k+1)/2$. From this it is seen that this approach fails for $n = 2t + 1$, unless $k = 1$ – which is exactly broadcast.

In general, the performance of a message transmission protocol is assessed by its transmission rate, which is the number of bits transmitted in total divided by the number of message bits. Thus, broadcast has a transmission rate of $\Theta(n)$, and a lower transmission rate is better. In [PCRS10], it was shown that the transmission rate is at least $\Omega(1)$, and provided a protocol attaining this rate under certain circumstances. Hence, the bound is tight, and a protocol that attains this bound is called optimal. Note, however, that the optimality is understood in an asymptotic sense. Even tough a protocol is optimal, it may still be possible to improve its performance, for instance by lowering the constants hidden by the Θ-notation, or by allowing smaller field sizes.

The improved protocols

In [Chr17], I propose two protocols for reliable message transmission, which are optimal for sufficiently large messages. Both use only a single round to transmit the message, and they are included as Protocols 1 and 2 on page 8. The protocols are based on [BPRW16] and [PCRS10], respectively, and allow larger messages or smaller field sizes than the originals. These improvements stem from a careful analysis of the underlying coding theory, which allows improved choices of parameters. In both cases, the protocols rely on an integrity check based on a family of hash functions. More precisely, I use *ε-almost universal* hash families as introduced by [Sti94]. If

[†] In the usual terminology, this is called *secure* rather than *private*.

Protocol 1: One-round RMT (using list-decoding) [Chr17]

This protocol allows a sender S to reliably send a symbols of $\mathbb{F}_q$ to a receiver R by using $n = 2t+1$ channels, t of which may be controlled by an adversary. The parameter a must be sufficiently small such that applying the Guruswami-Sudan algorithm on an $[n,a]$ Reed-Solomon code $\mathcal{C}$ allows correction of t errors with a list of size $L = \mathcal{O}(1)$. The protocol relies on an ε-AU hash family $\mathcal{H} = \{h_{\mathbf{k}} \colon \mathbb{F}_q^a \to \mathbb{F}_q^\eta \mid \mathbf{k} \in \mathbb{F}_q^\eta\}$.

1. The message $\mathbf{m} \in \mathbb{F}_q^a$ is encoded using $\mathcal{C}$, yielding the codeword $(s_1, s_2, \ldots, s_n)$.
2. For $i = 1,2,\ldots,n$, S samples a random key $\mathbf{k}_i \in \mathbb{F}_q^\eta$, and computes $\mathbf{v}_i = h_{\mathbf{k}_i}(\mathbf{m})$.
3. Across the i'th channel, S transmits $\{s_i, \mathbf{k}_i, \mathbf{v}_i\}$.
4. R receives the possibly modified values $\{s_i', \mathbf{k}_i', \mathbf{v}_i'\}$ for $i = 1,2,\ldots,n$. It uses the Guruswami-Sudan algorithm on the word $(s_1', s_2', \ldots, s_n')$ to obtain a list of L potential messages $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_L$.
5. For each of the L messages, R checks that $\mathbf{m}_i$ agrees with at least $t+1$ of the hash values $\mathbf{v}_j'$. If not, it removes $\mathbf{m}_i$ from the list.
6. If only a single $\mathbf{m}_i$ remains, R outputs this message. Otherwise, the protocol has failed.

Protocol 2: One-round RMT-protocol (using erasure-decoding) [Chr17]

This protocol allows a sender S to reliably send ab symbols of $\mathbb{F}_q$ to a receiver R in one round by using $n = 2t+1$ channels, t of which may be controlled by an adversary. Beforehand, S and R have agreed upon a parameter $e \in \mathbb{N}$, which satisfies $e \leq t+1-b$. Additionally, they agree on an ε-AU hash family $\mathcal{H} = \{h_{\mathbf{k}} \colon \mathbb{F}_q^a \to \mathbb{F}_q^\eta \mid \mathbf{k} \in \mathbb{F}_q^\eta\}$.

1. The message is represented as a matrix $M \in \mathbb{F}_q^{a\times b}$ and each row is encoded using an $[n,b]$ Reed-Solomon code over $\mathbb{F}_q$. Denote by Σ the matrix whose rows are given by the corresponding codewords.
2. For each column $\mathbf{s}_i$ of Σ, S samples uniformly and independently n keys $(\mathbf{k}_{i1}, \mathbf{k}_{i2}, \ldots, \mathbf{k}_{in})$, where $\mathbf{k}_{ij} \in \mathbb{F}_q^\eta$, and computes the tag $\mathbf{v}_{ij} = h_{\mathbf{k}_{ij}}(\mathbf{s}_i)$ for each $j \in \{1,2,\ldots,n\}$.
3. Across the i'th channel, S transmits $\{\mathbf{s}_i\} \cup \{\mathbf{k}_{ji}, \mathbf{v}_{ji}\}_{j=1,2,\ldots,n}$.
4. R receives the possibly modified values $\{\mathbf{s}_i'\} \cup \{\mathbf{k}_{ji}', \mathbf{v}_{ji}'\}_{j=1,2,\ldots,n}$ for $i = 1,2,\ldots,n$. For each i, it compares the tag $\mathbf{v}_{ij}'$ received from the j'th channel to the hash value $h_{\mathbf{k}_{ij}'}(\mathbf{s}_i')$. If these disagree for more than t channels, R will mark $\mathbf{s}_i$ as modified.
5. For each row in Σ', R computes the syndrome to check if it contains errors. Depending on the result, it proceeds with one of the three following steps.
   a) **The syndrome is zero:** Σ contains no errors, meaning that R can simply use polynomial interpolation to recover the message.
   b) **The syndrome is nonzero, and Σ contains at least $t-e$ erased columns:** R uses a decoding algorithm for Reed-Solomon codes to correct the erasures and errors, hereby recovering the message.
   c) **The syndrome is nonzero, and Σ contains less than $t-e$ erased columns:** Too many modified channels have passed the integrity checks. The protocol has failed.

$\mathcal{H}$ is a family of functions from a message set $\mathcal{M}$ to a tag set $\mathcal{T}$, it is called ε-almost universal, or ε-AU for short, if

$$\Pr_{h \leftarrow \mathcal{H}}[h(m) = h(m')] \leq \varepsilon,$$

whenever $m, m' \in \mathcal{M}$, and $m \neq m'$. That is, when sampling a function from $\mathcal{H}$ uniformly at random we have an upper bound on the risk of a hash collision happening.

As a specific instantiation, I consider the following family based on polynomial evaluation, which is a generalization of the family used in [BPRW16].

2.1.1 Definition (Chr17; Def. 2.2):
Consider the finite field $\mathbb{F}_q$, and let $\mathcal{K} \subseteq \mathbb{F}_q$. For every pair of positive integers $\eta \leq a$, define the map $\mathsf{PEval}^\eta : \mathbb{F}_q^a \times \mathcal{K}^\eta \to \mathbb{F}_q^\eta$ *by*

$$\mathsf{PEval}^\eta(\mathbf{m}, \mathbf{k}) = (f_{\mathbf{m}}(k_1), f_{\mathbf{m}}(k_2), \dots, f_{\mathbf{m}}(k_\eta)),$$

where $f_{\mathbf{m}}(x) = \sum_{i=1}^{a} m_i x^i$. *We use the notation* $\mathsf{PEval}_{\mathbf{k}}^\eta(\mathbf{m}) = \mathsf{PEval}^\eta(\mathbf{m}, \mathbf{k})$.

This means that the hash function $\mathsf{PEval}_{\mathbf{k}}^\eta$ produces its tags by evaluating the message in the points specified by $\mathbf{k}$. Throughout this work, the key space $\mathcal{K}$ will always be taken as the full field, $\mathbb{F}_q$.

The first protocol is merely the robust distributed storage from [BPRW16] translated into the setting of reliable message transmission. However, whereas Bishop, Pastro, Rajaraman and Wichs were primarily interested in asymptotic performance, and used the list-decoding algorithm by Sudan [Sud97], I apply the improved algorithm by Guruswami and Sudan [GS98]. This increases the possible message size from $\lfloor n/8 \rfloor + 1$ to $\lfloor n/5 \rfloor + 1$, while still retaining a constant list size. The idea of the protocol is to use a Reed-Solomon code to encode the message on the sender side, and let the receiver use list-decoding to correct any errors introduced by the adversary. Since this gives a list of possible messages, the sender will also use an almost universal hash-family to create tags for the message. The receiver can then use these tags to reduce the list of potential messages to a single message, which is in fact guaranteed to be the correct one. As shown in [Chr17; Sec. 4.3], when using the hash family from Definition 2.1.1, the protocol is optimal as long as the message is of size $\Theta(n)$. If only a single evaluation point is used in the hash functions – that is, $\eta = 1$ – the field size depends quadratically on n. Choosing a greater value of η will allow smaller field sizes, but this comes at the cost of sending additional field symbols.

The properties of Protocol 1 are summarized in the following propositions based on [Chr17; Prop. 4.2 and Sec. 4.2–4.3].[‡]

[‡]The presentation in [Chr17] is more general, and omits the assumption that $\mathcal{K} = \mathbb{F}_q$.

2.1.2 Proposition:
Let $n = 2t + 1$, the parameter a be as in Protocol 1, and $\mathcal{H}$ be an ε-AU hash family. If an adversary controls t out of n channels, Protocol 1 allows reliable transmission of a elements of $\mathbb{F}_q$ in a single round. The probability of failure is at most $nL\varepsilon$, where L is the maximal list size.

If $\mathcal{H}$ is the hash family of Definition 2.1.1, the protocol sends a total of $n(1 + 2\eta)$ elements of $\mathbb{F}_q$, and the probability of failure is at most $nL(a/q)^\eta$.

Remark:
The robust distributed storage from [BPRW16] is a special case of Protocol 1. It is recovered by setting (i) the field size to be a power of 2, (ii) $\mathcal{H}$ to be the hash-family from Definition 2.1.1 with $\eta = 1$, and (iii) by using Sudan list-decoding rather Guruswami-Sudan.

The second protocol is based on the work of [PCRS10], which in essence relies on erasure decoding of Reed-Solomon codes. Much like Protocol 1, the first step of the protocol is to use a Reed-Solomon code to encode each row of the message. Now, rather than hashing the message itself, the sender will hash the codeword entries sent across each channel. The receiver can use this information to identify some of the channels controlled by the adversary. The information delivered by these channels can then be treated as erasures when recovering the message.

In the original protocol by [PCRS10], the choice of parameters requires the receiver to identify all the corrupt channels during the integrity checks since the protocol will otherwise fail. What I propose in [Chr17] is to adjust the size of the message to allow some number of corrupt channels to pass the integrity checks, and then use a combination of erasure and error correction to recover the message. More precisely, [PCRS10] use $b = t + 1$, whereas I introduce the error parameter e, and require $b \leq t + 1 - e$. An erroneous channel might pass the integrity check by chance, and e describes the maximal number of such channels that we are willing to tolerate. By choosing the parameters appropriately, this yields an optimal protocol for messages of size $\Theta(n^2)$, when using the family defined in Definition 2.1.1. This is the same as [PCRS10], but where they require the field size to be cubic in n in order to obtain δ-reliability, my proposed protocol lowers this requirement to a quadratic dependence for $\eta = 1$. Again, greater values of η allow smaller field sizes, but increases the number of symbols transmitted.

Similar to Proposition 2.1.2, the results in [Chr17; Prop. 5.2 and Sec. 5.2–5.3] can be collected in a single proposition, summarizing the properties of Protocol 2.

2.1.4 Proposition:
Let $n = 2t + 1$, the parameters a, b, and e be as in Protocol 2, and let $\mathcal{H}$ be an ε-AU hash family. If an adversary controls t out of n channels, Protocol 2 allows reliable transmission of ab elements of $\mathbb{F}_q$ in a single round with probability of failure at most $t(t + 1)\varepsilon/(e + 1)$.

If $\mathcal{H}$ is the hash family of Definition 2.1.1, the protocol sends a total of $2\eta n^2 + an$ *elements of* $\mathbb{F}_q$*, and the probability of failure is at most* $t(t+1)a^\eta/((e+1)q^\eta)$.

Remark:
Although [PCRS10] has a more fine-grained adversary structure, the original protocol can be considered as a special case of Protocol 2. This is seen by setting (i) the error parameter $e = 0$, (ii) the message dimensions to $a = n$ and $b = t+1$, and (iii) $\mathcal{H}$ to be the hash-family from Definition 2.1.1 with $\eta = 1$. It is worth mentioning, however, that [PCRS10] use a systematic Reed-Solomon encoding, but that this is equivalent to the one used in Protocol 2.

In both cases, the proposed protocols are only optimal as long as the message has a certain size. In the light of this, it is natural to ask if the optimal transmission rate of $\Theta(1)$ can be achieved for any message size. In [Chr17; Prop. 3.2], I answer this in the negative by proving that any protocol for reliable message transmission requires a transmission rate of at least $\Theta(n)$ when transmitting messages of constant size. This result not only holds true for one-round protocols, but protocols using any number of rounds.

Concrete examples

Since the above considerations apply to the asymptotic performance, [Chr17] also includes a number of concrete examples, which may be found in Tables 1–4 on page 12. Due to the number of parameters in the construction, the shown examples were found using a computer search, the source code of which may be found in Appendix A. Given the number of corrupt channels and the message size in bits, the program will search through the possible parameter values for each protocol to determine the ones leading to the lowest transmission rate. This is not an exhaustive search since it for instance only considers fields of sizes 2^{2^k} for Protocol 2. This is enough, however, to illustrate the differences between the two protocols. Studying the tables reveals that Protocol 1 is suited for relatively small messages, whereas Protocol 2 performs better for larger message sizes. It is also worth noting that Protocol 1 often uses a very large field. This is caused by the dimension restrictions imposed by the list decoding algorithm.

The code in Appendix A also includes a method `beatBroadcast`, which was not used directly in the construction of the examples. Instead, the output of this function is the reason why $t = 1$ and 2048 bits was considered; this is the smallest example where Protocol 2 has a lower total transmission cost than the naïve solution of broadcasting.

Implications

Message transmission protocols are important in secure multiparty computation since any such computation requires the participants to communicate. Depending on the context, this communication may not need to be secure, but merely reliable. The relevance of reliable message transmission is also witnessed by the strong connections to the reliable distributed storage used in [BPRW16]. Here, the method is used as a sub-procedure for creating a robust secret sharing scheme. As such, it is not inconceivable that these techniques and their improvements may be applicable in other problems of secret sharing and secure multiparty computation.

Protocol	Field size	Message dimension	Parameters	Bits transmitted	Percent of broadcast
1	2^{2048}	1	$\eta = 1$	18432	300.0 %
2	2^{64}	$(16, 2)$	$\eta = 2, e = 0$	5376	87.5 %
2	2^{32}	$(32, 2)$	$\eta = 3, e = 0$	4800	78.1 %
2	2^{16}	$(64, 2)$	$\eta = 9, e = 0$	5664	92.2 %

Table 1. [Chr17] Examples of performance of Protocols 1 and 2 when sending a message consisting of 2048 bits in the case $t = 1$. All protocols achieve a reliability of 2^{-80}. Broadcast costs 6144 bits.

Protocol	Field size	Message dimension	Parameters	Bits transmitted	Percent of broadcast
1	$2^{8,000,000}$	1	$\eta = 1$	72,000,000	300.0 %
2	2^{64}	$(62500, 2)$	$\eta = 2, e = 0$	12,002,304	50.0 %
2	2^{32}	$(125000, 2)$	$\eta = 6, e = 0$	12,003,456	50.0 %

Table 2. [Chr17] Examples of performance of Protocols 1 and 2 when sending a 1 megabyte message (8,000,000 bits) in the case $t = 1$. All protocols achieve a reliability of 2^{-80}. Broadcast costs 24,000,000 bits.

Protocol	Field size	Message dimension	Parameters	Bits transmitted	Percent of broadcast
1	2^{26}	10	$\eta = 4$	47,034	91.4 %
2	2^{16}	$(1, 101)$	$\eta = 6, e = 0$	7,760,208	1508.1 %
2	2^{8}	$(1, 62)$	$\eta = 11, e = 39$	7,112,184	1382.2 %

Table 3. [Chr17] Examples of performance of Protocols 1 and 2 when sending a 256 bit message in the case $t = 100$. All protocols achieve a reliability of 2^{-80}. Broadcast costs 51,456 bits.

Protocol	Field size	Message dimension	Parameters	Bits transmitted	Percent of broadcast
1	$2^{195,122}$	41	$\eta = 1$	117,658,566	7.3 %
2	2^{64}	$(1238, 101)$	$\eta = 2, e = 0$	26,268,288	1.6 %
2	2^{32}	$(2476, 101)$	$\eta = 5, e = 0$	28,853,952	1.8 %
2	2^{16}	$(5000, 100)$	$\eta = 25, e = 1$	48,400,800	3.0 %

Table 4. [Chr17] Examples of performance of Protocols 1 and 2 when sending a 1 megabyte message (8,000,000 bits) in the case $t = 100$. All protocols achieve a reliability of 2^{-80}. Broadcast costs 1,608,000,000 bits.

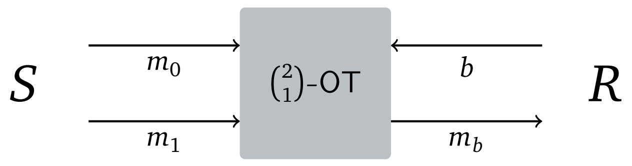


Figure 1. Diagram representing the oblivious transfer-functionality. The receiver R inputs $b \in \{0,1\}$, and receives the corresponding message m_b.

2.2 Extending oblivious transfer

Multiparty computation relies on several so-called primitives, which can be used to implement larger, more complicated protocols. One such primitive is *oblivious transfer*, and in its simplest form it allows a sender to input two messages m_0 and m_1, and a receiver to learn a message m_b of his choice. This must be done in such a way that the receiver learns nothing about m_{1-b}, and the sender nothing about b. More generally, the sender will input N messages, and the receiver will choose K of these. In this case, we call the primitive a K out of N-oblivious transfer, which is denoted by $\binom{N}{K}$-OT. Figure 1 illustrates the OT-primitive in the case of 1 out of 2.

As a simple motivational example for the use of OT, consider the following. The sender and receiver each hold a value in $\mathbb{F}_2$, say x and y, respectively. By using the OT-primitive, the participants can compute the product xy in a secure manner. In particular, if the sender inputs $m_0 = 0$ and $m_1 = x$, and the receiver $b = y$, the OT will output $m_0 = 0 \cdot x$ or $m_1 = 1 \cdot x$, depending on the input of the receiver. In either case, the receiver has learned xy. By changing this procedure slightly, the OT can also be used to construct a secret sharing of the product. Namely, if the sender inputs $m_0 = r$ and $m_1 = r + x$ for some random $r \in \mathbb{F}_2$, then the receiver will learn the share $r + xy$. When combined with the sender's share r, the product can be recovered.

Oblivious transfer is among the most important primitives in multiparty computation, and finds use in well-known protocols such as Yao's garbled circuits [Yao82] and the GMW-compiler [GMW87]. Perhaps surprisingly, [Kil88] even showed that the OT-primitive is enough to implement any cryptographic protocol. Using OT, however, comes at a price; the work of [IR89] suggests that any implementation of OT is highly likely to require expensive public key cryptosystems.

In order to reduce the cost of OT, [Bea96] showed that a relatively small number of base OT's can be 'extended' to produce the same output as a much larger number of OT's, and [IKNP03] presented the first efficient protocol for doing so. Via this procedure, the computational cost is reduced to the cost of the base OT's, and hence we aim to reduce the number of base OT's as much as possible. This procedure is called OT-extension.

A q-ary protocol

As part of a course on *Cryptographic Computing* at Aarhus University, Jaron S. Gundersen and I did a course project on OT-extension using q-ary codes. More precisely, we considered the actively secure OT-extension protocol from [OOS17], and showed that the use of q-ary codes allows a significantly smaller number of base OT's than in the binary case. Afterwards, we pursued this further in collaboration with Ignacio Cascudo, which lead to [CCG18]. In this paper, we propose the protocol included as Protocol 3 on page 15, which implements m $\binom{N}{1}$-OT's using n $\binom{2}{1}$-OT's. Both sets of OT's have string length κ, which is also one of the security parameters.

The protocol relies on several components. First of all, it requires a functionality which implements the n base OT's with string length κ as mentioned above. This functionality is denoted by $\mathcal{F}^{\kappa,n}_{2\text{-OT}}$. Similarly, the resulting OT-functionality is denoted by $\mathcal{F}^{\kappa,m}_{N\text{-OT}}$. In addition, the protocol uses a pseudorandom generator, PRG, which takes a seed as input and produces a string of field elements that are indistinguishable from a uniformly sampled string. Finally, the protocol relies on a hash function with certain properties, H. The details of this function and its properties are not essential for this exposition. These may be found in [CCG18]. Note also that we use the notation $\Delta_{\mathbf{b}}$, where $\mathbf{b} \in \{0,1\}^n$. This denotes the diagonal matrix, whose i'th diagonal entry is given by b_i.

The following theorem captures the main result of our paper.

2.2.1 Theorem (CCG18; Thm. 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 $\mathcal{F}^{\kappa,n}_{2\text{-OT}}$, then Protocol 3 on page 15 implements the functionality $\mathcal{F}^{\kappa,m}_{N\text{-OT}}$.

The protocol is computationally secure against an actively corrupt adversary.

Assuming for a moment that neither participant is actively corrupt, the basic idea of the passive protocol is to use the base OT's in such a way that the sender learns a subset of randomness chosen by the receiver. The receiver will then encode his selection inputs via a linear q-ary code, and mask the actual values using the randomness unknown to the sender. Combining the encoded inputs and the outputs of the base OT's, the sender is able to construct a set of vectors, only one of which the receiver can compute. By applying the hash function H to this set of vectors, the sender obtains bit-strings suitable as one-time pads of his inputs. He can then send these to the receiver, who can decode only the ones corresponding to his inputs.

In a little more detail, the purpose of the first two phases can be seen as giving the sender one out of two shares for the matrix C containing the encoding of the receivers inputs. That is, at the end of Phase II, each column of $Q = T_0 + C\Delta_{\mathbf{b}}$ contains either $(T_0)_i$ or $(T_0 + C)_i$, where the subscript denotes the i'th column. In Phase IV, S can then construct all possible matrices T_0 that agree with his set of shares. One of these matrices – the true T_0 – is known by R since he sampled it, and this allows him to compute the messages corresponding to his selection inputs.

Protocol 3: OT-Extension [CCG18]

This protocol implements the functionality $\mathcal{F}^{\kappa,m}_{N\text{-OT}}$ having access to $\mathcal{F}^{\kappa,n}_{2\text{-OT}}$. 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 every $t \in \{n-d+1, n-d+2, \ldots, n\}$. 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 $\mathbf{w} \in \mathbb{F}_q^k$.

I. 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 $i = 0, 1$.
3. The participants call the functionality $\mathcal{F}^{\kappa,n}_{2\text{-OT}}$, where S acts as the receiver with input $\mathbf{b}$, and R acts as the sender with input pair (N_0, N_1). S receives $N = N_0 + (N_1 - N_0)\Delta_{\mathbf{b}}$, and by using PRG, he can compute $T = T_0 + (T_1 - T_0)\Delta_{\mathbf{b}}$.

II. 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 $Q = T_0 + C\Delta_{\mathbf{b}}$.

III. 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.

IV. 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 $\mathbf{v}_{\mathbf{w}_i,i} = \mathbf{y}_{\mathbf{w}_i,i} \oplus \mathsf{H}(\mathbf{t}_i)$.

If the receiver R is actively corrupt, however, it cannot be guaranteed that the matrix C sent in step 2. of phase II consists of codewords. As was argued in [IKNP03; Sec. 4], an adversary who has obtained additional knowledge – for instance via a side-channel attack – can send non-codewords to extract information about the sender's inputs. Therefore, it is necessary to include a consistency check such as the one in phase III. Here, the receiver is asked to open a linear combination of his inputs, which allows the sender to verify that the rows of C are indeed codewords. To ensure the privacy of the inputs, R must also encode a number of random dummy inputs – contained in W'' – which are used to mask his actual inputs.

In order to reduce the cost of the protocol, we show in [CCG18; Sec. 4] that our proposed protocol is still secure, even if the matrix M' used in the consistency check is chosen uniformly over a subfield. The benefit of doing so is twofold: the number of bits required to transmit M' is reduced by a factor equal to the extension degree, and the consistency check can be performed using the cheaper subfield operations.

Comparison with the binary protocol

Comparing the performance of the q-ary protocol against the binary in a precise manner is difficult. The computational cost depends heavily on the number of base OT's, but the literature does not contain good estimates on the cost of a single OT-instance. Such an estimate is also difficult because of the many different known constructions. Analysing the computational cost of our protocol is impeded by the number of parameters involved. In an attempt to still obtain a useful comparison, we argue in [CCG18] that the greatest difference in communication cost between the two protocols arises from the encoding phase. This is dominated by the term mn' in the binary case and $mn\log_2 q$ in the q-ary, where n' and n denote the number of base OT's. This leads us to consider the relative reduction in base OT's, n'/n, and the relative increase in communication cost $\log_2 q \cdot n'/n$. In the most extreme example, the use of q-ary codes more than halves the number of base OT's, while increasing the communication cost by only 33%. Table 5 shows all the examples presented in [CCG18].

Relation to reliable message transmission

In some sense, the techniques of Protocols 2 and 3 are very similar. In both cases, the input is encoded using what is essentially an interleaved code.[§] In both protocols, an interleaved codeword is sent together with additional information that can be used to determine the correctness of the codeword. In the case of Protocol 2 this extra information is the tags produced by the hash functions, and in the case Protocol 3 it is the encoding of the dummy inputs. Yet, the way the information is used to check correctness differs. Since Protocol 2 uses only a single round, the receiver can check the integrity of the interleaved codeword by performing local computations. In Protocol 3, on the other hand, the consistency check can be seen as an interactive zero-knowledge proof, thus requiring multiple communication rounds.

Implications

As already mentioned, the proposed protocol provides a trade-off between the number of base OT's used, and the total number of bits transmitted. Presumably, the binary logic in computer circuits of today is the main reason that binary arithmetic – and hence binary codes – is often considered in the literature. Theoretically, there

[§]If $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a code, the interleaved code $\mathcal{C}^{\odot m}$ is the space of $m \times n$-matrices whose rows are codewords of $\mathcal{C}$

Code	N	n (Base OT's)	d	Comparison	
				n	CC
Walsh-Had. [KK13]	256	256	128		
Juxt. simplex code over $\mathbb{F}_4$	256	170	128	$\div$ 1.51	$\times$ 1.33
Punct. Walsh-Had. [OOS17]	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 [OOS17]	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 [OOS17]	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 5. [CCG18] 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.

is no reason to steer clear of q-ary codes and restrict the theory to a computational model based on binary.

Additionally, it may be noted that even though the code used in Protocol 3 is q-ary, the used base OT's are still binary, and they are used in a black box-way. Hence, there are already a wide variety of existing constructions that can be used to implement them.

2.3 Construction of nested code pairs

As mentioned in the preliminaries, the nested code pair construction $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q$ has proved useful both in secret sharing and in the construction of asymmetric quantum codes. In both cases, the relative distances of the codes themselves, $d(\mathcal{C}_1, \mathcal{C}_2)$, and of their duals, $d(\mathcal{C}_2^\perp, \mathcal{C}_1^\perp)$, determine the properties of the associated secret sharing scheme and quantum code.

In [CG18], Olav Geil and I present two constructions of nested code pairs similar to the ones presented in [GGHR18]. In the latter paper, codes are defined from a Cartesian product of points, whereas our proposed construction is based on the Hermitian codes. We show that our new constructions lead to nested codes, whose parameters are better than previous, comparable constructions. This will become evident from Tables 6 and 7 later in this section. Bear in mind that [CG18] is a manuscript in progress, and that given references apply to its current, unfinished state. Hence, the specific proposition numbers or given examples may change in the published version.

The Hermitian codes are defined from the Hermitian function field H, whose rational places can be written as $P_1, P_2, \ldots, P_n, Q$, where $n = q^3$. Let $D = P_1 + P_2 + \cdots + P_n$ be the divisor consisting of all rational places except Q, and define the evaluation map $\text{ev}_D : H \to \mathbb{F}_{q^2}$ by $\text{ev}(f) = (f(P_1), f(P_2), \ldots, f(P_n))$, which is $\mathbb{F}_q$-linear. Given $\lambda \geq 0$, the associated Hermitian code is given by

$$\mathcal{C}_{\mathcal{L}}(D, \lambda Q) = \{\mathrm{ev}_D(f) \mid f \in \mathcal{L}(\lambda Q)\},$$

where $\mathcal{L}(\lambda Q)$ is the Riemann-Roch space associated to the divisor λQ.

The codes constructed in [CG18] are defined from a certain subset of the Weierstraß semigroup of Q denoted by $H(Q)$. This subset is given by

$$H^*(Q) = \{\lambda \in H(Q) \mid \mathcal{C}_{\mathcal{L}}(D, \lambda Q) \neq \mathcal{C}_{\mathcal{L}}(D, (\lambda - 1)Q)\},$$

and it may be shown[††] that in fact

$$H^*(Q) = \left\{iq + j(q+1) \mid 0 \le i < q^2, 0 \le j < q\right\}. \tag{2.1}$$

By the Weierstraß gap theorem [Sti09; Thm. 1.6.8], the dimensions of the Riemann-Roch spaces $\mathcal{L}((\lambda - 1)Q)$ and $\mathcal{L}(\lambda Q)$ can differ by at most one. Additionally, the dimension grows if and only if $\lambda \in H(Q)$. Hence, for each $\lambda \in H^*(Q)$ it is possible to fix an element $f_\lambda \in \mathcal{L}(\lambda Q) \setminus \mathcal{L}((\lambda-1)Q)$, which will, when combined with a basis for $\mathcal{L}((\lambda-1)Q)$, span $\mathcal{L}(\lambda Q)$ as a vector space over $\mathbb{F}_{q^2}$. By the definition of $H^*(Q)$ it now follows that the set

$$\{\mathrm{ev}_D(f_\lambda) \mid \lambda \in H^*(Q)\} \tag{2.2}$$

is linearly independent, and since $|H^*(Q)| = n$, it forms a basis of $\mathbb{F}_{q^2}$.

Using the results from [Gei03], we define the mappings $\sigma \colon H^*(Q) \to \{1, 2, \ldots, q^3\}$ and $\mu \colon H^*(Q) \to \{1, 2, \ldots, q^3\}$, given by

$$\sigma\big(iq + j(q+1)\big) = \begin{cases} q^3 - iq - j(q+1) & \text{if } 0 \le i < q^2 - q \\ (q^2-1)(q-j) & \text{if } q^2 - q \le i < q^2 \end{cases},$$

and $\mu\big(iq + j(q+1)\big) = \sigma\left((q^2 - 1 - i)q + (q-1-j)(q+1)\right)$. These can be used to bound the distances of the codes $\mathcal{C}_{\mathcal{L}}(D, \lambda Q)$, and their duals $\mathcal{C}_\Omega(D, \lambda Q)$. More precisely, the order bound for primary and dual codes [DP10; HLP98] imply that

$$d(\mathcal{C}_{\mathcal{L}}(D, \lambda Q)) \ge \min\{\sigma(\gamma) \mid 0 \le \gamma \le \lambda, \gamma \in H^*(Q)\}, \tag{2.3}$$

and

$$d(\mathcal{C}_\Omega(D, \lambda Q)) \ge \min\{\mu(\gamma) \mid \lambda < \gamma, \gamma \in H^*(Q)\}. \tag{2.4}$$

Now, letting $\lambda_1, \lambda_2 \in H^*(Q)$ with $\lambda_1 > \lambda_2$, we can use the bounds above to gain information about the relative distances of the codes:

$$\begin{aligned} d\big(\mathcal{C}_{\mathcal{L}}(D, \lambda_1 Q), \mathcal{C}_{\mathcal{L}}(D, \lambda_2 Q)\big) &\ge \min\{\sigma(\gamma) \mid \lambda_2 < \gamma \le \lambda_1, \gamma \in H^*(Q)\} \\ d\big(\mathcal{C}_\Omega(D, \lambda_2 Q), \mathcal{C}_\Omega(D, \lambda_1 Q)\big) &\ge \min\{\mu(\gamma) \mid \lambda_2 < \gamma \le \lambda_1, \gamma \in H^*(Q)\}. \end{aligned} \tag{2.5}$$

It turns out, however, by the results in [Gei03] that the inequalities in (2.3), (2.4), and (2.5) are in fact equalities in the Hermitian case, meaning that we can determine

[††]For instance by combining [Sti09; Lem. 6.4.4(e)] with [Sti09; Prop. 8.8.3(a)]

the relative and non-relative distances explicitly, without checking each individual codeword.

The way we construct the codes in [CG18] is to decide on a designed distance δ, and then only choose those codewords whose weights are guaranteed to be at least δ. Thus, the improved codes are given by

$$\tilde{E}(\delta) = \operatorname{span}_{\mathbb{F}_{q^2}} \left\{ \operatorname{ev}_D(f_\lambda) \mid \sigma(\lambda) \geq \delta \right\}$$
$$\tilde{C}(\delta) = \left(\operatorname{span}_{\mathbb{F}_{q^2}} \left\{ \operatorname{ev}_D(f_\lambda) \mid \mu(\lambda) < \delta \right\} \right)^{\perp}.$$

The results in [Gei03; Sec. 6] imply that the two codes $\tilde{E}(\delta)$ and $\tilde{C}(\delta)$ are, in fact, equal. This means that we can consider the codes both from a primary perspective using the values from σ and from a dual perspective via the values from μ.

Dimension of the codes

Since (2.2) is a basis of $\mathbb{F}_{q^2}$, the dimension of the code $\tilde{E}(\delta)$ – and hence also $\tilde{C}(\delta)$ – is exactly the number of $\lambda \in H^*(Q)$ satisfying $\sigma(\lambda) \geq \delta$. This number, however, cannot always be determined immediately from the value of δ, whence we present the following bounds on the dimension.

2.3.1 Proposition (CG18; Prop. 14):
Given $q < \delta \leq q^2 - q$, write

$$q^3 - \delta = q^3 - q^2 + aq + b(q+1),$$

where $-q < a < q$ and $0 \leq b < q$. If $a \geq 0$, then

$$\dim(\tilde{E}(\delta)) \geq q^3 - \delta - g + 1 - \sum_{s=0}^{a+b}(s+1) + a + q^2 - \lfloor \delta + \delta \ln(q^2/\delta) \rfloor.$$

If $a < 0$ then

$$\dim(\tilde{E}(\delta)) \geq q^3 - \delta - g + 1 - \sum_{s=0}^{a+b}(s+1) + q^2 - \lfloor \delta + \delta \ln(q^2/\delta) \rfloor.$$

2.3.2 Proposition (CG18; Prop. 15):
Given $1 \leq \delta \leq q$, the dimension of the code $\tilde{E}(\delta)$ satisfies

$$\dim(\tilde{E}(\delta)) \geq q^3 - \lfloor \delta + \delta \ln(\delta) \rfloor.$$

In the remaining cases, the code $\tilde{E}(\delta)$ coincides with the usual Hermitian codes, and the dimension is well-known; see for instance [Sti09].

Pairs with large codimension

Using the improved codes from above, we can construct a pair of codes $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$ such that $\mathcal{C}_1$ is the code $\tilde{E}(\delta_1)$ of designed distance δ_1, and $\mathcal{C}_2$ is $\tilde{C}(\delta_2)^{\perp}$. That is, $\mathcal{C}_2$ is the dual of a code of designed distance δ_2. In this way, we obtain a pair $\mathcal{C}_2 \subsetneq \mathcal{C}_1$ of nested codes with relative distances

$$d\left(\tilde{E}(\delta_1), \tilde{C}(\delta_2)^\perp\right) \geq \delta_1 \quad \text{and} \quad d\left(\tilde{C}(\delta_2), \tilde{E}(\delta_1)^\perp\right) \geq \delta_2 \tag{2.6}$$

by the bounds in (2.5). Furthermore, the observations below (2.5) imply that equality occur in (2.6) whenever $\delta_1, \delta_2 \in \sigma(H^*(Q))$. However, not all pairs (δ_1, δ_2) of minimal distances are suitable since we must ensure that $\tilde{C}(\delta_2)^\perp$ is indeed contained in $\tilde{E}(\delta_1)$. Determining such pairs is not a trivial exercise, and complicated formulae are needed. In [CG18], we prove such formulae.

2.3.3 Proposition (CG18; Prop. 18–22):
Consider the Hermitian curve, and let $2 \leq \delta_1 \leq q^3$. Then $\tilde{C}(\delta_2)^\perp \subseteq \tilde{E}(\delta_1)$ if and only if one of the following cases occur.

(i) $2 \leq \delta_1 \leq q$, *and* $\delta_2 \leq q^3 - (\delta_1 - 2)(q+1)$

(ii) $q < \delta_1 \leq q^2 - q$, *and*

$$\delta_2 \leq \begin{cases} q^3 - q^2 + q - \delta_1 + 2 & \text{if } 0 \leq b \leq a \\ q^3 - q^2 - a(q+1) & \text{if } b > a \end{cases},$$

where $\delta_1 - (q+1) = aq + b$.

(iii) $q^2 - q < \delta_1 \leq q^3 - 2q^2 + 2q$, *and* $\delta_2 \leq q^3 - q^2 + q + 2 - \delta_1$

(iv) $q^3 - 2q^2 + 2q < \delta_1 \leq q^3 - q^2$, *and*

$$\delta_2 \leq \begin{cases} (a+1)q + b + 2 & \text{if } b < a \\ (a+2)q & \text{if } a \leq b < q-1 \\ (a+2)q + 1 & \text{if } b = q-1 \end{cases},$$

where $q^3 - q^2 - \delta_1 = aq + b$.

(v) $q^3 - q^2 \leq \delta_1 \leq q^3$, *and*

$$\delta_2 \leq \begin{cases} a+1 & \text{if } b < a \\ a+2 & \text{if } b \geq a \end{cases},$$

where $q^3 - \delta_1 = aq + b$.

We can summarize this construction by the following proposition.

2.3.4 Proposition:
Let $\delta_1, \delta_2 \in H^*(Q)$ satisfy one of the cases in Proposition 2.3.3. Then $\tilde{C}(\delta_2)^\perp \subsetneq \tilde{E}(\delta_1)$ is a nested code pair with relative distances

$$d\left(\tilde{E}(\delta_1), \tilde{C}(\delta_2)^\perp\right) = \delta_1 = d\left(\tilde{E}(\delta_1)\right)$$

and

$$d\left(\tilde{C}(\delta_2), \tilde{E}(\delta_1)^\perp\right) = \delta_2 = d\left(\tilde{C}(\delta_2)\right).$$

The codimension is given by the cardinality of $\{\lambda \in H^*(Q) \mid \sigma(\lambda) \geq \delta_1, \mu(\lambda) \geq \delta_2\}$.

2.3.5 Example:

To illustrate the construction of Proposition 2.3.4, consider $q = 4$, and set $\delta_1 = 12$ and $\delta_2 = 5$. The nested code pair $\tilde{C}(5)^\perp \subsetneq \tilde{E}(12)$ then has codimension 40, see Figure 2. By (2.6), the relative distances are 12 and 5, respectively, meaning that the corresponding quantum code has parameters $[[64, 40, 12/5]]_{16}$. This code is pure since the relative and non-relative distances agree when using this construction.

The corresponding secret sharing scheme has secrets of size 40, and the reconstructing and privacy numbers are $r = 64 - 12 + 1 = 53$ and $t = 5 - 1 = 4$. ◄

49	45	41	37	33	29	25	21	17	13	9	5	4	3	2	1
54	50	46	42	38	34	30	26	22	18	14	10	8	6	4	2
59	55	51	47	43	39	35	31	27	23	19	15	12	9	6	3
64	60	56	52	48	44	40	36	32	28	24	20	16	12	8	4

4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
3	6	9	12	15	19	23	27	31	35	39	43	47	51	55	59
2	4	6	8	10	14	18	22	26	30	34	38	42	46	50	54
1	2	3	4	5	9	13	17	21	25	29	33	37	41	45	49

Figure 2. The nested codes from Example 2.3.5. The upper grid shows $\sigma(H^*(Q))$, and the lower $\mu(H^*(Q))$. The dark shaded region shows the elements included in $\tilde{C}(5)^\perp$, and $\tilde{E}(12)$ includes the lightly shaded region as well.

Pairs with small codimension

In [CG18], we also present a second construction of nested code pairs, which is in essence based on the same technique as found in [GGHR18; Sec. IV]. Rather than using the improved codes $\tilde{E}(\delta)$ and $\tilde{C}(\delta)$ as in the previous section, this construction is based on the usual one-point Hermitian codes $C_{\mathcal{L}}(D, \lambda Q)$ and $C_\Omega(D, \lambda Q)$. The trick is to choose these in such a way that the relative distance exceeds the minimal distance of one of the codes. In this way, the associated quantum codes are impure. For ease of notation, the codes $C_{\mathcal{L}}(D, \lambda Q)$ and $C_\Omega(D, \lambda Q)$ will be denoted by C_λ and $C_\lambda^\perp$, respectively.

2.3.6 Proposition (CG18; Prop. 27):

Let $\lambda_1 = iq + j(q+1) \in H^*(Q)$ where $i \le j < q$, and define $\lambda_2 = jq + i(q+1) - 1$. Then $C_{\lambda_2} \subsetneq C_{\lambda_1}$ is a nested code pair with codimension $\ell = j - i + 1$, whose relative distances satisfy

$$d(C_{\lambda_1}, C_{\lambda_2}) = q^3 - \lambda_1 = d(C_{\lambda_1}),$$

and

$$d(C_{\lambda_2}^\perp, C_{\lambda_1}^\perp) = (i+1)(j+1) \ge d(C_{\lambda_2}^\perp). \tag{2.7}$$

The inequality in (2.7) *is strict if and only if $i \neq 0$ and $j \neq q - 1$.*

In terms of the rectangular structure of $H^*(Q)$ given in (2.1), the above proposition considers codes in the "bottom left corner" of $H^*(Q)$. In general, this gives $d(C_{\lambda_1}, C_{\lambda_2}) \geq d(C_{\lambda_2}^\perp, C_{\lambda_1}^\perp)$, and hence it is primarily this result that is of interest to quantum coding. Yet, the same technique can be applied at the other end of $H^*(Q)$ as well, which may be beneficial for constructing secret sharing schemes.

2.3.7 Proposition (CG18; Prop. 28):
Let $\lambda_1 = (q^2-1-i)q + (q-1-j)(q+1) \in H^*(Q)$ where $i \leq j < q$, and define $\lambda_2 = (q^2-1-j)q + (q-1-i)(q+1) - 1$. Then $C_{\lambda_2} \subsetneq C_{\lambda_1}$ is a nested code pair with codimension $\ell = j-i+1$, whose relative distances satisfy

$$d(C_{\lambda_1}, C_{\lambda_2}) = (i+1)(j+1) \geq d(C_{\lambda_1}), \tag{2.8}$$

and

$$d(C_{\lambda_2}^\perp, C_{\lambda_1}^\perp) = q^3 - jq - i(q+1) = d(C_\lambda^\perp).$$

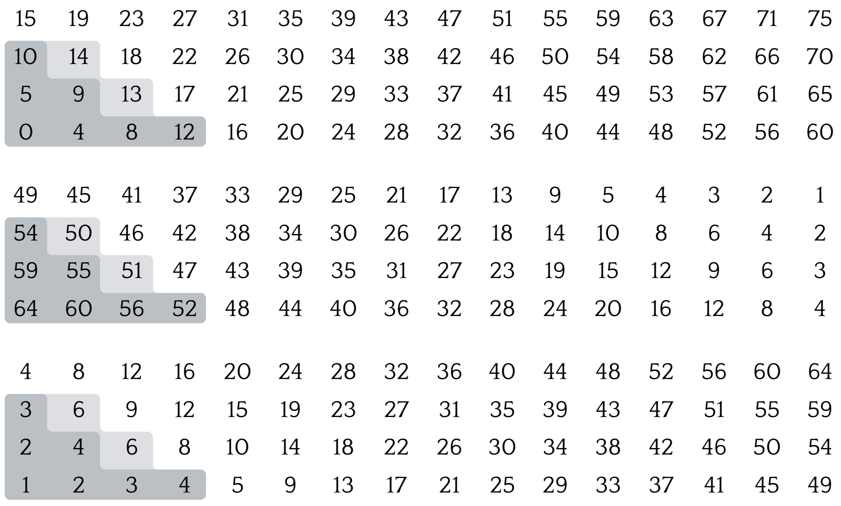
The inequality in (2.8) *is strict if $i \neq 0$ and $j \neq q-1$.*

2.3.8 Example:
As an example of the construction defined in Proposition 2.3.6, let $q = 4$ and $\lambda_1 = 1q + 2(q+1) = 14$. This implies $\lambda_2 = 2q + 1(q+1) - 1 = 12$, and the associated code pair is $C_{12} \subsetneq C_{14}$. As is seen in the top grid of Figure 3, the elements in $C_{14} \setminus C_{12}$ lie along a line of slope -1. Starting at the midpoint of this line segment and moving in either direction, the value of μ decreases. The idea of the construction is that as long as the segment corresponding to $C_{14} \setminus C_{12}$ does not stretch to the edges of $H^*(Q)$, there is an element outside C_{14} with a lower value of μ than any element inside $C_{14} \setminus C_{12}$. In the current example, this value is 4, implying that $d(C_{12}) = 4 < 6 = d(C_{14}, C_{12})$. The corresponding quantum code has parameters $[[64, 2, 50/6]]_{16}$, and by the above observations it is impure.

In the secret sharing perspective, we can share a secret of size 2 such that any $r = 64 - 50 + 1 = 15$ participants can reconstruct, and any set of at most $t = 6 - 1 = 5$ participants have no information. ◄

Comparison with existing codes

In [CG18], we also compare the two construction with existing codes from the literature. Instead of using the bounds in Propositions 2.3.1 and 2.3.2, we have computed the actual dimensions. For the first construction, we compare it with a construction from [LaG12] which gives asymmetric quantum generalized Reed-Solomon codes. The results of this comparison are seen in Table 6. For each quantum code created from the construction of [LaG12], we give up to two comparable quantum codes from the construction of Proposition 2.3.4. For the first, we fix the dimension of the code and the distance d_x, and present the code with the maximal



15	19	23	27	31	35	39	43	47	51	55	59	63	67	71	75
10	14	18	22	26	30	34	38	42	46	50	54	58	62	66	70
5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60

49	45	41	37	33	29	25	21	17	13	9	5	4	3	2	1
54	50	46	42	38	34	30	26	22	18	14	10	8	6	4	2
59	55	51	47	43	39	35	31	27	23	19	15	12	9	6	3
64	60	56	52	48	44	40	36	32	28	24	20	16	12	8	4

4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
3	6	9	12	15	19	23	27	31	35	39	43	47	51	55	59
2	4	6	8	10	14	18	22	26	30	34	38	42	46	50	54
1	2	3	4	5	9	13	17	21	25	29	33	37	41	45	49

Figure 3. Nested codes from Example 2.3.8. The upper grid shows $H^*(Q)$, and the two below show $\sigma(H^*(Q))$ and $\mu(H^*(Q))$, respectively. The dark shaded region shows the elements included in C_{12}, and C_{14} includes the lightly shaded region as well.

value of d_z. For the second, we fix both of the distances, and seek to maximize the dimension. A dash indicates that our construction cannot be used to obtain a code under the given restrictions. It may however be possible to construct a better code. For instance, we cannot construct a code with minimal distances 5/4 to compare with the code $[[27, 6, 5/4]]_9$. Instead, our construction can for instance be used to obtain a $[[27, 15, 6/4]]_9$ code, which still beats the parameters of the original.

Another set of comparable asymmetric quantum codes can be found in [EJS15]. The constructions in Propositions 2.3.4 and 2.3.6 do not improve on the parameters of these, but in almost all cases, they match the parameters in [EJS15].

Since the nested code pairs of Propositions 2.3.6 and 2.3.7 have a relative distance that exceeds one of the non-relative distances, these can be used to illustrate the advantage of analysing relative distances rather than relying on the bounds in (1.1).

Construction of [LaG12]	Ours, same dimension	Ours, same distances
$[[27, 3, 8/2]]_9$	$[[27, 3, 21/2]]_9$	$[[27, 16, 8/2]]_9$
$[[27, 3, 7/3]]_9$	$[[27, 3, 19/3]]_9$	$[[27, 15, 7/3]]_9$
$[[27, 6, 7/2]]_9$	$[[27, 6, 18/2]]_9$	$[[27, 17, 7/2]]_9$
$[[27, 3, 6/4]]_9$	$[[27, 3, 17/4]]_9$	$[[27, 15, 6/4]]_9$
$[[27, 6, 6/3]]_9$	$[[27, 6, 16/3]]_9$	$[[27, 17, 6/3]]_9$
$[[27, 9, 6/2]]_9$	$[[27, 9, 15/2]]_9$	$[[27, 19, 6/2]]_9$
$[[27, 6, 5/4]]_9$	$[[27, 6, 14/4]]_9$	-
$[[27, 9, 5/3]]_9$	$[[27, 9, 13/3]]_9$	-
$[[27, 12, 5/2]]_9$	$[[27, 12, 12/2]]_9$	-
$[[27, 12, 4/3]]_9$	$[[27, 12, 10/3]]_9$	$[[27, 19, 4/3]]_9$
$[[27, 15, 4/2]]_9$	$[[27, 15, 9/2]]_9$	$[[27, 21, 4/2]]_9$
$[[27, 18, 3/2]]_9$	-	$[[27, 23, 3/2]]_9$

Table 6. [CG18] Asymmetric quantum codes of length 27 over $\mathbb{F}_9$. The constructions in columns two and three stem from Proposition 2.3.3.

This can be done in the following way [EJS15; GGHR18]. First, fix a length n, a codimension ℓ, and a minimal distance δ_1. Then use the tables of best known codes [Gra07; SS15] to find codes $\mathcal{C}', \mathcal{C} \subseteq \mathbb{F}_q^n$ with $\dim \mathcal{C} - \dim \mathcal{C}' = \ell$, $d(\mathcal{C}) \geq \delta_1$, and $d(\mathcal{C}'^{\perp}) \geq \delta_2$. Denote the largest such δ_2 by $g(\ell, \delta_1)$. For a nested code pair $\mathcal{C}_2 \subsetneq \mathcal{C}_1 \subseteq \mathbb{F}_q^n$ of codimension ℓ, we can compare $d(\mathcal{C}_2^{\perp})$ to the value $g(\ell, d(\mathcal{C}_1))$ in order to see how it compares against the best known codes – even though these codes are in no way guaranteed to be nested. Remarkably, the construction in Proposition 2.3.6 will in most cases match the parameters of the hypothetical code pairs. By analysing the relative distances, it may even exceed the parameters of the best known codes which use only information about the non-relative distances.

Table 7 shows the possible asymmetric quantum codes from Proposition 2.3.6 and the comparison with $g(\ell, \delta_1)$ as described above. The tables of [Gra07] only contains alphabets up to $\mathbb{F}_9$, meaning that we have to rely on [SS15] for any larger field sizes. The codes in the latter, however, are usually not as optimized as the ones found in [Gra07]. For this reason, it is not clear whether the examples with $d_x > g(\ell, \delta_1)$ indicate a true improvement, or whether the tables are deficient. Yet, in either case even matching the best parameters *and* ensuring inclusion of the codes highlights the power of using Proposition 2.3.6 and its information on relative distances.

Implications

As mentioned in Sections 1.2 and 1.4, pairs of nested code give rise to secret sharing schemes and asymmetric quantum codes, where the reconstruction and privacy numbers or the minimal distances are determined by the relative distances of the used codes. The constructions given in [CG18] provide good pairs of nested codes with designed relative distances. Subsequently, the results can be applied in both of the above settings, and as shown in the previous paragraphs, the resulting quantum codes compare favourably against other known constructions.

Field	(i,j)	Parameters	$g(\ell,\delta_1)$
	$(2,2)$	$[[27,1,13/9]]_9$	9
	$(1,1)$	$[[27,1,20/4]]_9$	4
$q=3$	$(1,2)$	$[[27,2,16/6]]_9$	6
	$(0,1)$	$[[27,2,23/2]]_9$	2
	$(0,2)$	$[[27,3,19/3]]_9$	3
	$(3,3)$	$[[64,1,37/16]]_{16}$	16
	$(2,2)$	**$[[64,1,46/9]]_{16}$**	8
	$(1,1)$	$[[64,1,55/4]]_{16}$	4
	$(2,3)$	**$[[64,2,41/12]]_{16}$**	11
$q=4$	$(1,2)$	$[[64,2,50/6]]_{16}$	6
	$(0,1)$	$[[64,2,59/2]]_{16}$	2
	$(1,3)$	**$[[64,3,45/8]]_{16}$**	7
	$(0,2)$	$[[64,3,54/3]]_{16}$	3
	$(0,3)$	$[[64,4,49/4]]_{16}$	5
	$(4,4)$	$[[125,1,81/25]]_{25}$	25
	$(3,3)$	**$[[125,1,92/16]]_{25}$**	14
	$(2,2)$	$[[125,1,103/9]]_{25}$	–
	$(1,1)$	$[[125,1,114/4]]_{25}$	–
	$(3,4)$	**$[[125,2,86/20]]_{25}$**	19
	$(2,3)$	**$[[125,2,97/12]]_{25}$**	10
$q=5$	$(1,2)$	$[[125,2,108/6]]_{25}$	–
	$(0,1)$	$[[125,2,119/2]]_{25}$	–
	$(2,4)$	**$[[125,3,91/15]]_{25}$**	13
	$(1,3)$	$[[125,3,102/8]]_{25}$	–
	$(0,2)$	$[[125,3,113/3]]_{25}$	–
	$(1,4)$	$[[125,4,96/10]]_{25}$	10
	$(0,3)$	$[[125,4,107/4]]_{25}$	–
	$(0,4)$	$[[125,5,101/5]]_{25}$	6

Table 7. [CG18] In the case $q=3$, [Gra07] is used to determine $g(\ell,\delta_1)$, whereas [SS15] is used for other values of q. Note that the codes marked with bold have $\delta_2 > g(\ell,\delta_1)$.

Future Plans CHAPTER 3

3.1 Research stay abroad

In the spring of 2019, I am scheduled to do a research stay at a university abroad. Although it is still too early to make specific arrangements, we are hoping to find an agreement with Associate Professor Ryutaroh Matsumoto at Nagoya University in Japan. Matsumoto, who is also an adjunct professor of our department, is a long-time collaborator of the "Reliable and Secure Communication"-research group. With his expertise on quantum information processing, a collaboration with him is a good way to proceed with the work on quantum error correcting codes.

3.2 Research ideas

Within the near future, I expect to conclude the work on [CG18], the initial results of which were presented in Section 2.3. This may include a comparison between the codes in Tables 6 and 7 and a recent bound on the threshold gap found in [CGR18]. This bound gives information on secret sharing schemes, but since secret sharing schemes and quantum codes are related via pairs of nested codes, the bound may help to quantify how close the codes in [CG18] are to being optimal. After submitting [CG18], Olav Geil and I have discussed some further ideas within the area of quantum error correcting codes that are worth exploring. It is, however, still too early to reveal the details of this endeavour.

In recent work [CCXY18], Cascudo et al. have considered a coding theoretic alternative to the hyperinvertible matrices defined in [BTH08]. A matrix $M \in \mathbb{F}_q^{n\times n}$ is hyperinvertible if all $s \times s$-submatrices are invertible for $1 \leq s \leq n$. Such a matrix has the property that if $\mathbf{y} = M\mathbf{x}$ for $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$, and $a, b \in \mathbb{N}$ such that $a + b = n$, then any set containing a components of $\mathbf{x}$ and b components of $\mathbf{y}$ uniquely determines the remaining components of $\mathbf{x}$ and $\mathbf{y}$. What is proposed in [CCXY18] is that a code of length $2n$ and dimension n can be used to obtain a weaker, yet similar, construction over a smaller field than allowed by hyperinvertible matrices. This requires that the distance of the code, d, and that of its dual, $d^\perp$, satisfy certain properties. One candidate for finding such codes over small fields is the class of Hermitian codes, and Ignacio has suggested that some of the techniques from [CG18] may be relevant in this setting.

Before Diego Ruano became a 'Ramón y Cajal'-fellow at the University of Valladolid, we had started a joint work with Ignacio Cascudo on the topic of multicyclic codes and their squares. In multiparty computation, there is an interest in multiplicative secret sharing schemes since these allow the participants to multiply shared secrets, without revealing their values. If a scheme is constructed from a code $\mathcal{C}$, it is possible to determine if it is multiplicative or not by considering the span of all componentwise products of codewords in $\mathcal{C}$; see for instance [CCCX09; CCX11]. This span of componentwise products is what is called the *square* of $\mathcal{C}$. Especially, we are interested in codes, whose dual and whose square both have high minimal distances [CCCX09; Cor. 2]. The hope is to combine the insights in multicyclic codes from [GGHR17; GHR15] with the ideas of [Cas17] to obtain codes with the aforementioned distance properties. As of now, however, the research project initiated with Olav Geil is my main focus, and Diego, Ignacio, and myself have agreed that I shall treat the squares of multicyclic codes as a side project. In the future, time may be devoted to this research topic again.

Bibliography

[AKS06] **S. A. Aly**, **A. Klappenecker and P. K. Sarvepalli**. 'Remarkable Degenerate Quantum Stabilizer Codes Derived from Duadic Codes'. In: *2006 IEEE International Symposium on Information Theory*. July 2006, pp. 1105–1108. DOI: `10.1109/ISIT.2006.261955`.

[Bea96] **Donald Beaver**. 'Correlated Pseudorandomness and the Complexity of Private Computations'. In: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*. STOC '96. ACM, 1996, pp. 479–488. ISBN: 0-89791-785-5. DOI: `10.1145/237814.237996`.

[BPRW16] **Allison Bishop**, **Valerio Pastro**, **Rajmohan Rajaraman and Daniel Wichs**. 'Essentially Optimal Robust Secret Sharing with Maximal Corruptions'. In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*. 2016, pp. 58–86. DOI: `10.1007/978-3-662-49890-3_3`.

[BTH08] **Zuzana Beerliová-Trubíniová and Martin Hirt**. 'Perfectly-Secure MPC with Linear Communication Complexity'. In: *Theory of Cryptography*. Springer Berlin Heidelberg, 2008, pp. 213–230. ISBN: 978-3-540-78524-8. DOI: `10.1007/978-3-540-78524-8_13`.

[Cas17] **I. Cascudo**. 'On squares of cyclic codes'. In: *ArXiv e-prints* (Mar. 2017). arXiv: `1703.01267 [cs.IT]`.

[CCCX09] **Ignacio Cascudo**, **Hao Chen**, **Ronald Cramer and Chaoping Xing**. 'Asymptotically Good Ideal Linear Secret Sharing with Strong Multiplication over Any Fixed Finite Field'. In: *Advances in Cryptology - CRYPTO 2009*. Springer Berlin Heidelberg, 2009, pp. 466–486. ISBN: 978-3-642-03356-8. DOI: `10.1007/978-3-642-03356-8_28`.

[CCG18] **Ignacio Cascudo**, **René Bødker Christensen and Jaron Skovsted Gundersen**. 'Actively Secure OT-Extension from q-ary Linear Codes'. Submitted to 11th Conference on Security and Cryptography for Networks. 2018.

[CCX11] **Ignacio Cascudo, Ronald Cramer and Chaoping Xing**. 'The Torsion-Limit for Algebraic Function Fields and Its Application to Arithmetic Secret Sharing'. In: *Advances in Cryptology – CRYPTO 2011*. Springer Berlin Heidelberg, 2011, pp. 685–705. ISBN: 978-3-642-22792-9. DOI: `10.1007/978-3-642-22792-9_39`.

[CCXY18] **Ignacio Cascudo, Ronald Cramer, Chaoping Xing and Chen Yuan**. 'Amortized Complexity of Information-Theoretically Secure MPC Revisited'. Cryptology ePrint Archive, Report 2018/429. 2018. URL: `https://eprint.iacr.org/2018/429`.

[CDN15] **Ronald Cramer, Ivan Bjerre Damgård and Jesper Buus Nielsen**. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. ISBN: 978-1-107-04305-3.

[CG18] **René Bødker Christensen and Olav Geil**. 'On nested code pairs from the Hermitian curve'. Not yet published. 2018.

[CGR18] **Ignacio Cascudo, Jaron Skovsted Gundersen and Diego Ruano**. 'Improved Bounds on the Threshold Gap in Ramp Secret Sharing'. Cryptology ePrint Archive, Report 2018/099. 2018. URL: `https://eprint.iacr.org/2018/099`.

[Chr17] **René Bødker Christensen**. 'On one-round reliable message transmission'. Cryptology ePrint Archive, Report 2017/1042. Submitted to Inf. Proc. Letters. 2017. URL: `https://eprint.iacr.org/2017/1042`.

[CRSS97] **A. R. Calderbank, E. M. Rains, P. W. Shor and N. J. A. Sloane**. 'Quantum Error Correction and Orthogonal Geometry'. In: *Phys. Rev. Lett.* 78 (3 Jan. 1997), pp. 405–408. DOI: `10.1103/PhysRevLett.78.405`.

[CRSS98] **A. R. Calderbank, E. M. Rains, P. W. Shor and N. J. A. Sloane**. 'Quantum error correction via codes over GF(4)'. In: *IEEE Trans. Inform. Theory* 44.4 (1998), pp. 1369–1387. DOI: `10.1109/ISIT.1997.613213`.

[DDWY93] **Danny Dolev, Cynthia Dwork, Orli Waarts and Moti Yung**. 'Perfectly Secure Message Transmission'. In: *J. ACM* 40.1 (Jan. 1993), pp. 17–47. ISSN: 0004-5411. DOI: `10.1145/138027.138036`.

[DP10] **Iwan M. Duursma and Seungkook Park**. 'Coset bounds for algebraic geometric codes'. In: *Finite Fields and Their Applications* 16.1 (2010), pp. 36 –55. ISSN: 1071-5797. DOI: `10.1016/j.ffa.2009.11.006`.

[EJS15] **Martianus Frederic Ezerman, Somphong Jitman and Patrick Solé**. 'Xing–Ling codes, duals of their subcodes, and good asymmetric quantum codes'. In: *Designs, Codes and Cryptography* 75.1 (Apr. 2015), pp. 21–42. ISSN: 1573-7586. DOI: `10.1007/s10623-013-9885-5`.

[FW00] **Matthew Franklin and Rebecca N. Wright**. 'Secure Communication in Minimal Connectivity Models'. In: *Journal of Cryptology* 13.1 (Jan. 2000), pp. 9–30. ISSN: 1432-1378. DOI: `10.1007/s001459910002`.

[Gei03] **Olav Geil**. 'On codes from norm–trace curves'. In: *Finite Fields and Their Applications* 9.3 (2003), pp. 351 –371. ISSN: 1071-5797. DOI: `10.1016/S1071-5797(03)00010-8`.

[GGHR17] **Carlos Galindo**, **Olav Geil**, **Fernando Hernando and Diego Ruano**. 'On the distance of stabilizer quantum codes from J-affine variety codes'. In: *Quantum Information Processing* 16.4 (Mar. 2017), p. 111. ISSN: 1573-1332. DOI: `10.1007/s11128-017-1559-1`.

[GGHR18] **C. Galindo**, **O. Geil**, **F. Hernando and D. Ruano**. 'Improved Constructions of Nested Code Pairs'. In: *IEEE Transactions on Information Theory* 64.4 (Apr. 2018), pp. 2444–2459. ISSN: 0018-9448. DOI: `10.1109/TIT.2017.2755682`.

[GHR15] **Carlos Galindo**, **Fernando Hernando and Diego Ruano**. 'Stabilizer quantum codes from J-affine variety codes and a new Steane-like enlargement'. In: *Quantum Information Processing* 14.9 (Sept. 2015), pp. 3211–3231. ISSN: 1573-1332. DOI: `10.1007/s11128-015-1057-2`.

[GMW87] **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*. STOC '87. ACM, 1987, pp. 218–229. ISBN: 0-89791-221-7. DOI: `10.1145/28395.28420`.

[Go18] **The Go Project**. *Go Programming Language*. Version 1.10. Feb. 2018. URL: `https://golang.org/`.

[Gra07] **Markus Grassl**. *Bounds on the minimum distance of linear codes and quantum codes*. Online available at `http://www.codetables.de`. Accessed on 2017-11-14. 2007.

[GS98] **V. Guruswami and M. Sudan**. 'Improved decoding of Reed-Solomon and algebraic-geometric codes'. In: *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*. Nov. 1998, pp. 28–37. DOI: `10.1109/SFCS.1998.743426`.

[HLP98] **T. Høholdt**, **J. H. van Lint and R. Pellikaan**. 'Algebraic Geometry Codes'. In: *Handbook of Coding Theory*. Vol. 1. Elsevier, 1998, pp. 871–961.

[HP03] **W. Cary Huffman and Vera Pless**. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003. ISBN: 978-0-521-78280-7.

[IKNP03] **Yuval Ishai**, **Joe Kilian**, **Kobbi Nissim and Erez Petrank**. 'Extending Oblivious Transfers Efficiently'. In: *Advances in Cryptology - CRYPTO 2003:*

23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings. Springer Berlin Heidelberg, 2003, pp. 145–161. ISBN: 978-3-540-45146-4. DOI: `10.1007/978-3-540-45146-4_9`.

[IM07] **Lev Ioffe and Marc Mézard**. 'Asymmetric quantum error-correcting codes'. In: *Phys. Rev. A* 75 (3 Mar. 2007), p. 032345. DOI: `10.1103/PhysRevA.75.032345`.

[IR89] **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*. STOC '89. ACM, 1989, pp. 44–61. ISBN: 0-89791-307-8. DOI: `10.1145/73007.73012`.

[Kil88] **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. ISBN: 0-89791-264-0. DOI: `10.1145/62212.62215`.

[KK13] **Vladimir Kolesnikov and Ranjit Kumaresan**. 'Improved OT Extension for Transferring Short Secrets'. In: *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Springer Berlin Heidelberg, 2013, pp. 54–70. ISBN: 978-3-642-40084-1. DOI: `10.1007/978-3-642-40084-1_4`.

[KUM12] **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* E95.A.11 (2012), pp. 2067–2075. DOI: `10.1587/transfun.E95.A.2067`.

[LaG12] **Giuliano G. La Guardia**. 'Asymmetric quantum Reed-Solomon and generalized Reed-Solomon codes'. In: *Quantum Information Processing* 11.2 (Apr. 2012), pp. 591–604. ISSN: 1573-1332. DOI: `10.1007/s11128-011-0269-3`.

[NC10] **Michael A. Nielsen and Isaac L. Chuang**. *Quantum Computation and Quantum Information*. 10th Anniversary Edition. Cambridge University Press, 2010. ISBN: 978-1-107-00217-3.

[OOS17] **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: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14–17, 2017,* *Proceedings*. Springer International Publishing, 2017, pp. 381–396. ISBN: 978-3-319-52153-4. DOI: `10.1007/978-3-319-52153-4_22`.

[PCRS10] **Arpita Patra**, **Ashish Choudhury**, **C. Pandu Rangan and Kannan Srinathan**. 'Unconditionally reliable and secure message transmission in undirected synchronous networks: Possibility, feasibility and optimality'. In: *International Journal of Applied Cryptography* 2.2 (2010), pp. 159–197. DOI: `10.1504/IJACT.2010.038309`.

[SKR09] **Pradeep Kiran Sarvepalli**, **Andreas Klappenecker and Martin Rötteler**. 'Asymmetric quantum codes: constructions, bounds and performance'. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. Vol. 465. 2105. The Royal Society. 2009, pp. 1645–1672. DOI: `10.1098/rspa.2008.0439`.

[SS15] **Wolfgang Ch. Schmid and Rudolf Schürer**. *MinT: the online database for optimal parameters of (t,m,s)-nets, (t,s)-sequences, orthogonal arrays, linear codes, and OOAs*. 2015. URL: `http://mint.sbg.ac.at/`.

[Sti09] **Henning Stichtenoth**. *Algebraic Function Fields and Codes*. 2nd ed. Graduate Texts in Mathematics. Springer, 2009. ISBN: 978-3-540-76877-7.

[Sti94] **D.R. Stinson**. 'Universal hashing and authentication codes'. In: *Designs, Codes and Cryptography* 4.3 (1994), pp. 369–380. ISSN: 1573-7586. DOI: `10.1007/BF01388651`.

[Sud97] **Madhu Sudan**. 'Decoding of Reed Solomon Codes beyond the Error-Correction Bound'. In: *Journal of Complexity* 13.1 (1997), pp. 180 –193. ISSN: 0885-064X. DOI: `10.1006/jcom.1997.0439`.

[Yao82] **Andrew C. 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`.

Appendix A
Source Code for Parameter Search

The following pages contain the source code used to find the examples given in Tables 1–4. It is written in the Go language [Go18]. The parameters of the two protocols 1 and 2 are implemented as two separate structs, `bprwProtocol` and `pcrsProtocol`, respectively.

```go
package main

import "fmt"
import "math"
import "flag"

// The maximal field size considered. Expected to be a power of 2
const maxFieldSize = uint(128)

// logCeil returns ceil(log[2](a)). Essentially, the number of bits needed to store a
func logCeil(a uint) (res uint) {
  for a != 0 {
    res++
    a >>= 1
  }
  return res
}

// Compute the smallest q such that qb>=a
func quoCeil(a, b uint) (q uint) {
  q = a / b
  if a%b != 0 {
    q++
  }
  return q
}

// Class defining the basic RMT-problem and its parameters.
type rmtInstance struct {
  t           uint // Number of corrupt channels
  n           uint // Total number of channels
  messageSize uint // Message size in bits
  relParam    uint // -log2 of reliability parameters. Eg. 80 gives reliability 2^(-80)
}

func newRmtInstance(t, messageSize, relParam uint) *rmtInstance {
  obj := new(rmtInstance)
  obj.t = t
  obj.n = 2*obj.t + 1
  obj.messageSize = messageSize
  obj.relParam = relParam
  return obj
}

func (obj *rmtInstance) broadcastCost() uint {
  return obj.n * obj.messageSize
}

// Compute the maximal dimension using the bound from Chr17. This also implies
// a list size of 5
func (obj *rmtInstance) bprwMaxDim() uint {
  return obj.n/5 + 1
}

// Container for parameters of the BPRW16-protocol
type bprwProtocol struct {
  problem   *rmtInstance // The parameters of the problem
  fieldSize uint         // log_2 of field size
  dim       uint         // Message length
  eta       uint         // Number of evaluation points in hash
}

// minEta computes the minimal eta needed to achieve the desired
// reliability when using the protocol by BPRW16
func (obj *bprwProtocol) computeEta() {
  // Using that list size is 5
  numerator := float64(obj.problem.relParam) + math.Log2(float64(obj.problem.n*5))
  denominator := float64(obj.fieldSize) - math.Log2(float64(obj.dim))
  obj.eta = uint(math.Ceil(numerator / denominator))
}

// Returns the minimal k such that BPRW16 can encode the message
// using symbols from F_(2^k)
func (obj *bprwProtocol) fieldNeeded() (res uint) {
  res = quoCeil(obj.problem.messageSize, obj.dim)
  // Ensure that the field size is greater than n
  if i := logCeil(obj.problem.n); res < i {
    res = i
  }
  return
}

// Compute the cost of running a bprwProtocol
func (obj *bprwProtocol) bitsTransmitted() uint {
  return obj.problem.n * (1 + 2*obj.eta) * obj.fieldSize
}

// Determine if the given parameters are likely to cause overflow in bitsTransmitted()
func (obj *bprwProtocol) overflowSafe() bool {
  if logCeil(obj.problem.n)+logCeil((1+2*obj.eta))+logCeil(obj.fieldSize) >= 64 {
    return false
  }
  return true
}

func (obj *bprwProtocol) print() {
  fmt.Printf("F_(2^%d)\tDimension: %d\tEta: %d\tTransmits %d bits\n",
    obj.fieldSize, obj.dim, obj.eta, obj.bitsTransmitted())
}

```

```go
// Construct a BPRW16-instance. Returns abort if overflow is likely
func newBprwProtocol(problem *rmtInstance, dim uint) (_ *bprwProtocol, abort bool) {
  obj := new(bprwProtocol)
  obj.problem = problem
  obj.dim = dim
  obj.fieldSize = obj.fieldNeeded()
  obj.computeEta()
  if !obj.overflowSafe() {
    return obj, true
  }
  return obj, false
}

// Search for possible parameter choices when using a bprwProtocol to
// reliably send a message. Will attempt increasing the field size until
// the overflows occur
func (obj *rmtInstance) getCostsForBprw(verbose bool) *bprwProtocol {
  best, abort := newBprwProtocol(obj, obj.bprwMaxDim())
  if verbose {
    fmt.Println("BPRW16:")
    defer best.print()
  }
  if abort {
    return best
  }
  for i := obj.bprwMaxDim() - 1; i > 0; i-- {
    contender, abort := newBprwProtocol(obj, i)
    if contender.bitsTransmitted() < best.bitsTransmitted() {
      *best = *contender
    }
    if abort {
      break
    }
  }
  return best
}

// Container for PCRS10-protocols
type pcrsProtocol struct {
  problem   *rmtInstance // The parameters of the problem
  fieldSize uint         // log_2 of field size
  dimA      uint         // Number of message rows
  dimB      uint         // Number of message columns
  eta       uint         // Number of evaluation points in hash
  e         uint         // Correctable errors
}

// Compute the eta necessary to achieve the desired reliability
func (obj *pcrsProtocol) computeEta() {
  numerator := float64(obj.problem.relParam) +
    math.Log2(float64(obj.problem.t*(obj.problem.t+1))/float64(obj.e+1))
  denominator := float64(obj.fieldSize) - math.Log2(float64(obj.dimA))
  obj.eta = uint(math.Ceil(numerator / denominator))
}

// Computes the dimension A needed to transmit the message given
// the desired dimension B and field size. Returns false if the computed
// dimension is greater than the field size.
func (obj *pcrsProtocol) computeDimA() bool {
  symbolsNeeded := quoCeil(obj.problem.messageSize, obj.fieldSize)
  dimA := quoCeil(symbolsNeeded, obj.dimB)
  if logCeil(dimA) >= obj.fieldSize {
    // Ensure that dimA is smaller than the field. Otherwise, the hash family
    // provides no security
    return false
  }
  obj.dimA = dimA
  return true
}

// Construct a PCRS10-instance. Return ok=false if the number of message rows
// is greater than the field size
func newPcrsProtocol(
  problem *rmtInstance,
  fieldSize uint,
  dimB uint,
) (_ *pcrsProtocol, ok bool) {
  obj := new(pcrsProtocol)
  obj.problem = problem
  obj.fieldSize = fieldSize
  if dimB > obj.problem.t+1 {
    // Decrease dimB if it exceeds the maximal value
    obj.dimB = obj.problem.t + 1
  } else {
    obj.dimB = dimB
  }
  obj.e = obj.problem.t + 1 - dimB
  ok = obj.computeDimA()
  if !ok || !obj.overflowSafe() {
    // Either dimA is greater than n, or bits transmitted risks overflowing uint64
    return obj, false
  }
  obj.computeEta()
  return obj, true
}

func (obj *pcrsProtocol) bitsTransmitted() uint {
  return obj.problem.n * (obj.dimA + 2*obj.problem.n*obj.eta) * obj.fieldSize
}

// Determine if the given parameters are likely to cause overflow in bitsTransmitted()
func (obj *pcrsProtocol) overflowSafe() bool {
  if logCeil(obj.problem.n)+logCeil((obj.dimA+2*obj.problem.n*obj.eta))+
    logCeil(obj.fieldSize) >= 64 {
    return false
  }
  return true
}

func (obj *pcrsProtocol) print() {
  fmt.Printf("F_(2^%d)\tDimension: (%d, %d)\te: %d\tEta: %d\tTransmits %d bits\n",
    obj.fieldSize, obj.dimA, obj.dimB, obj.e, obj.eta, obj.bitsTransmitted())
}

// Compares the transmission rates of different parameter choices
// in the protocol by PCRS10
func (obj *rmtInstance) findBestPcrs(fieldSize uint) (_ *pcrsProtocol, ok bool) {
  // Start with the greatest possible value of dimB
  best, ok := newPcrsProtocol(obj, fieldSize, obj.t+1)
  if !ok {
    return best, false
  }
  for dimB := obj.t; dimB > 0; dimB-- {
    contender, ok := newPcrsProtocol(obj, fieldSize, dimB)
    if !ok {
      break
    } else if contender.bitsTransmitted() < best.bitsTransmitted() {
      *best = *contender
    }
  }
  return best, true
}

func (obj *rmtInstance) getCostsForPcrs(verbose bool) (_ *pcrsProtocol, ok bool) {
  if verbose {
    fmt.Println("PCRS10:")
  }
  best, ok := obj.findBestPcrs(maxFieldSize)
  if !ok {
    return best, false
  }
  if verbose {
    best.print()
  }
  for i := maxFieldSize >> 1; i >= logCeil(obj.n); i >>= 1 {
    contender, ok := obj.findBestPcrs(i)
    if !ok {
      break
    } else if contender.bitsTransmitted() < best.bitsTransmitted() {
      *best = *contender
    }
    if verbose {
      contender.print()
    }
  }
  return best, true
}

func (obj *rmtInstance) runComparisons() {
  fmt.Printf(
    "\nRunning RMT-comparisons with parameters\n"+
      "\tParticipants: %d\n"+
      "\tMessage size: %d\n"+
      "\tReliability:  %d\n\n",
    obj.n, obj.messageSize, obj.relParam,
  )
  fmt.Printf("Cost of broadcasting: %d bits\n\n", obj.broadcastCost())
  obj.getCostsForBprw(true)
  fmt.Println("")
  obj.getCostsForPcrs(true)
}

// Compute the message size needed for the two protocols to
// beat broadcast
func beatBroadcast(
  t, reliability uint,
) (bprw, pcrs uint, bprwProt *bprwProtocol, pcrsProt *pcrsProtocol) {
  bprwProt = new(bprwProtocol)
  pcrsProt = new(pcrsProtocol)
  // Loop over all possible message sizes. To avoid overflow in broadcastCost(),
  // we need n*messageSize<2^64
  for m := uint(1); m <= uint(1)<<(64-logCeil(2*t+1)); m <<= 1 {
    obj := newRmtInstance(t, m, reliability)
    if bprw == 0 {
      bestBprw := obj.getCostsForBprw(false)
      if bestBprw.overflowSafe() && bestBprw.bitsTransmitted() <= obj.broadcastCost() {
        bprw = m
        *bprwProt = *bestBprw
      }
    }
    if pcrs == 0 {
      bestPcrs, ok := obj.getCostsForPcrs(false)
      if !ok {
        break
      }
      if bestPcrs.overflowSafe() && bestPcrs.bitsTransmitted() <= obj.broadcastCost() {
        pcrs = m
        *pcrsProt = *bestPcrs
      }
    }
    if bprw != 0 && pcrs != 0 {
      break
    }
  }
  return
}

func main() {
  // Define and handle command line arguments
  tPtr := flag.Uint("t", 0, "Number of corrupt channels")
  mPtr := flag.Uint("message", 0, "Size of the message (in bits)")
  relPtr := flag.Uint("reliability", 80, "Reliability parameter")
  beatPtr := flag.Bool("beat", false, "Compute message size where the protocols"+
      "outperform broadcast")
  flag.Parse()
  if *tPtr == 0 {
    fmt.Print("Please specify a parameter t.")
    return
  }
  if !*beatPtr && *mPtr == 0 {
    fmt.Print("Please specify message size when flag -beat is not given. ")
    return
  }
  if *beatPtr {
    bprw, pcrs, bprwProt, pcrsProt := beatBroadcast(*tPtr, *relPtr)
    fmt.Printf("BPRW16: %d\nPCRS10: %d\n\n", bprw, pcrs)
    fmt.Println("The protocol parameters are")
    if bprw == 0 {
      fmt.Printf("BPRW16 did not outperform broadcast (for messages up to 2^%d bits)\n",
        63-logCeil(2**tPtr+1))
    } else {
      bprwProt.print()
    }
    if pcrs == 0 {
      fmt.Printf("PCRS10 did not outperform broadcast (for messages up to 2^%d bits)\n",
        63-logCeil(2**tPtr+1))
    } else {
      pcrsProt.print()
    }
    return
  }
  problem := newRmtInstance(*tPtr, *mPtr, *relPtr)
  problem.runComparisons()
}
```