PROGRAMA DE DOCTORADO EN MATEMÁTICAS

TESIS DOCTORAL:

ALGEBRAIC CODING THEORY FOR PRIVATE INFORMATION RETRIEVAL

Presentada por Şeyma Bodur para optar al grado de
Doctora por la Universidad de Valladolid

Dirigida por:
Dr. Diego Ruano
Dr. Edgar Martínez-Moro

PHD PROGRAMME IN MATHEMATICS

DOCTORAL THESIS:

ALGEBRAIC CODING THEORY FOR PRIVATE INFORMATION RETRIEVAL

Submitted by Şeyma Bodur in fulfilment of the requirements for the PhD degree by the Universidad de Valladolid

Supervised by:
Dr. Diego Ruano
Dr. Edgar Martínez-Moro

Acknowledgments

I would first like to thank my supervisors, Edgar Martínez-Moro and Diego Ruano Benito. Their support and guidance have greatly influenced how I've grown as a researcher over the years. They were always generous with their time and willing to assist me. Their encouragement made a real difference, especially in the more uncertain moments. What I appreciated most was the freedom they gave me to explore, make mistakes, and find my own way. I feel truly fortunate to have worked with them.

Next, I would like to express my gratitude to Félix Delgado de la Mata, my tutor during this period. I truly appreciate his support throughout the process.

I would also like to thank Fernando Hernando for the opportunity to collaborate on our work. Working with him was a valuable experience, and I truly appreciated his insight, clarity, and patience throughout the process.

I'm also grateful to the members of the examination committee for reviewing my thesis and for their helpful suggestions and comments. Many thanks as well to my pre-examiners for taking the time to read my work carefully and share their valuable feedback.

I'm thankful to have conducted this research at the University of Valladolid. I appreciate the friendly and supportive environment there, and I'm grateful to everyone who made my time more enjoyable, both academically and personally.

I'd especially like to thank Camilla Hollanti, who kindly hosted me at Aalto University for three months. I feel lucky to have had the chance to work with her. I always felt her support, and working with her and her group really gave me a fresh perspective on research—it was one of the most inspiring parts of this journey. I would also like to thank Ragnar Freij-Hollanti for his support and helpful discussions during my stay at Aalto.

To my family—my mother, Fatma Gul, my father, Murat, my brother, Mehmet, and my uncle, Hasan—thank you for your endless love, patience, and encouragement. Your support has been everything to me.

To my dear Matematik Kantini friends—Gizem, Gonca, Tahir, and Sahika— and Cennet, Bilge thank you for always being there with your humor, warmth, and energy. You made the tough parts easier and the good parts even better.

And to Álvaro, Dani, and Maria—thank you for making Valladolid feel like home. Our time together made this whole experience unforgettable. I'm also grateful to Adrian, Mario, Nacho, Bea, Alberto, and Elvira for their kindness and for all the good times we shared during this journey. I warmly thank Giorgia, a dear friend who made me feel supported and helped create wonderful memories not only in Valladolid but also in Rome.

This research was conducted while I was awarded a Predoctoral Research Contract at the University of Valladolid **CONTPR-2019-385**, this contract was funded by Universidad de

Valladolid and Banco Santander. I also gratefully acknowledge the financial support provided by the research projects

1. **PGC2018-096446-B-C21** *Singularidades y geometría algebraica. Semigrupos y AG-códigos correctores* – funded by the Spanish Research Agency (AEI).

2. **TED2021-130358B-I00** *secureCAT Coding theory and Algebraic Trends for Cryptography, Distributed Data Storage, Machine Learning and Quantum Information* – funded by MICIU/AEI/10.13039/501100011033 and by the "European Union Next Generation EU/PRTR"

3. **PID2022-138906NB-C21** *Singularidades. Métodos valorativos y combinatorios. Códigos algebraicos y de red.* funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

which enabled me to attend several conferences, schools, and workshops. I had access to these projects thanks to the support of my supervisors.

Şeyma Bodur
Universidad de Valladolid, June 2025

Contents

List of publications related to the thesis

1. **Ş. Bodur**, E. Martínez-Moro, D. Ruano, *Private Information Retrieval Schemes Using Cyclic Codes*, in: International Workshop on the Arithmetic of Finite Fields, Lecture Notes in Computer Science, vol. 13100, pp. 194–207, 2022.
   https://doi.org/10.1007/978-3-031-06675-1_14

   ArXiv preprint arXiv:2111.09060, 2021. https://doi.org/10.48550/arXiv.2111.09060

   Reference [8]

2. **Ş. Bodur**, E. Martínez-Moro, D. Ruano, *Single Server Private Information Retrieval Protocols With Codes Over Rings*, Journal of Algebra and its Applications.
   https://doi.org/10.1142/S021949882541012X

   ArXiv preprint arXiv:2311.04688, 2023. https://arxiv.org/abs/2311.04688

   Reference [9]

3. **Şeyma Bodur**, Fernando Hernando, Edgar Martínez-Moro, Diego Ruano, *The Schur product of evaluation codes and its application to CSS-T quantum codes and private information retrieval.* (2025) Submitted.

   ArXiv preprint arXiv:2505.10068, 2025. https://doi.org/10.48550/arXiv.2505.10068

   Reference [7]

List of Figures

List of Tables

Extended Abstract & Introduction

The increasing demand for privacy in digital communications and data access has led to the development of protocols that protect users from third-party surveillance. While numerous protocols have been proposed to ensure secure communication between users and servers, a critical limitation remains: in most cases, the server itself is fully aware of the data being accessed by the user. This exposure poses significant privacy risks, particularly when the data being accessed is sensitive or when the server is not fully trusted.

To address this limitation, *Private Information Retrieval* (PIR) protocols have been developed. These cryptographic protocols allow a user to retrieve an entry from a database without revealing to the database server which entry was retrieved [20]. PIR has numerous applications in privacy-preserving technologies, including private search engines, anonymous communication systems, and more recently, privacy-preserving queries in machine learning and artificial intelligence systems such as ChatGPT [62], which are based on large language models.

The design and analysis of PIR protocols generally follow two principal approaches, depending on the storage configuration of the data: **single-server** or **multi-server** settings. In the multi-server setting, the database is distributed across several servers, and privacy can be ensured using *information-theoretic techniques*, which guarantee privacy even against adversaries with unlimited computational power. In contrast, the single-server case cannot achieve such guarantees unless the entire database is downloaded, which is highly inefficient. Therefore, single-server PIR protocols rely on computational assumptions for privacy.

Data storage models and their role in PIR

In multi-server environments, *Data Storage Systems* (DSS) play a key role in ensuring data availability, fault tolerance, and security. Data can be either *replicated*—meaning exact copies are stored on multiple servers—or *encoded* using error-correcting codes. Replicated storage improves access speed and reliability, as any single server failure can be mitigated by retrieving data from another server. Encoded storage, on the other hand, distributes pieces of encoded information across servers, which enhances fault tolerance and storage efficiency. In such schemes, the original data can be reconstructed even in the presence of data loss or corruption.

The design of PIR protocols in multi-server settings often assumes *collusion* among a subset of servers. If up to t servers are allowed to collude, the protocol must guarantee that even the combined knowledge of those t servers reveals nothing about the user's query. This leads to the notion of *t-private PIR protocols*, which are constructed using tools from *coding theory*. Specifically, the privacy and efficiency of these protocols depend on the *parameters of linear*

codes and their *componentwise (Schur) products*.

A key result by Shah et al. [56] demonstrated that PIR privacy can be achieved if the user downloads just one bit more than the size of the requested file. However, this method requires a large number of servers. Additionally, when servers may fail or refuse to respond, PIR protocols must account for *communication between servers*. In such cases, the assumption of possible collusion becomes even more realistic and pressing.

Coding theory and PIRs

A major focus of recent research has been on designing PIR schemes using coding theory. *Linear codes*—and in particular, their *Schur products*—are central to constructing PIR schemes that are both efficient and secure. The *Schur product* of two codes is formed by taking the componentwise product of their codewords, and it plays a critical role in determining the privacy and rate of PIR protocols.

Several types of codes have been studied in this context, including *Generalized Reed–Solomon (GRS)* codes, *Reed–Muller (RM)* codes, and *cyclic codes*. GRS codes are particularly useful in multi-server PIR settings with collusion [24], where the rate of the scheme depends on the minimum distance of the Schur product between the storage and retrieval codes.

However, using GRS or other *Maximum Distance Separable (MDS)* codes requires working over large finite fields, which may hinder practical implementations. To address this, recent work has explored PIR protocols based on codes over *binary fields*, which are more suitable for implementation. For instance, [23] proposed a binary PIR scheme using *Reed–Muller codes*, which benefit from a *transitive automorphism group*—a desirable property for PIR schemes with good privacy-rate trade-offs.

Single-server PIRs and computational privacy

In the single-server setting, information-theoretic privacy is unattainable unless the entire database is retrieved. As a result, *computational PIR protocols* have been developed, relying on assumptions such as the hardness of factoring large integers [41], or the use of *fully homomorphic encryption* (FHE) [48, 32]. However, many of these schemes suffer from high computational cost or communication overhead, particularly for large databases. Moreover, cryptographic protocols based on number-theoretic assumptions are believed to be vulnerable to *quantum attacks* [57].

Recent efforts in this area have also explored *code-based computational PIR protocols*. One notable construction by Holzbaur, Hollanti, and Wachter-Zeh [37] proposed a code-based PIR protocol with low complexity. However, this protocol was later shown to be insecure under a *polynomial-time linear algebra attack* [10].

To overcome these vulnerabilities, this thesis proposes a novel *single-server PIR scheme* based on *codes over rings*, designed to resist linear algebra attacks. This new construction modifies and strengthens the earlier approach by incorporating *cyclic inner codes* over $\mathbb{Z}_m$ and *matrix-product outer codes* over $R = \mathbb{Z}_m[x]/\langle x^n - 1\rangle$, providing both efficiency and resistance to known attacks (See Chapter 3 and [9]).

Componentwise product and evaluation codes

Beyond PIR, the *Schur product of linear codes* has emerged as a powerful concept in both *classical and quantum coding theory* [52]. Various families of evaluation codes, such as *cyclic*, *Reed–Muller*, *hyperbolic*, and *toric codes*, have been studied in this context. Evaluation codes are constructed by evaluating polynomials over finite point sets, and their algebraic structure makes them suitable for PIR and quantum error correction.

This thesis focuses on the study of *monomial–Cartesian codes* and their Schur products. These codes are particularly relevant for the construction of *CSS–T quantum codes*, which are capable of fault-tolerantly implementing non-Clifford gates like the T-gate. We generalize previous results and prove that *J-affine variety codes*—a general class of evaluation codes—support efficient componentwise multiplication and subfield-subcodes (See Chapter 4 and [7]). Moreover, we also consider Cartesian codes and their Schur products in the context of *secure multi-party computation* protocols. In secure multi-party computation, the subfield subcode of the component-wise square of an evaluation code and its dual code must be taken into account. Controlling the various parameters from a single code can be quite challenging. We will present some strategies for addressing these parameters while constructing codes from well-known families of Cartesian product codes.

Thesis organization

The thesis is organized as follows:

- **Chapter 1** provides some basic notions and references on coding theory and related introductory topics in PIR schemes and quantum codes.
- **Chapter 2** focuses on multi-server PIR protocols using binary cyclic codes. We construct PIR schemes with improved privacy guarantees by carefully selecting storage and retrieval codes with optimal or near-optimal parameters, comparing our schemes with known Reed–Muller-based constructions.
- **Chapter 3** presents our single-server PIR protocol, designed to resist linear algebra attacks. The scheme utilizes codes over rings, offering computational privacy and low complexity.
- **Chapter 4** investigates the Schur product of monomial–Cartesian codes. We explore applications to both CSS–T quantum codes, multi-server PIR, and secure Multi-Party Computation protocols, including constructions with better rate-privacy trade-offs than known schemes.

Results and contributions

Private Information Retrieval Schemes Using Cyclic Codes, Chapter 2

We address the construction of t-private information retrieval (PIR) schemes in distributed storage systems with colluding servers. Building upon prior work that utilized Reed-Muller

codes, the authors explore the use of binary cyclic codes due to their transitive automorphism groups and efficient encoding and decoding properties.

Key Contributions

- **Cyclic Codes for PIR:** We show that cyclic codes can be effectively employed in PIR schemes, providing a broader set of parameters and potentially outperforming Reed-Muller-based constructions in certain scenarios.

- **Enhanced Parameter Flexibility:** By leveraging the structure of cyclic codes, the proposed schemes offer greater flexibility in choosing code parameters, which can lead to improved PIR rates and privacy guarantees.

Single Server PIR Protocols With Codes Over Rings, Chapter 3

This part introduces a novel single-server PIR protocol based on coding theory over rings, aiming to provide computational security against linear algebra attacks, a known vulnerability in previous code-based PIR schemes.

Key Contributions

- **Use of Codes Over Rings:** The protocol employs two types of codes over different rings: an inner non-free linear code used to distinguish elements added to the query matrix, and an outer code for generating the query matrix.

- **Resistance to Linear Algebra Attacks:** By utilizing non-free modules over rings, the scheme mitigates the effectiveness of rank-based attacks that exploit linear dependencies in the query structure.

- **Modular Arithmetic Implementation:** The protocol's operations are confined to modular arithmetic, enhancing computational efficiency and simplifying implementation, especially when the base ring is $\mathbb{Z}_m$.

- **Trade-off Between Rate and Security:** While the scheme may exhibit a lower PIR rate compared to some existing protocols, it offers enhanced security features, making it a viable option when privacy is paramount.

The Schur Product of Evaluation Codes and Its Application to CSS-T Quantum Codes and Private Information Retrieval, Chapter 4

In this last chapter, we study the componentwise (Schur) product of monomial-Cartesian codes by exploiting its correspondence with the Minkowski sum of their defining exponent sets. They show that J-affine variety codes are well-suited for such products, generalizing earlier results for cyclic, Reed-Muller, hyperbolic, and toric codes.

Key Contributions

- **Schur Product and Minkowski Sum Correspondence:** We establish a connection between the Schur product of evaluation codes and the Minkowski sum of their defining exponent sets, enabling a deeper understanding of the structural properties of these codes.
- **Construction of CSS-T Quantum Codes:** Utilizing the established correspondence, CSS-T quantum codes are constructed from weighted Reed-Muller codes and binary subfield-subcodes of J-affine variety codes, leading to codes with better parameters than previously known.
- **PIR Schemes for Multiple Colluding Servers:** We present Private Information Retrieval constructions for multiple colluding servers based on hyperbolic codes and subfield-subcodes of J-affine variety codes, demonstrating that they outperform existing PIR schemes in terms of efficiency and privacy guarantees.
- **Secure Multi-Party Computation:** Finally, some approaches are presented to address the parameter constraints through constructions based on well-known families of Cartesian product codes.

Resumen Extendido e Introducción

La creciente demanda de privacidad en las comunicaciones digitales y el acceso a datos ha impulsado el desarrollo de protocolos que protegen a los usuarios frente a la vigilancia de terceros. Si bien existen numerosos protocolos que garantizan una comunicación segura entre el usuario y el servidor, persiste una limitación crítica: en la mayoría de los casos, el propio servidor conoce exactamente qué datos están siendo accedidos por el usuario. Esta exposición representa un riesgo significativo, especialmente cuando se accede a información sensible o cuando el servidor no es completamente confiable.

Para abordar este problema, se han desarrollado los *Protocolos Privados de Recuperación de Información* (PIR, por sus siglas en inglés). Estos protocolos criptográficos permiten a un usuario recuperar una entrada de una base de datos sin revelar al servidor cuál fue la entrada consultada [20]. El PIR tiene numerosas aplicaciones en tecnologías orientadas a la privacidad, incluyendo motores de búsqueda privados, sistemas de comunicación anónimos y, más recientemente, consultas privadas a sistemas de inteligencia artificial como ChatGPT [62], basados en modelos de lenguaje de gran escala.

El diseño y análisis de protocolos PIR generalmente sigue dos enfoques principales, dependiendo de la configuración de almacenamiento de los datos: entornos con **un único servidor** o con **múltiples servidores**. En el caso de múltiples servidores, la base de datos está distribuida entre varios servidores, y la privacidad puede asegurarse utilizando técnicas de *teoría de la información*, que garantizan privacidad incluso frente a adversarios con capacidad de cómputo ilimitada. En cambio, en el caso de un único servidor, no es posible alcanzar este nivel de privacidad sin descargar toda la base de datos, lo cual es ineficiente. Por ello, los protocolos PIR de un solo servidor se basan en suposiciones computacionales para preservar la privacidad.

Modelos de almacenamiento de datos y su papel en PIR

En entornos de múltiples servidores, los *Sistemas de Almacenamiento de Datos* (DSS, por sus siglas en inglés) desempeñan un papel fundamental para garantizar la disponibilidad, tolerancia a fallos y seguridad de los datos. La información puede almacenarse de manera *replicada* —es decir, copias exactas en varios servidores— o *codificada*. El almacenamiento replicado mejora la velocidad de acceso y la confiabilidad, ya que si un servidor falla, otro puede proporcionar los datos. El almacenamiento codificado, por otro lado, distribuye fragmentos codificados de la información entre varios servidores, lo cual mejora la eficiencia y la tolerancia a errores.

El diseño de protocolos PIR en este contexto muchas veces asume la posibilidad de *colusión* o acuerdo entre algunos servidores. Si hasta t servidores pueden acordar compartir su información

referente a la petición del usuario, el protocolo debe garantizar que incluso el conocimiento combinado de esos t servidores no revele nada sobre la consulta del usuario. Esto da lugar a los denominados *protocolos PIR t-privados*, cuya construcción depende de herramientas de la *teoría de códigos*. En particular, la privacidad y eficiencia de estos protocolos están determinadas por los *parámetros de códigos lineales* y sus *productos componente a componente* (también conocidos como productos de Schur).

Un resultado fundamental de Shah et al. [56] muestra que es posible garantizar privacidad si el usuario descarga solo un bit más que el tamaño del archivo deseado. No obstante, este enfoque requiere muchos servidores. Además, ante posibles fallos o inactividad de algunos servidores, los protocolos PIR deben permitir cierto grado de *comunicación entre servidores*. Por tanto, resulta natural considerar que los servidores puedan *coludirse* compartiendo la información recibida del usuario.

Teoría de códigos y PIR

Un enfoque moderno en el diseño de esquemas PIR se basa en el uso de *teoría de códigos*. Los *códigos lineales*, y en particular el *producto de Schur* de dos códigos, son fundamentales para construir esquemas PIR que sean eficientes y seguros. El *producto de Schur* entre dos códigos se define como el producto componente a componente de sus palabras código, y tiene un rol clave en la determinación de la privacidad y la tasa del protocolo.

Diversos tipos de códigos han sido estudiados con este propósito, como los *códigos Generalizados de Reed–Solomon (GRS)*, los *códigos de Reed–Muller (RM)* y los *códigos cíclicos*. Los códigos GRS son particularmente útiles en escenarios con múltiples servidores que pueden coludirse [24], donde la tasa del esquema está relacionada con la distancia mínima del producto de Schur entre el código de almacenamiento y el de recuperación.

No obstante, el uso de códigos GRS u otros códigos *MDS (Máxima Distancia Separable)* requiere operar sobre cuerpos finitos grandes, lo cual dificulta la implementación práctica. Para solucionar esto, investigaciones recientes han explorado esquemas PIR sobre *campos binarios*, más apropiados para aplicaciones prácticas. Por ejemplo, [23] propuso un esquema binario basado en códigos de Reed–Muller, los cuales poseen un *grupo de automorfismos transitivo*, una propiedad deseable para esquemas PIR con buenas relaciones entre tasa y privacidad.

PIR con servidor único y privacidad computacional

En el caso de servidor único, la privacidad informacionalmente teórica no es alcanzable a menos que se descargue toda la base de datos. Por ello, se han desarrollado *protocolos PIR computacionales*, basados en supuestos como la dificultad de factorizar enteros grandes [41], o en el uso de *cifrado homomórfico completo* (FHE) [48, 32]. Sin embargo, muchos de estos esquemas presentan altos costos computacionales o de comunicación, especialmente para bases de datos grandes. Además, los protocolos criptográficos basados en suposiciones aritméticas clásicas son vulnerables a *ataques cuánticos* [57].

En años recientes, también se ha explorado la posibilidad de construir protocolos PIR computacionales usando *códigos correctores de errores*. Un ejemplo notable es el protocolo presentado por Holzbaur, Hollanti y Wachter-Zeh [37], que es eficiente y de baja complejidad. Sin

embargo, este protocolo fue atacado con éxito mediante un *ataque lineal de álgebra matricial* en tiempo polinomial [10].

Para superar estas vulnerabilidades, en esta tesis se propone un nuevo *protocolo PIR de servidor único*, basado en *códigos sobre anillos*, diseñado para resistir dichos ataques. La nueva construcción modifica y fortalece el enfoque anterior mediante el uso de un *código cíclico interno* sobre $\mathbb{Z}_m$ y un *código producto de matrices externo* sobre $R = \mathbb{Z}_m[x]/\langle x^n - 1\rangle$. Este esquema proporciona eficiencia computacional y privacidad computacional frente a ataques conocidos (Ver Capítulo 3 y [9]).

Producto Schur de códigos de evaluación

Más allá del PIR, el *producto de Schur entre códigos lineales* se ha consolidado como una herramienta poderosa tanto en la *teoría clásica de códigos* como en la *teoría cuántica de corrección de errores* [52]. Diversas familias de códigos de evaluación, como los *códigos cíclicos*, *Reed–Muller*, *hiperbólicos* y *tóricos*, han sido estudiadas bajo esta perspectiva. Los códigos de evaluación se construyen evaluando polinomios sobre conjuntos finitos de puntos, y su estructura algebraica los hace adecuados para aplicaciones en PIR y codificación cuántica.

Esta tesis estudia los *códigos monomiales–Cartesianos* y sus productos de Schur, con aplicaciones tanto en PIR como en códigos cuánticos del tipo *CSS–T*. En particular, se demuestra que los *códigos de variedades J-afines*, una clase general de códigos de evaluación, permiten multiplicaciones eficientes y compatibilidad con subcódigos en subcuerpos. (Ver Capítulo 4 y [7]). El último apartado de la tesis es relativo a los códigos cartesianos y sus productos de Schur en aplicación a los protocolos de computación multiparte. En este caso, dado un código de evaluación, los códigos que provienen de él sobre subcuerpos, su cuadrado de Schur y el dual del código deben ser tenidos en cuenta. El control de todos esos parámetros para un código dado puede presentar dificultades de optimización, por lo que se presentan algunas estrategias para mejorarlos basándose en algunas familias bien conocidas obtenidas a partir de códigos cartesianos.

Estructura de la tesis

La tesis se organiza de la siguiente manera:

- **Capítulo 1** ofrece algunas nociones básicas y referencias sobre teoría de códigos, así como temas introductorios relacionados con los esquemas PIR y los códigos cuánticos.
- **Capítulo 2** se centra en los protocolos PIR con múltiples servidores utilizando códigos cíclicos binarios. Construimos esquemas PIR con garantías de privacidad mejoradas, seleccionando cuidadosamente códigos de almacenamiento y recuperación con parámetros óptimos o casi óptimos, y comparamos nuestros esquemas con construcciones conocidas basadas en códigos de Reed–Muller.
- **Capítulo 3** presenta nuestro protocolo PIR para un solo servidor, diseñado para resistir ataques de álgebra lineal. El esquema utiliza códigos sobre anillos y proporciona privacidad computacional y baja complejidad.

- **Capítulo 4** estudia el producto de Schur de códigos monomiales–cartesianos. Exploramos aplicaciones tanto a códigos cuánticos CSS–T como a esquemas PIR con múltiples servidores y a protocolos de computación multiparte, incluyendo construcciones con mejores relaciones tasa–privacidad que los esquemas conocidos.

Resultados y contribuciones

Esquemas PIR usando Códigos Cíclicos, Capítulo 2

Este capítulo aborda la construcción de esquemas de Recuperación de Información Privada (t-PIR) en sistemas de almacenamiento distribuido con servidores coludidos. Basándose en trabajos previos que utilizaban códigos de Reed-Muller, exploramos el uso de códigos cíclicos binarios debido a sus grupos de automorfismos transitivos y a sus propiedades de codificación/decodificación eficientes.

Contribuciones principales

- **Códigos cíclicos para PIR:** Se demuestra que los códigos cíclicos pueden emplearse eficazmente en esquemas PIR, proporcionando un conjunto más amplio de parámetros y potencialmente superando a las construcciones basadas en Reed-Muller en ciertos escenarios.

- **Mayor flexibilidad en los parámetros:** Aprovechando la estructura de los códigos cíclicos, los esquemas propuestos ofrecen mayor libertad en la elección de parámetros, lo cual puede derivar en mejores tasas de PIR y garantías de privacidad mejoradas.

Protocolos PIR con un único servidor basados en códigos sobre Anillos, Capítulo 3

En esta parte se introduce un nuevo protocolo PIR de servidor único basado en teoría de códigos sobre anillos, con el objetivo de proporcionar seguridad computacional frente a ataques de álgebra lineal, una vulnerabilidad conocida en esquemas PIR basados en códigos.

Contribuciones principales

- **Uso de códigos sobre anillos:** El protocolo emplea dos tipos de códigos sobre distintos anillos: un código lineal interno no libre que permite distinguir los elementos añadidos a la matriz de consulta, y un código externo que genera dicha matriz.

- **Resistencia a ataques de álgebra lineal:** Utilizando módulos no libres sobre anillos, el esquema mitiga la efectividad de ataques basados en el rango que explotan dependencias lineales en la estructura de la matriz de consulta.

- **Implementación mediante aritmética modular:** Las operaciones del protocolo se limitan a la aritmética modular, lo cual mejora la eficiencia computacional y simplifica la implementación, especialmente cuando el anillo base es $\mathbb{Z}_m$.

- **Equilibrio entre tasa y seguridad:** Aunque el esquema puede presentar una tasa PIR inferior a la de algunos protocolos existentes, ofrece mejores características de seguridad, lo que lo convierte en una opción viable cuando la privacidad es prioritaria.

El Producto de Schur de Códigos de Evaluación y su Aplicación a Códigos Cuánticos CSS-T y Recuperación de Información Privada, Capítulo 4

Aquí se estudia el producto componente a componente (producto de Schur) de códigos monomiales-Cartesianos, aprovechando su correspondencia con la suma de Minkowski de sus conjuntos de exponentes definitorios. Se muestra que los códigos de variedad J-afin son adecuados para tales productos, generalizando resultados anteriores sobre códigos cíclicos, Reed-Muller, hiperbólicos y toricos.

Contribuciones principales

- **Correspondencia entre el producto de Schur y la suma de Minkowski:** Se establece una conexión entre el producto de Schur de códigos de evaluación y la suma de Minkowski de sus exponentes definitorios, lo que permite una comprensión más profunda de sus propiedades estructurales.
- **Construcción de códigos cuánticos CSS-T:** Aprovechando esta correspondencia, los autores construyen códigos cuánticos CSS-T a partir de códigos de Reed-Muller ponderados y de subcódigos de subcampos binarios de códigos de variedad J-afin, obteniendo así códigos con mejores parámetros que los conocidos previamente.
- **Esquemas PIR para múltiples servidores coludidos:** Se presentan construcciones de esquemas PIR para múltiples servidores coludidos basadas en códigos hiperbólicos y subcódigos de subcampos de códigos de variedad J-afin, mostrando que superan a los esquemas existentes en términos de eficiencia y garantías de privacidad.
- **Protocolos para computación segura multiparte:** Para finalizar, se presentan algunas aproximaciones concretas al diseño de protocolos seguros de computación multiparte que permiten controlar los parámetros del protocolo de forma efectiva a través del uso de familias conocidas de códigos cartesianos.

Chapter 1

Preliminaries

1.1 Coding theory

This section provides an overview of fundamental definitions related to linear codes, which play a central role in the construction of PIR schemes. The material covered is biased due to our interest; for a full treatment of linear codes, see, for example [38].

Let q be a prime power. We denote by $\mathbb{F}_q$ the finite field with q elements. A linear code C is a k-dimensional subspace of $\mathbb{F}_q^n$. We will denote G as a generator matrix of code C. A codeword is a vector in the code space that can be expressed as a linear combination of the rows of G. In other words, every codeword lies within the row span of G and the code C has q^k codewords. An information set of the linear code C is a set of k coordinates such that the elements in C restricted to that set are all the elements in $\mathbb{F}_q^k$. More precisely, $I \subseteq [n] := \{1, \ldots, n\}$, with $|I| = k$, is an information set of C if the $k \times k$ submatrix of G, whose columns are indexed by I, is a full rank matrix. If the set $I = \{1, 2, \ldots, k\}$, representing the first k positions, then G is said to be in standard form: $G = [Id_k | A_{k \times n-k}]$, where Id_k is the $k \times k$ identity matrix, and $A_{k \times (n-k)}$ is the parity part of G.

Linear codes play a crucial role in ensuring the correct message is transmitted. By encoding messages, one can correct the errors when data is sent through a noisy channel. The encoding of a message $\mathbf{m} \in \mathbb{F}_q^k$ is performed as follows:

$$\begin{aligned} \text{Enc} : \mathbb{F}_q^k &\to \mathbb{F}_q^n, \\ \mathbf{m} &\mapsto \mathbf{m}G = \mathbf{c}. \end{aligned}$$

The k symbols in $\mathbf{c}$ represent the *information symbols*, and the remaining $n-k$ symbols represent the *check symbols* that enable data recovery even if it has errors.

The *Hamming weight* of a codeword $c \in C$ is the number of entries that are different from zero, denoted by $w_H(c)$. The Hamming distance between two codewords $c, c' \in C$ is the number of entries at which the corresponding entries are different, and it is given by

$$d_H(c, c') = | \{i : c_i \neq c'_i\} | .$$

The Hamming distance between two vectors corresponds to the Hamming weight of their difference, i.e., $wt_H(c - c') = d_H(c, c')$. The minimum distance of the code C is defined as the

smallest Hamming distance between all distinct codewords in C , denoted as d :

$$d := min\{d_H(c, c') : c \neq c'\}.$$

We denote the parameters of the code C by $[n, k, d]_q$.

A $(n-k) \times n$ matrix H, of which the elements are in $\mathbb{F}_q$, is a *parity check matrix* of a code C if it satisfies $Hc^{\perp} = 0$ for all $c \in C$. The *dual code* $C^{\perp}$ consists of all vectors in $\mathbb{F}_q^n$ that are orthogonal to every codeword in the code C. In other words, a vector $d \in \mathbb{F}_q^n$ belongs to $C^{\perp}$ if and only if it satisfies $d \cdot c = 0$ for all $c \in C$. Notice that the parity check matrix of code C is a generator matrix of $C^{\perp}$.

Definition 1 (Singleton Bound [45])**.** The *Singleton bound* indicates that for any $[n, k, d]$ linear code over $\mathbb{F}_q$, the minimum distance satisfies $d \leq n-k+1$. A linear code that meets this bound with equality, that is, $d = n - k + 1$, is called a *maximum distance separable (MDS)* code.

One of the aims of this thesis is to investigate the component-wise product of several code families and its applications in cryptography and CSS-T codes. The component-wise product, also known as the Schur product or star product, is defined as follows.

Definition 2. Given two linear codes C and D of length n over $\mathbb{F}_q$, we define their *componentwise product* (or *Shur product*) $C \star D$ as the linear code in $\mathbb{F}_q^n$ spanned by the set $\{c \star d \mid c \in C, d \in D\}$, where $\star$ denotes the component-wise product $c \star d = (c_1 d_1, \ldots, c_n d_n)$. We will denote $C^{\star 2} = C \star C$.

Definition 3 (Punctured Code)**.** Let C be an $[n, k]$ linear code over $\mathbb{F}_q$. The *punctured code* of C at coordinate i denoted $C^{\bullet}(i)$, is obtained by deleting the i^{th} coordinate from all codewords of C and a generator matrix of $C^{\bullet}(i)$ is formed by removing the i^{th} column from a generator matrix of C.
More generally, puncturing can be performed on a set of coordinates $\{i_1, \ldots, i_t\}$ with $t < n$, by removing the corresponding positions from every codeword in C. The resulting code, denoted $C^{\bullet}(i_1, \ldots, i_t)$, is again a linear code over $\mathbb{F}_q$ with parameters $[n-t, k^{\bullet}, d^{\bullet}]$ where $k^{\bullet} \geq k - t$ and $d^{\bullet} \geq d - t$.

Example 4. Consider a binary $[5, 3, 2]$ linear code C with the generator matrix G given by:

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

By puncturing the coordinates $\{1, 2\}$ of the code C, we obtain the punctured code $C^{\bullet}(1, 2)$ which has parameters $[3, 2, 1]$ and the following generator matrix:

$$G^{\bullet} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Definition 5 (Shortened Code)**.** Let C be an $[n, k]$ linear code over $\mathbb{F}_q$ and let $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ be a set of coordinate positions. To construct the *shortened code* of C, we first consider the set of codewords in C that are zero at all positions indexed by $\{i_1, \ldots, i_t\}$. Then, we remove these coordinates from each of the selected codewords.

The resulting code, denoted $C_{\bullet}(i_1, \ldots, i_t)$, is called the shortened code of C with respect to $\{i_1, \ldots, i_t\}$, and it has length $n_{\bullet} = n - t$.

Example 6. Consider again the $[5, 3, 2]$ binary linear code C in Example 4. We now construct a shortened code $C_\bullet(1, 2)$ which has parameters $[3, 1, 2]$ and the following generator matrix:

$$G_\bullet = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}.$$

Shortened and punctured codes are closely related, as illustrated by the following identity:

Theorem 7. *Let C be an $[n, k, d]$ linear code over $\mathbb{F}_q$, and let $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ be a set of coordinates. Then the following equality holds:*

$$(C^\perp)^\bullet(i_1, \ldots, i_t) = (C_\bullet(i_1, \ldots, i_t))^\perp \quad \textit{and} \quad (C^\perp)_\bullet(i_1, \ldots, i_t) = (C^\bullet(i_1, \ldots, i_t))^\perp.$$

Example 8. In light of Theorem 7, if we consider Examples 4 and 6, we observe that

$$(C^\perp)^\bullet(1, 2) = (C_\bullet(1, 2))^\perp \quad \text{and} \quad (C^\perp)_\bullet(1, 2) = (C^\bullet(1, 2))^\perp,$$

Let $C \subseteq \mathbb{F}_q^n$ be a linear code of length n. Let S_n denote the symmetric group on the set $\{1, 2, \ldots, n\}$. The group S_n acts on C by permuting the coordinate positions of codewords. More precisely, for a permutation $\pi \in S_n$, the action is defined as

$$\pi(\boldsymbol{c}) = \left(c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(n)}\right).$$

A permutation $\pi \in S_n$ is said to preserve the code C if $\pi(C) = C$, which means that it maps each codeword to another codeword in C. The set of all such permutations forms a subgroup of S_n, commonly referred to as the *permutation automorphism group* of the code, and is denoted by

$$\mathrm{Aut}(C) = \{\pi \in S_n \mid \pi(C) = C\}.$$

This group is usually referred to as the *automorphism group* of the code, since it consists of all coordinate permutations that preserve the code structure.

Theorem 9 ([38, Theorem 1.6.4])**.** *Let C be a linear code over $\mathbb{F}_q$. Then* $\mathrm{Aut}(C) = \mathrm{Aut}(C^\perp)$.

Definition 10. A code C is called transitive if its automorphism group acts transitively on all of its codewords.

Theorem 11 ([38, Theorem 1.6.6(i)])**.** *Let C be an $[n, k, d]$ linear code over $\mathbb{F}_q$. If the permutation automorphism group* $\mathrm{Aut}(C)$ *is transitive. Then, for any coordinate position $i, j \in \{1, \ldots, n\}$, the punctured code $C^\bullet(i)$ is permutation equivalent to the punctured code $C^\bullet(j)$.*

1.1.1 Reed-Muller Codes

The binary r^{th} order *Reed-Muller code*, denoted by $RM(r, m)$, is defined as

$$RM(r, m) = \{ev(f) \mid f \in \mathbb{F}_2[x_1, ..., x_m],\ deg(f) \leq r\}, \tag{1.1}$$

where $ev(f)$ is the evaluation of f at all points in $\mathbb{F}_2^m$ in a given order.

Remark 12. $RM(r, m)$ is a linear code of length $n = 2^m$, dimension $k = \sum_{i=0}^{r} \binom{m}{i}$, and minimum distance 2^{m-r} [45].

- The code $RM(0,m)$ is a repetition code that is generated by the all-one vector.
- The code $RM(m,m)$ contains all possible binary vectors of length 2^m, which means that it spans the entire space $\mathbb{F}_2^n$.
- It is well known fact that

$$RM(r_1,m) \star RM(r_2,m) = RM(r_1+r_2,m),$$

where $r_1+r_2 \leq m$.

Definition 13. The shortened code at the position given by the evaluation at the point $\mathbf{0}$ of a binary Reed-Muller code is denoted by the linear $[2^m-1, k-1, 2^{m-r}]$ code $C_\bullet$. The punctured code at the position evaluating at $\mathbf{0}$ of a binary Reed-Muller code is denoted by the linear $[2^m-1, k, 2^{m-r}-1]$ code $C^\bullet$.

Note that it is easy to check that Reed-Muller codes $RM(r,m) \subseteq \mathbb{F}_q^m$, are transitive codes for all m and all $r \leq m$.

1.1.2 Cyclic Codes

In this section, we will be concerned with basic cyclic code definitions and computing the star product of two cyclic codes.

Definition 14. A $[n,k]$ linear code C is said to be a *cyclic code* if for any codeword $(c_0, c_1, \ldots, c_{n-1}) \in C$, its cyclic shift is also codeword in C, that is $(c_{n-1}, c_0, \ldots, c_{n-2}) \in C$.

Theorem 15 ([42])**.** *A linear code C is a cyclic code if and only if C is isomorphic, as a $\mathbb{F}_q$-linear spaces, to an ideal in the ring $R_n = \mathbb{F}_q[x]/\langle x^n-1\rangle$.*

If the ring R_n is a semisimple algebra over $\mathbb{F}_q$, we will require non-repeated roots for the polynomial x^n-1 and hence there will be a 1-1 correspondence between its factors and its roots. From now on we will require that $\gcd(n,q)=1$, which ensures semisimplicity.

Definition 16. Let C be a cyclic code in R_n. We call $g(x)$ a *generator polynomial* of C if $C = \langle g(x)\rangle$. Clearly, $g(x)$ must be a divisor of x^n-1 in $\mathbb{F}_q[x]$.

Definition 17. A set $J \subseteq \{0,\ldots,n-1\}$ is said to be *the defining* set of $C = \langle g(x)\rangle$ if

$$J = \{\, j \in \mathbb{Z}/n\mathbb{Z} \mid g(\alpha^j) = 0 \,\}.$$

A set I is called the *generating set* of $C = \langle g(x)\rangle$ if

$$I = \{\, j \in \mathbb{Z}/n\mathbb{Z} \mid g(\alpha^j) \neq 0 \,\},$$

where α is a primitive element of $\mathbb{F}_q$.

Remark 18. Let $g(x)$ be a generator polynomial of cyclic code C then $g(x) = \prod_{j\in J}(x-\alpha^j)$ and

$$g(x) = \frac{x^n-1}{\prod_{i\in I}(x-\alpha^i)}.$$

Furthermore, one has that $dim(C) = n - |J| = |I|$.

Remark 19. Assume that J is the defining set of the code C, then the generator polynomial of $C^{\perp}$ is $h(x) = \prod_{i \in -I}(x - \alpha^i)$, where $-I$ is the set of additive inverses in $\mathbb{Z}/n\mathbb{Z}$ of the elements in I.

Definition 20. The *cyclotomic coset* containing s, denoted by U_s, is defined to be the set

$$\{s, sq, \ldots, sq^i\} \pmod{n}$$

where i is the smallest integer such that $q^i \equiv 1 \pmod{n}$.

We have the following result about the star product of cyclic codes.

Theorem 21. *Let I_1 and I_2 be the generating sets of the cyclic codes C and D, respectively. The star product of $C \star D$ is generated by*

$$g_{C\star D} = \frac{x^n - 1}{\prod_{j \in I_1 + I_2}(x - \alpha^j)}, \tag{1.2}$$

where $+$ denotes the Minkowski sum on sets, that is,

$$I_1 + I_2 := \{i_1 + i_2 \mid i_1 \in I_1,\ i_2 \in I_2\}.$$

Proof. We will follow a similar way to [17, Theorem III.3], which proves this result for $C \star C$. It is well known from [6] that a cyclic code can be defined as follows, consider $\mathbb{K}$ the extension field of $\mathbb{F}_q$ such that $x^n - 1$ splits in linear factors in $\mathbb{K}[x]$. For a set $M \subseteq \{1, \ldots, n-1\}$ let $\mathcal{B}(M)$ be the $\mathbb{K}$-vector space

$$\mathcal{B}(M) = \left\{ (f(\alpha^0), f(\alpha^1), \ldots, f(\alpha^{n-1})) \mid f = \sum_{i \in M} f_i x^i \in \mathbb{K}[x] \right\}.$$

For a cyclic code C with defining set I, as a byproduct of Delsarte's theorem, one has that C is equal to the subfield subcode $\mathcal{B}(-I)|_{\mathbb{F}_q^n} = B(-I) \cap \mathbb{F}_q^n$ (see [17, Lemma 5]). Now note that the vector space obtained by the extension of scalars of C, denoted by $\mathbb{K} \otimes C$, is a $\mathbb{K}$-cyclic code with the same dimension as $\mathcal{B}(-I)$ (given by $|I|$) and, henceforth $(\mathbb{K} \otimes C) = \mathcal{B}(-I)$. Note that the extension by scalars commutes with the star product (see [52, Lemma 2.23]) thus it is clear that $C \star D = (\mathbb{K} \otimes (C \star D))|_{\mathbb{F}_q^n} = (\mathbb{K} \otimes C \star \mathbb{K} \otimes D)|_{\mathbb{F}_q^n} = (\mathcal{B}(-(I_1)) \star \mathcal{B}(-(I_2)))\,|_{\mathbb{F}_q^n} = \mathcal{B}(-(I_1 + I_2))|_{\mathbb{F}_q^n}$. □

Proposition 22 (BCH Bound)**.** *Let J be a defining set of a cyclic code C with minimum distance d. If J contains $\delta - 1$ consecutive elements $\{i, \ldots, i + \delta - 2\} \subseteq J$, where $i, \delta \in \mathbb{Z}/n\mathbb{Z}$, then $d \geq \delta$.*

Example 23. Set $q = 2$, $n = 31$. Let I_C be a generating set and let J_C be a defining set of the code C. The first cyclotomic cosets (modulo 31) are:

$$U_0 = \{0\},\ U_1 = \{1, 2, 4, 8, 16\},\ U_3 = \{3, 6, 12, 17, 24\},\ U_5 = \{5, 9, 10, 18, 20\}.$$

Consider C the cyclic code with defining set $J_C = U_0 \cup U_1 \cup U_3$. One has that J_C contains $\{0, 1, 2, 3, 4\}$, thus the BCH bound of C is equal to 6. The dimension of C is equal to $k = |I_C| = 31 - 11 = 20$. Therefore, the parameters of this code are $[31, 20, \geq 6]$. The Minkowski sum of J_C and J_C is given by:

$$J_C + J_C = U_0 \cup U_1 \cup U_3 \cup U_5.$$

Thus, one can determine that the parameters of $C \star C$ are $[31, 15, \geq 8]$.

Note that even though a RM code C is not cyclic, $C_\bullet$ and $C^\bullet$ are cyclic codes [65]. Cyclic codes are invariant under the action of the n-cycle $\sigma = (1\ 2\ \ldots\ n)$, it follows that $\sigma \in \mathrm{Aut}(C)$, and therefore the group $\mathrm{Aut}(C)$ is transitive. That is, for any $i, j \in \{1, 2, \ldots, n\}$, there exists a permutation in $\mathrm{Aut}(C)$ mapping i to j.

1.1.3 Quasicyclic Codes

A linear code $C \subseteq \mathbb{F}_q^n$ of length $n = n_0\ell$ is called *quasi-cyclic* of the index ℓ if shifting a codeword by ℓ positions is again a codeword in C. The code C can be represented by generator matrices formed of ℓ circulant blocks, each of size $n_0 \times n_0$, where each circulant block is a matrix whose rows are cyclic shifts of its first row, such that

$$\begin{bmatrix} c_0 & c_1 & \cdots & c_{n_0-1} \\ c_{n_0-1} & c_0 & \cdots & c_{n_0-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{bmatrix}.$$

Example 24. Consider the following generator matrix of a binary $[10, 5]$ quasi-cyclic code with $\ell = 2$:

$$G = \begin{bmatrix} C_{1,1} & C_{1,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

where $C_{1,1}$, $C_{1,2}$ are circulant matrices. This matrix consists of two circulant submatrices of size 5×5.

Note that quasi-cyclic codes can be seen as a generalization of cyclic codes; in particular, when $\ell = 1$, it is a cyclic code.

1.1.4 Matrix Product Codes

Let $C_1, \ldots, C_s \subset \mathbb{F}_q^n$ be linear codes of length n and dimension k_i, and let M be an $s \times \ell$ matrix with entries in $\mathbb{F}_q$. The *matrix-product code* $C = [C_1, .., C_s]M$ is defined as the set of all products $[c_1, .., c_s]M$ where $c_i \in C_i$ for $i \in \{1, .., s\}$. Note that the code C has length $n\ell$, and if the matrix M has dimension s, then C has dimension $\mathrm{k_1} + \cdots + \mathrm{k_s}$.

For instance, the well-known *Plotkin construction* for two codes of the same length C_1, C_2, $(u \mid u + v)$, is given by the matrix

$$M = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

as the matrix product code $[C_1, C_2]M$. Consider C_1 being a $[n, k_1, d_1]$ code and C_2 an $[n, k_2, d_2]$ code. Applying the $(u \mid u + v)$ construction yields a new code of length $2n$, dimension k_1k_2, and minimum distance $\min\{2d_1, d_2\}$. This construction also provides a recursive method for building binary Reed–Muller codes.

1.1.5 Berman Codes

The r^{th} order *Berman code*, denoted by $B_n(r,m)$, is a binary linear codes of length n^m, where $n \geq 2$, $m \geq 1$, and $0 \leq r \leq m$. Each codeword in $B_n(r,m)$ is constructed by concatenating n vectors from $B_n(r-1,m-1)$, such that their coordinate-wise sum belongs to $B_n(r,m-1)$:

$$B_n(r,m) = \left\{ (\mathbf{v}_0 \mid \mathbf{v}_1 \mid \cdots \mid \mathbf{v}_{n-1}) : \mathbf{v}_\ell \in B_n(r-1,m-1) \text{ for all } \ell,\ \sum_{\ell=0}^{n-1} \mathbf{v}_\ell \in B_n(r,m-1) \right\}.$$

When $r = m$, the code consists of the zero vector in $\mathbb{F}_2^{n^m}$. The dimension of $B_n(r,m)$ is given by

$$\dim(B_n(r,m)) = \sum_{w=r+1}^{m} \binom{m}{w} (n-1)^w,$$

and its minimum distance is 2^{r+1}. The dual code, denoted $DB_n(r,m)$, has dimension

$$\dim(DB_n(r,m)) = \sum_{w=0}^{r} \binom{m}{w} (n-1)^w,$$

and minimum distance n^{m-r}. The recursive definition of $DB_n(r,m)$ is given by

$$DB_n(r,m) = \{(\mathbf{u}_0 + \mathbf{u} \mid \mathbf{u}_1 + \mathbf{u} \mid \cdots \mid \mathbf{u}_{n-2} + \mathbf{u} \mid \mathbf{u}) : \mathbf{u}_\ell \in DB_n(r-1,m-1),\ \mathbf{u} \in DB_n(r,m-1)\}.$$

When $r = m$, the code is the full space $\mathbb{F}_2^{n^m}$; when $r = 0$, it is a repetition code. The automorphism groups of the Berman code and Dual Berman code are transitive [49]. Note that Reed-Muller codes are a special case of Berman code when $n = 2$.

1.1.6 (Affine variety) Monomial Cartesian codes and J-affine variety codes

Let $\mathbb{F}_q$ be the finite field with q elements, a power of the prime p. For $j = 1, \ldots, m$, let $Z_j \subseteq \mathbb{F}_q$, with $z_j = \#Z_j \geq 2$. We consider the set $Z = Z_1 \times \cdots \times Z_m \subseteq \mathbb{F}_q^m$. Consider now $E_j = \{0, 1, \ldots, z_j - 1\}$, for $j = 1, \ldots, m$, and $E = E_1 \times \cdots \times E_m \subseteq \mathbb{F}_q^m$. Note that $n = \#Z = \#E = \prod_{j=1}^m z_j$. For any $\boldsymbol{e} = (e_1, \ldots, e_m) \in E$, set $X^{\boldsymbol{e}} = X_1^{e_1} \cdots X_m^{e_m}$.

Let I be the ideal of $\mathbb{F}_q[X_1, \ldots, X_m]$ generated by the polynomials $Q_j(X_j) = \prod_{\beta \in Z_j}(X_j - \beta)$, for $j = 1, \ldots, m$. Note that the field equations $X_j^q - X_j \in I$, for all $j = 1, \ldots, m$. We consider the quotient ring $R := \mathbb{F}_q[X_1, \ldots, X_m]/I$, then any class of polynomials a unique representative of the form

$$f(X_1, \ldots, X_m) = \sum_{\boldsymbol{e} \in E} f_{\boldsymbol{e}} X^{\boldsymbol{e}} = \sum_{(e_1, \ldots, e_m) \in E} f_{e_1, \ldots, e_m} X_1^{e_1} \cdots X_m^{e_m},$$

with $f_{e_1,\ldots,e_m} \in \mathbb{F}_q$. Abusing notation, we will refer to f as a polynomial in R. Let the zero locus of the ideal I be equal to $Z = \{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n\}$, with the points of Z ordered in a specific way. Consider the linear evaluation map given by

$$\mathrm{ev}_Z \colon R \longrightarrow \mathbb{F}_q^n, \qquad \mathrm{ev}_Z(f) = (f(\boldsymbol{P}_i))_{i=1,\ldots,n}.$$

It is well-defined and bijective. For $\Delta \subseteq E$, the *(Affine variety) Monomial Cartesian code (MCC)* $C_{\Delta,Z}$ is the image of $\{f \in R \mid \mathrm{supp}(f) \subseteq \Delta\}$ under the evaluation map ev_Z, that is,

$$C_{\Delta,Z} := \mathrm{Span}\{\mathrm{ev}_Z(X^{\boldsymbol{e}}) \mid \boldsymbol{e} \in \Delta\} \subseteq \mathbb{F}_q^n.$$

Since the evaluation map is injective, the dimension of $C_{\Delta,Z}$ is equal to the cardinality of Δ.

The minimum distance of $C_{\Delta,Z}$, $d(C_{\Delta,Z})$, can be estimated mainly in two different ways. The first way is by the foot-print bound [30], which follows from considering a Gröbner basis of the ideal I [21]:

$$d(C_{\Delta,Z}) \geq \delta_{\mathrm{FB}}(C_{\Delta,Z}) = \min\left\{\prod_{i=1}^{m}(z_j - e_j) : \boldsymbol{e} \in \Delta\right\}. \tag{1.3}$$

The second one is considering the multiplicative nature of the minimum distance of monomial Cartesian codes [58, 43]. Let $\Delta \subseteq \Delta_1 \times \cdots \times \Delta_m$, then

$$d(C_{\Delta,Z}) \geq \prod_{i=1}^{m} d(C_i), \tag{1.4}$$

where $C_i = C_{\Delta_i,Z_i}$ is a uni-variate evaluation code. Hyperbolic codes [31], and in general monomial-decreasing Cartesian codes (see definition 32), are optimal for the foot-print bound [28]. For the multiplicative bound, are optimal codes coming from Δ sets that are a Cartesian product ($\Delta = \Delta_1 \times \cdots \times \Delta_m$) [44]. We introduce now these families of codes.

Consider $s \geq 0$ and let $\Delta = \{\boldsymbol{e} \in \{0,\ldots,q-1\}^m : e_1 + \cdots + e_m \leq s\}$ and $Z = \mathbb{F}_q^m$, then the code $C_{\Delta,Z}$ is a *Reed-Muller code* and it is denoted by $\mathrm{RM}_q(s,m)$. The minimum distance of a Reed-Muller code is equal to $(q-b)q^{m-1-a}$, where $s = a(q-1)+b$, with $0 \leq b \leq q-1$ [29]. Let $s, s_1, \ldots, s_m > 0$ and $\Delta = \{\boldsymbol{e} \in \{0,\ldots,q-1\}^m : s_1e_1 + \ldots + s_me_m \leq s\}$. Then, the code $C_{\Delta,Z}$ is a *weighted Reed-Muller*, with $\mathcal{S} = (s_1,\ldots,s_m)$, and we denote it by $\mathrm{WRM}_q(s,m,\mathcal{S})$. If $s_1 = \ldots = s_m = 1$, then the $\mathrm{WRM}_q(s,m,\mathcal{S})$ is a Reed-Muller code, $\mathrm{RM}_q(\lfloor s \rfloor, m)$. Finally, let $s \geq 0$ and set $\Delta = \{\boldsymbol{e} \in \{0,\ldots,q-1\}^m : (q-e_1)\cdots(q-e_m) \geq s\}$ and $Z = \mathbb{F}_q^m$, then the code $C_{\Delta,Z}$ is known as a *hyperbolic code* [31]. The foot-print bound is sharp for these families of codes. Actually, one has that the footprint bound is sharp if and only if all the elements $(\beta_1,\ldots,\ldots,\beta_m)$ with $0 \leq \beta_i \leq \alpha_i$ belong to Δ, where $\delta_{\mathrm{FB}}(C_{\Delta,Z}) = \prod_{i=1}^{m}(q-\alpha_i)$ [28], these codes are known as monomial-decreasing Cartesian codes.

We denote $C^{\star 2} = C \star C$. The *Minkowski sum* of two sets $A, B \subset \mathbb{N}^m$ is equal to

$$A + B = \{\boldsymbol{a} + \boldsymbol{b} = (a_1 + b_1, \ldots, a_m + b_m) \mid \boldsymbol{a} \in A, \boldsymbol{b} \in B\}.$$

There is a strong connection between the componentwise product of evaluation codes and the Minkowski sum. Indeed, in one direction one has that $\mathrm{ev}_Z(f_1) \star \mathrm{ev}_Z(f_2) = \mathrm{ev}_Z(f_1 f_2)$, if $f_1 f_2 \in E$. However, on the other side if $f_1 f_2 \notin E$, then we have to reduce this polynomial by the ideal I and this reduction is not explicit, in general. Actually, for $\mathbf{e} \notin E$, the evaluation $\mathrm{ev}_Z(X^{\mathbf{e}})$ is equivalent to $\mathrm{ev}_Z(h)$ where $X^{\mathbf{e}} \equiv h \mod I$ and the $\mathrm{supp}(h) \subset E$. Nevertheless, we do not know the $\mathrm{supp}(h)$ for the reduced polynomial of $X^{\mathbf{e}}$ modulo I, in general.

However, the product of polynomials can sometimes be explicitly reduced by I, that is, we may reduce the Mikowski sum of $\Delta_1 + \Delta_2$ in E, as can be found in the literature for the componentwise product of evaluation codes. For instance, this is the case when $Z = (\mathbb{F}_q^*)^m$, $E = \{0,\ldots,q-2\}$, and the codes are called *toric codes* [47, 34]. In this later case of codes, the ideal I is generated by $X_j^{q-1} - 1$, for $j = 1,\ldots,m$. Hence, for $\mathbf{e} \in \mathbb{N}^m$ we have that $\mathrm{ev}_Z(X^{\mathbf{e}}) = \mathrm{ev}_Z(X^{\overline{\mathbf{e}}})$, where $\overline{\mathbf{e}} = (e_1 \mod q-1, \ldots, e_m \mod q-1)$ and the reduction modulo $q-1$ is taken in $\{0,\ldots,q-2\}$. That is, $\overline{\mathbf{e}} \in \mathbb{Z}_{q-1}^m = \{0,\ldots,q-2\}^m$. Therefore, if we define $\overline{A} = \{\overline{\mathbf{a}} \mid \mathbf{a} \in A\} \subseteq E$, for $A \subset \mathbb{N}^m$, then $C_{\Delta_1,(\mathbb{F}_q^*)^m} \star C_{\Delta_2,(\mathbb{F}_q^*)^m}$ is equal to $C_{\overline{\Delta_1+\Delta_2},(\mathbb{F}_q^*)^m}$, where $\overline{\Delta_1+\Delta_2} \subset E$.

Another known case is $Z = E = \mathbb{F}_q^m$. For instance, for Reed-Muller and Hyperbolic codes [28]. Now the reduction is a bit more involved but still explicit. The ideal I is generated by the field equations $X_j^q - X_j$, for $j = 1, \dots, m$. Hence, for $\mathbf{e} \in \mathbb{N}^m$ we have that $\mathrm{ev}_Z(X^{\mathbf{e}}) = \mathrm{ev}_Z(X^{\overline{\mathbf{e}}})$, where $\overline{\mathbf{e}} = (e_1 \mod q-1, \dots, e_m \mod q-1)$ and the reduction modulo $q-1$ is taken in $\{1, \dots, q-1\}$ if $e_i > 0$ and it is not reduced otherwise. For instance, $0 \mod q-1 = 0$, $q-1 \mod q-1 = q-1$, and $q \mod q-1 = 1$. Thus, $\overline{\mathbf{e}} \in \{\{0\} \cup \mathbb{Z}_{q-1}\}^m = \{\{0\} \cup \{1, \dots, q-1\}\}^m$. If we define now $\overline{A} = \{\overline{\mathbf{a}} \mid \mathbf{a} \in A\} \subseteq E$, for $A \subset \mathbb{N}$. And we have that $C_{\Delta_1, \mathbb{F}_q^m} \star C_{\Delta_2, \mathbb{F}_q^m}$ is equal to $C_{\overline{\Delta_1 + \Delta_2}, \mathbb{F}_q^m}$, where $\overline{\Delta_1 + \Delta_2} \subset E$.

1.1.7 J-Affine variety codes

In this section, we aim to consider (Affine variety) Monomial Cartesian codes where one can find an explicit reduction of $\Delta_1 + \Delta_2$ in E. That is, we extend the previous two cases considered in the literature [34, 28]. We propose to consider J-affine variety codes, and their subfield subcodes, that have been successfully used to obtain quantum codes [27, 25], and LCD codes [26], with excellent parameters. This family of codes considers sets Z with a cyclic structure that is well posed for considering subfield subcodes, and that, moreover, they can also be useful for computing their componentwise product. The results concerning the componentwise product in this section are new.

Let us introduce J-affine variety codes. Fix m integers $N_j > 1$ such that $N_j - 1$ divides $q-1$ for $j = 1, \dots, m$. Consider a subset $J \subseteq \{1, \dots, m\}$ and the ideal I_J in $\mathcal{R}$ generated by the binomials $X_j^{N_j} - X_j$ when $j \notin J$ and by $X_j^{N_j - 1} - 1$ otherwise. We have that Z_j is the zero locus of the corresponding binomial, and we have hence defined the set Z. Note that the j-th coordinate, for $j \in J$, of the points in Z_J is different from zero, and the length is given by $n_J = \prod_{j \notin J} N_j \prod_{j \in J} (N_j - 1)$. Moreover, denote $T_j = N_j - 2$ when $j \in J$ and $T_j = N_j - 1$ otherwise, then define

$$E_J = \{0, 1, \dots, T_1\} \times \cdots \times \{0, 1, \dots, T_m\},$$

which agrees with the definition of the set E at the beginning of this section. Then, from this definition of Z (and E) we have a particular class of (Affine variety) Monomial Cartesian codes known as *J-affine variety codes*. In particular, for $N_j = q$, we recover the two previous situations for $J = \{1, \dots, m\}$ (toric case), and $J = \emptyset$ ($Z = \mathbb{F}_q^m$ case). We will consider a simpler notation for J-affine codes that ease the reading and strengths the dependence on the set J: the quotient ring will be denoted by $R_J = R$, the evaluation map will be denoted by $\mathrm{ev}_J = \mathrm{ev}_Z$ and the *J-affine variety code given by $\Delta \subset E$* is the $\mathbb{F}_q$-vector subspace C_Δ^J of $\mathbb{F}_q^{n_J}$ generated by $\mathrm{ev}_J(X^{\boldsymbol{a}})$, $\boldsymbol{a} \in \Delta$. The dual code can be computed using the following result [27].

Proposition 25. *Let $J \subseteq \{1, 2, \dots, m\}$, consider $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{H}_J$ and let $X^{\boldsymbol{a}}$ and $X^{\boldsymbol{b}}$ be two monomials representing elements in $\mathcal{R}_J$. Then, $\mathrm{ev}_J(X^{\boldsymbol{a}}) \cdot \mathrm{ev}_J(X^{\boldsymbol{b}})$ is different from 0 if, and only if, the following two conditions are satisfied.*

- *For every $j \in J$, it holds that $a_j + b_j \equiv 0 \mod (N_j - 1)$, (i.e., $a_j = N_j - 1 - b_j$ when $a_j + b_j > 0$ or $a_j = b_j = 0$).*
- *For every $j \notin J$, it holds that*
  - *either $a_j + b_j > 0$ and $a_j + b_j \equiv 0 \mod (N_j - 1)$, (i.e., $a_j = N_j - 1 - b_j$ if $0 < a_j, b_j < N_j - 1$ or $(a_j, b_j) \in \{(0, N_j - 1), (N_j - 1, 0), (N_j - 1, N_j - 1)\}$ otherwise),*

– *or* $a_j = b_j = 0$ *and* $p \not| N_j$.

If we set $E' := E_{\{1,2,\ldots,m\}}$ and pick $\Delta \subseteq E_\emptyset$. Let us define $\Delta^\perp$ as

$$E_J \setminus \{(N_1 - 1 - a_1, N_2 - 1 - a_2, \ldots, N_m - 1 - a_m) \mid \boldsymbol{a} \in \Delta\},$$

if $\Delta \subseteq E'$. When $\Delta \not\subseteq E'$ define $\Delta^\perp$ as

$$E_J \setminus \left\{\{(N_1 - 1 - a_1, N_2 - 1 - a_2, \ldots, N_m - 1 - a_m) | \boldsymbol{a} \in \Delta \cap E'\} \cup \{\boldsymbol{a}' | \boldsymbol{a} \in \Delta, \boldsymbol{a} \notin E'\}\right\},$$

where we set $a'_j = N_j - 1 - a_j$ if $a_j \neq N_j - 1$ and a'_j equals either $N_j - 1$ or 0 otherwise.

Next, we state the result about the dual code and self-orthogonality of a J-affine code that follows from the previous result.

Proposition 26. *With notations as above, let* Δ *be a subset of* E_J. *Then* $\left(C^J_\Delta\right)^\perp = C^J_{\Delta^\perp}$ *whenever* $\Delta \subseteq E'$. *Otherwise, it holds that* $\left(C^J_\Delta\right)^\perp \subseteq C^J_{\Delta^\perp}$.

For $\Delta_1, \Delta_2 \subseteq E_J$, we will consider their Mikowski sum $\Delta_1 + \Delta_2 \subset \mathbb{N}^m$ and then reduce it E_J in the following way. For $\mathbf{a} \in \mathbb{N}^m$ we define $\overline{\mathbf{a}} = (\overline{a}_1, \ldots, \overline{a}_m) \in E_J$, where, if $j \in J$, $\overline{a}_j = a_j \mod N_j - 1$, that is $\overline{a}_j \in \{0, \ldots, N_j - 2\} = \{0, 1, \ldots, T_j\}$. Otherwise, if $j \notin J$, then $\overline{a}_j$ is equal to 0 if $a_j = 0$, and it is equal to $a_j \mod N_J - 1$, where the reduction modulo $N_j - 1$ is taken in $\{1, \ldots N_J - 1\}$, i.e. $\overline{a}_j \in \{0, 1, \ldots, N_j - 1\} = \{0, 1, \ldots, T_j\}$. Furthermore, $\mathrm{ev}_Z(X^{\mathbf{a}}) = \mathrm{ev}_Z(X^{\overline{\mathbf{a}}})$, since we evaluate classes of polynomials modulo the ideal I_J in $\mathcal{R}$ generated by the binomials $X_j^{N_j} - X_j$ when $j \notin J$ and by $X_j^{N_j - 1} - 1$ otherwise. Thus, for $A \subset \mathbb{N}^m$, we define $\overline{A} = \{\overline{\mathbf{a}} \mid \mathbf{a} \in A\} \subset E_J$. Thus, the componentwise product of J-affine variety codes is given by the following result.

Theorem 27. *Let* $N_j > 1$ *such that* $N_j - 1$ *divides* $q - 1$ *for* $j = 1, \ldots, m$ *and* $J \subseteq \{1, \ldots, m\}$. *Let* $\Delta_1, \Delta_2 \subseteq E_J$. *Then, the componentwise product of* $C^J_{\Delta_1}$ *and* $C^J_{\Delta_2}$ *is given by*

$$C^J_{\Delta_1} \star C^J_{\Delta_2} = C^J_{\overline{\Delta_1 + \Delta_2}}.$$

Proof. The result follows from the previous disccusion and the fact that $\mathrm{ev}_Z(f_1) \star \mathrm{ev}_Z(f_2) = \mathrm{ev}_Z(f_1 f_2)$ and that $\mathrm{ev}_Z(f) = \mathrm{ev}_Z(f \mod I)$. □

Notice how this extends the computations in [34, 28] (for $N_j = q$ for all j).

1.1.8 Subfield subcodes of J-Affine codes

Given a linear code C of lenght n over $\mathbb{F}_q$ and $\mathbb{F}_{q'} \subseteq \mathbb{F}_q$, the *subfield-subcode* over $\mathbb{F}_{q'}$ is $S(C) = C \cap \mathbb{F}_{q'}^n$, i.e., the set of codewords in C with all the coordinates over the subfield $\mathbb{F}_{q'}$.

We are going to consider subfield subcodes of J-Affine codes. A subset I of the Cartesian product E_J is called a cyclotomic set with respect to p if $p \cdot \boldsymbol{x} \in I$ for every element $\boldsymbol{x} = (x_1, \ldots, x_m) \in I$, where we define $p \cdot \boldsymbol{x} = (px_1, \ldots, px_m)$. A cyclotomic set I is said to be minimal (with respect to p) if it consists exactly of all elements expressible as $p^i \cdot \boldsymbol{x}$ for some fixed element $\boldsymbol{x} \in I$ and some nonnegative integer i. In the case of one variable, they are usually called cyclotomic cosets.

Within each minimal cyclotomic set I, we select a representative element $\boldsymbol{a} = (a_1, \ldots, a_m)$ consisting of nonnegative integers as follows: first, a_1 is chosen as the minimum first coordinate among all nonnegative representatives of elements in I; next, a_2 is the minimum second coordinate among those elements having first coordinate equal to a_1; and similarly, we define coordinates $a_3, \ldots, a_m$. We denote by $I_{\boldsymbol{a}}$ the cyclotomic set with representative $\boldsymbol{a}$, and by $\mathcal{A}$ the set of representatives of all minimal cyclotomic sets. Thus, the set of minimal cyclotomic sets is given by $\{I_{\boldsymbol{a}}\}_{\boldsymbol{a}\in\mathcal{A}}$. In addition, we denote its cardinality by $i_{\boldsymbol{a}} = \mathrm{card}(I_{\boldsymbol{a}})$.

The subfield-subcode associated to a given J-affine variety code C_Δ^J over the finite field $\mathbb{F}_{q'} = \mathbb{F}_{p^r}$ is defined as:

$$C_\Delta^{J,\sigma} = S(C_\Delta^J) = C_\Delta^J \cap \mathbb{F}_{p^r}^{n_J}.$$

Consider now the following maps: $\mathrm{tr} : \mathbb{F}_q \to \mathbb{F}_p$, $\mathrm{tr}(x) = x + x^p + \cdots + x^{p^{r-1}}$; $\mathbf{tr} : \mathbb{F}_q^{n_J} \to \mathbb{F}_p^{n_J}$ given componentwise by $\mathrm{tr}(x)$, and $\mathcal{T} : \mathcal{R}_J \to \mathcal{R}_J$ defined by $\mathcal{T}(f) = f + f^p + \cdots + f^{p^{r-1}}$.

The dimension of $C_\Delta^{J,\sigma}$ is given in [26, Theorem 11]. Note that, when computing the dimension, only those cyclotomic sets that are complete will contribute, that is, when a cyclotomic coset $I_{\boldsymbol{a}} \subseteq \Delta$. The minimum distance of $C_\Delta^{J,\sigma}$ is lower bounded by the minimum distance of C_Δ^J.

Theorem 28. *Let Δ be a subset of $\mathcal{H}_J$ and set $\xi_{\boldsymbol{a}}$ a primitive element of the field $\mathbb{F}_{p^{i_{\boldsymbol{a}}}}$. Then, a basis of the vector space $C_\Delta^{J,\sigma}$ is given by the images under the map ev_J of the set of classes in R_J*

$$\bigcup_{\boldsymbol{a}\in\mathcal{A}|I_{\boldsymbol{a}}\subseteq\Delta} \{\mathcal{T}_{\boldsymbol{a}}(\xi_{\boldsymbol{a}}^s X^{\boldsymbol{a}}) | 0 \le s \le i_{\boldsymbol{a}} - 1\},$$

and the dimension of $C_\Delta^{J,\sigma}$ is given by the cardinality of $\Delta^\sigma := \cup_{\boldsymbol{a}\in\mathcal{A}|\mathfrak{I}_{\boldsymbol{a}}\subseteq\Delta} I_{\boldsymbol{a}}$

Computing the componentwise product of subfield subcodes can be tricky, as the next remark shows.

Remark 29. Let $q = 4$, α a primitive element ($\alpha^3 = 1$), and subfield-subcodes over $\mathbb{F}_2$. Let C_1 and C_2 the linear codes generated by $(1, \alpha, 0)$, and $(0, \alpha^2, 1)$, respectively. One can easily check that $S(C_1) = S(C_2) = \{(0, 0, 0)\}$, but $S(C_1 \star C_2)$ is generated by $(1, 1, 1)$.

However, the componentwise product of the subfield subcodes of J-affine variety codes can be explicitely computed if we consider their defining set to be a union of complete cyclotomic cosets.

Lemma 30. *Let $N_j > 1$ such that $N_j - 1$ divides $q - 1$, for $j = 1, \ldots, m$, and $J \subseteq \{1, \ldots, m\}$. Let $\Delta_1, \Delta_2 \subseteq E_J$ be a union of complete cyclotomic cosets. Then, the componentwise product of $C_\Delta^{J,\sigma}$ and $C_{\Delta_2}^{J,\sigma}$ is given by*

$$S(C_{\Delta_1}^J \star C_{\Delta_2}^J) = C_{\Delta_1}^{J,\sigma} \star C_{\Delta_2}^{J,\sigma} = C_{\overline{\Delta_1+\Delta_2}}^{J,\sigma}.$$

Proof. By Theorem 27, it follows that $S(C_{\Delta_1}^J \star C_{\Delta_2}^J) = S(C_{\overline{\Delta_1+\Delta_2}}^J) = C_{\overline{\Delta_1+\Delta_2}}^{J,\sigma}$.

On the other hand, under the asumption that Δ_1 and Δ_2 are complete cyclotomic cosets, reasoning as before Theorem 27, one hast that $S(C_{\Delta_1}^J) \star S(C_{\Delta_2}^J) = C_{\Delta_1}^{J,\sigma} \star C_{\Delta_2}^{J,\sigma}$ is equal to $C_{\overline{\Delta_1+\Delta_2}}^{J,\sigma}$, and the result holds. □

The next example shows that it is not possible, even though one works with a union of complete cyclotomic cosets, since the equality $S(C_1 \star C_2)^\perp = S(C_1)^\perp \star S(C_2)^\perp$ does not hold, in general.

Remark 31. Let $N_1 = 16$ and $J = \{1\}$. Consider $\Delta_1 = \{1, 2, 4, 8\}$ and $\Delta_2 = \{0\}$, both a union of complete cyclotomic cosets. One has that

$$\Delta_1^\perp = \{0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 12\}, \text{ and } \Delta_2^\perp = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}.$$

Then the codes $C^J_{\Delta_1}$ and $C^J_{\Delta_2}$ have length 15 and are defined over $\mathbb{F}_{16}$. We consider subfield subcodes over $\mathbb{F}_2$. Moreover, notice that $\Delta_1 + \Delta_2 = \Delta_1$.

Then, $S(C_1^J \star C_2^J)^\perp = S(C_1^J)^\perp$, but $S(C^J_{\Delta_1})^\perp \star S(C^J_{\Delta_2})^\perp$ is equal to $S(C^J_\Delta)$, with $\Delta = \Delta_1^\perp + \Delta_2^\perp = \{0, 1, \ldots, 14\}$. Thus we have that

$$S(C^J_{\Delta_1} \star C^J_{\Delta_2})^\perp \neq S(C^J_{\Delta_1})^\perp \star S(C^J_{\Delta_2})^\perp.$$

1.1.9 Transitivity

In [14], a new family of monomial-Cartesian codes, known as decreasing monomial-Cartesian codes, was introduced. These codes are defined by evaluating monomials in a manner analogous to the construction of Reed–Solomon and Reed–Muller codes. However, they impose additional conditions on the functions to be evaluated, specifically in terms of divisibility.

Definition 32 (Decreasing monomial-Cartesian code)**.** A *decreasing monomial set* is a set of monomials $\mathcal{M} \subseteq R$ such that, if $\mathbf{m} \in \mathcal{M}$ and $\mathbf{m}'$ divides $\mathbf{m}$, then $\mathbf{m}' \in \mathcal{M}$. The code $C_{\Delta,Z}$ is called a *decreasing monomial-Cartesian code* if the set of monomials $\{\mathbf{x}^\mathbf{a} \mid \mathbf{a} \in \Delta\}$ forms a decreasing monomial set.

In [14, Theorem 3.9], the minimum distance and dimension of a decreasing monomial-Cartesian code are computed using a minimal generating set of $\mathcal{M}$. Additionally, in [14, Theorem 3.3], it is shown that the dual of a decreasing monomial-Cartesian code is equivalent to another decreasing monomial-Cartesian code. The following lemma shows that a decreasing monomial-Cartesian code is transitive.

Lemma 33. *Let $C_{\Delta,Z}$ be a decreasing monomial-Cartesian code, where Z is an additive subgroup of $\mathbb{F}_q^n$. Then $C_{\Delta,Z}$ is transitive.*

Proof. Recall that each coordinate $i \in \{1, \ldots, n\}$ can be identified with a point $\mathbf{P}_i \in Z$. Without loss of generality, we assume that a codeword $(c_{\mathbf{P}_1}, \ldots, c_{\mathbf{P}_n})$ is obtained by evaluating a monomial $f(\mathbf{x}) = \mathbf{x}^\mathbf{a}$ (a general codeword is simply a linear combination of such evaluations). Given two distinct points $\mathbf{P}_i, \mathbf{P}_j \in Z$, we aim to show that there exists a permutation $\pi : Z \to Z$ such that $\pi(\mathbf{P}_i) = \mathbf{P}_j$ and that the permuted codeword $(c_{\pi(\mathbf{P}_1)}, \ldots, c_{\pi(\mathbf{P}_n)})$ still lies in $C_{\Delta,Z}$.

Consider the map $\pi(z) = z - \mathbf{P}_i + \mathbf{P}_j$. Clearly, $\pi(\mathbf{P}_i) = \mathbf{P}_j$, and π defines a permutation on Z since Z is an additive subgroup. Now, define the polynomial

$$g(\mathbf{x}) = \prod_{s=1}^{n} \left(x_s - \mathbf{P}_{is} + \mathbf{P}_{j_s}\right)^{\mathbf{a}_s}.$$

Since the monomial set defining the code is decreasing, $g(\mathbf{x})$ is a linear combination of monomials in Δ, as each such monomial divides $f(\mathbf{x}) = \mathbf{x}^{\mathbf{a}}$. Therefore, the permuted codeword satisfies

$$(c_{\pi(\mathbf{P}_1)}, \ldots, c_{\pi(\mathbf{P}_n)}) = (f \circ \pi(\mathbf{P}_1), \ldots, f \circ \pi(\mathbf{P}_n)) = (g(\mathbf{P}_1), \ldots, g(\mathbf{P}_n)) \in C_{\Delta,Z}.$$

□

In the context of J-affine codes, being a decreasing monomial-Cartesian code follows from considering *consecutive* cyclotomic sets and it is widely used (see, for example, [27, 25]). That is, $\Delta_{\boldsymbol{a}_i} = I_{\boldsymbol{a}_0} \cup I_{\boldsymbol{a}_1} \cup \cdots \cup I_{\boldsymbol{a}_i}$.

Lemma 34. *Let C_Δ^J be a J-affine code defined by a union of consecutive cyclotomic sets $\Delta = \Delta_{\boldsymbol{a}_i} = I_{\boldsymbol{a}_0} \cup I_{\boldsymbol{a}_1} \cup \cdots \cup I_{\boldsymbol{a}_i}$. Then Δ is a decreasing monomial set and C_Δ^J is transitive.*

Proof. The proof follows similar reasoning to the previous lemma, taking into account that

$$g(\mathbf{x}) = \prod_{s=1}^{n} \left(x_s - \mathbf{P}_{i_s} + \mathbf{P}_{j_s}\right)^{\mathbf{a}_s \cdot q^k} = \prod_{s=1}^{n} \left(x_s - \mathbf{P}_{i_s} + \mathbf{P}_{j_s}\right)^{\mathbf{a}_s},$$

since the exponents are taken modulo the size of the field. □

Lemma 35. *Let $C_{\Delta_1,Z}$ and $C_{\Delta_2,Z}$ be two transitive monomial-Cartesian codes. Then the componentwise (or star) product code $C_{\Delta_1,Z} \star C_{\Delta_2,Z}$ is also transitive.*

Proof. This follows directly from the fact, noted in Section 1.1.6, that the $\star$-product code corresponds to the evaluation code defined by the sum of the monomial sets $\Delta_1 + \Delta_2$. □

Remark 36. Note that a code and its dual share the same automorphism group, see for example [38]. Therefore, if a code is transitive, so it is its dual. This observation, together with the preceding lemmas, allows us to consider decreasing monomial-Cartesian codes, hyperbolic codes, and their duals for the construction of PIR schemes in coding theory framework of [23] that provide the parameters in Theorem 65.

1.2 Private Information Retrieval

A PIR scheme consists of three stages: Data Storage, File Request, and Response Process. For the multiple-server case, files are uploaded to a distributed storage system (DSS) during the data storage process. In the File Request, users select the file they want to retrieve, called the desired file, and based on that, they choose queries that are sent to the servers. In the final Response Process, servers 'operate' the files with the queries generating a matrix of responses that are sent back to the user. It should be noted that the servers do not have any information about the file the user requested.

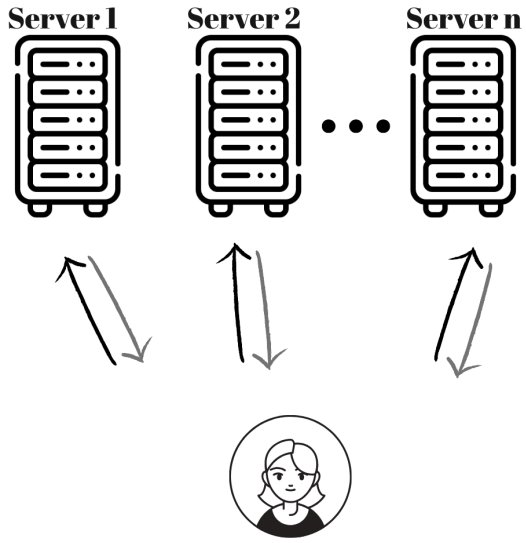


Figure 1.1: Private Information Retrieval Schemes

1.2.1 Data Storage Process

In the case where files are stored on multiple servers, a distributed storage system (DSS) can split the data across those servers. In order to upload the files into the servers, files are encoded by a k-dimensional storage code $C \subseteq \mathbb{F}_q^n$ with parameters $[n, k, d]$.

Assume that there are r-files, each file has ρ-rows, k-columns, and the elements of the files are in $\mathbb{F}_q$. Since the number of files is r, the total file can be understood as $r\rho \times k$ matrix represented by A, and each file is denoted by $\boldsymbol{a}^i$, where $i \in \{1, \ldots, r\}$. Concerning encoding, we multiply the matrix A, which covers all files, by G_C the generator matrix of the linear code C and we obtain the matrix $B := A \cdot G_C$. Since A is a $\rho r \times k$ matrix, B has ρr rows and n columns.

Example 37. Assume that there are two files, $\boldsymbol{a}^1, \boldsymbol{a}^2$, given by

$$\boldsymbol{a}^1 = \begin{bmatrix} a^1_{1,1} & a^1_{1,2} \\ a^1_{2,1} & a^1_{2,2} \end{bmatrix}, \quad \boldsymbol{a}^2 = \begin{bmatrix} a^2_{1,1} & a^2_{1,2} \\ a^2_{2,1} & a^2_{2,2} \end{bmatrix},$$

where each file consists of $\rho = 2$ rows and $k = 2$ columns and $A = \begin{bmatrix} \boldsymbol{a}^1 \\ \boldsymbol{a}^2 \end{bmatrix}$.

Let $C \subseteq \mathbb{F}_2^3$ be a storage code with parameters $[3, 2, 2]_2$. The files are then encoded with the generator matrix G of C, multiplying each row of a^i with G. For example

$$\boldsymbol{a}^1_{1,\cdot} \cdot G = \begin{bmatrix} a^1_{1,1} & a^1_{1,2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a^1_{1,1} & a^1_{1,2} & a^1_{1,3} \end{bmatrix},$$

where $a^1_{1,3} = a^1_{1,1} + a^1_{1,2}$ and $\boldsymbol{a}^1_{1,\cdot}$ represents the first row of the first file. Each column of the encoded file is stored on a different server in the following form:

Server 1	Server	Server 3
$a^1_{1,1}$	$a^1_{1,2}$	$a^1_{1,1} + a^1_{1,2}$
$a^1_{2,1}$	$a^1_{2,2}$	$a^1_{2,1} + a^1_{2,2}$
$a^2_{1,1}$	$a^2_{1,2}$	$a^2_{1,1} + a^2_{1,2}$
$a^2_{2,1}$	$a^2_{2,2}$	$a^2_{2,1} + a^2_{2,2}$

1.2.2 Request and Response Process

Assume that the user wishes to retrieve the file $\boldsymbol{a}^d$. Then the user chooses a random query Q^i and sends this query to the servers. Each server computes the inner product of Q^i_j and B_j, where j is the server index, i.e, j^{th}-server computes $\langle Q^i_j, B_j \rangle$. Then, the servers send the response vectors back to the user. This communication process is repeated m times, once for each iteration $\gamma \in [m]$. In the final round, all parts of the file are completed. The parts of the file from several servers are gathered to get the whole file.

Example 38.

Consider that the storage code C is a repetition code with parameters $[3, 1, 3]_2$. Three files, a^1, a^2, $a^3 \in \mathbb{F}_2$, are stored on the $n = 3$ servers. Each file has $\rho = 1$ rows and $k = 1$ columns. Since the storage code is a repetition code, each server stores the same information. Assume that the user wants to obtain the first file, a^1. In order to retrieve the first file, the user generates query vectors consisting of
$\boldsymbol{q}_1 = (0, 1, 1)$, $\boldsymbol{q}_2 = (1, 1, 0)$, $\boldsymbol{q}_3 = (0, 0, 1)$. The servers return a response
$r_j = \sum_{i=1}^{3} q_{j,i} \cdot a^i$ for $j \in \{1, 2, 3\}$, which gives
$r_1 = a^2 + a^3$, $r_2 = a^1 + a^2$ and $r_3 = a^3$. The user can recover the desired file, a^1, by adding all responses.

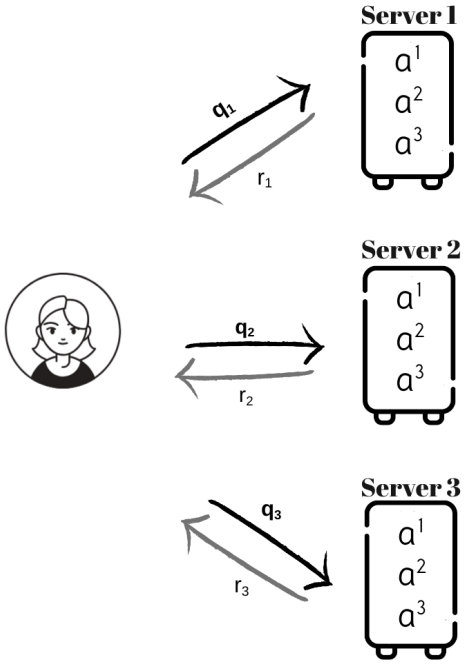


Figure 1.2: Example Private Information Retrieval Schemes

Let us now define two key concepts: the rate and privacy of a PIR scheme, which measures the efficiency of the PIR scheme.

- **PIR Rate:** During the retrieval process, the user downloads both the symbols needed to reconstruct the desired file and additional redundant information to preserve privacy. This provides that the server cannot determine the desired file. Therefore, the PIR rate is defined as the ratio of the information obtained during the process to the downloaded information.

- **Privacy:** In the query generation process, the user selects the query vectors uniformly at random. A t-PIR scheme guarantees that the identity of the requested file remains hidden against any set of up to t colluding servers. Even if these servers share the queries they receive, they cannot determine any information about the file index. More precisely, for every subset $T \subset [n]$ of size t, the scheme ensures that the queries observed by the servers in T do not reveal any information about the desired file index. Formally, this can be expressed as

$$I\left(\{q_i^d : i \in T\}; d\right) = 0,$$

  where $I(\cdot\,;\,\cdot)$ denotes the mutual information, which measures the amount of information one random variable provides about another.

1.3 CSS Construction of Quantum Codes Over Finite Fields

The Calderbank–Shor–Steane (CSS) construction of Quantum Codes from binary linear codes [12, 61] can be generalized to arbitrary finite fields $\mathbb{F}_q$, where q is a prime power, see [3, 40]. In this section, we give a glance of how quantum stabilizer codes can be obtained from pairs of classical linear codes over $\mathbb{F}_q$ satisfying certain orthogonality conditions with respect to the Euclidean or Hermitian inner product, for a detailed account see [64, 50, 55]. This general framework allows the construction of quantum codes over qudits, extending the binary case to broader settings relevant in quantum information theory. The CSS construction over $\mathbb{F}_q$ requires that $C_2^{\perp} \subseteq C_1$ with respect to a chosen inner product. Two common choices are:

- The **Euclidean inner product**, defined as

$$\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{n} u_i v_i \in \mathbb{F}_q,$$

  for $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^n$, typically used when q is a prime or odd power of a prime.

- The **Hermitian inner product**, for $q = p^{2m}$, defined as

$$\langle \mathbf{u}, \mathbf{v} \rangle_H = \sum_{i=1}^{n} u_i v_i^{p^m} \in \mathbb{F}_q,$$

  commonly used in constructions involving $\mathbb{F}_{q^2}$.

The construction follows from this result.

Theorem 39 (Quantum Stabilizer Codes from Classical Linear Codes)**.** *Let $C_1, C_2 \subseteq \mathbb{F}_q^n$ be two linear codes such that $C_2 \subseteq C_1$. Given such a pair (C_1, C_2), there is a CSS code encodes $k = \dim C_1 - \dim C_2$ qudits into n and can correct all errors affecting fewer than $\lfloor (d-1)/2 \rfloor$ positions, where*

$$d = \min\left\{ \mathrm{wt}(C_1 \setminus C_2), \mathrm{wt}(C_2^\perp \setminus C_1^\perp) \right\}.$$

Let us outline the explicit construction:

1. Choose linear codes $C_1, C_2 \subseteq \mathbb{F}_q^n$ such that $C_2 \subseteq C_1$ and $C_2^\perp \subseteq C_1$ with respect to the chosen inner product (i.e., C_1 is self-orthogonal with respect to $C_2^\perp$).
2. The stabilizer S of the quantum code is generated by the parity-check matrix of C_1 for X-type operators and the parity-check matrix of C_2 for Z-type operators.
3. The logical code space $\mathcal{Q}$ is defined as the joint $+1$ eigenspace of all stabilizer generators. Basis states of $\mathcal{Q}$ are superpositions of coset representatives of C_2 in C_1, that is, for $\mathbf{u} \in C_1$:

$$|\mathbf{u} + C_2\rangle := \frac{1}{\sqrt{|C_2|}} \sum_{\mathbf{v} \in C_2} |\mathbf{u} + \mathbf{v}\rangle.$$

This produces a quantum stabilizer code with parameters $[[n, k = \dim C_1 - \dim C_2, d]]_q$, often written as $[[n, k, d]]_q$.

The field size q affects both the choice of classical codes and the resulting quantum code parameters:

- Over $\mathbb{F}_2$, common constructions include Reed-Muller, BCH, and cyclic codes.
- For larger q, one can use generalized Reed–Solomon (GRS) codes, AG codes, or J-affine variety codes, ensuring the required orthogonality conditions are met.
- If $C_1 = C$ and $C_2 = C^\perp$, the resulting CSS code is said to be *self-dual*. In that case, $k = 0$ and the code is used to detect but not transmit logical information.

Example 40. CSS Codes from Reed–Solomon Codes over $\mathbb{F}_q$

Let $C_1 = \mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{v})$, and $C_2 = \mathrm{GRS}_{k'}(\boldsymbol{\alpha}, \boldsymbol{v})$ with $k' < k$ and $\boldsymbol{\alpha}, \boldsymbol{v} \in \mathbb{F}_q^n$ satisfying orthogonality conditions:

$$\mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{v})^\perp = \mathrm{GRS}_{n-k}(\boldsymbol{\alpha}, \boldsymbol{v}^{-1} \star \boldsymbol{g}),$$

for a suitable weight vector $\boldsymbol{g}$, where $\star$ denotes componentwise multiplication. Then the inclusion $C_2^\perp \subseteq C_1$ can be verified algebraically, ensuring the CSS conditions.

1.4 Secure Multi-Party Computation

Multi-Party Computation (MPC) is a cryptographic protocol that allows multiple parties to collaboratively compute a function over their private inputs while ensuring that none of the participants learns anything about the others' data except what can be inferred from the output. The primary goal of MPC is to preserve the privacy of each participant's input throughout the computation process while still guaranteeing the correctness of the final result.

The key Stages of MPC are as follows:

- **Input Sharing:** Each participant splits their private input into several shares using a suitable secret sharing scheme, such as Shamir's Secret Sharing or additive secret sharing. Then, the shares are distributed among the other participants. This step ensures that no single party can reconstruct private input of another participant from the shares they receive.
- **Computation Phase:** The participants jointly perform computations on the distributed shares. The computations are structured so that each party only accesses partial information, and, ideally, no participant can deduce the original inputs of the others.
- **Output Reconstruction:** Once the computations are complete, the participants combine their outputs to reconstruct the final output. This reconstruction is done in such a way that no private input is revealed to any party beyond what is necessary for obtaining the correct result.

MPC protocols rely on several cryptographic techniques, among which secret sharing, homomorphic encryption, and oblivious transfer are key components. In particular, secret sharing methods—especially Shamir's scheme—are fundamental and have strong ties to algebraic coding theory. This branch of mathematics offers tools for encoding and decoding data with error correction, which is directly relevant for building robust and secure computations.

Polynomial-based secret sharing, inspired by Reed–Solomon codes, is widely used; these codes work by evaluating polynomials over finite fields, as explained in [22] and other related works. Within MPC, such techniques allow the secure distribution of a secret among several participants, ensuring that the secret can only be reconstructed when a minimum number of shares are combined. This threshold mechanism resembles how certain error-correcting codes can recover lost or corrupted data.

More recently, developments like packed and replicated secret sharing have emerged, drawing on ideas from multidimensional coding theory to improve efficiency. Additionally, verifiable secret sharing has become important in MPC scenarios where participants need guarantees about the correctness of their shares. This enhancement is particularly desirable, especially in adversarial settings, as discussed in [18] and the references therein.

Chapter 2

Private Information Retrieval Schemes for Several Servers Based on Cyclic Codes

In this chapter, we propose using cyclic codes to construct PIR schemes in the same fashion as [23]. Cyclic codes have a transitive automorphism group and can also be defined over binary (or small) finite fields. Moreover, the star product of two cyclic codes is a cyclic code whose parameters can be computed [17]. Namely, the star product of two cyclic codes is determined by the sum of their generating sets. We can compute its dimension and estimate its minimum distance by considering cyclotomic cosets.

We aim to optimize the number of servers that can collude without disclosing information about the identity of the data retrieved to the server. In order to show the advantages of cyclic codes for PIR schemes, we first provide pairs of cyclic codes C and D, the storage code and retrieval code, such that the parameters of C, D, $D^{\perp}$, $C \star D$ and $(C \star D)^{\perp}$ are -at the same time- optimal or the best known. As we will recall in Section 2.1, their parameters determine the performance of the PIR scheme defined by C and D. Since a punctured RM code is a cyclic code, we may obtain PIR schemes using punctured RM codes by using cyclic codes. Moreover, we show that we obtain a larger constellation of possible parameters of binary PIR schemes by using cyclic codes. The construction of PIR schemes and the computations of their parameters follow from a detailed analysis of cyclotomic cosets. Next, we focus on the privacy and the rate of a PIR scheme since the upload cost in a PIR scheme can be neglected [4]. More concretely, in the case that the storage code C has dimension 2, we obtain binary PIR schemes that greatly outperform the ones obtained using RM codes; more concretely, they protect against a more significant number of colluding servers. Finally, we compare our schemes with shortened RM codes and show that, in this case, the PIR schemes using cyclic codes outperform them as well, namely, they offer more privacy for a fixed rate.

2.1 The Scheme

Let $C \subseteq \mathbb{F}_q^n$ be a k-dimensional linear code with generator matrix G that we will call the *storage code*, and $D \subseteq \mathbb{F}_q^n$ be another linear code that we will call *retrieval code*. The *database* in the server consists of r files, where each file is considered as a matrix $\boldsymbol{a}^i \in \mathbb{F}_q^{\rho \times k}$. The elements in the database are encoded by the linear code C, i.e., $\boldsymbol{b}^i = \boldsymbol{a}^i G$, where $i \in \{1, \cdots, r\}$. The

encoded data is distributed among the n servers. Each column of the encoded file $\boldsymbol{b}^i$ is stored in a different server, that is, entry $b^i_{\ell,j}$ is stored on the j^{th} server where $\ell \in \{1, \cdots, \rho\}$. Given a matrix A, we write $A_{i,\cdot}$ to denote the i^{th} row with all columns and write $A_{\cdot,j}$ to denote j^{th} column with all rows.

To retrieve file $\boldsymbol{a}^d$, the user constructs the queries $\boldsymbol{q}^i = \boldsymbol{w}^i + \boldsymbol{u}^i \in \mathbb{F}_q^{\rho \times n}$ for $i \in \{1, \cdots, r\}$, as follows.

- The rows of the matrix $\boldsymbol{w}^i$ are chosen uniformly at random from codewords at the retrieval code D,
- Let d be the index of the file that the user wants to access. The matrix $\boldsymbol{u}^i \in \mathbb{F}_q^{\rho \times n}$ is all zero matrix if $i \neq d$, and $\boldsymbol{u}^d$ is a matrix whose rows are not in the retrieval code D and has $d_{C \star D} - 1$ non-zero entries, where $d_{C \star D}$ is the minimum distance of the Schur product of C and D. Each of these entries is used to recover one symbol from the desired file, and they are distributed in separate columns for different file parts, so no column is used more than once. Thus,
$$\boldsymbol{u}^i = \begin{cases} \boldsymbol{u}^d & \text{if } i = d, \\ \boldsymbol{0} & \text{otherwise.} \end{cases}$$
To fully recover the desired file $\boldsymbol{a}^d$, the query generalization process is repeated m times. In each iteration $\gamma \in \{1, \ldots, m\}$, the user selects a new set of $d_{C \star D} - 1$ columns, which are not used in previous rounds. For each selected coordinate $j \in \{1, \ldots, n\}$, the user assigns it to a row index $\rho_j \in \{1, \ldots, \rho\}$ such that:
  - $j \in I_{\rho_j}$, where I_{ρ_j} is an information set of C used to decode row ρ_j, and
  - the pair (j, ρ_j) has not been used before.

The process above allows us to construct a *query matrix* Q is given by

$$Q = \begin{pmatrix} \boldsymbol{q}^1 \\ \boldsymbol{q}^2 \\ \vdots \\ \boldsymbol{q}^r \end{pmatrix} = \begin{pmatrix} \boldsymbol{w}^1 + \boldsymbol{u}^1 \\ \boldsymbol{w}^2 + \boldsymbol{u}^2 \\ \vdots \\ \boldsymbol{w}^d + \boldsymbol{u}^d \\ \vdots \\ \boldsymbol{w}^r + \boldsymbol{u}^r \end{pmatrix}, \qquad \boldsymbol{q}^i = \begin{pmatrix} q^i_{1,1} & q^i_{1,2} & \cdots & q^i_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ q^i_{\rho,1} & q^i_{\rho,2} & \cdots & q^i_{\rho,n} \end{pmatrix}, \tag{2.1}$$

where each $\boldsymbol{q}^i$ is a matrix in $\mathbb{F}_q^{\rho \times n}$.

In the communication part of the scheme, each server $j \in \{1, \ldots, n\}$, receives from the user the j-th column of Q, denoted $q^i_{\cdot,j}$, for all i, and computes the answers

$$r_j = \sum_{i=1}^{r} \langle q^i_{\cdot,j}, b^i_{\cdot,j} \rangle = \sum_{\ell=1}^{\rho} \sum_{i=1}^{r} w^i_{\ell,j} \cdot b^i_{\ell,j} + \sum_{\ell=1}^{\rho} u^d_{\ell,j} \cdot b^d_{\ell,j},$$

where $\langle , \rangle$ denotes the inner product, and send it back to the user.

Let $\boldsymbol{r} = (r_1, r_2, \ldots, r_n) \in \mathbb{F}_q^n$ denote the vector of responses received by the user from the servers for each iteration $\gamma \in \{1, \ldots, m\}$. One can see the response vector $\boldsymbol{r}$ as an element in

$C \star D + C \star u$. Therefore, the user can extract the desired file from $\mathbf{r}$ by applying the projection given by

$$\begin{aligned} \text{proj} : \mathbb{F}_q^n &\longrightarrow \mathbb{F}_q^{\dim(C\star D)^\perp} \\ \boldsymbol{r} &\longmapsto \boldsymbol{r} \cdot H_{C\star D}^\top, \end{aligned}$$

where $H_{C\star D}$ is a parity check matrix of $C\star D$. Since the support of $\boldsymbol{u}$ is chosen, the corresponding positions of the desired file can be recovered. After running this process m times, the user is able to fully recover the desired file.

In each iteration, only $d_{C\star D} - 1$ coordinates are accessed. The set of coordinates queried in each round is disjoint from those used in previous rounds, so there is no repeated access to the same information. To guarantee decodability, the symbols retrieved in each iteration lie in information sets of the storage code C. The following lemma ensures the number of required disjoint information sets.

Lemma 41 ([23, Lemma 1])**.** *Let C be a linear code with parameters $[n, k, d]$. Then C contains at least $\lceil \frac{d}{k} \rceil$ disjoint information sets.*

Therefore, the rate of the scheme is $(d_{C\star D} - 1)/n$. The privacy of the scheme is based on the fact that all rows of the matrix $\boldsymbol{w}^i$ are chosen uniformly at random from the retrieval code D. If $t = d_{D^\perp} - 1$, then the requested file cannot be determined if t servers communicate with each other. As a result, we have the following theorem, as stated in [23].

Theorem 42 ([23, Theorem 2])**.** *Let $C \subseteq \mathbb{F}^n$ be an $[n, k, d]$ linear storage code and let $D \subseteq \mathbb{F}^n$ be any linear code. Then there exists a private information retrieval scheme with a PIR rate $\frac{d_{C\star D}-1}{n}$, which resists a $d_{D^\perp} - 1$ collusion attack.*

So far, we have explained a general PIR scheme from [23, Theorem 2], where the construction is based on information sets and carefully selected support positions for the error matrix.

As discussed earlier, this approach guarantees correctness if the code has enough disjoint information sets. However, choosing such support sets becomes problematic. To address this, the following theorem also proposes an alternative method based on the automorphism group of the code.

Theorem 43 ([23, Theorem 4])**.** *If the automorphism groups of C and $C \star D$ are transitive on the set $\{1, \ldots, n\}$, then there exists a PIR scheme with rate $\frac{dim(C\star D)^\perp}{n}$ that resists a $(d_{D^\perp} - 1)$–collusion attack. That is, the privacy is $t = d_{D^\perp} - 1$.*

According to Theorem 43, instead of selecting new information sets in each iteration, the scheme constructs them by applying automorphisms to a single information set of size $dim(C \star D)^\perp$. It is the key idea for an improved scheme presented in [23]. The automorphism group ensures that the query positions are distributed uniformly across servers while satisfying the decoding requirements since the code is transitive. It also gives a better PIR rate. Theorem 43 is the key to finding the number of colluding servers and the system's PIR rate.

2.2 PIR Schemes from Cyclic Codes

Now, we focus on cyclic codes towards obtaining PIR schemes over small fields and compute the code parameters with cyclotomic cosets, as described in Section 1. First, we will analyze

the codes obtained from computer search, their cyclotomic cosets, PIR rates, and the number of colluding servers.

The formulation of the number of colluding servers and the PIR rate is given in Theorem 43. This theorem is valid for PIR schemes arising from cyclic codes since the automorphism group of a cyclic code is also transitive (see Section 1). Table 2.1 gives some cyclic codes, the rate of the corresponding PIR scheme arising from them, and the maximum number of servers that may collude, that is, the privacy parameter t.

C	D	$D^{\perp}$	$C * D$	$(C * D)^{\perp}$	Privacy	Rate
$[127, 8, 63]$	$[127, 29, 43]$	$[127, 98, 10]$	$[127, 113, 5]$	$[127, 14, 56]$	9	14/127
$[127, 8, 63]$	$[127, 42, 32]$	$[127, 85, 13]$	$[127, 112, 6]$	$[127, 15, 55]$	12	15/127
$[127, 15, 55]$	$[127, 15, 55]$	$[127, 112, 6]$	$[127, 106, 7]$	$[127, 21, 48]$	5	21/127
$[127, 15, 55]$	$[127, 21, 48]$	$[127, 106, 7]$	$[127, 112, 6]$	$[127, 15, 55]$	6	15/127
$[127, 21, 48]$	$[127, 21, 48]$	$[127, 106, 7]$	$[127, 112, 6]$	$[127, 15, 55]$	6	15/127

Table 2.1: Computer search experiments

For instance, the first row in Table 2.1 considers C as a storage code with parameters $[127, 8, 63]$ and D as a retrieval code with parameters $[127, 29, 43]$. Applying Theorem 43, we can conclude that this scheme is secure against 9-colluding servers since $d(D^{\perp}) = 10$ and that the PIR's rate is $\frac{dim(C \star D)^{\perp}}{n} = \frac{14}{127}$.

We have obtained the codes in Table 2.1 by computer search, their generating set can be found in Table 2.2. For instance, Consider the codes in the first row, the generating set of the code C consists of the union of the cyclotomic cosets U_1 and U_{31}, and the one of D consists of U_0, U_5, U_{23}, U_{27}, U_{31}. As mentioned before, the generating set of star products of cyclic codes are given by the Minkowski sum of their generating sets. Hence, the generating set of $C \star D$ consists of all cyclotomic cosets except U_{13} and U_{47}.

From now on, for sake of brevity, we will denote as $U_{\{s_1,\ldots,s_t\}}$ the union of the t cyclotomic cosets given by $U_{\{s_1,\ldots,s_t\}} = \bigcup_{j=1}^{t} U_{s_i}$.

C	D
$U_{\{0,31\}}$	$U_{\{0,5,23,27,31\}}$
$U_{\{0,11\}}$	$U_{\{1,3,11,23,43,55\}}$
$U_{\{0,5,43\}}$	$U_{\{0,23,43\}}$
$U_{\{0,23,63\}}$	$U_{\{19,31,55\}}$
$U_{\{1,10,29\}}$	$U_{\{7,31,55\}}$

Table 2.2: Cyclotomic cosets used for codes in Table 2.1.

Table 2.3 classifies the codes in Table 2.1 according to the best-known linear codes in the database [33], which gives lower and upper bounds on the parameters of linear codes. As it is shown in the following table, their parameters are the best known or optimal.

C	D	$D^{\perp}$	$C \star D$	$(C \star D)^{\perp}$
$optimal$	$best-known$	$best-known$	$optimal$	$optimal$
$optimal$	$best-known$	$best-known$	$optimal$	$best-known$
$best-known$	$best-known$	$optimal$	$best-known$	$best-known$
$best-known$	$best-known$	$best-known$	$optimal$	$best-known$
$best-known$	$best-known$	$best-known$	$optimal$	$best-known$

Table 2.3: Classification of codes in Table 2.1

2.2.1 Comparison with Punctured and Shortened RM Codes

We will show now why cyclic codes may provide better performance than RM codes. Since punctured and shortened RM codes, denoted as $C_{\bullet}$ and $C^{\bullet}$, are cyclic codes as discussed in Section 1.1.2, we will compare the PIR rate and privacy given by a cyclic code with the corresponding punctured and shortened RM codes.

First, let us focus on the comparison with punctured RM codes. For length 127 and 255, we fixed as storage code a $[127, 8, 63]$, $[255, 9, 172]$ cyclic code and collected the star product of some codes in Table 2.4 and Table 2.6, respectively. We remark that in Table 2.4 the BCH bound of the retrieval codes (D) equal to their minimum distance.

C	D	$D^{\perp}$	$C * D$	$(C * D)^{\perp}$	Privacy	Rate
$[127, 8, 63]$	$[127, 8, 63]$	$[127, 119, 4]$	$[127, 29, 31]$	$[127, 98, 7]$	3	98/127
[127,8,63]	**[127,22,47]**	**[127,105,8]**	**[127,64,15]**	**[127,63,16]**	**7**	**63/127**
$[127, 8, 63]$	$[127, 29, 31]$	$[127, 98, 8]$	$[127, 64, 15]$	$[127, 63, 16]$	7	63/127
[127,8,63]	**[127,50,27]**	**[127,77,16]**	**[127,99,7]**	**[127,28,32]**	**15**	**28/127**
[127,8,63]	**[127,57,23]**	**[127,70,16]**	**[127,99,7]**	**[127,28,32]**	**15**	**28/127**
$[127, 8, 63]$	$[127, 64, 15]$	$[127, 63, 16]$	$[127, 99, 7]$	$[127, 28, 32]$	15	28/127
[127,8,63]	**[127,85,13]**	**[127,42,32]**	**[127,120,3]**	**[127,7,64]**	**31**	**7/127**
[127,8,63]	**[127,92,11]**	**[127,35,32]**	**[127,120,3]**	**[127,7,64]**	**31**	**7/127**
$[127, 8, 63]$	$[127, 99, 7]$	$[127, 28, 32]$	$[127, 120, 3]$	$[127, 7, 64]$	31	7/127

Table 2.4: Comparison with punctured RM codes (Shadow rows)

C	D
$U_{\{0,1\}}$	$U_{\{0,1\}}$
	$U_{\{0,1,5,9\}}$
	$U_{\{0,1,5,9,3\}}$
	$U_{\{0,1,5,9,3,11,19,21\}}$
	$U_{\{0,1,5,9,3,11,19,21,7\}}$
	$U_{\{0,1,5,9,3,11,19,21,7,13\}}$
	$U_{\{0,1,5,9,3,11,19,21,7,13,23,27,43\}}$
	$U_{\{0,1,5,9,3,11,19,21,7,13,23,27,43,29\}}$
	$U_{\{0,1,5,9,3,11,19,21,7,13,23,27,29,43,15\}}$

Table 2.5: Cyclotomic cosets used for codes in Table 2.4.

C	D	$D^{\perp}$	$C * D$	$(C * D)^{\perp}$	Privacy	Rate
$[255, 9, 127]$	$[255, 9, 127]$	$[255, 246, 4]$	$[255, 37, 63]$	$[255, 218, 8]$	3	$\frac{218}{255}$
[255,9,127]	$[\mathbf{255, 25, \geq 63}]$	$[\mathbf{255, 230, \geq 8}]$	$[\mathbf{255, 93}]$	$[\mathbf{255, 162}]$	$\geq \mathbf{7}$	$\mathbf{\frac{162}{255}}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 33, \geq 63}]$	$[\mathbf{255, 222, \geq 8}]$	$[\mathbf{255, 93}]$	$[\mathbf{255, 162}]$	$\geq \mathbf{7}$	$\mathbf{\frac{162}{255}}$
$[255, 9, 127]$	$[255, 37, 63]$	$[255, 218, 8]$	$[255, 93, 31]$	$[255, 162, 16]$	7	$\frac{162}{255}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 77, \geq 31}]$	$[\mathbf{255, 178, \geq 16}]$	$[\mathbf{255, 161}]$	$[\mathbf{255, 94}]$	$\geq \mathbf{15}$	$\mathbf{\frac{94}{255}}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 85, \geq 31}]$	$[\mathbf{255, 170, \geq 16}]$	$[\mathbf{255, 163}]$	$[\mathbf{255, 92}]$	$\geq \mathbf{15}$	$\mathbf{\frac{92}{255}}$
$[255, 9, 127]$	$[255, 93, 31]$	$[255, 162, 16]$	$[255, 163, 15]$	$[255, 92, 32]$	15	$\frac{92}{255}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 133, \geq 15}]$	$[\mathbf{255, 122, \geq 32}]$	$[\mathbf{255, 219}]$	$[\mathbf{255, 36}]$	$\geq \mathbf{31}$	$\mathbf{\frac{36}{255}}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 141, \geq 15}]$	$[\mathbf{255, 114, \geq 32}]$	$[\mathbf{255, 219}]$	$[\mathbf{25536}]$	$\geq \mathbf{31}$	$\mathbf{\frac{36}{255}}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 149, \geq 15}]$	$[\mathbf{255, 106, \geq 32}]$	$[\mathbf{255, 219}]$	$[\mathbf{255, 36}]$	$\geq \mathbf{31}$	$\mathbf{\frac{36}{255}}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 153, \geq 15}]$	$[\mathbf{255, 102, \geq 32}]$	$[\mathbf{255, 219}]$	$[\mathbf{25536}]$	$\geq \mathbf{31}$	$\mathbf{\frac{36}{255}}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 161, \geq 15}]$	$[\mathbf{255, 94, \geq 32}]$	$[\mathbf{255, 219}]$	$[\mathbf{255, 36}]$	$\geq \mathbf{31}$	$\mathbf{\frac{36}{255}}$
$[255, 9, 127]$	$[255, 163, 15]$	$[255, 92, 32]$	$[255, 219, 7]$	$[255, 36, 64]$	31	$\frac{36}{255}$
$[\mathbf{255, 9, 127}]$	$[\mathbf{255, 211, \geq 7}]$	$[\mathbf{255, 44, \geq 64}]$	$[\mathbf{255, 247}]$	$[\mathbf{255, 8}]$	$\geq \mathbf{63}$	$\mathbf{\frac{8}{255}}$
$[255, 9, 127]$	$[255, 219, 7]$	$[255, 36, 64]$	$[255, 247, 3]$	$[255, 8, 128]$	63	$\frac{8}{255}$

Table 2.6: Comparison with punctured RM codes (Shadow rows)

Unbold rows in Table 2.4 and Table 2.6 display the parameters of those codes obtained by the star product of two cyclic codes, equivalent to the punctured RM codes. **Bold** rows are obtained by the star product of a cyclic code and the fixed code C. Consequently, when the rate and the storage codes are fixed, cyclic codes provide the same parameters as punctured RM ones except D with parameters $[255, 77, 31]$ which gives a better rate than RM codes. However, for a fixed-length n, the dimension of RM codes overgrow, thus for a fixed $C \star D$, there are not

C	D
$U_{\{0,1\}}$	$U_{\{0,1\}}$
	$U_{\{0,1,3,5\}}$
	$U_{\{0,1,3,5,9\}}$
	$U_{\{0,1,3,5,9,17\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,25\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37,15,23,27,29,39\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37,15,23,27,29,39,53\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37,15,23,27,29,39,45,53\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37,15,23,27,29,39,45,51,53\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37,15,23,27,29,39,43,45,51,53\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37,15,23,27,29,39,43,45,51,53,85\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37,15,23,27,29,39,43,45,51,53,85,31,47,55,59,61,87\}}$
	$U_{\{0,1,3,5,9,17,7,11,13,19,21,25,37,15,23,27,29,39,43,45,51,53,85,31,47,55,59,61,87,91\}}$

Table 2.7: Cyclotomic cosets used for codes in Table 2.6.

many values that the dimensions of the code C and D may take. Hence, the first advantage of using cyclic codes in the PIR scheme is to easily provide a larger constellation of parameters.

As an illustration of this fact, in the fourth and fifth rows of Table 2.4, the dimension of D can be 50 or 57 other than 64, or the dimension of D can be 85 or 92, different than 99. Thus, we have different options for the same rate and privacy. The following remark will show the method we used for obtaining the codes in Table 2.4 and Table 2.6.

Remark 44. The r-th order punctured generalized RM code is the cyclic code length $n = q^m - 1$ with generator polynomial

$$g(x) := \prod_{i \in I} (x - \alpha^i), \; where \; I = \{i : \; w_q(i) \leq (q-1)c\}, \tag{2.2}$$

for some $c \in \mathbb{Z}^+$ and $w_q(i)$ is the number of non-zeros in the q-ary expression of i. Now using Equation (2.2), we have created the unbolded row in Table 2.4 and Table 2.6. Namely, if we add or remove some cyclotomic classes to the punctured RM code's generating sets, we can get another cyclic code, which provides the same rate and privacy. While making these additions and removals of cosets, we use Remark 19 to decide the heuristics of which cyclotomic cosets we select. Note that the minimum distance of the code $D^\perp$ provides the privacy of the scheme, so we wish $d(D^\perp) - 1$ being as big as possible. For this purpose, we set $-I$ to be a large set of consecutive elements.

For instance, the third row in Table 2.4, the generating set of C is comprised of U_0 and U_1, and D is comprised of U_0, U_1, U_3, U_5, U_9. The generating set of code D in the second row consists of U_0, U_1, U_5, U_9 by removing U_3. We have removed U_3, because U_3 does not change the BCH bound of the code. Moreover, in the fourth row in Table 2.6, the generating set of D

is comprised of U_0, U_1, U_{17}, U_9, U_5, U_3. Removing U_{17} is not affecting the BCH bound of the code and, thus we obtain the parameter in the third row.

We also achieved a second advantage, more privacy, by reducing the dimension of the storage code C, which is not equivalent to punctured Reed-Muller codes. We remark that the upload cost in the PIR scheme can be neglected [4], thus we focus on the value $d(D^{\perp}) - 1$, which provides privacy, and on $dim(C \star D)^{\perp}/n$, which gives the PIR rate. Therefore, we can reduce the dimension of the code C.

Example 45. Consider the punctured RM codes C_{RM} and D_{RM} with parameters $[63, 7, 31]$ and $[63, 42, 7]$, respectively. One has that the product code $C_{RM} \star D_{RM}$ has parameters $[63, 57, 3]$. The PIR scheme given using C_{RM} and D_{RM} protects against $d_{D^{\perp}} - 1 = 15$ collusions. Consider now the cyclic code C with parameters $[63, 2, 42]$, where the generating set of C is equal to U_{21}, and the cyclic code D with parameters $[63, 51, 3]$. One has that $C \star D = C_{RM} \star D_{RM}$. In this case $d_{D^{\perp}} - 1 = 19$. Therefore, our cyclic code proposal protects against a more significant number of colluding servers for the same rate.

Remark 46. Note that for length 63, there are two good cyclic codes in terms of the PIR parameters, one has dimension 2 (see Example 45) and the second one, D with parameters $[63, 40, 7]$, provides the same rate and privacy of a RM code. In the case of length 31, there are no binary cyclic codes that improve the rate or privacy of a RM code other than the ones equivalent to them. This is why we consider binary cyclic codes of length greater than or equal to 127.

C	D	$D^{\perp}$	$(C \star D)^{\perp}$	Privacy	Rate
$[255, 2, 170]$	$[255, 192]$	$[255, 63, \mathbf{20}+\mathbf{45}]$	$[255, \mathbf{19}]$	64	19/255
$[255, 2, 170]$	$[255, 195]$	$[255, 60, \mathbf{15}+\mathbf{51}]$	$[255, \mathbf{8}]$	65	8/255
$[255, 2, 170]$	$[255, 198]$	$[255, 57, \mathbf{15}+\mathbf{53}]$	$[255, \mathbf{8}]$	67	8/255
$[255, 2, 170]$	$[255, 200]$	$[255, 55, \mathbf{29}+\mathbf{41}]$	$[255, \mathbf{11}]$	69	11/255
$[255, 2, 170]$	$[255, 201]$	$[255, 54, \mathbf{40}+\mathbf{32}]$	$[255, \mathbf{9}]$	71	9/255
$[255, 2, 170]$	$[255, 202]$	$[255, 53, \mathbf{28}+\mathbf{44}]$	$[255, \mathbf{8}]$	71	8/255
$[255, 2, 170]$	$[255, 204]$	$[255, 51, \mathbf{14}+\mathbf{60}]$	$[255, \mathbf{11}]$	73	11/255

Table 2.8: Reducing the dimension of the storage

Table 4.8 contains more examples where, by using cyclic codes, the dimension of the storage code has been reduced. In this table, the minimum distance of $D^{\perp}$, which is related to the privacy, was first evaluated by the BCH bound and then its real value was computed using the powerful minimum distance algorithm in [35] (for instance, by $20 + 15$ we mean that the BCH bound is equal to 20 and the real minimum distance is equal to 35). The table displays the privacy (number of colluding servers) and rate of the PIR scheme obtained using a code C with length 255 and dimension 2. Note that the PIR scheme obtained using the Punctured RM codes C_{RM} and $D_{RM}^{\perp}$ with parameters $[255, 9, 127]$ and $[255, 36, 64]$, respectively, protects against a maximum of 63 colluding servers. Moreover, the PIR rate of this scheme is equal to 8/255. In Table 2.8, the code pairs in all rows protect against more than 63 collusions. The cyclotomic cosets used for constructing the codes in Table 2.8 are given in Table 2.9.

C	D
U_{85}	$V \setminus (U_{\{0,1,11,13,17,21,25,61,85,87\}})$
	$V \setminus (U_{\{1,13,25,27,29,31,45,119\}})$
	$V \setminus (U_{\{0,1,7,13,25,31,39,45\}})$
	$V \setminus (U_{\{0,1,13,17,25,29,31,63,85\}})$
	$V \setminus (U_{\{39,55,61,63,85,87,119,127\}})$
	$V \setminus (U_{\{0,1,9,13,25,31,111,119\}})$
	$V \setminus (U_{\{0,1,11,13,29,47,85,111\}})$

Table 2.9: Cyclotomic cosets used for codes in Table 2.8, where V is the set of all cyclotomic cosets for $q = 2$, modulo $n = 255$.

In Table 2.10, shortened RM codes at the evaluation of **0** and cyclic codes with length 127 are analyzed. Again, we specify **bold** rows for star product of cyclic codes and unbold rows for star product of shortened RM codes. The storage code C with parameters $[127, 7, 64]$, equivalent to a shortened RM, is fixed. The only difference with respect to Table 2.4 is that we do not include the cyclotomic coset U_0 in the generating set of C.

C	D	$D^{\perp}$	$C * D$	$(C * D)^{\perp}$	Privacy	Rate
$[127, 7, 64]$	$[127, 7, 64]$	$[127, 120, 3]$	$[127, 28, 32]$	$[127, 99, 7]$	2	$\frac{99}{127}$
$\mathbf{[127, 7, 64]}$	$\mathbf{[127, 22, 47]}$	$\mathbf{[127, 105, 8]}$	$\mathbf{[127, 63, 16]}$	$\mathbf{[127, 64, 15]}$	**7**	$\mathbf{\frac{64}{127}}$
$[127, 7, 64]$	$[127, 28, 32]$	$[127, 99, 7]$	$[127, 63, 16]$	$[127, 64, 15]$	6	$\frac{64}{127}$
$\mathbf{[127, 7, 64]}$	$\mathbf{[127, 50, 23]}$	$\mathbf{[127, 77, 16]}$	$\mathbf{[127, 98, 8]}$	$\mathbf{[127, 29, 31]}$	**15**	$\mathbf{\frac{29}{127}}$
$\mathbf{[127, 7, 64]}$	$\mathbf{[127, 57, 23]}$	$\mathbf{[127, 70, 16]}$	$\mathbf{[127, 98, 8]}$	$\mathbf{[127, 29, 31]}$	**15**	$\mathbf{\frac{29}{127}}$
$[127, 7, 64]$	$[127, 63, 16]$	$[127, 64, 15]$	$[127, 98, 8]$	$[127, 29, 31]$	14	$\frac{29}{127}$
$\mathbf{[127, 7, 64]}$	$\mathbf{[127, 85, 13]}$	$\mathbf{[127, 42, 32]}$	$\mathbf{[127, 119, 4]}$	$\mathbf{[127, 8, 63]}$	**31**	$\mathbf{\frac{8}{127}}$
$\mathbf{[127, 7, 64]}$	$\mathbf{[127, 92, 11]}$	$\mathbf{[127, 35, 32]}$	$\mathbf{[127, 119, 4]}$	$\mathbf{[127, 8, 63]}$	**31**	$\mathbf{\frac{8}{127}}$
$[127, 7, 64]$	$[127, 98, 8]$	$[127, 29, 31]$	$[127, 119, 4]$	$[127, 8, 63]$	30	$\frac{8}{127}$

Table 2.10: Comparison with shortened RM codes (Shadow rows)

One has that the PIR schemes using cyclic codes protect against one more colluding server than shortened RM codes, as it can be seen at Table 2.10. Moreover, in this way we may increase the constellation of possible parameters. For instance, for a case rate equal to 29/127, the PIR scheme coming from a cyclic code protects against 15-collusion, but the one from a shortened RM code protects against 14-collusion. The cyclotomic cosets used for constructing the codes in Table 2.10 are given in Table 2.11.

C	D
$U_{\{1\}}$	$U_{\{1\}}$
	$U_{\{0,1,5,9\}}$
	$U_{\{1,5,9,3\}}$
	$U_{\{0,1,5,9,3,11,19,21\}}$
	$U_{\{0,1,5,9,3,11,19,21,7\}}$
	$U_{\{1,5,9,3,11,19,21,7,13\}}$
	$U_{\{0,1,5,9,3,11,19,21,7,13,23,27,43\}}$
	$U_{\{0,1,5,9,3,11,19,21,7,13,23,27,43,29\}}$
	$U_{\{1,5,9,3,11,19,21,7,13,23,27,29,43,15\}}$

Table 2.11: Cyclotomic cosets used for codes in Table 2.10.

Chapter 3

Private Information Retrieval Schemes for Single-Server Systems Based on Codes over Rings

This chapter provides a single server PIR protocol based on coding theory over rings which is resistant to the attack in [10] based on detecting a rank difference in the query matrix by deleting rows. Our protocol can be understood as a modification of the protocol in [37] and it will be based on codes over finite rings. We take advantage of the fact that some of the involved codes are non-free as ring modules, which makes linear algebra attacks non-feasible. Non-free codes are linear codes that cannot be generated by a basis consisting of linearly independent codewords. In other words, for a non-free code, its codewords cannot be expressed uniquely as linear combinations of a set of basis codewords. This lack of a basis with linearly independent codewords complicates the encoding and decoding procedures compared to free codes (free codes is the case for codes over fields). However, the lack of a simple basis representation will be an advantage for us since simple linear algebra can not be used to describe them without some side information. For further information on these codes, we refer to [5] and the references therein.

We will require two finite rings $R \subset \mathcal{R}$, such that $\mathcal{R}$ can be seen as a R-submodule. We will define in $\mathcal{R}$ an R-linear code called inner code and we will also establish a $\mathcal{R}$-linear code as the outer code. All the transmitted information and the computations needed will be made with R as the alphabet whereas the information of the codes will be used in the query generation and response processing, both by the user.

We propose to consider $R = \mathbb{Z}_m$, the set of integers modulo a composite number m, and

$$\mathcal{R} = \mathbb{Z}_m[x]/\langle x^n - 1\rangle,$$

with $\gcd(n, m) = 1$. As inner code, we will use a $\mathbb{Z}_m$-cyclic code that can be seen as a Chinese remainder construction of its prime components (depending on the prime factorization of m) and the outer code will be a matrix-product code with constituents in $\mathcal{R}$ that turns out to be a quasi-cyclic code when it is seen as a code over $\mathbb{Z}_m$. Note that the condition $\gcd(n, m) = 1$ is usually known as the semisimplicity condition and it ensures that the factorization of $x^n - 1$ in $\mathbb{Z}_m$ does not have multiple roots and that we can control the factorization just in terms of the cyclotomic cosets. In the language of coding theory, this means that there is a one-to-one

correspondence between a given cyclic code and its defining set as the disjoint union of several cyclotomic cosets.

3.1 Basic Single Server PIR Scheme

Before detailing the protocols in 3.1.1 and introducing our proposed protocol, we first give the following basic code base single server PIR scheme, which is illustrated in 3.1 and explained in [1].

The server has t files denoted as $\mathrm{db}^i \in \mathbb{F}_q$. For $i \in \{1, \ldots, t\}$. Let $\mathbf{a^i} \in \mathbb{F}_q^k$ be randomly chosen vectors, encoded by the generator matrix, G, of the code C, such that $\mathrm{Enc}(\mathbf{a}^i) = \mathbf{a}^i G = \mathbf{w}^i$. Suppose the user aims to retrieve the file d. The user selects random vectors $\mathbf{u}^i \in \mathbb{F}_q^n$ that are zero in the positions corresponding to the information set I of the code C. To retrieve the desired file, the user selects a column position $j \in \{1, \cdots, n\} \setminus I$ such that $u_j^i = 0$ for all $i \neq d$, this means that the u_j^i has a nonzero entry only in the row corresponding to the desired file. The queries are then formed as $\mathbf{q^i} = \mathbf{w^i} + \mathbf{u^i}$. The query matrix $\mathbf{Q}$ consists of rows $\mathbf{q^i}$ given by:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q^1} \\ \vdots \\ \mathbf{q^d} \\ \vdots \\ \mathbf{q^t} \end{bmatrix} = \begin{bmatrix} w_1^1 & \cdots & w_j^1 & \cdots & w_n^1 \\ & & \vdots & & \\ w_1^d & \cdots & w_j^d + u_j^d & \cdots & w_n^d \\ & & \vdots & & \\ w_1^t & \cdots & w_j^t & \cdots & w_n^t \end{bmatrix}.$$

The user sends Q to the server. After receiving the query, the server computes

$$\mathbf{r} = \sum_{i=1}^{t} db^i \mathbf{q}^i$$

and returns the response $\mathbf{r}$ to the user. To extract the desired file, the user first focuses on the response values at the positions of the information set. Since the error vector $\mathbf{u^i}$ is zero in these positions, the response restricted to I can be expressed as:

$$\mathbf{r_I} = \left(\sum_{i=1}^{t} db^i \cdot \mathbf{a^i} \right)_I \cdot G_I$$

where G_I is the submatrix of the generator matrix G, consisting of the columns indexed by I. The user multiply $\mathbf{r_I}$ by G_I^{-1} and get:

$$\mathbf{r_I} G_I^{-1} = \left(\sum_{i=1}^{t} db^i \cdot \mathbf{a}^i \right)_I.$$

Re-encoding this using the generator matrix G, we obtain:

$$(\mathbf{r_I} G_I^{-1}) G = \sum_{i=1}^{t} db^i \cdot \mathbf{w}^i.$$

Subtract this term from the original response:

$$\mathbf{r} - \mathbf{r_I} G_I^{-1} G = \sum_{i=1}^{t} db^i \cdot \mathbf{u}^i.$$

Since $\mathbf{u^i}$ is a zero vector except at the selected column position j corresponding to the desired file, the remaining terms reduce to $db^d \cdot u_j^d$. Therefore, the user can successfully recover the desired file db^d.

Server		**User**
$db^1, db^2, \ldots, db^t$		**Query Generation**
		Choose random vector $\mathbf{a}^i \in \mathbb{F}_q^k$,
	$\xleftarrow{\mathbf{Q}}$	Compute $\mathbf{q}^i = \text{Enc}(\mathbf{a}^i) + \mathbf{u}^i$,
Response		
Compute $\mathbf{r} = \sum_{i=1}^{t} \mathbf{q^i} \cdot db^i$	$\xrightarrow{\mathbf{r}}$	**Decoding & File Retrieval**
		Calculate: $\mathbf{r} - \mathbf{r_I} G_I^{-1} G$

Table 3.1: Single-Server PIR Process

The rows of matrix $\mathbf{W}$ correspond to $\mathbf{w^i}$, while the rows of matrix $\mathbf{U}$ correspond to $\mathbf{u^i}$ in Figure 3.1.

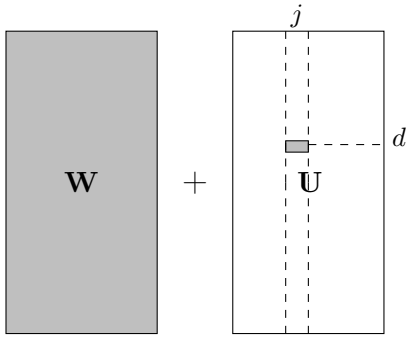


Figure 3.1: Query matrix in the protocol 3.1. White places means entries equal to 0.

The query matrix is constructed as $\mathbf{Q} = \mathbf{W} + \mathbf{U}$, the column span satisfies:

$$\text{columnspan}(\mathbf{Q}) = \text{columnspan}(\mathbf{W} + \mathbf{U}).$$

An adversary can analyze the column span of $\mathbf{Q}$ and identify differences among the columns, particularly by detecting a unit vector, $(0, 0, \ldots, 1, 0, \ldots, 0)^\top$, within the column space. This occurs because only $u_j^d \neq 0$ while all other $u_j^i = 0$ for $i \neq d$. Therefore, the adversary can determine the index of the requested file in polynomial time, and the scheme does not ensure privacy.

3.1.1 HHWZ PIR Protocol

In the single server HHWZ PIR protocol [37], the setup is the following. There are t files stored on the server, m_i for $i \in \{1, \ldots, t\}$. For simplicity, we consider here that the files are a vector of length L with entries in $\mathbb{F}_q$.

Let $s > 0$ be an integer and let $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_s\}$ be a basis of $\mathbb{F}_{q^s}$ as a vector space over $\mathbb{F}_q$. Let us consider $V = \langle \mathbf{b}_1, \ldots, \mathbf{b}_v \rangle_{\mathbb{F}_q}$ and $V^c = \langle \mathbf{b}_{v+1}, \ldots, \mathbf{b}_s \rangle_{\mathbb{F}_q}$, two linear subspaces. The user selects a random $[n, k]_{q^s}$ code $C \subseteq \mathbb{F}_{q^s}^n$, and let $\mathbf{W}$ be a matrix of size $t \times n$ over $\mathbb{F}_{q^s}$ whose rows are selected uniformly at random from codewords in C. For an information set I, of the code C, the user selects a matrix $\mathbf{E}$ of size $t \times n$ and entries in the subspace V, such that a column in $\mathbf{E}$ is the all-zero vector if its index is in I. Finally, the user chooses a $t \times n$ matrix $\mathbf{U}$ whose elements $\mathbf{U}_{i,j}$ are in V^c and $\mathbf{U}_{i,j}$ is a non-zero element when i points to the desired file and $j \in I$; and the entries $\mathbf{U}_{i,j}$ are zero elsewhere. The query matrix is given by $\mathbf{Q} = \mathbf{W} + \mathbf{E} + \mathbf{U}$, which the user sends to the server. The server computes $\mathbf{r} = \sum_{i=1}^{t} m_i \cdot \mathbf{q}_i$ where $\mathbf{q}_i$ represents rows of the query matrix and then sends the response $\mathbf{r}$ to the user.

Based on the previous fixed information set I, the user can eliminate the codewords of C from the response matrix:

$$\mathbf{r} - \mathbf{r}_I G_I^{-1} G = \sum_{j \in I} m_i \cdot \mathbf{i}_j,$$

where $\mathbf{i}_j$ is the j-th coordinate vector in $\mathbb{F}_{q^s}^n$. Consider the projection over the subspace V given by

$$\begin{array}{rcl} \text{proj} : \mathbb{F}_{q^s} & \longrightarrow & \mathbb{F}_{q^s} \\ \sum_{i=1}^{s} x_i \cdot \mathbf{b}_i & \longmapsto & \sum_{i=1}^{v} x_i \cdot \mathbf{b}_i. \end{array}$$

Then, the desired file can be recovered by applying the projection of $\mathbb{F}_{q^s}$ over V^c for the j^{th} coordinate.

$$\text{proj}(\sum_{i=1}^{t} m_i \cdot \mathbf{e}_i^j) = m_d \cdot \mathbf{e}_j^d,$$

where $\mathbf{e}^d$ is the d^{th} row of the matrix $\mathbf{E}$. As the user knows $\mathbf{e}_j^d$, the desired file can be recovered.

It was shown in [10] an attack based on removing a row from the query matrix and checking whether the resulting matrix has a lower rank than the original matrix. In this way, the server may determine the desired file with a high probability. Our protocol avoids this attack.

3.1.2 Single Server PIR protocol Over Rings

In this section, we describe our PIR protocol. We will consider two rings $R \subset \mathcal{R}$ such that $\mathcal{R}$ is an R-module, and two ring-linear codes, namely a code over $\mathcal{R}$ that will play a similar role as the code C in HHWZ PIR protocol and a R-code in $\mathcal{R}$ as an R-submodule. We will call the code over $\mathcal{R}$ the *outer code* and the R-subcode of $\mathcal{R}$ the *inner code*.

From now on we will fix the following notation: a vector (respectively a matrix) with entries in $\mathcal{R}$ will be denoted with round brackets, ( ). Note that, since $\mathcal{R}$ is an R-submodule if we fix an R-generating set of $\mathcal{R}$ any element $r \in \mathcal{R}$ can be expressed as a row vector in R^n, where n is the number of generators of $\mathcal{R}$ as R-module, we will denote that expansion with square brackets,

[]. In the same fashion, this can be done componentwise to vectors and matrices with entries in $\mathcal{R}$ and we will also use square brackets for that row-wise expansion of the entries of matrices and vectors.

More concretely, throughout this chapter, we will fix the ring $\mathcal{R}$, i.e. the alphabet for the query generation process, to be $\mathcal{R} = R[x]/\langle x^n - 1\rangle$. The alphabet for the projection will be $R = \mathbb{Z}_m$, the set of integers modulo m for a fixed $m \in \mathbb{Z}$. Moreover, we will take $C_{\mathtt{IN}}$ to be an ideal in $\mathcal{R}$, that is, a $\mathbb{Z}_m$ cyclic code of length n. As usual, one can describe cyclic codes in $\mathcal{R}$ using the Chinese remainder theorem (CRT), that is, if $m = \Pi_{i=1}^{\ell} p_i^{e_i}$ is the prime decomposition of m, then

$$\begin{aligned}\Phi : \frac{\mathbb{Z}_m[x]}{\langle x^n-1\rangle} &\longrightarrow \bigoplus_{i=1}^{\ell} \frac{\mathbb{Z}_{p_i^{e_i}}[x]}{\langle x^n-1\rangle} \\ \sum_{j=0}^{n-1} c_j x^j &\longmapsto (\sum_{j=0}^{n-1} c_j (\mathrm{mod} p_1^{e_1}) x^j, \ldots, \sum_{j=0}^{n-1} c_j (\mathrm{mod} p_\ell^{e_\ell}) x^j),\end{aligned}$$

is a module isomorphism. Moreover, if we let $C_1, C_2, \ldots, C_s$ be cyclic codes in $\mathbb{Z}_{p_i^{e_i}}[x]/\langle x^n - 1\rangle$, for $i \in \{1, \ldots, s\}$, then the set $\mathrm{CRT}(C_1, \ldots, C_s) = \{\Phi^{-1}(v_1, v_2, \ldots, v_s) \mid v_i \in C_i\}$ is a $\mathbb{Z}_m$-cyclic code of length n. That is, it corresponds to an ideal in $\mathcal{R} = \mathbb{Z}_m[x]/\langle x^n - 1\rangle$. From [5, Remark 2], we have that $\mathrm{CRT}(C_1, \ldots, C_s)$ is a non-free $\mathbb{Z}_m$-module if and only if there is at least a pair $i, j \in \{1, 2, \ldots, s\}$, with $i \neq j$, such that $\mathrm{rank}_{p_i^{e_i}}(C_i) \neq \mathrm{rank}_{p_j^{e_j}}(C_j)$, where $\mathrm{rank}_{p_i^{e_i}}$ denotes the rank as $\mathbb{Z}_{p_i^{e_i}}$-module.

The inner code in our system $C_{\mathtt{IN}}$ will be a non-free cyclic code over $\mathbb{Z}_m$ of length n that will be used for encoding. Besides the code being non-free there is a matrix that can play the role of a parity check matrix (see [51]) that can be used as a distinguisher of whether an element in $\mathcal{R}$ is in $C_{\mathtt{IN}}$ or in $\mathcal{R} \setminus C_{\mathtt{IN}}$. In the case that $C_{\mathtt{IN}}$ is an LCD code there is a projection in the alphabet as in the HHWZ PIR protocol [37]. Therefore, in our example, $C_{\mathtt{IN}}$ will then be a non-free LCD, that type codes has been characterized in [5].

As an outer code $C_{\mathtt{OUT}}$, we will consider a linear code of length s over $\mathcal{R}$. Note that, since $\mathcal{R}$ is chosen to be a cyclic ambient space, the set $[C_{\mathtt{OUT}}] = \{[\mathbf{v}] \mid (\mathbf{v}) \in C_{\mathtt{OUT}}\}$ is a quasi-cyclic code over $\mathbb{Z}_m$ of length ns. A linear code C of length ns over the ring R is said to be a quasi-cyclic code of index s if it is invariant under the cyclic shift of codewords by s positions and s is the smallest number with this property. We will define the outer code through its expansion, $[C_{\mathtt{OUT}}]$ over $\mathbb{Z}_m$, using a matrix-product code.

Let $\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_s \subset \mathcal{R}$, where $\tilde{C}_i$ is a cyclic code over $\mathbb{Z}_m$ for $i \in \{1, \ldots, s\}$. Let $[C_{\mathtt{OUT}}]$ be the matrix-product code $[C_{\mathtt{OUT}}] = [\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_s]M$, where M is an $s \times s$ matrix over $\mathbb{Z}_m$. Hence, the outer code $C_{\mathtt{OUT}}$ is a $\mathcal{R}$ submodule in $\mathcal{R}^s$ that is a s-generator quasi-cyclic code. We will denote by $G_{\mathtt{OUT}}$ the generator matrix over $\mathcal{R}$.

For technical reasons in the recovery stage and the security analysis, we will require that our codes fulfill the following technical conditions:

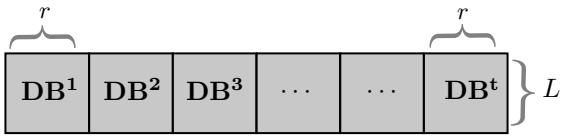
- The constituent codes are nested: $\tilde{C}_1 \supseteq \tilde{C}_2 \supseteq \cdots \supseteq \tilde{C}_s$.
- For $i \in \{1, \ldots, s\}$, $\tilde{C}_i \cap C_{\mathtt{IN}} \neq \{0\}$, and $\tilde{C}_i \cap (C_{\mathtt{IN}}^{\perp} \setminus C_{\mathtt{IN}}) \neq \{0\}$.
- $e_i > 1$, for all $i = 1, \ldots, \ell$.

- The projections of the codes $\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_s$ over $\mathbb{Z}_{p_i^{e_i}}$ are non-Hensel lifts (see Remark 52 for a definition), for $i = 1, \ldots, \ell$.

Remark 47. Note that other possibilities for inner and outer codes may be chosen. That is, $C_{\mathtt{IN}} \subset \mathcal{R}$, and $C_{\mathtt{OUT}}$ being a code with alphabet $\mathcal{R}$, can be achieved in many other situations. For example, the broader class of codes over affine algebras with a finite commutative chain coefficient ring in [46], that includes uni-(multivariate) codes (cyclic, negacyclic, constacyclic, polycyclic, abelian ...), quasi-cyclic, quasi-abelian, and many other can be used in this protocol.

3.1.2.1 Data Storage Process

We assume that the data alphabet in the server is $\mathbb{Z}_{m'} \subset \mathbb{Z}_m$ where $m' = \Pi_{i=1}^{\ell} p_i$. The server will contain t files stored as an $L \times r$ matrix of elements in $\mathbb{Z}_{m'}$ denoted as $\mathbf{DB^i} = [D^i_{j\ell}]$ where $i \in \{1, \ldots, t\}$, $j \in \{1, \ldots, L\}$, $\ell \in \{1, \ldots, r\}$ and all files are stored concatenated in the matrix $\mathbf{DB} = [\mathbf{DB^1} \parallel \mathbf{DB^2} \parallel \cdots \parallel \mathbf{DB^t}]$, where the number of columns r of each file is lower than or equal to s, the number of generators of $C_{\mathtt{OUT}}$, i.e. $r \leq s$. Throughout the chapter, $\mathbf{DB^d}$ will denote the desired file that the user wants to retrieve.



3.1.2.2 Query Generation

First, the user sets up the codes. The inner code $C_{\mathtt{IN}} \subset \mathcal{R} = \mathbb{Z}_m[x]/\langle x^n - 1\rangle$ is a cyclic $\mathbb{Z}_m$-linear code of length n. The user also chooses $C_{\mathtt{OUT}}$ as an s-generator quasi-cyclic code in $\mathcal{R}^s$ arising from a matrix product code $[\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_s]M$, such that $\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_s \subset \mathcal{R}$, where each $\tilde{C}_i$ is a cyclic code over $\mathbb{Z}_m$ for $i \in \{1, \ldots, s\}$ with the technical conditions stated above (see Algorithm 1) We will denote by $G_{\mathtt{OUT}}$ a generator matrix of $C_{\mathtt{OUT}}$ as an $\mathcal{R}$-linear code.

The query generation begins with the user randomly selecting $s \cdot t \cdot r$ elements a^i_{kj} in the $m'\mathcal{R}$. The user arranges these elements as t matrices $\mathbf{a}^i$ of size $r \times s$, where the rows consists of s-uples in $\mathcal{R}^s$

$$\mathbf{a}^i = \begin{pmatrix} \mathrm{a}^{\mathrm{i}}_{11} & \mathrm{a}^{\mathrm{i}}_{12} & \ldots & \mathrm{a}^{\mathrm{i}}_{1\mathrm{s}} \\ \mathrm{a}^{\mathrm{i}}_{21} & \mathrm{a}^{\mathrm{i}}_{22} & \ldots & \mathrm{a}^{\mathrm{i}}_{2\mathrm{s}} \\ \vdots & \vdots & & \vdots \\ \mathrm{a}^{\mathrm{i}}_{\mathrm{r}1} & \mathrm{a}^{\mathrm{i}}_{\mathrm{r}2} & \ldots & \mathrm{a}^{\mathrm{i}}_{\mathrm{rs}} \end{pmatrix},$$

where $i \in \{1, \ldots, t\}$, $k \in \{1, \ldots, r\}$, and $j \in \{1, \ldots, s\}$. Then the user encodes $\mathbf{a}^i$ as

$$\mathbf{w}^i = \mathbf{a}^i \cdot G_{\mathtt{OUT}},$$

i.e. the rows in $\mathbf{w}^i$ are codewords of $C_{\mathtt{OUT}}$. Note that, in this encoding, all the multiplications are carried out in the ring $\mathcal{R}$.

Now the user selects t matrices $\mathbf{e}^i$ of size $r \times s$, for $i \in \{1, \ldots, t\}$, where each entry $e^i_{kj} \in \mathrm{nf}(C_{\mathtt{IN}})$, for each $i \in \{1, \ldots, t\}$, $k \in \{1, \ldots, r\}$ and $j \in \{1, \ldots, s\}$, where $\mathrm{nf}(C)$ denotes the non-free part of the code C. Moreover, the user randomly selects a column position $\gamma \in \{1, .., s-r+1\}$

and constructs t matrices $\mathbf{u}^i$ of size $r \times s$, for $i \in \{1, \ldots, t\}$, with all zero entries but

$$\mathrm{u}^{\mathrm{d}}_{1+\lambda,\gamma+\lambda} \in \mathrm{nf}(\tilde{C}_s \cap (C^{\perp}_{\mathtt{IN}} \setminus C_{\mathtt{IN}})), \text{ for all } \lambda = 0, \ldots, r-1. \tag{3.1}$$

We recall that d is the index of the file the user wants to retrieve from the server. Thus, $\mathbf{u}^i$ has zeros in all positions except in those entries belonging to the desired file. Let $\boldsymbol{\delta}^i = (\mathbf{w^i} + \mathbf{e^i} + \mathbf{u^i})$, for $i \in \{1, \ldots, t\}$, and $\boldsymbol{\Delta}$ and $\mathbf{A}$ the matrices with entries in $m'\mathcal{R}$ with $r \cdot t$ rows given by

$$\boldsymbol{\Delta} = \begin{pmatrix} \boldsymbol{\delta}^1 \\ \boldsymbol{\delta}^2 \\ \vdots \\ \boldsymbol{\delta}^d \\ \vdots \\ \boldsymbol{\delta}^t \end{pmatrix} = \begin{pmatrix} \mathbf{w^1} + \mathbf{e^1} + \mathbf{u^1} \\ \mathbf{w^2} + \mathbf{e^2} + \mathbf{u^2} \\ \vdots \\ \mathbf{w^d} + \mathbf{e^d} + \mathbf{u^d} \\ \vdots \\ \mathbf{w^t} + \mathbf{e^t} + \mathbf{u^t} \end{pmatrix} = \begin{pmatrix} \mathbf{w^1} + \mathbf{e^1} \\ \mathbf{w^2} + \mathbf{e^2} \\ \vdots \\ \mathbf{w^d} + \mathbf{e^d} + \mathbf{u^d} \\ \vdots \\ \mathbf{w^t} + \mathbf{e^t} \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} \mathbf{a}^1 \\ \mathbf{a}^2 \\ \vdots \\ \mathbf{a}^d \\ \vdots \\ \mathbf{a}^t \end{pmatrix}. \tag{3.2}$$

The user generates the query matrix $\mathbf{Q}$ with entries in $\mathbb{Z}_m$ by expanding in the ring $\mathbb{Z}_m$ the concatenation $(\mathbf{A} \parallel \boldsymbol{\Delta})$, that is $\mathbf{Q} = [\mathbf{A} \parallel \boldsymbol{\Delta}]$. For a fixed $i \in \{1, \ldots, t\}$, we will denote by $[\mathbf{q^i}]$, the submatrix of the query matrix whose rows are given by the rows $(i-1)r + 1, \ldots, ir$ in $\mathbf{Q}$, that is $[\mathbf{q^i}] = [\mathbf{a}^i \parallel \boldsymbol{\delta}^i]$. This procedure has been summarized in Algorithm 2.

3.1.2.3 Response

Note that in the database, for a fixed $i \in \{1, \ldots, t\}$, the i^{th} file, is an $r \times L$ matrix with entries in $\mathbb{Z}_m$ that we denote by $\mathbf{DB^i}$. Then the response is just the matrix multiplication over $\mathbb{Z}_m$ given by

$$\mathbf{R} = \mathbf{DB} \cdot \mathbf{Q}, \quad \text{where } \mathbf{DB} = [\mathbf{DB^1} \parallel \mathbf{DB^2} \parallel \cdots \parallel \mathbf{DB^t}]. \tag{3.3}$$

Note that, since $\mathbb{Z}_m \subset \mathcal{R}$, the response matrix $\mathbf{R}$ that the server computes is just the $\mathbb{Z}_m$ expansion of the matrix

$$\sum_{i=1}^{t} \mathbf{DB^i} \cdot [\mathbf{a}^i \parallel \boldsymbol{\delta}^i] = \sum_{i=1}^{t} [\mathbf{DB^i} \cdot \mathbf{a}^i \parallel \mathbf{DB^i} \cdot [\mathbf{w}^i + \mathbf{e}^i + \mathbf{u}^i]] = [\mathbf{R_1} \parallel \mathbf{R_2}], \tag{3.4}$$

which is a matrix with entries in $\mathcal{R}$, but the server does not know such decomposition since the ring $\mathcal{R}$ is kept secret.

3.1.2.4 Recovering Stage

We will use a recovering method that resembles the technique used in [37] but without information sets. The user can get the matrix $(\mathbf{R_1} \parallel \mathbf{R_2})$ in Equation (3.4) from the matrix $\mathbf{R}$ in Equation (3.3) since the user knows the ring $\mathcal{R}$ and henceforth n. Thus, the user can compute

$$\mathbf{R_2} - (\mathbf{R_1} \cdot G_{\mathtt{OUT}}) = \sum_{i=1}^{t} (\mathbf{DB^i} \cdot \mathbf{w}^i + \mathbf{DB^i} \cdot (\mathbf{e}^i + \mathbf{u}^i) - \mathbf{DB^i} \cdot \mathbf{w}^i) = \sum_{i=1}^{t} (\mathbf{DB^i} \cdot \mathbf{e}^i) + (\mathbf{DB^i} \cdot \mathbf{u^i}). \tag{3.5}$$

Let us denote by $\Gamma_s(C) = [C, \ldots, C]\mathrm{Id}_s$ the matrix product code of a cyclic code $C \subset \mathcal{R}$, where Id_s is the $s \times s$ the identity matrix and let us denote by $H_{\Gamma_r(C^{\perp}_{\mathrm{IN}})}$ a parity check matrix of the code $\Gamma(C_{\mathrm{IN}})$ over $\mathbb{Z}_m$.

Then the user computes

$$\begin{aligned}[\mathbf{R_2} - (\mathbf{R_1} \cdot G_{\mathtt{OUT}})]H^{\top}_{\Gamma_r(C^{\perp}_{\mathrm{IN}})} &= \sum_{i=1}^{t}[\mathbf{DB^i} \cdot \mathbf{e}^i + \mathbf{DB^i} \cdot \mathbf{u}^i]H^{\top}_{\Gamma_r(C^{\perp}_{\mathrm{IN}})} \\ &= \sum_{i=1}^{t}\underbrace{[\mathbf{DB^i} \cdot \mathbf{e}^i H^{\top}_{\Gamma_r(C^{\perp}_{\mathrm{IN}})}]}_{=\mathbf{0} \text{ since } \mathbf{e}^i_{kj} \in C_{\mathtt{IN}}} + \underbrace{[\mathbf{DB^i} \cdot \mathbf{u}^i H^{\top}_{\Gamma_r(C^{\perp}_{\mathrm{IN}})}]}_{=\mathbf{0}, \text{ when } t \neq d} \\ &= [\mathbf{DB^d} \cdot \mathbf{u}^d H^{\top}_{\Gamma_r(C^{\perp}_{\mathrm{IN}})}] = \mathbf{M}.\end{aligned}$$

Note that the inner code provides us with a projection-like map in $\mathcal{R}$ (see Section 3.1.1). Now, in order to retrieve the desired file the user must solve the following system of equations in $\mathbb{Z}_m$.

$$\mathbf{M} = \begin{bmatrix} \mathbf{m_1} & & \\ & \ddots & \\ & & \mathbf{m_r} \end{bmatrix} = \begin{bmatrix} \mathrm{D}^{\mathrm{d}}_{11} & \cdots & \mathrm{D}^{\mathrm{d}}_{1\mathrm{r}} \\ \vdots & & \vdots \\ \mathrm{D}^{\mathrm{d}}_{\mathrm{L}1} & \cdots & \mathrm{D}^{\mathrm{d}}_{\mathrm{Lr}} \end{bmatrix} \cdot \begin{bmatrix} \mathrm{u}^{\mathrm{d}}_{1,\gamma} H^{\top}_{\mathrm{IN}} & & \\ & \mathrm{u}^{\mathrm{d}}_{2,\gamma+1} H^{\top}_{\mathrm{IN}} & \\ & & \ddots \\ & & & \mathrm{u}^{\mathrm{d}}_{\mathrm{r},\gamma+\mathrm{r}-1} H^{\top}_{\mathrm{IN}} \end{bmatrix} \tag{3.6}$$

We have that this system of equations over $\mathbb{Z}_m$ can be seen as several systems of linear equations, each one can be expressed as

$$\mathbf{m_i} = \begin{bmatrix} \mathrm{D}^{\mathrm{d}}_{1\mathrm{i}} \\ \vdots \\ \mathrm{D}^{\mathrm{d}}_{\mathrm{Li}} \end{bmatrix} \mathrm{u}^{\mathrm{d}}_{\mathrm{i},\gamma} H^{\top}_{\mathrm{IN}}, \text{ for } i \in \{1, \cdots, r\}.$$

Note that $\mathbf{m_i}$ is a codeword in the code $\mathrm{u}^{\mathrm{d}}_{\mathrm{i},\gamma}(C^{\perp}_{\mathrm{IN}})$, hence, once we fix a generator matrix for the code $C^{\perp}_{\mathrm{IN}}$ given by H_{IN}, there is a unique expression for $\mathbf{m_i}$ in terms of the rows of the matrix $\mathrm{u}^{\mathrm{d}}_{\mathrm{i},\gamma} H_{\mathrm{IN}}$ since the database elements are in $\mathbb{Z}_{m'}$, which is the desired solution of the system. For constructing the matrix H_{IN} we use the standard-like generator matrix for codes over that type of rings [51] to build it via the inverse of the CRT. The recovering stage above has been outlined as pseudocode in Algorithm 3

Remark 48 (The finite field case). Note that, if we consider $C_{\mathtt{IN}}$ as a $\mathbb{F}_q$-code in $\mathbb{F}_q[x]/\langle f(x)\rangle$, with f a primitive polynomial (i.e. an $\mathbb{F}_q$ subspace of $\mathbb{F}_q^{\deg(f)}$), and $C_{\mathtt{OUT}}$ as a code with alphabet $\mathbb{F}_{q^s} = \mathbb{F}_q[x]/\langle f(x)\rangle$, both codes operate within finite fields. However, the arithmetic ceases to be modular if q is not a prime. Note that this is nothing else than choosing the generators of $C_{\mathtt{IN}}$ as the basis of V in HHWZ PIR protocol in Section 3.1.1. All the processes will follow as above and this will provide a PIR protocol with a database whose elements belong to the field $\mathbb{F}_q$. In this case, the query matrix will be larger than the one in HHWZ PIR protocol but it will not be safe against the rank difference attack in [10] as we will show in Section 3.1.2.5.3. The primary distinction between the two protocols lies in the utilization of a generating set A, which eliminates the necessity for information sets. Consequently, the information contained in $\mathbf{u}$ can be distributed across the entire query matrix $\mathbf{Q}$, as shown in Figure 3.2.

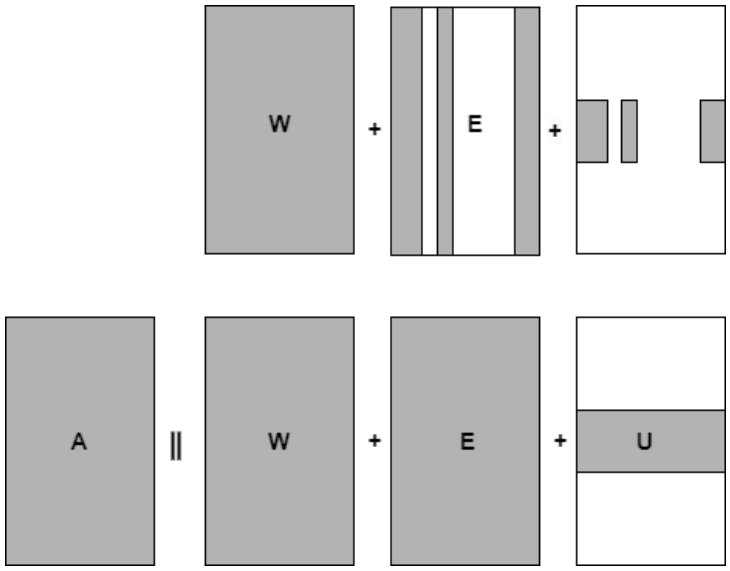


Figure 3.2: Query matrix in the original HHWZ PIR protocol vs. the modified one. White places means entries equal to 0

Example 49 (Toy example)**.** For simplicity we will take $m' = m = 3{\cdot}5$, thus $R = \mathbb{Z}_{15}$. Note that the value m does not fulfill the technical conditions for security and that we are considering that $m' = m$, which is not allowed in our protocol. The aim is to illustrate the protocol with a toy example and simple computations. Consider the 3-file in database $\mathbf{DB} = [\mathbf{DB}^1, \mathbf{DB}^2, \mathbf{DB}^3] = [1, 2, 1] \in \mathbb{Z}_{15}^3$, and DB^1 is the desired file, i.e. $d = 1$.

- **Setup**. Let C_1 be a cyclic code in $\mathbb{Z}_3[x]/\langle x^{13}-1\rangle$ and C_2 be a cyclic code in $\mathbb{Z}_5[x]/\langle x^{13}-1\rangle$ with generator polynomial $g_1(x) = x^7 + x^5 + x^4 + 2x^3 + 2x^2 + 2$ and $g_2(x) = x^9 + 2x^8 + 4x^7 + 3x^5 + 2x^4 + x^2 + 3x + 4$, respectively. One can check that $C_{\mathtt{IN}} = CRT(C_1, C_2)$ is a cyclic code in $\mathcal{R} = \mathbb{Z}_{15}[x]/\langle x^{13} - 1\rangle$ with generator polynomial $g_{\mathtt{IN}} = 6x^9 + 12x^8 + 4x^7 + 13x^5 + 7x^4 + 5x^3 + 2x^2 + 3x + 14$.

  Let $\tilde{C}_1$ be a cyclic code in $\mathcal{R}$ with generator polynomial $\tilde{g_1} = 10x^7+5x^5+6x^4+x^3+14x^2+x+11$ and let $\tilde{C}_2$ be a cyclic code with generator polynomial $\tilde{g_2} = 10x^7 + 11x^5 + 13x^3 + 2x^2 + 10x + 14$. We have that $\tilde{C}_2 \subseteq \tilde{C}_1$. Finally, we select the 2-generator quasi-cyclic code $C_{\mathtt{OUT}} = [\tilde{C}_1, \tilde{C}_2]\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ with generator matrix over $\mathcal{R}$ given by

$$G_{\mathtt{OUT}} = \begin{pmatrix} 10x^7 + 5x^5 + 6x^4 + x^3 + 14x^2 + x + 11, & 10x^7 + 5x^5 + 6x^4 + x^3 + 14x^2 + x + 11 \\ 0 & 10x^7 + 11x^5 + 13x^3 + 2x^2 + 10x + 14 \end{pmatrix}.$$

- **Query generation**. Now, consider random polynomials that provide the user the entries of $\mathbf{A}$ in $\mathcal{R} = \mathbb{Z}_{15}[x]/\langle x^{13} - 1\rangle$, the elements $e_{1j}^i(x) \in C_{\mathtt{IN}}$ for $i \in \{1, 2, 3\}$, $j \in \{1, 2\}$, and $u_{11}^1(x)$ in $\tilde{C}_2 \cap (C_{\mathtt{IN}}^{\perp} \setminus C_{\mathtt{IN}})$ given by

$$
\begin{array}{ll}
a_{11}^1(x) = & 5x^{12} + 11x^{11} + 10x^{10} + 13x^9 + 7x^8 + x^7 + 3x^6 + 14x^5 + 14x^4 + 7x^2 + \\
& 6x + 4 \\
a_{12}^1(x) = & 8x^{12} + 8x^{11} + 12x^{10} + 13x^9 + 11x^8 + 11x^7 + 6x^6 + x^5 + 2x^4 + 6x^3 + \\
& 5x^2 + 9x + 3 \\
a_{11}^2(x) = & 5x^{12} + 13x^{11} + 4x^9 + 2x^8 + 14x^7 + 12x^6 + 10x^5 + 6x^3 + 5x^2 + 12x \\
a_{12}^2(x) = & 10x^{12} + 4x^{11} + 4x^9 + 14x^8 + 8x^7 + 6x^5 + 7x^4 + 12x^3 + 11x^2 + 6x + 9 \\
a_{11}^3(x) = & 5x^{12} + 3x^{11} + 4x^{10} + 10x^9 + 4x^8 + 9x^7 + 14x^6 + 12x^5 + 8x^4 + 9x^3 + \\
& 6x^2 + 6x \\
a_{12}^3(x) = & 9x^{12} + 14x^{11} + x^{10} + 4x^9 + x^8 + 13x^7 + 8x^6 + 3x^5 + 13x^4 + 11x^3 + \\
& 2x^2 + 14x + 10 \\
\hline
e_{11}^1(x) = & 11x^{12} + 9x^{11} + 14x^{10} + x^9 + 14x^8 + 12x^7 + 12x^6 + 6x^5 + 12x^4 + 11x^3 + \\
& 7x^2 + 2x + 9 \\
e_{12}^1(x) = & 6x^{12} + 14x^{11} + 10x^{10} + 13x^9 + 3x^8 + 7x^7 + 5x^6 + 3x^5 + 2x^4 + 2x^3 + \\
& 10x^2 + 6x + 9 \\
e_{11}^2(x) = & 7x^{12} + 6x^{11} + 9x^{10} + 4x^9 + 13x^8 + 3x^7 + 10x^6 + 2x^5 + 2x^4 + 11x^3 + \\
& x^2 + 9x + 13 \\
e_{12}^2(x) = & 5x^{12} + 3x^{11} + 13x^{10} + 4x^9 + 9x^8 + 6x^7 + 12x^6 + 13x^5 + 14x^4 + 6x^3 + \\
& 6x^2 + 12x + 2 \\
e_{11}^3(x) = & 6x^{12} + 14x^{11} + 4x^{10} + 2x^9 + x^8 + x^7 + 7x^6 + 14x^5 + 14x^4 + 11x^3 + \\
& 13x^2 + 5x + 13 \\
e_{12}^3(x) = & 10x^{12} + 5x^{11} + 6x^{10} + 3x^9 + 11x^8 + 9x^7 + 13x^6 + 12x^4 + x^3 + 14x^2 + \\
& 7x + 14 \\
\hline
u_{11}^1(x) = & 13x^{12} + 14x^{11} + x^{10} + 3x^9 + 2x^8 + 2x^7 + 7x^6 + 7x^5 + 13x^4 + 6x^3 + \\
& 4x^2 + 3x
\end{array}
$$

Then the user can compute $\mathbf{\Delta}$ given by

$$
\begin{array}{ll}
\delta_{11}^1(x) = & 10x^{12} + 9x^{11} + 10x^{10} + 5x^9 + 9x^8 + 13x^7 + 9x^6 + 9x^5 + 12x^4 + 12x^3 + \\
& 11x^2 + 6x + 5 \\
\delta_{12}^1(x) = & 13x^{12} + 7x^{11} + 14x^{10} + 13x^9 + 6x^8 + 3x^7 + 5x^6 + 13x^5 + 7x^4 + x^3 + \\
& 12x^2 + 11x \\
\delta_{11}^2(x) = & 6x^{12} + 6x^{11} + 12x^{10} + 8x^9 + 2x^8 + 10x^7 + 13x^6 + 13x^5 + 10x^4 + 10x^3 + \\
& 6x^2 + 9x + 9 \\
\delta_{12}^2(x) = & 9x^{12} + 4x^{11} + 14x^{10} + 2x^9 + 4x^8 + 4x^7 + 6x^6 + 4x^5 + x^3 + \\
& 3x^2 + 7x + 11 \\
\delta_{11}^3(x) = & 4x^{12} + 9x^{11} + 12x^{10} + 8x^9 + 10x^8 + 14x^7 + 5x^6 + 7x^5 + 11x^4 + 3x^2 + \\
& 8x + 14 \\
\delta_{12}^3(x) = & 6x^{12} + 10x^{11} + 7x^{10} + 8x^9 + 11x^7 + 11x^6 + 13x^5 + 11x^4 + 6x^3 + 5x + 2
\end{array}
$$

and the query matrix $\mathbf{Q}$ is the result of expanding in $\mathbb{Z}_{15}$ the entries in $(\mathbf{A} \parallel \mathbf{\Delta})$

$$\left[\begin{array}{l} 4,6,7,0,14,14,3,1,7,13,10,11,5,3,9,5,6,2,1,6,11,11,13,12,8,8,5,6,11,12,12,9,\\ \quad 9,13,9,5,10,9,10,0,11,12,1,7,13,5,3,6,13,14,7,13\\ 0,12,5,6,0,10,12,14,2,4,0,13,5,9,6,11,12,7,6,0,8,14,4,0,4,10,9,9,6,10,10,13,\\ \quad 13,10,2,8,12,6,6,11,7,3,1,0,4,6,4,4,2,14,4,9\\ 0,6,6,9,8,12,14,9,4,10,4,3,5,10,14,2,11,13,3,8,13,1,4,1,14,9,14,8,3,0,11,7,\\ \quad 5,14,10,8,12,9,4,2,5,0,6,11,13,11,11,0,8,7,10,6 \end{array}\right].$$

- **Server response**. The server computes $\mathbf{R} = \mathbf{DB} \cdot \mathbf{Q}$, which is equal to

$$\left[\begin{array}{c} 4,6,8,6,7,1,11,8,0,1,14,10,5,1,5,14,11,14,1,14,10,10,10,13,0,7,7,\\ 2,11,2,13,12,10,2,8,14,1,0,11,9,0,3,9,3,4,13,7,14,10,4,10,7 \end{array}\right].$$

- **Recovering stage**. Once received $\mathbf{R}$, the user can extract the submatrices $\mathbf{R_1} = (r^1_{1,1}, r^1_{1,2})$ and $\mathbf{R_2} = (r^2_{1,1}, r^2_{1,2})$ with elements in $\mathcal{R} = \mathbb{Z}_{15}[x]/\langle x^{13}-1\rangle$ given by the following elements

$\mathbf{R_1}$	$r^1_{1,1} = 5x^{12} + 10x^{11} + 14x^{10} + x^9 + 8x^7 + 11x^6 + x^5 + 7x^4 + 6x^3 + 8x^2 + 6x + 4$
	$r^1_{1,2} = 7x^{12} + 13x^{10} + 10x^9 + 10x^8 + 10x^7 + 14x^6 + x^5 + 14x^4 + 11x^3 + 14x^2 + 5x + 1$
$\mathbf{R_2}$	$r^2_{1,1} = 11x^{12} + x^{10} + 14x^9 + 8x^8 + 2x^7 + 10x^6 + 12x^5 + 13x^4 + 2x^3 + 11x^2 + 2x + 7$
	$r^2_{1,2} = 7x^{12} + 10x^{11} + 4x^{10} + 10x^9 + 14x^8 + 7x^7 + 13x^6 + 4x^5 + 3x^4 + 9x^3 + 3x^2 + 9$

and compute $\mathbf{R_2} - (\mathbf{R_1} \cdot G_{\texttt{OUT}})$, which is a 1×2 matrix whose elements are

$14x^{12} + 4x^{11} + 7x^{10} + 14x^9 + 13x^8 + 6x^7 + x^6 + x^5 + 13x^4 + 5x^3 + 11x^2 + 13x + 3$
$11x^{12} + 10x^{11} + 12x^{10} + 9x^9 + 2x^8 + 13x^7 + 12x^6 + 14x^5 + 12x^4 + 6x^2 + 7x + 12$

The user has selected $\delta = 1$ as a column position and the files have one column, hence the user does not need the second position. Thus, the user multiplies each entry of the matrix (written as a vector in $\mathbb{Z}_{15}^{13}$) by $H^\top_{\texttt{IN}}$, the parity check matrix of the code $C_{\texttt{IN}}$, given by

$$H^\top_{\texttt{IN}} = \left[\begin{array}{cccccccccccc} 1&0&0&0&0&0&0&1&0&2&11&2&0\\ 0&1&0&0&0&0&0&0&1&0&2&11&2\\ 0&0&1&0&0&0&0&1&0&0&2&4&2\\ 0&0&0&1&0&0&0&1&1&14&2&4&10\\ 0&0&0&0&1&0&0&2&1&8&0&6&13\\ 0&0&0&0&0&1&0&2&2&5&0&4&6\\ 0&0&0&0&0&0&1&0&2&11&2&0&1\\ 0&0&0&0&0&0&0&3&0&3&9&6&6\\ 0&0&0&0&0&0&0&0&3&9&0&9&3 \end{array}\right]^T.$$

Therefore, the user computes

$$[3,13,11,5,13,1,1,6,13,14,7,4,14] \cdot H^\top_{\texttt{IN}} = [2,7,0,0,11,14,14,6,3]$$

Since the user knows $\mathbf{DB}^1 \cdot \mathbf{U} \cdot H_{\mathtt{IN}}^\top$ and $\mathbf{U} \cdot H_{\mathtt{IN}}^\top$, the user can solve the two systems of equations, originally over $\mathbb{Z}_{15}$, that arise when solving the system of equations $(\mathbf{DB}^1 \cdot \mathbf{U} \cdot H_{\mathtt{IN}}^\top) = x \cdot (\mathbf{U} \cdot H_{\mathtt{IN}}^\top)$ over $\mathbb{Z}_3$ and $\mathbb{Z}_5$, respectively. Then the user lifts both solutions to a single solution in $\mathbb{Z}_{15}$ via the Chinese remainder theorem, obtaining the desired file $x = \mathbf{DB}^1 = 1$.

3.1.2.5 Analysis

The hardness of a brute force attack will rely on the capacity of the server to guess the inner code, thus it depends on the knowledge of n as well, which is not a public parameter, and on the fact that there is a big enough number of possible cyclotomic cosets that define different cyclic codes in $\mathcal{R} = \mathbb{Z}_m[x]/\langle x^n - 1\rangle$. The protocol we propose can resist the rank difference attack known for single-server PIR protocols based on codes whereas our rate information is worse than the one in [37]. To enhance security against the rank difference attack, we somewhat compromise the rate in the PIR protocol. Furthermore, as stated before, all the computations will be made as modular operations (modulo m) and therefore they are less computationally intense than other protocols that require large field extensions. Moreover, the server only has to perform a modular multiplication of matrices.

3.1.2.5.1 PIR Rate

We recall that the database is an $L \times tr$ matrix whose elements are in $\mathbb{Z}_{m'}$. The entries of the query matrix are elements in the polynomial ring $\mathcal{R}$, but they are sent to the server as elements in R since $\mathcal{R}$ is an algebra over R, therefore the size of the matrix $\mathbf{Q}$ is $tr \times 2ns$ and its elements are in $\mathbb{Z}_m$, thus the upload cost is

$$H(\mathbf{Q}) = 2 \cdot t \cdot r \cdot n \cdot s \log(m).$$

On the other hand, the desired file has L rows and r columns. Therefore, the size of the desired file is $Lr \log(m')$. The server computes the matrix product of the database and the query matrix, and hence the download cost is

$$H(\mathbf{R}) = 2 \cdot L \cdot n \cdot s \log(m).$$

The **PIR rate** is the ratio of the desired file size over the sum of the download and upload costs, in the protocol we propose is given by

$$\frac{Lr \log(m')}{2 \cdot t \cdot r \cdot n \cdot s \log(m) + 2 \cdot L \cdot n \cdot s \log(m)} = \frac{L \cdot r}{t \cdot r + L} \left(\frac{\log(m')}{2 \cdot n \cdot s \log(m)} \right). \tag{3.7}$$

As usual, we assume that the size of the files is much larger than the number of files, $L \gg tr$, hence the PIR rate of the protocol is approximately equal to

$$\text{PIR rate} \approx \frac{r}{2 \cdot n \cdot s} \cdot \frac{\log(m')}{\log(m)}. \tag{3.8}$$

Remark 50. A recent proposal [63] presented a modification of single-server PIR scheme in [37]. Although their rate outperforms ours, they execute multiplications in large finite fields. In contrast, our protocol employs simpler computations using modular arithmetic, which satisfies lower computational complexity compared to protocols requiring a large field extension.

m	n	s	r	k	*Rate*	S
36	91	5	4	377	$\frac{1}{455}$	$\geq (2^{28})^6$
36	91	5	5	377	$\frac{1}{364}$	$\geq (2^{28})^6$
36	91	6	6	435	$\frac{1}{364}$	$\geq (2^{28})^7$
36	91	10	10	607	$\frac{1}{364}$	$\geq (2^{28})^{11}$
216	91	5	5	377	$\frac{1}{546}$	$\geq (2^{28})^6$

Table 3.2: Rate and work factor based on parameter selection. The column labeled 'Rate' shows rates corresponding to the respective parameters, while in the 'S' column, the inverse probability is displayed as the work factor.

3.1.2.5.2 Computational cost

The cost is dominated by the multiplication of matrices **DB** and **Q**, both of them with entries in the ring $R = \mathbb{Z}_m$, that is, the product is performed with modular arithmetic. In the HHWZ PIR protocol [37], they perform that multiplication in large finite fields as degree-s polynomials (where s is the degree of the extension of the chosen field over the base field). Thus, their multiplication complexity can be expressed as $\log(q) \cdot s\log(q)$. That is, they consider this multiplication as the product over $\mathbb{F}_{q^{\sqrt{s}}}$ in [37]. If we fix the number of rows L and the number of files, there are δn of such field multiplications. In our protocol, the usual multiplication in $\mathbb{Z}_m$ is performed $2rns$ times. When δ in [37] equals our defined value r, which corresponds to the number of columns in the files, and the code length is the same, we perform $2s$ times more multiplications than [37]. However, the multiplications in our protocol are simpler since we use modular arithmetic and [37] uses field arithmetic.

During the data recovery process, the procedures employed involve solving linear systems of equations and utilizing the Chinese remainder theorem (CRT). Thus, the complexity of the required operations to recover the desired file is the complexity of solving linear equations and performing CRT.

3.1.2.5.3 Security Analysis

The subspace attack in [37, Section V.A] could be also applied as a submodule attack in our case, but it has at least the same complexity as in [37], thus if we chose the same security parameter on the size of the module, such attack is still unfeasible. In [10], the authors showed that the HHWZ PIR protocol is not private, since the server can recover in polynomial time the index of the desired file with high probability. The central concept of the attack is that by eliminating rows from the query matrix that correspond to the desired file, it yields a large decrease in the dimension over the vector space spanned by the rows of this punctured matrix. The dimension loss only shows a low (almost negligible) probability when the rows unrelated to the requested file are deleted. Thus, the attack method relies on comparing the ranks of the query matrix after deleting different rows. Therefore, it is essential to construct the query matrix in such a way that deleting a row does not reveal information about the desired data. We will show that this is the case in our protocol.

Theorem 51. *Let $C_{\mathtt{OUT}}$ be a linear code in $\mathcal{R}^s$, and $C_{\mathtt{IN}} \subseteq \mathcal{R}$ an R-linear code. Let $\mathbf{\Delta}$ be chosen as in Section 3.1.2 and $\mathbf{\Delta} = \mathbf{W} + \mathbf{E} + \mathbf{U}$ the decomposition in Equation (3.2). Then we have*

that

$$\operatorname{rowspan}([\mathbf{\Delta}]) \subseteq C_{\mathtt{OUT}} + \Gamma_s(C_{\mathtt{IN}}), \tag{3.9}$$

where rowspan *is taken in* R *and* $\Gamma_s(C_{\mathtt{IN}}) = [C_{\mathtt{IN}}, \ldots, C_{\mathtt{IN}}]\mathrm{Id}_s$.

Proof. Let $[\mathbf{W}]$, $[\mathbf{E}]$ and $[\mathbf{U}]$ be the matrices over R containing the expansion of each element in $\mathcal{R}$ to a tuple in R of the matrices $\mathbf{W}$, $\mathbf{E}$ and $\mathbf{U}$ in Equation (3.2). Note that the entries of $[\mathbf{E}]$ are random elements in $C_{\mathtt{IN}}$. From our technical conditions, we have that $\tilde{C}_j \cap (C_{\mathtt{IN}}^{\perp} \setminus C_{\mathtt{IN}}) \neq \{\mathbf{0}\}$, for $j \in \{1, \ldots, s\}$, and the entries of $[\mathbf{U}]$ are elements in $\tilde{C}_s \cap (C_{\mathtt{IN}}^{\perp} \setminus C_{\mathtt{IN}})$. Hence, the components of $\mathbf{\Delta} = \mathbf{W} + \mathbf{E} + \mathbf{U}$ fulfill

$$\begin{aligned}
&\operatorname{rowspan}([\mathbf{W}]) \subseteq C_{\mathtt{OUT}},\\
&\operatorname{rowspan}([\mathbf{E}]) \subseteq \Gamma_s(C_{\mathtt{IN}}),\\
&\operatorname{rowspan}([\mathbf{U}]) \subseteq C_{\mathtt{OUT}} \cap \Gamma_s(C_{\mathtt{IN}}^{\perp}).
\end{aligned}$$

Thus we have

$$\operatorname{rowspan}([\mathbf{\Delta}]) = \operatorname{rowspan}([\mathbf{W} + \mathbf{E} + \mathbf{U}]) \subseteq C_{\mathtt{OUT}} + \Gamma_s(C_{\mathtt{IN}}).$$

□

Remark 52 (Choosing the base ring and the projection codes)**.** Let $\mathbf{Q}$ be the query matrix in our protocol, note that it is a matrix with entries in $\mathbb{Z}_m$. If we know the factorization of $m = \Pi_{i=1}^{\ell} p_i^{e_i}$, we can consider the matrices $\mathbf{Q}_{p_i^{e_i}}$, for $i = 1, \ldots, \ell$, given by reducing each entry of $\mathbf{Q}$ modulo $\mathbb{Z}_{p_i^{e_i}}$. For $i = 1, \ldots, \ell$, we can consider $\mathbf{Q}_{p_i^{e_i}}$ as a generator matrix of a code over $\mathbb{Z}_{p_i^{e_i}}$ and $\mathbf{Q}$ can be considered as a generator matrix of a code over $\mathbb{Z}_m$ which is the CRT code of the previous projection components. Suppose that one can apply linear algebra techniques in at least one of the projection codes, that is, there is a projection $i_0 \in \{1, \ldots, \ell\}$ such that $\mathbf{Q}_{p_{i_0}^{e_{i_0}}} = [\mathbf{A}_{p_{i_0}^{e_{i_0}}} \mid \mathbf{\Delta}_{p_{i_0}^{e_{i_0}}}]$ generates a free code over $\mathbb{Z}_{p_{i_0}^{e_{i_0}}}$. In this case, the attack in [10] can be applied successfully. That is, in that case, the positions given by $\mathbf{A}_{p_{i_0}^{e_{i_0}}}$ can be seen as a subset of the information set of the projection code and, provided that the attacker has enough rows, the attack in [10] may show a rank difference when one removes the rows with non-zero components in $\mathbf{U}$. Note that this is the case when $e_{i_0} = 1$, since $\mathbb{Z}_{p_{i_0}^{e_{i_0}}}$ is a field and then the projection code is free.
If $e_i > 1$, for each $i = 1, \ldots, \ell$, then $\mathbb{Z}_{p_i^{e_i}}$ is a chain ring with maximal ideal $\langle p_i \rangle$ and nilpotence index e_i. Furthermore, a cyclic code over a chain ring is free if and only if it is a Hensel lift, i.e. it is generated by a monic polynomial in $\mathbb{Z}_{p_i^{e_i}}[x]/\langle x^n - 1 \rangle$ (see [51, Proposition 4.11]). Therefore, we must consider non-free projection codes for our protocol that we will call non-Hensel lifts. They are provided by choosing the entries of $\mathbf{A}_{p_i^{e_i}}$ in $p_i \mathbb{Z}_{p_i^{e_i}}$, for each $i = 1, \ldots, \ell$, and cyclic codes whose generator set in standard form (see [51] for a definition) involves at least one monic polynomial multiplied by a non-zero power of p. Thus, we are ensured that the cyclic part is non-free by [51, Theorem 4.5].

Corollary 53. *There is no R-rank difference in the sub-matrix of $\mathbf{Q}$ that we obtain when removing a row of the original matrix if the projection codes are non-Hensel lifts. Therefore, the attack strategy relying on rank difference in [10] does not apply to our proposed protocol.*

Proof. The result follows from Theorem 51 and Remark 52. □

As pointed out in the introduction, single-server PIR protocols cannot be information-theoretical secure, thus there is always some information leakage that an attacker could use to infer which the index of the file the user wants to retrieve. In our protocol, the inner code, $C_{\mathtt{IN}}$, and the codes $\tilde{C}_i$, for $i = 1, \ldots, s$, are kept as secret information. The query matrix $\mathbf{Q}$ is public information, meaning the attacker knows $[\mathbf{\Delta}]$ the $\mathbb{Z}_m$-expansion of $\mathbf{\Delta}$. Therefore, the attacker may obtain some information on the constituent cyclic codes by considering their possible defining f-cyclotomic cosets where f is a factor of the length of $[\mathbf{\Delta}]$. However, it is still difficult for an attacker to find the inner and outer codes using brute force attacks. Indeed, since they are non-free codes, their generator matrix has linearly dependent vectors. Hence, there is no information set for these codes. Thus, straight linear algebra operations can not be applied to find these codes.

Remark 54. The cyclotomic coset containing $\theta \in \mathbb{Z}_n$, denoted by U_θ, is the set $\{\theta, \theta q, \ldots, \theta q^i\} \pmod n$ where i is the smallest integer such that $q^i \equiv 1 \pmod n$. It is well known that the number of irreducible factors of $x^n - 1$ over $\mathbb{F}_q$ is equal to the number of cyclotomic cosets of q modulo n and each of those factors is called a cyclotomic polynomial. They are $\Phi_n(x) = \prod_{d|n}(x^d - 1)^{\mu(\frac{n}{d})}$, where where μ is the Möbius function.

Let $(q, n) = 1$ and let $\varphi(n)$ be the Euler phi function. One has that $\Phi_n(x)$ can be factorized into $\varphi(n)/m$ distinct monic irreducible polynomials of degree m over $\mathbb{F}_q$, where m is the least positive integer such that $q^m \equiv 1 \pmod n$ and m is called the order of q modulo n and it is denoted as $\mathrm{ord}_d(q)$. Therefore, the number of cyclotomic cosets is $T = \sum_{d|n} \dfrac{\varphi(d)}{\mathrm{ord}_d(q)}$.

When generating the protocol, we can assume that the user selects $x^n - 1$ with a number of irreducible factors big enough. In this way, protection against brute force attacks can be provided.

From Theorem 51, one has that $\mathrm{rowspan}([\mathbf{\Delta}]) \subseteq C_{\mathtt{OUT}} + \Gamma_s(C_{\mathtt{IN}})$. Even though $C_{\mathtt{OUT}} + \Gamma_s(C_{\mathtt{IN}})$ is known, it is hard to get $C_{\mathtt{OUT}}$ and $C_{\mathtt{IN}}$. First, the attacker must use the Chinese remainder theorem to decompose the code $C_{\mathtt{OUT}} + \Gamma_s(C_{\mathtt{IN}})$ into a direct sum of ideals over $\mathbb{Z}_{p_i^{e_i}}[x]/\langle x^n - 1\rangle$. Then the attacker must compute the number of cyclotomic cosets in order to find the number of cyclic codes for each decomposition. For instance, if T is the number of cyclotomic cosets, then there are 2^T divisors of $x^n - 1$, equal to the number of cyclic codes of length n. As we mentioned, the attacker could get some cyclotomic polynomials from $\mathrm{rowspan}([\mathbf{\Delta}])$. Suppose we define $\tilde{T}$ as the number of cyclotomic polynomials that an attacker cannot obtain without a brute force attack and that there exists at least $2^{\tilde{T}}$ cyclic codes of length n. In that case, the probability of recovering the code is given by $Pr \leq \dfrac{1}{2^{\tilde{T}}}$. Note that we are bounding the total number of cyclic codes by the number of cyclotomic classes since the generating set in standard form of a cyclic code of length n over $\mathbb{Z}_{p^s}$ is of the form $\{f_0, pf_1, \ldots, p^{s-1}f_{s-1}\}$, where $f_{s-1}| \ldots |f_1|f_0|x^n - 1$ and they are monic polynomials (see [51, Definition 4.1]). In other words, the probability of finding the code is inversely proportional to the number of cyclotomic polynomials the attacker cannot obtain. Therefore, that probability exponentially decreases as $\tilde{T}$ increases. For example, considering the data presented in Table 3.2, wherein the inner code length $n = 91$ over the ring $\mathbb{Z}_4$, the number of cyclic codes is at least 2^{10}, and over $\mathbb{Z}_9$, the number of cyclic codes is at least 2^{18}. For a specific choice of component $s = 10$, the probability that an attacker could

successfully guess the outer codes is determined by $\Pr \leq (2^{-28})^{10}$ and the probability of guessing the inner codes is determined by $\Pr \leq 2^{-28}$. Therefore, the probability of guessing all codes is $\Pr \leq (2^{-28})^{11}$. This probability is notably low, underscoring the resilience of the system against brute force attacks.

After presenting a toy example to illustrate the protocol (see Example 49), we now turn to a more general and realistic construction.

Example 55 (Real example)**.** For simplicity we will take $m' = 3 \cdot 5$, thus $R = \mathbb{Z}_{15}$. Note that the value m does not fulfill the technical conditions for security and that we are considering that $m' = m$, which is not allowed in our protocol. The aim is to illustrate the protocol with a toy example and simple computations. Consider the 3-file in database $\mathbf{DB} = [\mathbf{DB}^1, \mathbf{DB}^2, \mathbf{DB}^3] = [1, 2, 1] \in \mathbb{Z}_{15}^3$, and DB^1 is the desired file, i.e. $d = 1$.

- **Set up**. Let C_1 be a cyclic code in $\mathbb{Z}_9[x]/\langle x^{13}-1\rangle$ and C_2 be a cyclic code in $\mathbb{Z}_{25}[x]/\langle x^{13}-1\rangle$ with generator polynomial $g_1(x) = x^7 + 3x^6 + 7x^5 + 4x^4 + 5x^3 + 8x^2 + 6x + 2$ and $g_2(x) = x^9 + 12x^8 + 14x^7 + 20x^6 + 18x^5 + 12x^4 + 20x^3 + 21x^2 + 13x + 19$, respectively. One can check that $C_{\mathtt{IN}} = CRT(C_1, C_2)$ is a cyclic code in $\mathcal{R} = \mathbb{Z}_{225}[x]/\langle x^{13}-1\rangle$ with generator polynomial $g_{\mathtt{IN}} = 126x^9 + 162x^8 + 64x^7 + 120x^6 + 43x^5 + 112x^4 + 95x^3 + 71x^2 + 213x + 119$. Let $\tilde{C}_1$ be a cyclic code in $\mathcal{R}$ with generator polynomial $\tilde{g_1} = 126x^{12} + 126x^{11} + 126x^{10} + x^9 + 206x^8 + 16x^7 + 86x^6 + 36x^5 + x^4 + 136x^3 + 36x^2 + 141x + 56$ and let $\tilde{C}_2$ be a cyclic code with generator polynomial $\tilde{g_2} = x^{12} + x^{11} + x^{10} + 76x^9 + 106x^8 + 166x^7 + 61x^6 + 136x^5 + 76x^4 + 211x^3 + 136x^2 + 91x + 106$. We have that $\tilde{C}_2 \subseteq \tilde{C}_1$. Finally, we select the 2-generator quasi-cyclic code $C_{\mathtt{OUT}} = [\tilde{C}_1, \tilde{C}_2] \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ with generator matrix over $\mathcal{R}$ given by

$$
\begin{aligned}
G_{\mathtt{OUT}_{11}} = {} & 126x^{12} + 126x^{11} + 126x^{10} + x^9 + 206x^8 + 16x^7 + 86x^6 + 36x^5 + x^4 + 136x^3 \\
& + 36x^2 + 141x + 56 \\
G_{\mathtt{OUT}_{12}} = {} & 126x^{12} + 126x^{11} + 126x^{10} + x^9 + 206x^8 + 16x^7 + 86x^6 + 36x^5 + x^4 + 136x^3 \\
& + 36x^2 + 141x + 56 \\
G_{\mathtt{OUT}_{21}} = {} & 0 \\
G_{\mathtt{OUT}_{22}} = {} & x^{12} + x^{11} + x^{10} + 76x^9 + 106x^8 + 166x^7 + 61x^6 + 136x^5 + 76x^4 + 211x^3 \\
& + 136x^2 + 91x + 106
\end{aligned}
$$

- **Query generation**. Now, consider random polynomials that provide the user the entries of $\mathbf{A}$ in $m'\mathcal{R}$, the elements $e_{kj}^1(x) \in \mathrm{nf}(C_{\mathtt{IN}})$ for $k, j \in \{1, 2\}$, and $u_{11}^1(x)$ in $\mathrm{nf}(\tilde{C}_2 \cap (C_{\mathtt{IN}}^{\perp} \setminus C_{\mathtt{IN}}))$ given by

$$
\begin{aligned}
a_{11}^1(x) = \;& 30x^{12} + 75x^{11} + 30x^{10} + 135x^9 + 165x^8 + 60x^7 + 90x^6 + 105x^5 + 135x^4 \\
& + 150x^3 + 90x^2 + 120x + 15 \\
a_{12}^1(x) = \;& 165x^{12} + 75x^{10} + 90x^9 + 150x^8 + 165x^7 + 105x^6 + 165x^5 + 150x^4 \\
& + 105x^3 + 180x^2 + 195 \\
a_{21}^1(x) = \;& 105x^{12} + 75x^{11} + 75x^{10} + 105x^9 + 150x^8 + 15x^7 + 60x^6 + 15x^5 + 30x^4 \\
& + 105x^3 + 15x^2 + 15x \\
a_{22}^1(x) = \;& 180x^{12} + 165x^{11} + 90x^{10} + 90x^9 + 120x^8 + 30x^7 + 150x^6 + 165x^5 + 165x^4 \\
& + 105x^3 + 30x + 195 \\
a_{31}^1(x) = \;& 120x^{12} + 195x^{10} + 210x^9 + 105x^8 + 45x^7 + 150x^6 + 45x^5 + 120x^3 \\
& + 210x^2 + 30x + 75 \\
a_{32}^1(x) = \;& 135x^{12} + 150x^{11} + 180x^{10} + 120x^9 + 165x^7 + 120x^6 + 165x^5 + 120x^4 \\
& + 210x^3 + 210x^2 + 120x
\end{aligned}
$$

$$
\begin{aligned}
e_{11}^1(x) = \;& 210x^{12} + 60x^{11} + 210x^{10} + 60x^9 + 195x^8 + 210x^7 + 15x^6 + 165x^5 + 120x^3 \\
& + 90x^2 + 75x + 165 \\
e_{12}^1(x) = \;& 15x^{12} + 165x^{11} + 210x^{10} + 180x^9 + 165x^7 + 210x^6 + 45x^5 + 135x^4 + 195x^3 \\
& + 75x^2 + 135x + 45 \\
e_{21}^1(x) = \;& 210x^{12} + 195x^{11} + 90x^{10} + 195x^9 + 60x^7 + 150x^6 + 75x^5 + 45x^4 + 150x^3 \\
& + 90x^2 + 90 \\
e_{22}^1(x) = \;& 60x^{12} + 30x^{11} + 180x^{10} + 165x^9 + 75x^8 + 15x^7 + 165x^5 + 15x^4 + 150x^2 + 45 \\
e_{31}^1(x) = \;& 195x^{12} + 45x^{11} + 210x^{10} + 30x^9 + 75x^8 + 60x^7 + 105x^6 + 90x^5 + 30x^4 + 210x^3 \\
& + 135x^2 + 15x + 150 \\
e_{32}^1(x) = \;& 165x^{11} + 90x^{10} + 60x^9 + 210x^8 + 120x^7 + 150x^6 + 60x^5 + 90x^4 + 135x^3 + 195x^2 \\
& + 120x + 180
\end{aligned}
$$

$$
\begin{aligned}
u_{11}^1(x) = \;& 45x^{12} + 45x^{11} + 15x^{10} + 15x^9 + 195x^8 + 30x^7 + 210x^6 + 210x^5 + 165x^4 + 120x^2 \\
& + 120x + 150
\end{aligned}
$$

Then the user can compute $\mathbf{\Delta}$ given by

$$
\begin{aligned}
\delta_{11}^1(x) = \;& 105x^{12} + 180x^{11} + 75x^{10} + 150x^9 + 15x^8 + 90x^7 + 75x^6 + 15x^4 + 195x^3 + 60x^2 \\
& + 45x + 165 \\
\delta_{12}^1(x) = \;& 60x^{12} + 210x^{11} + 30x^{10} + 45x^8 + 210x^7 + 30x^6 + 90x^5 + 180x^4 + 15x^3 + 120x^2 \\
& + 180x + 90 \\
\delta_{21}^1(x) = \;& 75x^{12} + 210x^{11} + 30x^{10} + 60x^9 + 90x^8 + 90x^6 + 90x^5 + 210x^4 + 90x^3 + 30x^2 \\
& + 90x + 105 \\
\delta_{22}^1(x) = \;& 60x^{12} + 180x^{11} + 30x^{10} + 165x^9 + 75x^8 + 90x^7 + 75x^6 + 90x^5 + 90x^4 + 75x^3 \\
& + 195 \\
\delta_{31}^1(x) = \;& 150x^{11} + 15x^{10} + 60x^9 + 105x^8 + 15x^7 + 210x^6 + 45x^5 + 135x^4 + 15x^3 + 90x^2 \\
& + 195x + 180 \\
\delta_{32}^1(x) = \;& 150x^{12} + 165x^{11} + 15x^{10} + 210x^9 + 135x^8 + 195x^7 + 150x^6 + 135x^5 + 90x^4 + 60x^3 \\
& + 45x^2 + 195x + 105
\end{aligned}
$$

and the query matrix $\mathbf{Q}$ is the result of expanding in $\mathbb{Z}_{15}$ the entries in $(\mathbf{A} \parallel \mathbf{\Delta})$

$$\left[\begin{array}{l} 15, 120, 90, 150, 135, 105, 90, 60, 165, 135, 30, 75, 30, 195, 0, 180, 105, 150, 165, 105, 165, 150, \\ \quad 90, 75, 0, 165, 165, 45, 60, 195, 15, 0, 75, 90, 15, 150, 75, 180, 105, 90, 180, 120, 15, 180, 90, \\ \quad\quad 30, 210, 45, 0, 30, 210, 60 \\ 0, 15, 15, 105, 30, 15, 60, 15, 150, 105, 75, 75, 105, 195, 30, 0, 105, 165, 165, 150, 30, 120, 90, 90, \\ \quad 165, 180, 105, 90, 3090, 210, 90, 90, 0, 90, 60, 30, 210, 75, 195, 0, 0, 75, 90, 90, 75, \\ \quad\quad 90, 75, 165, 30, 180, 60 \\ 75, 30, 210, 120, 0, 45, 150, 45, 105, 210, 195, 0, 120, 0, 120, 210, 210, 120, 165, 120, 165, 0, 120, \\ \quad 180, 150, 135, 180, 195, 90, 15, 135, 45, 210, 15, 105, 60, 15, 150, 0, 105, 195, 45, 60, 90, \\ \quad\quad 135, 150, 195, 135, 210, 15, 165, 150 \end{array}\right].$$

- **Server response**. The server computes $\mathbf{R} = \mathbf{DB} \cdot \mathbf{Q}$, which is equal to

$$\left[\begin{array}{l} 90, 180, 105, 30, 195, 180, 135, 135, 120, 105, 150, 0, 135, 135, 180, 165, 75, 150, 210, 75, 165, \\ \quad 165, 165, 210, 30, 210, 105, 195, 210, 165, 120, 0, 15, 105, 75, 105, 150, 75, 30, 135, 150, 165, \\ \quad\quad 0, 0, 180, 105, 135, 105, 90, 105, 60, 105 \end{array}\right].$$

- **Recovering stage**. Once received $\mathbf{R}$, the user can extract the submatrices $\mathbf{R_1} = (r^1_{1,1}, r^1_{1,2})$ and $\mathbf{R_2} = (r^2_{1,1}, r^2_{1,2})$ with elements in $\mathcal{R} = \mathbb{Z}_{225}[x]/\langle x^{13}-1\rangle$ given by the following elements

$\mathbf{R_1}$	$r^1_{1,1} = 135x^{12} + 150x^{10} + 105x^9 + 120x^8 + 135x^7 + 135x^6 + 180x^5 + 195x^4 + 30x^3$ $+105x^2 + 180x + 90$ $r^1_{1,2} = 210x^{12} + 30x^{11} + 210x^{10} + 165x^9 + 165x^8 + 165x^7 + 75x^6 + 210x^5 + 150x^4$ $+75x^3 + 165x^2 + 180x + 135$
$\mathbf{R_2}$	$r^2_{1,1} = 30x^{12} + 75x^{11} + 150x^{10} + 105x^9 + 75x^8 + 105x^7 + 15x^6 + 120x^4 + 165x^3$ $+210x^2 + 195x + 105$ $r^2_{1,2} = 105x^{12} + 60x^{11} + 105x^{10} + 90x^9 + 105x^8 + 135x^7 + 105x^6 + 180x^5 + 165x^2$ $+150x + 135$

and compute $\mathbf{R_2} - (\mathbf{R_1} \cdot G_{\mathtt{OUT}})$, which is a 1×2 matrix whose elements are

$195x^{12} + 90x^{11} + 165x^{10} + 45x^9 + 15x^8 + 195x^7 + 180x^6 + 165x^5 + 60x^4 + 180x^3 + 75x^2$ $+210x + 195,$
$135x^{12} + 165x^{11} + 210x^{10} + 120x^9 + 135x^8 + 90x^7 + 135x^6 + 210x^5 + 30x^4 + 105x^3 + 120x^2$ $+30x + 90$

The user has selected $\delta = 1$ as a column position and the files have one column, hence the user does not need the second position. Thus, the user multiplies each entry of the matrix

(written as a vector in $\mathbb{Z}_{225}^{13}$) by $H_{\mathtt{IN}}^{\top}$, the parity check matrix of the code $C_{\mathtt{IN}}$, given by

$$H_{\mathtt{IN}}^{\top} = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 & 11 & 4 & 18 & 26 & 15 & 202 & 37 & 57 \\ 0 & 1 & 0 & 0 & 0 & 5 & 11 & 25 & 42 & 98 & 9 & 73 & 82 \\ 0 & 0 & 1 & 0 & 1 & 13 & 7 & 20 & 8 & 192 & 169 & 5 & 124 \\ 0 & 0 & 0 & 1 & 1 & 2 & 0 & 22 & 12 & 155 & 167 & 141 & 41 \\ 0 & 0 & 0 & 0 & 3 & 0 & 6 & 24 & 42 & 54 & 66 & 45 & 153 \\ 0 & 0 & 0 & 0 & 0 & 15 & 0 & 30 & 30 & 75 & 0 & 60 & 90 \\ 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 30 & 165 & 30 & 0 & 15 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45 & 0 & 45 & 135 & 90 & 90 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45 & 135 & 0 & 135 & 45 \end{bmatrix}^{T}.$$

Therefore, the user computes

$$[195, 210, 75, 180, 60, 165, 180, 195, 15, 45, 165, 90, 195] \cdot H_{\mathtt{IN}}^{\top} = [75, 75, 90, 180, 0, 0, 0, 0, 0]$$

Since the user knows $\mathbf{DB}^1 \cdot \mathbf{U} \cdot H_{\mathtt{IN}}^{\top}$ and $\mathbf{U} \cdot H_{\mathtt{IN}}^{\top}$, the user can solve the two systems of equations, originally over $\mathbb{Z}_{225}$, that arise when solving the system of equations $(\mathbf{DB}^1 \cdot \mathbf{U} \cdot H_{\mathtt{IN}}^{\top}) = x \cdot (\mathbf{U} \cdot H_{\mathtt{IN}}^{\top})$ over $\mathbb{Z}_3$ and $\mathbb{Z}_5$, respectively. Then the user lifts both solutions to a single solution in $\mathbb{Z}_{15}$ via the Chinese remainder theorem, obtaining the desired file $x = \mathbf{DB}^1 = 1$.

Algorithm 1 Setup

Server Setup:

for $i \in \{1, \ldots, t\}$, $j \in \{1, \ldots, L\}$, $\ell \in \{1, \ldots, r\}$ **do**

$\mathbf{DB} = [\mathbf{DB^1} \parallel \mathbf{DB^2} \parallel \cdots \parallel \mathbf{DB^t}]$, where $\mathbf{DB^i} = [D^i_{j\ell}] \in \mathbb{Z}_{m'}$

end for

User Setup:

for $i \in \{1, \ldots, s\}$ **do**

Select non-Hensel lift codes $\tilde{C}_i \subseteq \mathcal{R}$ with $\tilde{C}_i \cap C_{\mathtt{IN}} \neq \{0\}$, and $\tilde{C}_i \cap (C_{\mathtt{IN}}^{\perp} \setminus C_{\mathtt{IN}}) \neq \{0\}$

end for

The user generates an outer code $[C_{\mathtt{OUT}}] = [\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_s]M$, where M is an $s \times s$ matrix over $\mathbb{Z}'_m$

Algorithm 2 Query Generation

Query Generation:
for $i \in \{1, \ldots, t\}$, $k \in \{1, \ldots, r\}$, $j \in \{1, \ldots, s\}$ **do**
Select randomly $r \times s$ matrices $\mathbf{a}^i$, where $a^i_{kj} \in m'\mathcal{R}$
Encode $\mathbf{a}^i$ as $\mathbf{w}^i = \mathbf{a}^i \cdot G_{\mathtt{OUT}}$, where $G_{\mathtt{OUT}}$ is the generator matrix of $C_{\mathtt{OUT}}$
Select randomly $r \times s$ matrices $\mathbf{e}^i$, where $e^i_{kj} \in \mathrm{nf}(C_{\mathtt{IN}})$
Select column position $\gamma \in \{1, .., s - r + 1\}$
for $\lambda \in \{0, .., r - 1\}$ **do**
Select $r \times s$ matrices $\mathbf{u}^i$ with all entries zero except

$$\mathrm{u}^{\mathrm{d}}_{1+\lambda,\gamma+\lambda} \in \mathrm{nf}(\tilde{C}_s \cap (C^{\perp}_{\mathtt{IN}} \setminus C_{\mathtt{IN}}))$$

end for
Create $r \times s$ matrices $\boldsymbol{\delta}^i = (\mathbf{w^i} + \mathbf{e^i} + \mathbf{u^i})$
Create $\boldsymbol{\Delta} = [\boldsymbol{\delta}^1 \parallel \boldsymbol{\delta}^1 \parallel \cdots \parallel \boldsymbol{\delta}^t]^\top$ and $\mathbf{A} = [\mathbf{a}^1 \parallel \mathbf{a}^1 \parallel \cdots \parallel \mathbf{a}^t]^\top$
Generate query matrix $\mathbf{Q} = [\mathbf{A} \parallel \boldsymbol{\Delta}]$
end for

Algorithm 3 Response and Recovering

Response:
Compute $\mathbf{R} = \mathbf{DB} \cdot \mathbf{Q} = [\mathbf{DBA} \parallel \mathbf{DB\Delta}] = [\mathbf{R_1} \parallel \mathbf{R_2}]$

Recovering Stage:
for $i \in \{1, \ldots, r\}$, **do**
$\mathbf{M} = [\mathbf{R_2} - (\mathbf{R_1} \cdot G_{\mathtt{OUT}})]H^{\top}_{\Gamma_r(C^{\perp}_{\mathrm{IN}})}$, where $H_{\Gamma_r(C^{\perp}_{\mathrm{IN}})}$ is a parity check matrix of $\Gamma_r(C_{\mathrm{IN}}) = [C_{\mathrm{IN}}, \ldots, C_{\mathrm{IN}}]\mathrm{Id}_s$
Solve the linear system of equations

$$\mathbf{M} = \mathrm{diag}[\mathbf{m_1}, \ldots, \mathbf{m_r}], \text{ where } \mathbf{m_i} = [D^d_{1i}, \ldots, D^d_{Li}]^{\top}\mathrm{u}^{\mathrm{d}}_{\mathrm{i},\gamma}H^{\top}_{\mathrm{IN}}$$

end for
return $\mathbf{DB^d}$

Chapter 4

The Schur product of evaluation codes and its applications

The intersection of algebraic coding theory with quantum information science and data privacy has led to the emergence of powerful techniques with broad applicability. One such technique is the Schur product—also known as the componentwise or Hadamard product—of linear codes, which has become increasingly relevant in the construction of quantum error-correcting codes and the design of private information retrieval (PIR) schemes.

This chapter presents a detailed study of the Schur product in the context of evaluation codes, particularly those arising from J-affine variety codes. This general framework encompasses various well-known families of codes, including Reed-Muller, toric, and hyperbolic codes. The main theoretical contribution lies in characterizing the Schur product of these evaluation codes via the Minkowski sum of the underlying sets of monomial exponents. This algebraic-geometric interpretation not only generalizes previous results but also facilitates the derivation of structural properties useful in both quantum and classical applications.

One of the primary applications explored here is the construction of CSS-T quantum codes, a family of Calderbank-Shor-Steane codes tailored to tolerate certain types of noise and beneficial in quantum fault-tolerant protocols. By exploiting the structure of weighted Reed-Muller codes and subfield-subcodes of J-affine variety codes, we can derive new families of quantum codes with parameters that improve upon those in the existing literature, particularly in the binary case. These constructions are grounded in a rigorous analysis of the duals and Schur powers of the evaluation codes involved, ensuring both robustness and efficiency.

Another significant application discussed in this chapter relates to PIR schemes, which are protocols that have already been introduced in previous chapters. Here, we propose new PIR schemes based on hyperbolic codes and subfield-subcodes of J-affine variety codes, demonstrating that these constructions achieve better download rates and stronger privacy guarantees than previously known methods.

Secure Multi-Party computation (MPC) protocols represent another important application of Schur products of linear codes. The central idea is to identify a high-dimensional linear code C such that both its dual code $C^{\perp}$ and its square (i.e., Schur product) code $C^{\star 2}$ have high minimum distance. These parameters are essential for ensuring both privacy and correctness in secure MPC protocols based on coding theory.

In summary, this chapter contributes both theoretical insights and practical tools for the use

of Schur products in the construction of advanced coding schemes by extending classical results to a more general algebraic framework.

4.1 CSS-T codes

We follow the convention of using $[[n, k, d]]$ to denote a quantum code encoding k qubits (known as logical qubits) into n physical qubits and that can correct less than d erasures. We consider the *CSS construction* named after their co-discoverers Robert Calderbank, Peter Shor, and Andrew Steane. [13, 60].

Theorem 56 (CSS construction)**.** *Let $C_1, C_2 \subset \mathbb{F}_2^n$ be linear codes with dimension k_1, k_2, respectively, and such that $C_2 \subset C_1$. Then, there is an $[[n, k_1 - k_2, d]]$ quantum code with*

$$d = \min\left\{\mathrm{wt}\left(C_1 \setminus C_2\right), \mathrm{wt}\left(C_2^{\perp} \setminus C_1^{\perp}\right)\right\},$$

where wt *denotes the minimum Hamming weight.*

CSS-T codes are a class of CSS codes that may implement the T gate transversally, which is a crucial step for achieving fault-tolerant quantum computation. They may reduce the overhead associated with magic state distillation, a common technique used to implement non-Clifford gates in fault-tolerant quantum circuits. CSS-T codes were introduced in [53, 54] and they were algebraically characterized in terms of the Schur product of the pair of binary classical linear codes that define them in [16], note that this definition specifically requires binary codes. More concretely, they are defined from a pair of binary linear codes (C_1, C_2), called a CSS-T pair, such that

$$C_2 \subset C_1 \cap (C_1^{\star 2})^{\perp}.$$

Moreover, we have that for a CSS-T pair (C_1, C_2) then $\min\{\mathrm{wt}(C_1), \mathrm{wt}(C_2^{\perp})\} = \mathrm{wt}(C_2^{\perp})$, and the parameters of the corresponding CSS-T code are ([16, Corollary 2.5])

$$[[n, k_1 - k_2, \geq \mathrm{wt}(C_2^{\perp})]].$$

4.1.1 CSS-T codes from Weighted Reed-Muller codes

CSS-T codes arising from Reed-Muller codes were considered in [2] and from cyclic codes in [16]. It was shown that the parameters of CSS-T codes coming from cyclic codes may outperform those coming from Reed-Muller codes. In this work, we show that we can define CSS-T codes from weighted Reed-Muller codes and that their parameters can improve the parameters in [2, 16]. Specifically, we will consider that C_1 is a weighted Reed-Muller code and C_2 is a Reed-Muller code (a Reed-Muller code being a weighted Reed-Muller code with trivial weights).

Initially, we require a lemma concerning the inclusion or nesting of Weighted Reed-Muller codes within Reed-Muller codes from [59].

Lemma 57. *One has that*

$$\mathrm{RM}(v_{\min}(s), m) \subseteq \mathrm{WRM}(s, m, \mathcal{S}) \subseteq \mathrm{RM}(v_{\max}(s), m),$$

where $v_{\min}(s) = \max\{v \mid s \geq \sum_{i=m-v+1}^{m} s_i\}$, $v_{\max}(s) = \max\{v \mid s \geq \sum_{i=1}^{v} s_i\}$.

We can now state the main result of this section.

Theorem 58. *For $m \geq 2$, let C_1 be binary weighted Reed-Muller $C_1 = \mathrm{WRM}(s, m, \mathcal{S})$ and C_2 be the binary Reed-Muller code $C_2 = \mathrm{RM}(r, m)$, with $r \leq \max\{v \mid s \geq \sum_{i=m-v+1}^{m} s_i\}$. Then (C_1, C_2) is a CSS-T pair if $a + r < m$, where $a = max\{j \mid 2s \geq \sum_{i=1}^{j} s_i\}$.*

The parameters of the associated CSS-T quantum code are $[[2^m, k_1 - k_2, 2^{r+1}]]$, where $k_1 = \dim(C_1)$, and $k_2 = \dim(C_2) = \sum_{i=0}^{s} \binom{m}{i}$,

Proof. From lemma 57, it follows that $C_2 \subset C_1$ because $r \leq \max\{v \mid s \geq \sum_{i=m-v+1}^{m} s_i\}$.

Note that in general, the Schur square of a weighted Reed-Muller code is not a weighted Reed-Muller code. However, we have the inclusion

$$\mathrm{WRM}(s, m, \mathcal{S})^{\star 2} \subseteq \mathrm{WRM}(2s, m, \mathcal{S}).$$

Combining this fact with Lemma 57, we can ensure that

$$\mathrm{WRM}(s, m, \mathcal{S})^{\star 2} \subseteq \mathrm{RM}(a, m). \tag{4.1}$$

The condition $C_2 \subset (C_1^{\star 2})^{\perp}$ translates to the inclusion

$$\mathrm{RM}(r, m) \subseteq (\mathrm{WRM}(s, m, \mathcal{S})^{\star 2})^{\perp},$$

which in turn is equivalent to

$$\mathrm{WRM}(s, m, \mathcal{S})^{\star 2} \subseteq \mathrm{RM}(r, m)^{\perp} = \mathrm{RM}(m - r - 1, m).$$

Furthermore, by combining the previous equation with equation (4.1), we have that $C_2 \subset (C_1^{\star 2})^{\perp}$ if

$$\mathrm{RM}(a, m) \subseteq \mathrm{RM}(m - r - 1, m),$$

which holds if $a \leq m - r - 1$. The parameters follow from [16, Corollary 2.5]. This completes the proof. □

Example 59. Consider the binary weighted Reed-Muller code $C_1 = \mathrm{WRM}(5, 7, (1, 2, 2, 2, 2, 2, 2))$ with the set $\Delta = \{(i_1, i_2, i_3, i_4, i_5, i_6, i_7) : i_1 + 2i_2 + 2i_3 + 2i_4 + 2i_5 + 2i_6 + 2i_7 \leq 5\}$, which has parameters $[128, 44, 16]$, and the binary Reed-Muller code $C_2 = \mathrm{RM}(1, 7)$ with parameters $[128, 8, 64]$. We have that $k((C_1^{\star 2})^{\perp}) = 11$ and $C_2 \subset C_1 \cap (C_1^{\star 2})^{\perp}$, as established by the previous result. Thus, (C_1, C_2) is a CSS-T pair whose associated CSS-T code has parameters $[[128, 36, 4]]$. This construction allows us to generate further examples, listed in Table 4.1. These examples outperform the CSS-T codes presented in [2, 16], as shown in table 4.3.

$\mathbf{C_2}$	$\mathbf{C_1}$	$\mathbf{C_1^{\star 2}}$	$(\mathbf{C_1^{\star 2}})^\perp$	$\mathbf{C_2^\perp}$	CSS-T
$[\mathbf{128, 8, 64}]$ RM$(1, 7)$	$[\mathbf{128, 44, 16}]$ WRM$(5, 7, (1, 2, 2, 2, 2, 2, 2))$	$[\mathbf{128, 117, 4}]$	$[\mathbf{128, 11, 32}]$	$[\mathbf{128, 120, 4}]$	$[[\mathbf{128, 36, 4}]]$
$[\mathbf{256, 37, 64}]$ RM$(2, 8)$	$[\mathbf{256, 58, 32}]$ WRM$(5, 8, (1, 2, 2, 2, 2, 2, 2, 2))$	$[\mathbf{256, 198, 8}]$	$[\mathbf{256, 58}]$	$[\mathbf{256, 219, 8}]$	$[[\mathbf{256, 21, 8}]]$
$[\mathbf{512, 10, 128}]$ RM$(1, 9)$	$[\mathbf{512, 186}]$ WRM$(7, 9, (1, 2, 2, 2, 2, 2, 2, 2, 2))$	$[\mathbf{512, 494}]$	$[\mathbf{512, 18}]$	$[\mathbf{512, 502, 4}]$	$[[\mathbf{512, 176, 4}]]$
$[\mathbf{1024, 56, 128}]$ RM$(2, 10)$	$[\mathbf{1024, 260}]$ WRM$(7, 10, (1, 2, 2, 2, 2, 2, 2, 2, 2, 2))$	$[\mathbf{1024, 932}]$	$[\mathbf{1024, 92}]$	$[\mathbf{1024, 968, 8}]$	$[[\mathbf{1024, 204, 8}]]$

Table 4.1: Parameters of CSS-T codes from Weighted Reed-Muller Codes C_1 and Reed-Muller Codes C_2

4.1.2 CSS-T codes from subfield subcodes of J-affine codes

Moreover, we can extend the previous result to J-affine codes, thereby increasing the range of possible parameters, particularly the length. This extension allows us to explore a broader constellation of codes, offering wider flexibility in code design.

Theorem 60. *Let q be a power of 2 and the sets $\Delta_1, \Delta_2 \subseteq E' \subset E_J$ be a union of complete cyclotomic cosets. The pair of binary codes $(C_{\Delta_1}^{J,\sigma}, C_{\Delta_2}^{J,\sigma})$ is a CSS-T pair if and only if*

1. $\Delta_2 \subseteq \Delta_1$, *and*
2. $\Delta_1 + \Delta_1 \subseteq \Delta_2^\perp$

The parameters of the associated CSS-T quantum code are

$$\left[\left[\prod_{j\notin J} N_j \prod_{j\in J}(N_j - 1), \#\Delta_1 - \#\Delta_2, \geq \mathrm{wt}((C_{\Delta_2}^{J,\sigma})^\perp) = \mathrm{wt}(C_{\Delta_2^\perp}^{J,\sigma})\right]\right].$$

Proof. By Proposition 26, $\left(C_{\Delta_i}^{J,\sigma}\right)^\perp = C_{\Delta_i^\perp}^{J,\sigma}$, for $i = 1, 2$, because $\Delta_1, \Delta_2 \subseteq E' \subset E_J$ and they are a union of complete cyclotomic cosets. Hence, $\Delta_2 \subseteq \Delta_1$ if and only if $C_{\Delta_2}^{J,\sigma} \subseteq C_{\Delta_1}^{J,\sigma}$. Moreover, since $\Delta_1 + \Delta_1 \subseteq \Delta_2^\perp$, then $(C_{\Delta_1}^{J,\sigma})^{\star 2} \subseteq (C_{\Delta_2}^{J,\sigma})^\perp$ which is equivalent to $C_{\Delta_2}^{J,\sigma} \subseteq ((C_{\Delta_1}^{J,\sigma})^{\star 2})^\perp$. The parameters follow from [16, Corollary 2.5] and the result holds. □

The previous result is neat; however, the hypothesis $\Delta_1, \Delta_2 \subseteq E'$ is impractical. The following one allows us to consider $\Delta_1, \Delta_2 \subseteq E_J$, that is, without the restriction $\Delta_1, \Delta_2 \subseteq E'$. Thus it is more flexible. However, in this case, one must be careful when computing the dual codes, as equality no longer holds by Proposition 26, but rather only a containment.

Theorem 61. *Let q be a power of 2 and the sets $\Delta_1, \Delta_2 \subset E_J$ be a union of complete cyclotomic cosets. The pair of binary codes $(C_{\Delta_1}^{J,\sigma}, C_{\Delta_2}^{J,\sigma})$ is a CSS-T pair if*

1. $\Delta_2 \subseteq \Delta_1$, *and*
2. *For all $\mathbf{a} \in \overline{\Delta_1 + \Delta_1 + \Delta_2}$ there exists $j \in \{1, \ldots, m\}$ such that,*
   - $a_j \neq 0$, *if $j \in J$,*

- $a_j \neq N_j - 1$, *if* $j \notin J$.

The parameters of the associated CSS-T quantum code are

$$\left[\left[\prod_{j\notin J} N_j \prod_{j\in J}(N_j-1), \#\Delta_1 - \#\Delta_2, \geq \mathrm{wt}((C_{\Delta_2}^{J,\sigma})^{\perp})\right]\right].$$

Proof. Since $\Delta_2 \subset \Delta_1$, it follows that $C_{\Delta_2}^{J,\sigma} \subset C_{\Delta_1}^{J,\sigma}$.

By Proposition 26, $\left(C_{\Delta_i}^{J,\sigma}\right)^{\perp} \subseteq C_{\Delta_i^{\perp}}^{J,\sigma}$, for $i = 1, 2$, because $\Delta_1, \Delta_2 \subseteq E' \subset E_J$ and they are unions of complete cyclotomic cosets.

Observe that $\mathbf{1} = (1\ldots,1) = \mathrm{ev}_Z(x_1^0 \cdots x_m^0)$, so $\mathbf{b} = \mathbf{0}$ in Proposition 25, we may conclude applying it that $\mathbf{1} = (1\ldots,1) \in \left(C_{\Delta}^{J}\right)^{\perp}$ if and only if for all $\mathbf{a} \in \Delta$ there exists $j \in \{1,\ldots,m\}$ such that ,

- $a_j \neq 0$, if $j \in J$,
- $a_j \neq N_j - 1$, if $j \notin J$. Notice that case $a_j = 0$ is excluded because $p = 2 \mid N_J$.

Furthermore, observe that $C_{\Delta_2}^{J,\sigma} \subseteq ((C_{\Delta_1}^{J,\sigma})^{\star 2})^{\perp}$ if and only if $\mathbf{1} \in (C_{\Delta_1}^{J,\sigma} \star C_{\Delta_1}^{J,\sigma} \star C_{\Delta_2}^{J,\sigma})^{\perp} = (C_{\overline{\Delta_1+\Delta_1+\Delta_2}}^{J,\sigma})^{\perp}$. This is supported by the previous claim and the hypothesis of the result. The parameters follow from [16, Corollary 2.5], and the result holds. □

Based on the previous results, we propose now a concrete construction of subfield subcodes of J-Affine variety codes that yield excellent families of CSS-T quantum codes. First, we will consider a one-variable subfield subcode of a J-affine variety code, then we will extend it to an m-variable J-affine variety code

Let $C^1 = C_{\Delta^1}^{J,\sigma}$ and $C^2 = C_{\Delta^2}^{J,\sigma}$ a pair of one-variable subfield subcodes with $N_1 - 1 \mid 2^r - 1$ and $J = \emptyset$ (we evaluate at zero). Assume that (C^1, C^2) is a CSS-T pair whose associated CSS-T code has parameters $[[N_1, \#\Delta^1 - \#\Delta^2, d]]$.

We will consider now the extension to m variables. Let $(N_i - 1) \mid 2^r - 1$, for $i = 2,\ldots m$, and $1 \leq m_1 \leq m$. Let $J = \{m_1 + 1, 3, \ldots, m\}$, that is, we evaluate at the zeros of $\prod_{j=1}^{m_1}(x_j^{N_j} - x_j)\prod_{j=m_1+1}^{m}(x_j^{N_j-1} - 1)$. Set Δ_1 to be

$$\Delta_1 = \Delta^1 \times \{0,1,\ldots,N_2-1\} \times \cdots \times \{0,1,\ldots,N_{m_1}-1\} \times \{0,1,\ldots,N_{m_1+1}-2\} \times \cdots \times \{0,1,\ldots,N_m-2\}.$$

Let $C \subseteq \mathbb{F}_{q^r}^n$, with $n = \prod_{j\notin J} N_j \prod_{j\in J}(N_j - 1)$, be a hyperbolic code with designed minimum distance d and let Δ_H such that $C_{\Delta_H}^J = C^{\perp}$. Now define Δ_2 to be the $\cup_{\boldsymbol{a}\in\Delta_H} I_{\mathbf{a}}$, i.e., for each element in Δ_H we append to Δ_2 the whole cyclotomic coset $I_{\mathbf{a}}$. Thus, Δ_2 is a union of complete cyclotomic cosets.

Let $C_i = C_{\Delta_i}^{J,\sigma}$, for $i = 1, 2$. Since $\Delta^2 \subset \Delta^1$ then $\Delta_2 \subset \Delta_1$ and therefore $C_2 \subseteq C_1$. Moreover, since by construction $N_1 - 1 \notin \overline{\Delta^1 + \Delta^1 + \Delta^2}$, it follows that there is no $\mathbf{a} \in \overline{\Delta_1 + \Delta_1 + \Delta_2}$ whose first coordinate, a_1, is equal to $N_1 - 1$. Thus, from Theorem 61 we have that (C_1, C_2) is a CSS-T pair whose associated quantum CSS-T codes has parameters given by the next result.

Corollary 62. *Let $q = 2^r$ amd $(N_i - 1) \mid 2^r - 1$, for $i = 2, \dots m$. Let $J = \{m_1 + 1, 3, \dots, m\}$ where $1 \leq m_1 \leq m$. Consider the construction of Δ_1 and Δ_2 designed before, then there exist a CSS-T codes with parameters*

$$\left[\left[\prod_{j=1}^{m_1} N_j \prod_{j=m_1+1}^{m} (N_j - 1), \#\Delta_1 - \#\Delta_2, d\right]\right].$$

We consider now several examples of the construction given in Corollary 62. The notation is as before. All the cyclotomic sets used to define the codes in Examples 63 and 64, and Table 4.3 are given explicitly in Table 4.2.

Example 63. We consider Corollary 62 in the case of three variables with $J = \emptyset$, i.e., $m = m_1 = 3$. Let $N_1 = 16$, $N_2 = 4$, and $N_3 = 2$. Then, the length of the CSS-T code is $n = N_1 \cdot N_2 \cdot N_3 = 128$.

Let $\Delta^1 = \Delta^2 = I_0 \cup I_1 = \{0, 1, 2, 4, 8\}$. Since Δ^2 contains three consecutive integers, the BCH bound implies that

$$d\left((C_{\Delta^2}^{J,\sigma})^{\perp}\right) \geq 4.$$

As described above, we define

$$\Delta_1 = \Delta^1 \times \{0, 1, 2, 3\} \times \{0, 1\},$$

and

$$\Delta_2 = I_{(0,0,0)} \cup I_{(1,0,0)} \cup I_{(0,1,0)} \cup I_{(0,0,1)}.$$

Note that $\Delta_2 \subset \Delta_1$, $\#\Delta_1 = 5 \cdot 4 \cdot 2 = 40$, and $\#\Delta_2 = 1 + 4 + 2 + 1 = 8$.

To estimate the minimum distance of $(C_{\Delta_2}^{J,\sigma})^{\perp}$, we observe that it is a subcode of a hyperbolic code with minimum distance 4, and hence

$$d\left((C_{\Delta_2}^{J,\sigma})^{\perp}\right) \geq 4.$$

Therefore, there exists a CSS-T code with parameters

$$[[128,\ 40 - 8 = 32,\ d \geq 4]].$$

Example 64. We consider now Corollary 62 in the case of two variables, with $J = \{2\}$. Hence, $m = 2$ and $m_1 = 1$. Let $N_1 = 64$ and $N_2 = 4$. Then, the length of the CSS-T code is $n = N_1(N_2 - 1) = 192$.

Let $\Delta^1 = I_0 \cup I_1 \cup I_3 \cup I_5 \cup I_7 \cup I_9$, and $\Delta^2 = I_0 \cup I_1 = \{0, 1, 2, 4, 8\}$. Again, by the BCH bound

$$d\left((C_{\Delta^2}^{J,\sigma})^{\perp}\right) \geq 4.$$

Now, let

$$\Delta_2 = I_{(0,0)} \cup I_{(1,0)} \cup I_{(0,1)},$$

which has cardinality $\#\Delta_2 = 9$. Since the dual code $(C_{\Delta_2}^{J,\sigma})^{\perp}$ is a subcode of the corresponding hyperbolic code of disctance 4, it follows that

$$d((C_{\Delta_2}^{J,\sigma})^{\perp}) = 4.$$

Finally, let

$$\Delta_1 = \Delta^1 \times \{0, 1, 2\},$$

with cardinality $\#\Delta_1 = 66$.

Therefore, there exists a CSS-T code with parameters

$$[[192,\ 66 - 9 = 57,\ d \geq 4]].$$

In Table 4.2, we list the selections of Δ_1 and Δ_2 for other values of N_2. For instance, replacing $N_2 - 1 = 3$ with $N_2 - 1 = 7$ yields a quantum code with parameters $[[448, 141, 4]]$, and replacing it with $N_2 - 1 = 9$ yields one with parameters $[[576, 183, 4]]$.

Δ_2	Δ_1	CSS-T
$I_{(0,0,0)} \cup I_{(1,0,0)} \cup I_{(0,1,0)} \cup I_{(0,0,1)}$	$\{0,1,2,4,8\} \times \{0,1,2,3\} \times \{0,1\}$	$[[128, 32, 4]]$
$I_{(0,0)} \cup I_{(1,0)} \cup I_{(0,1)}$	$\{0,1,2,4,8,16,32,3,6,12,24,48,33,5,10,20,40,17,34,9,18,36\} \times \{0,1,2\}$	$[[192, 57, 4]]$
$I_{(0,0)} \cup I_{(0,1)} \cup I_{(0,1)} \cup I_{(1,1)} \cup I_{(3,0)} \cup I_{(5,0)}$	$\{0,1,2,4,8,3,6,12,24,48,96,65,5,10,20,40,80,33,66,9,18,36,72,17,34,68\} \times \{0,1\}$	$[[256, 28, 8]]$
$I_{(0,0)} \cup I_{(1,0)} \cup I_{(0,1)}$	$\{0,1,2,4,8,16,32,3,6,12,24,48,33,5,10,20,40,17,34,9,18,36\} \times \{0,1,2,3,4,5,6\}$	$[[448, 141, 4]]$
$I_{(0,0,0,0)} \cup I_{(0,0,0,1)} \cup I_{(0,0,1,0)} \cup I_{(0,1,0,0)} \cup I_{(1,0,0,0)}$	$\{0,1,2,4,8,16,32,3,6,12,24,48,33,5,10,20,40,17,34,9,18,36\} \times \{0,1\} \times \{0,1\} \times \{0,1\}$	$[[512, 166, 4]]$
$I_{(0,0)} \cup I_{(1,0)} \cup I_{(0,1)}$	$\{0,1,2,4,8,16,32,3,6,12,24,48,33,5,10,20,40,17,34,9,18,36\} \times \{0,1,2,3,4,5,6,7,8\}$	$[[576, 183, 4]]$
$I_{(0,0)} \cup I_{(0,1)} \cup I_{(1,0)} \cup I_{(1,1)} \cup I_{(3,0)}$	$\{0,1,2,4,8,16,32,64,128,256,3,6,12,24,48,96,192,384,257,5,10,20,40,80,160,320,129,258,7,14,28,56,112,224,448,385,259,9,18,36,72,144,288,65,130,260,11,22,44,88,176,352,193,386,261,13,26,52,104,208,416,321,131,262,17,34,68,136,272,33,66,132,264,19,38,76,152,304,97,194,388,265,21,42,84,168,336,161,322,133,266,25,50,100,200,400,289,67,134,268,35,70,140,280,49,98,196,392,273,37,74,148,296,81,162,324,137,274,41,82,164,328,145,290,69,138,276,73,146,292\} \times \{0,1\}$	$[[1024, 231, 6]]$
$I_{(0,0)} \cup I_{(0,1)} \cup I_{(1,0)} \cup I_{(1,1)} \cup I_{(3,0)} \cup I_{(5,0)}$	$\{0,1,2,4,8,16,32,64,128,256,3,6,12,24,48,96,192,384,257,5,10,20,40,80,160,320,129,258,7,14,28,56,112,224,448,385,259,9,18,36,72,144,288,65,130,260,11,22,44,88,176,352,193,386,261,13,26,52,104,208,416,321,131,262,17,34,68,136,272,33,66,132,264,19,38,76,152,304,97,194,388,265,21,42,84,168,336,161,322,133,266,25,50,100,200,400,289,67,134,268,35,70,140,280,49,98,196,392,273,37,74,148,296,81,162,324,137274,41,82,164,328,145,290,69,138,276,73,146,292\} \times \{0,1\}$	$[[1024, 222, 8]]$

Table 4.2: Cyclotomic cosets used in Examples 63 and 64, and Table 4.3.

To conclude this section, Table 4.3 compares our codes with those in [2, 16]. The length and minimum distance of the codes in each row coincide, while our codes have larger dimension. Note also that the CSS-T codes from the J-affine variety construction outperform those from the WRM construction; however, for length 128 the WRM CSS-T code surpasses the J-affine one. A heuristic procedure to increase the dimension of a CSS-T code without reducing its minimum distance was proposed in [16, Corollary 3.9]. The resulting codes are labeled "Improved Reed–Muller" and "Improved Extended Cyclic" in Table 4.3. We have not applied this heuristic, so there remains potential to enhance our parameters using [16, Corollary 3.9].

n	Reed-Muller	Improved Reed-Muller	Extended Cyclic	Improved Extended Cyclic	WRM	J-Affine
128	$[[128, 21, 4]]$	$[[128, 26, 4]]$	$[[128, 28, 4]]$		$[[\mathbf{128}, \mathbf{36}, \mathbf{4}]]$	$[[\mathbf{128}, \mathbf{32}, \mathbf{4}]]$
256			$[[256, 20, 8]]$	$[[256, 22, 8]]$	$[[\mathbf{256}, \mathbf{21}, \mathbf{8}]]$	$[[\mathbf{256}, \mathbf{28}, \mathbf{8}]]$
512	$[[512, 120, 4]]$	$[[512, 133, 4]]$	$[[512, 147, 4]]$	$[[512, 148, 4]]$	$[[\mathbf{512}, \mathbf{176}, \mathbf{4}]]$	$[[\mathbf{512}, \mathbf{166}, \mathbf{4}]]$
1024			$[[1024, 210, 6]]$	$[[1024, 217, 6]]$		$[[\mathbf{1024}, \mathbf{231}, \mathbf{6}]]$
1024	$[[1024, 120, 8]]$	$[[1024, 125, 8]]$	$[[1024, 190, 8]]$	$[[1024, 192, 8]]$	$[[\mathbf{1024}, \mathbf{204}, \mathbf{8}]]$	$[[\mathbf{1024}, \mathbf{222}, \mathbf{8}]]$

Table 4.3: Parameters of the CSS-T codes in [2, 16], and the codes given in this section.

4.2 Private Information Retrieval

A *Private Information Retrieval* (PIR) scheme is a cryptographic protocol that enables a user to retrieve an item from a database without revealing to the database owner which item is being accessed. When the data is stored across multiple servers, that is, in a distributed storage system, no individual server can determine the specific item requested by the user. In this latter case, one proposed approach for constructing PIR schemes involves encoding the data using a storage linear code $C \subset \mathbb{F}_q^n$, and employing a retrieval linear code $D \subset \mathbb{F}_q^n$ for the data retrieval process [23].

If any set of t servers cannot obtain any information about the requested item, the PIR scheme is said to resist a t-collusion attack, or equivalently, to provide privacy level t. The following result characterizes the security and efficiency of a PIR scheme under collusion attacks, where some servers may share their data in an attempt to infer the user's request based on such a pair C, D of linear codes. Note that a linear code is said to be *transitive* if its *automorphism group* acts transitively on the set of coordinates. That is, for any pair of coordinate positions $i, j \in \{1, \dots, n\}$, there exists a permutation π of the coordinate positions such that $\pi(i) = j$ and π is an automorphism of the code C. We will denote the automorphism group of C by $\mathrm{Aut}(C)$.

Theorem 65 ([23])**.** *If* $\mathrm{Aut}(C)$ *and* $\mathrm{Aut}(C \star D)$ *act transitively on the set of coordinates* $\{1, \dots, n\}$*, then there exists a PIR scheme with rate*

$$R = \frac{\dim(C \star D)^{\perp}}{n}$$

such that it resists a $(d(D^{\perp}) - 1)$*–collusion attack.*

Reed–Muller codes [23] and cyclic codes [8] have been successfully proposed to address the properties outlined in the theorem above and to construct PIR schemes. We will show that PIR schemes based on hyperbolic codes may outperform those constructed from Reed–Muller codes over non-binary fields. Moreover, we will consider J-variety codes and their subfield subcodes to construct PIR schemes with excellent parameters.

Since transitivity is required property for the PIR schemes constructions (as noted in the introduction 1), we focus now on new families of codes that satisfy this condition.

4.2.1 PIR from hyperbolic codes

We begin by addressing the case in which the storage code C is a Reed–Muller code and the retrieval code D is taken to be the dual of a hyperbolic code. We compare this with the classical setting where both C and D are Reed–Muller codes, as in [23]. Specifically, in our comparison, the storage code C is fixed as a Reed–Muller code in both scenarios, while the retrieval code D is either a Reed–Muller code (as in [23]) or the dual of a hyperbolic code with the same minimum distance.

It is important to note that the dimension of a hyperbolic code is greater than or equal to that of a Reed–Muller code with the same minimum distance. Furthermore, since the dual of a Reed–Muller code is again a Reed–Muller code, it follows that the dual of a hyperbolic code has dimension less than or equal to that of a Reed–Muller code with the same minimum distance. This implies that both retrieval codes offer the same level of privacy, resisting a t-collusion attack with $t = d(D^{\perp}) - 1$, as both codes have the same minimum distance. However,

if $\dim(\mathrm{Hyp}_q(s,m)) > \dim(\mathrm{RM}_q(s,m))$, the PIR scheme achieves a better rate when D is the dual of a hyperbolic code, since the dimension of $(C \star D)^\perp$ is smaller in this case.

We illustrate the aforementioned behavior over the finite field with 7 elements using two and three variables. In Table 4.4 and Table 4.5, we fix the storage code C as the Reed–Muller codes $\mathrm{RM}(1,2)_7$ and $\mathrm{RM}(1,3)_7$, respectively. For the retrieval code D, the shaded rows correspond to the case where D is a Reed–Muller code, while the bold rows correspond to the case where D is the dual of a hyperbolic code, both with the same minimum distance. We observe that the duals of hyperbolic codes consistently yield better rates than their Reed–Muller counterparts.

s	C	D	$D^\perp$	$C \star D$	$(C \star D)^\perp$	Privacy	R_{PIR}
3	$[49,3,42]_7$	$[49,10,28]_7$	$[49,39,5]_7$	$[49,15,21]_7$	$[49,34,6]_7$	4	34/49
5	$\mathbf{[49,3,42]_7}$	$\mathbf{[49,8,28]_7}$	$\mathbf{[49,41,5]_7}$	$\mathbf{[49,14,21]_7}$	$\mathbf{[49,35,6]_7}$	**4**	**35/49**
4	$[49,3,42]_7$	$[49,15,21]_7$	$[49,34,6]_7$	$[49,21,14]_7$	$[49,28,7]_7$	5	28/49
6	$\mathbf{[49,3,42]_7}$	$\mathbf{[49,10,21]_7}$	$\mathbf{[49,39,6]_7}$	$\mathbf{[49,18,14]_7}$	$\mathbf{[49,31,7]_7}$	**5**	**31/49**
5	$[49,3,42]_7$	$[49,21,14]_7$	$[49,28,7]_7$	$[49,28,7]_7$	$[49,21,14]_7$	6	21/49
7	$\mathbf{[49,3,42]_7}$	$\mathbf{[49,14,14]_7}$	$\mathbf{[49,35,7]_7}$	$\mathbf{[49,23,7]_7}$	$\mathbf{[49,26,12]_7}$	**6**	**26/49**
6	$[49,3,42]_7$	$[49,28,7]_7$	$[49,21,14]_7$	$[49,34,6]_7$	$[49,15,21]_7$	13	15/49
14	$\mathbf{[49,3,42]_7}$	$\mathbf{[49,25,7]_7}$	$\mathbf{[49,24,14]_7}$	$\mathbf{[49,32,6]_7}$	$\mathbf{[49,17,20]_7}$	**13**	**17/49**
7	$[49,3,42]_7$	$[49,34,6]_7$	$[49,15,21]_7$	$[49,39,5]_7$	$[49,10,28]_7$	20	10/49
21	$\mathbf{[49,3,42]_7}$	$\mathbf{[49,34,6]_7}$	$\mathbf{[49,15,21]_7}$	$\mathbf{[49,39,5]_7}$	$\mathbf{[49,10,28]_7}$	**20**	**10/49**

Table 4.4: Comparison of $D = \mathrm{RM}_7(s,2)$ codes (shaded rows) with $D = \mathrm{Hyp}_7(s,2)^\perp$ codes (boldface rows).

4.2.2 PIR with subfield-subcodes of J-affine variety codes

We propose two constructions that provide pairs of codes with the desired properties to construct a PIR scheme from subfield subcodes of J-affine variety codes.

4.2.2.1 Using subfield subcodes of J-Affine Variety Codes in One Variable

It is important to note that subfield subcodes of J-affine variety codes correspond to BCH codes when the evaluation does not include zero; however, this correspondence no longer holds when evaluation at zero is considered. Let q^r denote the order of the finite field, and q the order of the subfield considered, that is, the cyclotomic cosets are computed over $\mathbb{F}_q$. Let n be a divisor of $q^r - 1$, and let I_a denote the cyclotomic coset associated with a modulo n. In this setting, the code is the evaluation code at the n-th roots of unity, possibly including zero.

The subfield subcodes of one-variable J-affine codes may improve PIR schemes' performance. The following example presents some representative parameters that demonstrate this improvement. Using the method described in Section 4.2, where C is taken as the code $\mathrm{RM}_7(1,2)$ and D is chosen as the dual of a hyperbolic code, the parameters obtained by puncturing these codes are shown in the shaded rows of Table 4.6.

It should be noted that puncturing or shortening a transitive code is a valid form of comparison, as it preserves the minimum distance of the original code (see [38, Theorem 7.6.1]), and

s	C	D	$D^{\perp}$	$C \star D$	$(C \star D)^{\perp}$	Privacy	R_{PIR}
2	$[343, 4, 294]_7$	$[343, 10, 245]_7$	$[343, 333, 4]_7$	$[343, 20]_7$	$[343, 323]_7$	3	323/343
4	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,7,245]_7}$	$\mathbf{[343,336,4]_7}$	$\mathbf{[343,19]_7}$	$\mathbf{[343,324]_7}$	**3**	**324/343**
3	$[343, 4, 294]_7$	$[343, 20, 196]_7$	$[343, 323, 5]_7$	$[343, 35]_7$	$[343, 308]_7$	4	308/343
5	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,13,21]_7}$	$\mathbf{[343,330,5]_7}$	$\mathbf{[343,29]_7}$	$\mathbf{[343,314]_7}$	**4**	**314/343**
4	$[343, 4, 294]_7$	$[343, 35, 147]_7$	$[343, 308, 6]_7$	$[343, 56]_7$	$[343, 287]_7$	5	287/343
6	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,16,147]_7}$	$\mathbf{[343,327,6]_7}$	$\mathbf{[343,38]_7}$	$\mathbf{[343,305]_7}$	**5**	**305/343**
5	$[343, 4, 294]_7$	$[343, 56, 98]_7$	$[343, 287, 7]_7$	$[343, 84]_7$	$[343, 259]_7$	6	259/343
7	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,25,98]_7}$	$\mathbf{[343,318,7]_7}$	$\mathbf{[343,53]_7}$	$\mathbf{[343,290]_7}$	**6**	**290/343**
6	$[343, 4, 294]_7$	$[343, 84, 49]_7$	$[343, 259, 14]_7$	$[343, 117]_7$	$[343, 226]_7$	13	226/343
14	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,59,49]_7}$	$\mathbf{[343,284,14]_7}$	$\mathbf{[343,98]_7}$	$\mathbf{[343,245]_7}$	**13**	**245/343**
7	$[343, 4, 294]_7$	$[343, 117, 42]_7$	$[343, 226, 21]_7$	$[343, 153]_7$	$[343, 190]_7$	20	190/343
21	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,95,42]_7}$	$\mathbf{[343,248,21]_7}$	$\mathbf{[343,144]_7}$	$\mathbf{[343,199]_7}$	**20**	**199/343**
8	$[343, 4, 294]_7$	$[343, 153, 35]_7$	$[343, 190, 28]_7$	$[343, 190]_7$	$[343, 153]_7$	27	153/343
28	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,120,35]_7}$	$\mathbf{[343,223,28]_7}$	$\mathbf{[343,154]_7}$	$\mathbf{[343,169]_7}$	**27**	**169/343**
9	$[343, 4, 294]_7$	$[343, 190, 28]_7$	$[343, 153, 35]_7$	$[343, 226]_7$	$[343, 117]_7$	34	117/343
35	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,144,28]_7}$	$\mathbf{[343,199,35]_7}$	$\mathbf{[343,201]_7}$	$\mathbf{[343,142]_7}$	**34**	**142/343**
10	$[343, 4, 294]_7$	$[343, 226, 21]_7$	$[343, 117, 42]_7$	$[343, 259]_7$	$[343, 84]_7$	41	84/343
42	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,168,21]_7}$	$\mathbf{[343,175,42]_7}$	$\mathbf{[343,225]_7}$	$\mathbf{[343,118]_7}$	**41**	**118/343**
11	$[343, 4, 294]_7$	$[343, 259, 14]_7$	$[343, 84, 49]_7$	$[343, 287]_7$	$[343, 56]_7$	48	56/343
49	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,192,14]_7}$	$\mathbf{[343,151,49]_7}$	$\mathbf{[343,244]_7}$	$\mathbf{[343,99]_7}$	**48**	**99/343**
12	$[343, 4, 294]_7$	$[343, 287, 7]_7$	$[343, 56, 98]_7$	$[343, 308]_7$	$[343, 35]_7$	97	35/343
98	$\mathbf{[343,4,294]_7}$	$\mathbf{[343,265,7]_7}$	$\mathbf{[343,78,98]_7}$	$\mathbf{[343,295]_7}$	$\mathbf{[343,48]_7}$	**97**	**48/343**

Table 4.5: Comparison of $D = \mathrm{RM}_7(s, 3)$ codes (shaded rows) with $D = \mathrm{Hyp}_7(s, 3)^{\perp}$ codes (boldface rows).

results in a code of similar length to the subfield subcode being compared. Furthermore, the transitivity of the code ensures that the result of puncturing is independent of the specific positions chosen, since all punctured versions are permutation equivalent (see [38, Theorem 1.6.6]).

Remark 66. To simplify the notation, in the remainder of the chapter, we will denote the subfield subcode of the J-affine variety codes $C^{J,\sigma}_{\Delta_C}$ and $D^{J,\sigma}_{\Delta_D}$, by just C and D, respectively. We will also say that C and D are defined by Δ_C and Δ_D, respectively, and that C and D have defining set Δ_C and Δ_D, respectively.

Example 67. Consider $q = 7$, $r = 2$, $N_1 = 48$, and set $J = \{1\}$. Then $\mathcal{R}_J = \mathbb{F}_q[x_1]/\langle x_1^{48} - 1\rangle$. Let $\Delta_C = \{24, 25, 31\}$ and $\Delta_D = \{24, 25, 31, 32\}$. The code C is a $[48, 3]_7$ code and D is a $[48, 4]_7$ code, whose dual $D^{\perp}$ has parameters $[48, 44, 4]_7$. Therefore, the star product code $C \star D$ has parameters $[48, 8]_7$, and its dual $(C \star D)^{\perp}$ has parameters $[48, 40]_7$.

Since both C and D are cyclic codes, and the automorphism group of a cyclic code is transitive (see [38, Theorem 1.6.4]), Theorem 65 is applicable. Consequently, this PIR scheme is secure against 3 colluding servers and achieves a PIR rate of $\frac{40}{48}$.

At the same privacy level, the punctured code D, which is the dual of a hyperbolic code—specifically, $D^{\perp}$ is the shortened hyperbolic code $\mathrm{Hyp}_7(4, 2)$—has parameters $[48, 5]_7$ and provides a PIR rate of $\frac{38}{48}$. Hence, the subfield subcode of the J-affine code achieves a better PIR rate. These

parameters are shown in the first two rows of Table 4.6. The first row, shaded in gray, corresponds to the code generated using the method described in Section 4.2, while the bold-faced row represents the code obtained as a subfield subcode of a one-variable J-affine variety code.

In Table 4.6, except for the case corresponding to privacy level 23, all subfield subcodes of one-variable J-affine codes either achieve the same PIR rate, or outperform those based on hyperbolic duals for the same privacy level. Moreover, the use of cyclotomic cosets allows a larger set of design parameters, enabling the construction of PIR schemes with increased privacy levels and improved rates. In some cases, suitable parameter sets for comparison may not exist; however, the obtained parameters still contribute meaningfully to get a broader set of achievable PIR parameters configurations.

s	C	D	$D^\perp$	$C \star D$	$(C \star D)^\perp$	Privacy	R_{PIR}
4	$[48,3]_7$	$[48,5]_7$	$[48,43,4]_7$	$[48,10]_7$	$[48,38]_7$	3	38/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,4]_7}$	$\mathbf{[48,44,4]_7}$	$\mathbf{[48,8]_7}$	$\mathbf{[48,40]_7}$	**3**	**40/48**
5	$[48,3]_7$	$[48,8]_7$	$[48,40,5]_7$	$[48,14]_7$	$[48,34]_7$	4	34/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,7]_7}$	$\mathbf{[48,41,5]_7}$	$\mathbf{[48,14]_7}$	$\mathbf{[48,34]_7}$	**4**	**34/48**
6	$[48,3]_7$	$[48,10]_7$	$[48,38,6]_7$	$[48,18]_7$	$[48,30]_7$	5	30/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,9]_7}$	$\mathbf{[48,39,6]_7}$	$\mathbf{[48,15]_7}$	$\mathbf{[48,33]_7}$	**5**	**33/48**
8	$[48,3]_7$	$[48,16]_7$	$[48,32,8]_7$	$[48,25]_7$	$[48,23]_7$	7	23/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,13]_7}$	$\mathbf{[48,35,8]_7}$	$\mathbf{[48,23]_7}$	$\mathbf{[48,25]_7}$	**7**	**25/48**
9	$[48,3]_7$	$[48,18]_7$	$[48,30,9]_7$	$[48,27]_7$	$[48,21]_7$	8	21/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,16]_7}$	$\mathbf{[48,32,9]_7}$	$\mathbf{[48,26]_7}$	$\mathbf{[48,22]_7}$	**8**	**21/48**
12	$[48,3]_7$	$[48,21]_7$	$[48,27,12]_7$	$[48,29]_7$	$[48,19]_7$	11	19/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,18]_7}$	$\mathbf{[48,30,12]_7}$	$\mathbf{[48,28]_7}$	$\mathbf{[48,19]_7}$	**11**	**19/48**
	$\mathbf{[48,3]_7}$	$\mathbf{[48,20]_7}$	$\mathbf{[48,28,13]_7}$	$\mathbf{[48,29]_7}$	$\mathbf{[48,18]_7}$	**12**	**18/48**
14	$[48,3]_7$	$[48,25]_7$	$[48,23,14]_7$	$[48,32]_7$	$[48,16]_7$	13	16/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,22]_7}$	$\mathbf{[48,26,14]_7}$	$\mathbf{[48,31]_7}$	$\mathbf{[48,17]_7}$	**13**	**17/48**
	$\mathbf{[48,3]_7}$	$\mathbf{[48,27]_7}$	$\mathbf{[48,21,19]_7}$	$\mathbf{[48,34]_7}$	$\mathbf{[48,14]_7}$	**18**	**14/48**
20	$[48,3]_7$	$[48,32]_7$	$[48,16,20]_7$	$[48,38]_7$	$[48,10]_7$	19	10/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,29]_7}$	$\mathbf{[48,19,20]_7}$	$\mathbf{[48,36]_7}$	$\mathbf{[48,12]_7}$	**19**	**12/48**
21	$[48,3]_7$	$[48,34]_7$	$[48,14,21]_7$	$[48,39]_7$	$[48,9]_7$	20	9/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,31]_7}$	$\mathbf{[48,17,21]_7}$	$\mathbf{[48,38]_7}$	$\mathbf{[48,10]_7}$	**20**	**10/48**
	$\mathbf{[48,3]_7}$	$\mathbf{[48,33]_7}$	$\mathbf{[48,15,22]_7}$	$\mathbf{[48,40]_7}$	$\mathbf{[48,8]_7}$	**21**	**8/48**
24	$[48,3]_7$	$[48,36]_7$	$[48,12,24]_7$	$[48,41]_7$	$[48,7]_7$	23	7/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,35]_7}$	$\mathbf{[48,13,24]_7}$	$\mathbf{[48,42]_7}$	$\mathbf{[48,6]_7}$	**23**	**6/48**
	$\mathbf{[48,3]_7}$	$\mathbf{[48,40]_7}$	$\mathbf{[48,8,33]_7}$	$\mathbf{[48,44]_7}$	$\mathbf{[48,4]_7}$	**32**	**4/48**
	$\mathbf{[48,3]_7}$	$\mathbf{[48,42]_7}$	$\mathbf{[48,6,34]_7}$	$\mathbf{[48,45]_7}$	$\mathbf{[48,3]_7}$	**33**	**3/48**
35	$[48,3]_7$	$[48,43]_7$	$[48,5,35]_7$	$[48,46]_7$	$[48,2]_7$	34	2/48
	$\mathbf{[48,3]_7}$	$\mathbf{[48,43]_7}$	$\mathbf{[48,5,35]_7}$	$\mathbf{[48,46]_7}$	$\mathbf{[48,2]_7}$	**34**	**2/48**

Table 4.6: Comparison of shortened $D = \mathrm{Hyp}_7(s,2)^\perp$ code (shaded rows) with subfield subcode of J-affine code (boldface rows)

For dealing with only one variable, we will introduce an alternative approach. Let Δ_D be defined as the union of consecutive cyclotomic cosets, specifically $I_0 \cup I_1 \cup \cdots \cup I_{a_i}$. We know

Δ_1	Δ_2
$I_{24} \cup I_{25}$	$I_{24} \cup I_{25} \cup I_{32}$
	$I_{25} \cup I_{32} \cup I_{33} \cup I_{34}$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40}$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18}$
	$I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19}$
	$I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26}$
	$I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41}$
	$I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11}$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11} \cup I_4 \cup I_{27}$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11} \cup I_4 \cup I_{27} \cup I_6$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11} \cup I_4 \cup I_{27} \cup I_6 \cup I_{17}$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11} \cup I_4 \cup I_{27} \cup I_6 \cup I_{17} \cup I_{10}$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11} \cup I_4 \cup I_{27} \cup I_6 \cup I_{17} \cup I_{10} \cup I_{13}$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11} \cup I_4 \cup I_{27} \cup I_6 \cup I_{17} \cup I_{10} \cup I_{13} \cup I_{20} \cup I_0 \cup I_3$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11} \cup I_4 \cup I_{27} \cup I_6 \cup I_{17} \cup I_{10} \cup I_{13} \cup I_{20} \cup I_0 \cup I_3 \cup I_1$
	$I_{24} \cup I_{25} \cup I_{32} \cup I_{33} \cup I_{34} \cup I_{40} \cup I_5 \cup I_{18} \cup I_{12} \cup I_{19} \cup I_{26} \cup I_{41} \cup I_{11} \cup I_4 \cup I_{27} \cup I_6 \cup I_{17} \cup I_{10} \cup I_{13} \cup I_{20} \cup I_0 \cup I_3 \cup I_1 \cup I_{16}$

Table 4.7: Cyclotomic cosets used in constructing the boldface rows in Table 4.6.

that the set

$$\{0, 1, 2, \ldots, a_{i+1} - 1\} \subset \Delta_D,$$

which implies that the minimum distance satisfies $d(D^\perp) \geq a_{i+1} + 1$ (BCH bound). Assume that N is a divisor of $q^r - 1$, and define $\nu = \frac{q^r-1}{N}$. We propose two possible definitions for the set Δ_C:

1. $\Delta_C = \{0, N, 2N, \ldots, (\nu - 1)N\}$. Since Δ_1 is a union of cyclotomic cosets, the evaluation at these points increases the dimension of the subfield subcode by $\#\Delta_C$ units.

2. $\Delta_C = I_0 \cup I_N$. This is also a union of cyclotomic cosets and contributes to the dimension of the subfield subcode in the same manner as in the previous case.

Note that in some cases, both definitions coincide. Based on their definitions, we have that

$$\#(\Delta_C + \Delta_D) \leq \#\Delta_C \cdot \#\Delta_D.$$

Therefore, the dimension of the dual code satisfies the following inequality.

$$\dim((C \star D)^\perp) \geq n - \#\Delta_C \cdot \#\Delta_D.$$

We summarize these ideas in the following result.

Lemma 68. *With the construction given above. There exists a PIR scheme with a storage code C of length n and dimension v , privacy level a_{i+1}, and rate*

$$\frac{n - \#\Delta_C \cdot \#\Delta_D}{n}.$$

Example 69. Consider the parameters $q = 2$, $r = 8$, $n = 255$, and $N = 85$, which means $\nu = 3$. We define the sets $\Delta_D = I_0 \cup I_1 = \{0, 1, 2, 4, 8, 18, 32, 64, 128\}$ and $\Delta_C = \{0, 85, 170\}$.

Consequently, D is a $[255, 9]_2$ code, while its dual code $D^\perp$ has parameters $[255, 246, \geq 4]_2$. The code C has parameters $[255, 3]_2$. Therefore, the code $C \star D$ has parameters $[255, 27]_2$, and its dual $(C \star D)^\perp$ has parameters $[255, 288]_2$.

This framework provides a PIR scheme of length 255, with a privacy level of 3, and a rate of $\frac{228}{255}$. This rate is better than the one presented in Table 6 of [8] for the same privacy level, despite having a lower storage rate. Therefore, our approach improves the constellation of possible parameters.

Table 4.8 presents several code parameters following this method. We note that in Table 4.8, the BCH bound for the retrieval codes (D) is sharp and matches their minimum distance. Moreover, by evaluating at zero, we also obtain a PIR scheme of length 256, privacy level 3, and rate $\frac{229}{256}$.

C	D	$D^\perp$	$C \star D$	$(C \star D)^\perp$	Privacy	R_{PIR}
$[255, 3, 85]_2$	$[255, 9]_2$	$[255, 246, 4]_2$	$[255, 27]_2$	$[255, 246]_2$	3	228/255
$[255, 3, 85]_2$	$[255, 17]_2$	$[255, 238, 6]_2$	$[255, 51]_2$	$[255, 204]_2$	5	204/255
$[255, 3, 85]_2$	$[255, 25]_2$	$[255, 230, 8]_2$	$[255, 75]_2$	$[255, 180]_2$	7	180/255
$[255, 3, 85]_2$	$[255, 33]_2$	$[255, 222, 10]_2$	$[255, 99]_2$	$[255, 156]_2$	9	156/255
$[255, 3, 85]_2$	$[255, 49]_2$	$[255, 206, 14]_2$	$[255, 123]_2$	$[255, 132]_2$	13	132/255
$[255, 3, 85]_2$	$[255, 57]_2$	$[255, 198, 16]_2$	$[255, 147]_2$	$[255, 108]_2$	15	108/255
$[255, 3, 85]_2$	$[255, 65]_2$	$[255, 190, 18]_2$	$[255, 171]_2$	$[255, 84]_2$	17	84/255
$[255, 3, 85]_2$	$[255, 69]_2$	$[255, 186, 20]_2$	$[255, 183]_2$	$[255, 72]_2$	19	72/255

Table 4.8: Subfield subcodes of one-variable J-affine codes of length 255 (Example 69).

Δ_C	Δ_D
$\{0, 85, 170\}$	$I_0 \cup I_1$
	$I_0 \cup I_1 \cup I_3$
	$I_0 \cup I_1 \cup I_3 \cup I_5$
	$I_0 \cup I_1 \cup I_3 \cup I_5 \cup I_7$
	$I_0 \cup I_1 \cup I_3 \cup I_5 \cup I_7 \cup I_9 \cup I_{11}$
	$I_0 \cup I_1 \cup I_3 \cup I_5 \cup I_7 \cup I_9 \cup I_{11} \cup I_{13}$
	$I_0 \cup I_1 \cup I_3 \cup I_5 \cup I_7 \cup I_9 \cup I_1 1 \cup I_{13} \cup I_{15}$
	$I_0 \cup I_1 \cup I_3 \cup I_5 \cup I_7 \cup I_9 \cup I_{11} \cup I_{13} \cup I_{15} \cup I_{17}$

Table 4.9: Cyclotomic cosets used for codes in Table 4.8.

Lemma 70. *If $q = 2$ and r is even then $3 \mid q^r - 1$, therefore there exist a PIR scheme with $q^r - 1$ servers, privacy 3, and rate*

$$\frac{q^r - 1 - (3(r+1))}{q^r - 1}.$$

4.2.2.2 Using Hyperbolic Codes

Consider the dual of a hyperbolic code, $D' = \mathrm{Hyp}_q(s, m)^\perp$, with defining set $\Delta_{D'}$. For each point $P \in \Delta_{D'}$, we associate the corresponding cyclotomic coset I_P. We then define the new

set $\Delta_D = \bigcup_{P \in \Delta_{D'}} I_P$. Let D denote the linear code with defining set Δ_D. It is straightforward that $d(D^\perp) \geq s$.

Example 71. Consider the case of two variables with $m = 2$ and $q = 2$, i.e. $q^3 = 8$. We evaluate at all points with nonzero coordinates, i.e., $A_1 = A_2 = \{1, \alpha, \ldots, \alpha^{q-2}\}$, thus $n = 49$.

Let D' be an affine variety code defined by the set $\Delta_{D'} = \{(0,0), (1,0), (2,0), (0,1), (0,2)\}$. To ensure that the defining set includes complete cyclotomic cosets, we consider the code D with defining set

$$\Delta_D = \{(0,0), (1,0), (2,0), (0,1), (0,2)\} \cup \{(4,0), (0,4)\}.$$

Clearly, we have $d(D^\perp) \geq 4$.

Now, define C as the code with defining set $\Delta_C = \{(0,0), (1,0), (2,0), (4,0)\}$. Then

$$\begin{aligned} C \star D = \{&(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (6,0), (0,1), (0,2), (0,4), (1,1), (2,1), (4,1), \\ &(1,2), (2,2), (4,2), (1,4), (2,4), (4,4)\}. \end{aligned}$$

Therefore, the dimension of $(C \star D)^\perp$ is 30.

We now present a specific scenario where the parameters can be explicitly computed, providing PIR schemes with favorable parameters. Consider affine variety codes in two variables. Let $N_1 - 1 \mid q^r - 1$ and $N_2 - 1 \mid q - 1$. This choice of N_2 ensures that the points of the form $(0, a)$ form minimal cyclotomic cosets, facilitating a selection of the set Δ_1 that minimizes the number of elements in the star product $C \star D$.

As in the previous setting, consider the hyperbolic code $\mathrm{Hyp}_q(s,2)^\perp$ with defining set $\Delta_D = \bigcup_{P \in \Delta_{D'}} I_P$, and define D as the linear code with this set. It is clear that $d(D^\perp) \geq s$. Next, define C as the linear code with defining set

$$\Delta_C = \{(0,0), (0,1), (0,2), \ldots, (0,a)\}.$$

Note that Δ_C contains $a + 1$ elements, and since the maximum size is q, we have $a \leq q - 1$. Therefore

$$\dim(C \star D) \leq (a+1) \cdot \dim(D),$$

and this fact motivates the goal of minimizing this dimension.

Let $\mathcal{A} = \{a_0 = 0 < a_1 = 1 < a_2 < \cdots < a_\nu\}$ denote the ordered set of minimal representatives of the cyclotomic cosets.

Theorem 72. *Under the assumptions and notation above, consider $0, a_1, a_2 \in \mathcal{A}$, and define the sets:*

$$\Delta_C = I_{(0,0)} \cup I_{(0,1)} \cup I_{(0,2)} = \{(0,0), (0,1), (0,2)\},$$

$$\Delta_D = I_{(0,0)} \cup I_{(0,1)} \cup I_{(0,2)} \cup I_{(a_1,0)} \cup I_{(a_2,0)}.$$

Then, the codes C and D defined by Δ_C and Δ_D respectively provide a PIR scheme of length n_J, privacy level 3, and rate at least $\frac{n_J - (3 \cdot 2r + 5)}{n_J}$.

Proof. From the footprint bound, we find that $d(D^\perp) = 4$, giving a privacy level 3.

The set Δ_D contains two cyclotomic cosets $I_{(a_1,0)}$ and $I_{(a_2,0)}$, each of size at most r, and three singleton cosets, implying $\#\Delta_D \leq 2r + 3$, and hence $\dim(D) \leq 2r + 3$.

To compute $\dim(C \star D)$, we must consider $3 \cdot (2r)$ products between Δ_C and the nontrivial cosets in Δ_D, plus the Minkowski sum $\Delta_C + \Delta_C$, which includes at most 5 elements if $q > 4$. Therefore, $\#(C \star D) \leq 6r + 5$, and hence $\dim((C \star D)^\perp) \geq n_J - (6r + 5)$. □

Example 73. Let $\mathbb{F}_{7^2}$ be the ambient field and $r = 1$, that is the subfield is $\mathbb{F}_7$. Take $N_1 = 49$, $N_2 = 7$, and $J = \emptyset$, so $n_J = 343$. Then, by Theorem 72, we obtain a PIR scheme with length 343, privacy level 3, and rate $\frac{326}{343}$. Specifically, the code C has parameters $[343, 3]_7$, and the dual of D has parameters $[343, 236, 4]_7$.

Compared to the first two rows in Table 4.5, this setup improves the PIR rate. The corresponding rates are $\frac{323}{343}$ and $\frac{324}{343}$, respectively, for the same privacy level $t = 3$, though our scheme has a lower storage code rate $R_s = \frac{3}{343}$ versus $\frac{4}{343}$ in Table 4.5.

Proposition 74. *Let C and D be subfield subcodes of J-affine variety codes of lengths N_1 and N_2, with $J = \emptyset$. Assume $q = 2$, $q^r = 2^r$, $N_1 = 2^r$, and $N_2 = 2$. Consider PIR schemes where C is defined by $\Delta_C = I_{(0,0)}$, yielding a repetition code $[2^{r+1}, 1]$, i.e., a replicated database. We compare these schemes with those based on Reed–Muller codes under the same privacy level.*

(a) *Let*

$$\Delta_D = I_{(0,0)} \cup I_{(1,0)} \cup I_{(0,1)}, \quad \Delta_C = I_{(0,0)}.$$

Then, the star product PIR scheme has length $n_J = 2^{r+1}$, privacy $t = 3$, and rate

$$\frac{n_J - (r + 2)}{n_J}.$$

A Reed–Muller scheme with $C = \mathrm{RM}_2(0, r + 1)$ and $D = \mathrm{RM}_2(1, r + 1)$ gives $D^\perp = \mathrm{RM}_2(r - 1, r + 1)$ with the same privacy and rate.

(b) *Let*

$$\Delta_D = \{I_{(0,0)}, I_{(0,1)}, I_{(1,1)}, I_{(1,0)}, I_{(3,0)}, I_{(5,0)}\}.$$

Then, $\dim(D) \leq 4r + 2$, and $d(D^\perp) = 8$, ensuring privacy $t = 7$. The rate is at least:

$$\frac{n_J - (4r + 2)}{n_J}.$$

For a Reed–Muller construction with $C = \mathrm{RM}_2(0, r + 1)$, $D = \mathrm{RM}_2(2, r + 1)$, we also get privacy $t = 7$, and

$$\dim(D) = \binom{r+1}{0} + \binom{r+1}{1} + \binom{r+1}{2}.$$

Our construction achieves a better PIR rate whenever

$$\binom{r+1}{0} + \binom{r+1}{1} + \binom{r+1}{2} > 4r + 2, \quad \text{which holds for all } r > 5.$$

Example 75. We illustrate Proposition 74, item (b), by comparing the subfield subcode construction of J-affine variety codes with the Reed–Muller-based PIR scheme under the same level of privacy.

- For $r = 7$ and $q = 2$ (i.e. $n_J = 256$), the involved codes have the following parameters.

C	D	$D^\perp$	$(C \star D)^\perp$
$[256, 1]$	$[256, 37]$	$[256, 219, 8]$	$[256, 219]$
$[\mathbf{256, 1}]$	$[\mathbf{256, 30}]$	$[\mathbf{256, 228, 8}]$	$[\mathbf{256, 228}]$

Table 4.10: First row: RM-based construction; second row: subfield subcode of J-affine variety code.

The PIR rate of the scheme based on the first row is $\frac{219}{256}$, while the rate of the one based on the codes in the second row is $\frac{228}{256}$. This shows an improvement of the new construction over the one based on Reed–Muller construction.

- For $r = 8$ and $q = 2$, (i.e. $n_J = 512$), the involved codes have the following parameters.

C	D	$D^\perp$	$(C \star D^\perp$
$[512, 1]$	$[512, 46]$	$[512, 466, 8]$	$[512, 466]$
$[\mathbf{512, 1}]$	$[\mathbf{512, 34}]$	$[\mathbf{512, 478, 8}]$	$[\mathbf{512, 478}]$

Table 4.11: First row: RM-based construction; second row: subfield subcode of J-affine variety code.

The PIR rate of the scheme based on the first row is $\frac{466}{512}$; whereas the rate of the one based on the codes in the second row rises to $\frac{478}{512}$, again outperforming the construction based on Reed Muller codes.

4.2.3 Comparison with Berman Codes

In this final section, we compare the construction based on subfield subcodes of J-affine variety codes with a PIR scheme derived from Berman codes [39], which are defined over the binary field. We consider the same setting as in Example 71, with the following parameters:

$$q = 2, \quad m = 2, \quad q^3 = 8, \quad n_J = 49.$$

Let D be the code defined by

$$\Delta_D = \{(0,0), (1,0), (2,0), (0,1), (0,2)\} \cup \{(4,0), (0,4)\},$$

so that $d(D^\perp) \geq 4$, ensuring a privacy level of $t = 3$.

Now consider the defining set $\Delta_C = \{(0,0)\}$. In this case, the dimension of $(C \star D)^\perp$ is 42. This construction results in a higher PIR rate compared to the scheme based on Berman codes [39], in which the storage code $C = \mathrm{DB}_7(0,2)$ has parameters $[49,1]_2$, and the retrieval code $D = \mathrm{DB}_7(1,2)$ has parameters $[49,13]_2$. Both schemes share the same storage rate $R_s = 1/49$ and privacy level $t = 3$.

Specifically, our scheme achieves a PIR rate of 42/49, while the Berman code-based scheme attains a PIR rate of 36/49. In the table below, the bolded rows highlight the parameters obtained via our construction, whereas the shaded row corresponds to the scheme based on the duals of Berman codes. For the same storage rate and privacy level, our construction offers a superior PIR rate.

C	D	$D^{\perp}$	$C \star D$	$(C \star D)^{\perp}$	R_S	Privacy	R_{PIR}
$\mathbf{[49,1]_2}$	$\mathbf{[49,7,21]_2}$	$\mathbf{[49,42,4]_2}$	$\mathbf{[49,7]_2}$	$\mathbf{[49,42]_2}$	**1/49**	**3**	**42/49**
$\mathbf{[49,1]_2}$	$\mathbf{[49,10,20]_2}$	$\mathbf{[49,39,4]_2}$	$\mathbf{[49,10]_2}$	$\mathbf{[49,39]_2}$	**1/49**	**3**	**39/49**
$[49,1]_2$	$[49,13,16]_2$	$[49,36,4]_2$	$[49,13]_2$	$[49,36]_2$	1/49	3	36/49

Table 4.12: Comparison of Berman codes-based scheme (shaded rows) with J-affine variety codes-based scheme (boldface rows).

Δ_C	Δ_D
$\{(0,0)\}$	$\{(0,0),(1,0),(2,0),(0,1),(0,2),(4,0),(0,4)\}$
	$\{(0,0),(1,0),(2,0),(0,1),(0,2),(4,0),(0,4),(1,1),(2,2),(4,4)\}$

Table 4.13: Cyclotomic cosets used for codes in Table 4.12.

Remark 76. In general, when selecting the storage code $C = \mathrm{DB}_n(r_C, m)$ and the retrieval code $D = \mathrm{DB}_n(r_D, m)$ as given in [39, Table 1], and setting $m = 2$ with $(r_C, r_D) = (0, 1)$, the resulting scheme has $t = 3$, a storage code rate of $R_s = \frac{1}{n^2}$, and a retrieval rate of $\frac{(n-1)^2}{n^2}$. Specifically, the parameters of the involved codes are as follows:

$$C : [n^2, 1, 49], \quad D : [n^2, 13, 7], \quad D^{\perp} : [n^2, 36, 4].$$

For comparison reasons, when choosing $n = q^s - 1 = 2^s - 1$, to achieve the same privacy level and storage rate while obtaining a better retrieval rate, our scheme must satisfy the conditions

$$\dim((C \star D)^{\perp}) = \dim(D^{\perp}) > (n-1)^2$$

$$n^2 - \dim(D^{\perp}) < n^2 - (n-1)^2 = 2^{s+1} + 3$$

$$\dim(D) < 2^{s+1} - 3.$$

Since we fix $t = 3$, we know that $\Delta_{D'} = \{(i,j) | (i+1)(j+1) < 4\}$, so $\#\Delta_{D'} = 5$. Thus, if we have

$$\#\left(\cup_{P \in \Delta_{D'}} I_P\right) < 2^{s+1} - 8,$$

i.e., $\#\Delta_D < 2^{s+1} - 3$, then $\dim((C \star D)^{\perp}) > (n-1)^2$, which gives better retrieval rate than the scheme described in [39].

4.3 Applications to the Multi-Party Computation

Secure multi-party computation (MPC) allows n players to jointly compute a function while ensuring both output correctness and input privacy, even if some players cheat. On the other hand, secret sharing is a method of distributing a secret among a group of n participants so that no individual has meaningful information alone, but a sufficient part of the group can reconstruct it. Certain properties of linear codes help design secret-sharing schemes that allow secure computations, even if up to t participants try to compromise the system. The application of secret sharing in the context of unconditionally secure MPC protocols relies on an additional multiplication property, allowing players to multiply two secret-shared elements by locally converting their shares of the two secrets into a combined sharing of their product. Linear codes with good square properties can be used to create multiplicative secret-sharing schemes (see [22] for a formal definition). More specifically, if a linear code C satisfies the conditions that the minimum distance of its squared code $C^{\star 2}$ is at least $t+2$, and the minimum distance of its dual code $C^{\perp}$ is also at least $t+2$, then it enables the construction of a t-strongly multiplicative secret sharing scheme. This, in turn, is sufficient to build a multiparty computation protocol that is information-theoretically secure against up to t corrupted players. Threfore, it is desirable for creating secure MPC protocols based on a linear code C that the parameters of the code fulfill the dimension of the code and the minimum distance $d(C^{\star 2})$ are big enough and $\min\{d(C^{\perp}), d(C^{\star 2})\} \geq t+2$ to provide information-theoretical security.

Controlling such a series of parameters from a single code is not an easy task; in this section, we will provide some strategies for tackling those parameters while constructing the codes from some well-known families of cartesian product codes.

We will first consider subfield subcodes of cartesian Products codes given by some special chosen cyclotomic cosets. Consider the field $\mathbb{F}_{q^2} \supseteq \mathbb{F}_q$. For $i = 1, \ldots, m$ take $N_i \mid (q+1)$ and $a_i = N_i(q-1)$. Let $s_i \in \mathbb{Z}_{>0}$ be such that $1 \leq s_i < \frac{a_i}{N_i} = q-1$, and define the set $S_i = \{0, N_i, 2N_i, \ldots, s_i N_i\}$. We will denote by $C_{\Delta_{\mathbf{s}},Z}$ the affine variety code with a defining set

$$\Delta_{\mathbf{s}} = S_1 \times \cdots \times S_m \tag{4.2}$$

Lemma 77. *Let $(\alpha_1 \cdot N_1, \ldots, \alpha_m \cdot N_m) \in \Delta_{\mathbf{s}}$, then its cyclotomic coset has a unique point.*

Proof. If we compute $q \cdot (\alpha_1 N_1, \ldots, \alpha_m N_m) \mod (a_1, \ldots, a_m)$, then for each coordinate $i = 1, \ldots, m$, we have that $q\alpha_i N_i = \alpha_i N_i(q-1) + \alpha_i N_i$ thus $q\alpha_i N_i \equiv \alpha_i N_i \mod N_i(q-1)$. The process for q^k is similar just considering the k times multiplication. □

Remark 78. Note that, if $s_i < q-2$, then we are not considering all the multiples of N_i in the set S_i. As we have proved in Lemma 77, each point constitutes its own cyclotomic coset, and therefore the codes $C_{\Delta_{\mathbf{s}},Z}$ and $S(C_{\Delta_{\mathbf{s}},Z})$ have the same parameters. The reason for not considering all the multiples is that in the code $C^{\star 2}_{\Delta_{\mathbf{s}},Z}$ the minimum distance increases as we will show later.

Lemma 79. *The code $C^{\star 2}_{\Delta_{\mathbf{s}},Z}$ has defining set*

$$\hat{\Delta}_{\mathbf{s}} = \hat{S}_1 \times \cdots \times \hat{S}_m \tag{4.3}$$

where $\hat{S}_i = \{0, N_i, 2N_i, \ldots, 2s_i N_i\}$. The biggest gap in $\hat{S}_i$ is $\max\{N_i - 1, N_i(q-1) - 2s_i N_i - 1\}$, for $i \in \{1, \ldots, m\}$.

Note that, by Lemma 77, the codes $C^{\star 2}_{\Delta_{\mathbf{s}},Z}$ and $S(C^{\star 2}_{\Delta_{\mathbf{s}},Z})$ have the same parameters. Thus, from the previous lemma, we have the following result.

Theorem 80. *Consider a set $\Delta_{\mathbf{s}}$ defined as in Equation (4.2). Then*

$$k(S(C_{\Delta_{\mathbf{s}},Z})) = \prod_{i=1}^{m}(s_i+1) \quad \text{and} \quad d(S(C^{\star 2}_{\Delta_{\mathbf{s}},Z})) \geq \prod_{i=1}^{m_1} d_i,$$

where $d_i = \max\{N_i, N_i(q-1) - 2s_iN_i\}$.

Example 81. Let $q = 19$, $N_1 = N_2 = 2$ and $s_1 = s_2 = 5$. We have a code C of length $n = 36^2$ with $d(S(C \star C)) = 16^2$ and $k(S(C)) = 36$.

Let us now consider the cartesian product of Reed-Solomon Codes. Let $\mathbb{F}_q$ be the finite field with q elements. For $i = 1, \dots m$, choose $s_i, a_i \in \mathbb{Z}_{\geq}0$ such that $s_i < a_i < q-1$. For $i = 1, \dots, m$ consider $S_i = \{0, 1, \dots, s_i - 1\}$ and let C be the affine variety codes with defined set

$$\Delta_{\square} = S_1 \times \cdots \times S_m. \tag{4.4}$$

A particular example of this type of code is the cube code family in [15], where all the s_i are equal.

Theorem 82. *Let $C_{\Delta_{\square},Z}$ the monomial cartesian code with defining set $\Delta_{\square}$ as in Equation (4.4). We have that :*

$$n(C_{\Delta_{\square},Z}) = \prod a_i, \quad k(C_{\Delta_{\square},Z}) = \prod_{i=1} s_i, \quad d(C^{\star 2}_{\Delta_{\square},Z}) \geq \prod_{i=1}(a_i - 2s_i + 2), \quad d(C^{\perp}_{\Delta_{\square},Z}) \geq min\{s_i\} + 1.$$

Proof. The length and dimension are clear from the definition. Now, $C^{\star 2}_{\Delta_{\square},Z}$ has defining set $T_1 \times \cdots \times T_m$ where $T_i = \{0, 1, \dots, 2s_i - 2\}$, thus a bound for the minimum distance is $\prod d_i$, where $d_i = a_i - 2s_i + 2$. Finally, we will bound the distance of the dual code using the foot-print bound, i.e. $d(C^{\perp}_{\Delta_{\square},Z}) \geq min\{\prod(b_i + 1) \mid (b_1, \dots, b_m) \notin \Delta_{\square}\} = min\{s_i\} + 1$.

□

Now we are in conditions in providing a mix construction from the previous results. For constructing the code, in one of the variables, we will take a cyclic code (a subfield subcode of a Reed-Solomon code), and in the remaining variables, we use Reed-Solomon codes. Consider the field $\mathbb{F}_{q^r}$ and let C_{Δ} be the cyclic code over $\mathbb{F}_q$ with defining set Δ. For $j = 1, \dots, m$, we consider $2 \leq n_j \leq q$ and we choose C_j to be the code with parameters $[n_j, n_j, 1]$, i,e, its defining set is $S_j = \{0, 1, \dots, n_j - 1\}$. Let C be the affine variety codes with defined set

$$\Delta^{+} = \Delta \times S_1 \times \cdots \times S_m \tag{4.5}$$

Theorem 83. *Let $C_{\Delta^{+},Z}$ the monomial cartesian code where the set Δ^{+} is defined in Equation (4.5). Therefore we have a linear code over $\mathbb{F}_q$ with parameters :*

- $n(C_{\Delta^{+},Z}) = n(C_{\Delta,Z}) \prod_{j=1}^{m} n_j$,
- $k(C_{\Delta^{+},Z}) = k(C_{\Delta,Z}) \prod_{j=1}^{m} n_j$,

- $d(C^{\star^2}_{\Delta^+,Z}) \geq d(C^{\star^2}_{\Delta})$.

Proof. First of all, note that since it is a Cartesian product of complete cyclotomic cosets, thus the defining set is also a union of complete cyclotomic cosets. Therefore, the parameters remain the same over both fields $\mathbb{F}_q$ and $\mathbb{F}_{q^r}$. It is clear that the length is the product of the lengths, and the dimension is the product of the dimensions. Let us denote Δ^2 as the defining set of $C^{\star^2}_{\Delta}$, then the defining set of $C^{\star^2}_{\Delta^+,Z}$ is $\Delta^2 \times S_1 \times \cdots \times S_m$. Hence, from Equation (1.4), one concludes that the distance of $C^{\star^2}_{\Delta^+,Z}$ is also the product of the distances. □

Example 84. Let C_Δ be the de cyclic code defined in [17] with parameters $[127, 15, 55]$ and $d(C^{\star^2}_{\Delta}) \geq 19$. Consider $m = 2$ with $n_j = q = 2$, then we have a code $C_{\Delta^+,Z}$ with parameters $[508, 60]$ and $d(C^{\star^2}_{\Delta^+}) \geq 19$. This construction improves [19, Table 1] where they provide a $[510, 54]$ code C with $d(C^{\star^2}) \geq 18$, whereas we get better parameters from a shorter length code.

Note that, in a more general setting, we could have $m = m_1 + m_2$, so we will consider for m_1 variables $j = 1, \ldots, m_1$ and $2 \leq n_j \leq q$ and C_j the code with parameters $[n_j, n_j, 1]$ (i.e. its defining set is $S_j = \{0, 1, \ldots, n_j - 1\}$); and for for $j = 1, \ldots, m_2$ we consider $2 \leq n_j \leq q$ and C_j the code with parameters $[n_j, 1, n_j]$ (i.e. its defining set is $S_j = \{0\}$). If we denote by $\Delta_{m_1+m_2} = \Delta_{m_1} \times \Delta_{m_2}$ its defining set then it is clear that $C_{\Delta_{m_1+m_2}}$ is a linear code over $\mathbb{F}_q$ such that

- $n(C_{\Delta_{m_1+m_2}}) = \prod_{j=1}^{m} n_j$.
- $k(C_{\Delta_{m_1+m_2}}) = \prod_{j=1}^{m_1} n_j$.
- $d(C^{\star 2}_{\Delta_{m_1+m_2}}) \geq \prod_{j=1}^{m_2} n_j$.

In other words, we can balance $k(C_{\Delta_{m_1+m_2}})$ and $d(C^{\star 2}_{\Delta_{m_1+m_2}})$ in terms of the number component codes we chose for $\Delta_{m_1+m_2} = \Delta_{m_1} \times \Delta_{m_2}$.

Conclusion and Future Work

By leveraging the structure of cyclic codes, we have constructed binary Private Information Retrieval (PIR) schemes that accommodate the presence of colluding servers, following the general framework introduced in [23]. In particular, we present a new family of optimal binary PIR schemes. A key advantage of our constructions over classical PIR schemes derived from Maximum Distance Separable (MDS) codes is that our schemes operate over the binary field $\mathbb{F}_2$, thereby avoiding the need for large field extensions. This simplification not only improves computational efficiency but also enhances the practicality of implementation.

Moreover, our binary PIR schemes encompass a broader set of parameters compared to those based on binary Reed–Muller codes. In several cases, our constructions even outperform the latter in terms of rate and efficiency. An additional benefit of our schemes is the reduction in the cost associated with generating query vectors. Specifically, since the dimension of the retrieval code is smaller, the user requires less randomness to produce valid queries, leading to a more efficient query generation process.

In a different line of work, we investigated PIR in the single-server setting by exploiting the algebraic properties of linear codes defined over finite commutative rings. We proposed a novel PIR protocol based on such ring-linear codes, which is specifically designed to resist the rank difference attack described in [10]. This type of attack identifies structural vulnerabilities in the query by observing changes in rank when certain rows of the query matrix are removed. Our protocol may be viewed as a modification of the scheme introduced in [37], adjusted to provide enhanced robustness against this class of attacks.

While our proposed scheme does not attain the highest retrieval rate when compared to other rank-resistant protocols such as the one in [63], it possesses notable advantages in terms of computational complexity. All operations are performed modulo $\mathbb{Z}_m$, thereby avoiding costly arithmetic over large field extensions. This simplification leads to a significant reduction in the server's computational load, as only modular matrix multiplications are required.

Although the rate of our protocol is lower than that of [37], it offers greater security against adversaries who exploit the algebraic structure of the queries. Therefore, it serves as a viable alternative in scenarios where privacy is of greater concern than communication efficiency.

A natural direction for future work is to investigate whether the enhanced single-server PIR framework proposed in [36], based on the approach introduced in [37] to resist rank difference attacks in submatrices of the query, can be adapted to the ring-based setting.

Lastly, we explored the componentwise (Schur) product of monomial–Cartesian codes through its connection to the Minkowski sum of the exponent sets defining these codes. We demonstrated that J-affine variety codes are particularly well suited for this operation, thereby extending known results for cyclic, Reed–Muller, hyperbolic, and toric codes. This structural insight en-

abled us to construct CSS–T type quantum error-correcting codes from weighted Reed–Muller codes, as well as from binary subfield-subcodes of J-affine variety codes. The resulting quantum codes exhibit improved parameters relative to those previously available in the literature.

In addition, we proposed new PIR schemes for settings with multiple colluding servers, based on both hyperbolic codes and subfield-subcodes of J-affine variety codes. These new constructions demonstrate superior performance when compared with existing schemes, further highlighting the utility of our approach in both classical and quantum coding contexts.

We also comment on potential extensions to secure multi-party computation (MPC), which enables multiple parties to jointly compute a function while preserving both input privacy and output correctness. Multiplicative secret-sharing schemes based on linear codes require a large code dimension, a large minimum distance of the dual code, and a high minimum distance of the code's componentwise square [22]. As noted in Remark 31, the dual of the subfield-subcode of the componentwise product of two J–affine variety codes does not, in general, coincide with the componentwise product of their dual subfield-subcodes. We leave the precise conditions under which this equality holds for future work.

All the explicit examples provided throughout the thesis were obtained using the computer algebra system Magma, as detailed in the Appendix [11].

Bibliography

[1] Alfarano, G.N., Khathuria, K., Weger, V., 2023. A survey on single server private information retrieval in a coding theory perspective. Appl. Algebra Engrg. Comm. Comput. 34, 335–358. URL: https://doi.org/10.1007/s00200-021-00508-5, doi:10.1007/s00200-021-00508-5.

[2] Andrade, E., Bolkema, J., Dexter, T., Eggers, H., Luongo, V., Manganiello, F., Szramowski, L., 2025. Css-t codes from reed muller codes. URL: https://arxiv.org/abs/2305.06423, arXiv:2305.06423.

[3] Ashikhmin, A., Knill, E., 2001. Nonbinary quantum stabilizer codes. IEEE Transactions on Information Theory 47, 3065–3072. doi:10.1109/18.959288.

[4] Banawan, K., Ulukus, S., 2018. The capacity of private information retrieval from coded databases. IEEE Transactions on Information Theory 64, 1945–1956. doi:10.1109/TIT.2018.2791994.

[5] Bhowmick, S., Fotue-Tabue, A., Martínez-Moro, E., Bandi, R., Bagchi, S., 2020. Do non-free LCD codes over finite commutative frobenius rings exist? Designs, Codes and Cryptography 88, 825–840. doi:10.1007/s10623-019-00713-x.

[6] Bierbrauer, J., 2002. The theory of cyclic codes and a generalization to additive codes. Des. Codes Cryptogr. 25, 189–206. doi:10.1023/A:1013808515797.

[7] Şeyma Bodur, Hernando, F., Martínez-Moro, E., Ruano, D., 2025. The schur product of evaluation codes and its application to css-t quantum codes and private information retrieval. URL: https://arxiv.org/abs/2505.10068, arXiv:2505.10068.

[8] Bodur, S., Martínez-Moro, E., Ruano, D., 2023. Private information retrieval schemes using cyclic codes, in: Arithmetic of finite fields. Springer, Cham. volume 13638 of *Lecture Notes in Comput. Sci.*, pp. 194–207. URL: https://doi.org/10.1007/978-3-031-22944-2_12, doi:10.1007/978-3-031-22944-2_12.

[9] Bodur, Ş., Martínez-Moro, E., Ruano, D., 2025. Single server private information retrieval protocols with codes over rings. To appear in Journal of Algebra and Its Applications URL: https://doi.org/10.1142/S0219498825410129, doi:10.1142/S0219498825410129.

[10] Bordage, S., Lavauzelle, J., 2020. On the privacy of a code-based single-server computational pir scheme. Cryptogr. Commun. 13, 519–526. doi:10.1007/s12095-021-00477-z.

[11] Bosma, W., Cannon, J., Playoust, C., 1997. The Magma algebra system. I. The user language. J. Symbolic Comput. 24, 235–265. URL: http://dx.doi.org/10.1006/jsco.1996.0125, doi:10.1006/jsco.1996.0125. computational algebra and number theory (London, 1993).

[12] Calderbank, A.R., Shor, P.W., 1996a. Good quantum error-correcting codes exist. Physical Review A 54, 1098–1105. doi:10.1103/PhysRevA.54.1098.

[13] Calderbank, A.R., Shor, P.W., 1996b. Good quantum error-correcting codes exist. Phys. Rev. A 54, 1098–1105. doi:10.1103/PhysRevA.54.1098.

[14] Camps, E., López, H.H., Matthews, G.L., Sarmiento, E., 2021. Polar decreasing monomial-cartesian codes. IEEE Transactions on Information Theory 67, 3664–3674. doi:10.1109/TIT.2020.3047624.

[15] Camps-Moreno, E., García-Marco, I., López, H.H., Márquez-Corbella, I., Martínez-Moro, E., Sarmiento, E., 2025. On decoding hyperbolic codes, in: Arithmetic of finite fields, Springer, Cham. pp. 37–52. URL: https://doi.org/10.1007/978-3-031-81824-0_3, doi:10.1007/978-3-031-81824-0_3.

[16] Camps-Moreno, E., López, H.H., Matthews, G.L., Ruano, D., San-José, R., Soprunov, I., 2024. An algebraic characterization of binary CSS-T codes and cyclic CSS-T codes for quantum fault tolerance. Quantum Inf. Process. 23, Paper No. 230, 24. URL: https://doi.org/10.1007/s11128-024-04427-5, doi:10.1007/s11128-024-04427-5.

[17] Cascudo, I., 2019. On squares of cyclic codes. IEEE Trans. Inform. Theory 65, 1034–1047. URL: https://doi.org/10.1109/TIT.2018.2867873, doi:10.1109/TIT.2018.2867873.

[18] Cascudo, I., David, B., 2024. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO, in: Advances in cryptology—EUROCRYPT 2024. Part V. Springer, Cham. volume 14655 of *Lecture Notes in Comput. Sci.*, pp. 216–248. URL: https://doi.org/10.1007/978-3-031-58740-5_8, doi:10.1007/978-3-031-58740-5_8.

[19] Cascudo, I., Gundersen, J.S., Ruano, D., 2020. Squares of matrix-product codes. Finite Fields and Their Applications 62, 101606. URL: https://www.sciencedirect.com/science/article/pii/S1071579719301091, doi:https://doi.org/10.1016/j.ffa.2019.101606.

[20] Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M., 1998. Private information retrieval. J. ACM 45, 965–982. URL: https://doi.org/10.1145/293347.293350, doi:10.1145/293347.293350.

[21] Cox, D., Little, J., O'Shea, D., 1997. Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra. Undergraduate Texts in Mathematics. second ed., Springer-Verlag, New York. URL: https://doi.org/10.1007/978-3-319-16721-3, doi:10.1007/978-3-319-16721-3.

[22] Cramer, R., Damgård, I., Maurer, U., 2000. General secure multi-party computation from any linear secret-sharing scheme, in: Advances in cryptology—EUROCRYPT 2000

(Bruges). Springer, Berlin. volume 1807 of *Lecture Notes in Comput. Sci.*, pp. 316–334. URL: https://doi.org/10.1007/3-540-45539-6_22, doi:10.1007/3-540-45539-6_22.

[23] Freij-Hollanti, R., Gnilke, O.W., Hollanti, C., Horlemann-Trautmann, A.L., Karpuk, D., Kubjas, I., 2019. t-private information retrieval schemes using transitive codes. IEEE Transactions on Information Theory 65, 2107–2118. doi:10.1109/TIT.2018.2871050.

[24] Freij-Hollanti, R., Gnilke, O.W., Hollanti, C., Karpuk, D.A., 2017. Private information retrieval from coded databases with colluding servers. SIAM J. Appl. Algebra Geom. 1, 647–664. URL: https://doi.org/10.1137/16M1102562, doi:10.1137/16M1102562.

[25] Galindo, C., Geil, O., Hernando, F., Ruano, D., 2017. On the distance of stabilizer quantum codes from J-affine variety codes. Quantum Inf. Process. 16, Paper No. 111, 32. URL: https://doi.org/10.1007/s11128-017-1559-1, doi:10.1007/s11128-017-1559-1.

[26] Galindo, C., Geil, O., Hernando, F., Ruano, D., 2019. New binary and ternary LCD codes. IEEE Trans. Inform. Theory 65, 1008–1016. URL: https://doi.org/10.1109/TIT.2018.2834500, doi:10.1109/TIT.2018.2834500.

[27] Galindo, C., Hernando, F., Ruano, D., 2015. Stabilizer quantum codes from J-affine variety codes and a new Steane-like enlargement. Quantum Inf. Process. 14, 3211–3231. URL: https://doi.org/10.1007/s11128-015-1057-2, doi:10.1007/s11128-015-1057-2.

[28] García-Marco, I., Márquez-Corbella, I., Ruano, D., 2020. High dimensional affine codes whose square has a designed minimum distance. Des. Codes Cryptogr. 88, 1653–1672. URL: https://doi.org/10.1007/s10623-020-00764-5, doi:10.1007/s10623-020-00764-5.

[29] Geil, O., 2008. On the second weight of generalized Reed-Muller codes. Des. Codes Cryptogr. 48, 323–330. URL: https://doi.org/10.1007/s10623-008-9211-9, doi:10.1007/s10623-008-9211-9.

[30] Geil, O., Hø holdt, T., 2000. Footprints or generalized Bezout's theorem. IEEE Trans. Inform. Theory 46, 635–641. URL: https://doi.org/10.1109/18.825832, doi:10.1109/18.825832.

[31] Geil, O., Høholdt, T., 2001. On hyperbolic codes, in: Boztaş, S., Shparlinski, I.E. (Eds.), Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 159–171. URL: https://doi.org/10.1007/3-540-45624-4_17, doi:10.1007/3-540-45624-4_17.

[32] Gentry, C., Halevi, S., 2019. Compressible FHE with applications to PIR, in: Theory of cryptography. Part II, Springer, Cham. pp. 438–464. URL: https://doi.org/10.1007/978-3-030-36033-7_17, doi:10.1007/978-3-030-36033-7_17.

[33] Grassl, M., 2007. Bounds on the minimum distance of linear codes and quantum codes. Online available at http://www.codetables.de. Accessed on June 2025.

[34] Hansen, J.P., 2017. Secret sharing schemes with strong multiplication and a large number of players from toric varieties, in: Arithmetic, geometry, cryptography and coding theory. Amer. Math. Soc., Providence, RI. volume 686 of *Contemp. Math.*, pp. 171–185. URL: https://doi.org/10.1090/conm/686, doi:10.1090/conm/686.

[35] Hernando, F., Igual, F.D., Quintana-Ortí, G., 2019. Algorithm 994: fast implementations of the Brouwer-Zimmermann algorithm for the computation of the minimum distance of a random linear code. ACM Trans. Math. Software 45, Art. 23, 28. URL: https://doi.org/10.1145/3302389, doi:10.1145/3302389.

[36] Hollanti, C., Verma, N., 2025. Cb-cpir: Code-based computational private information retrieval. URL: https://arxiv.org/abs/2505.03407, arXiv:2505.03407.

[37] Holzbaur, L., Hollanti, C., Wachter-Zeh, A., 2020. Computational code-based single-server private information retrieval, in: 2020 IEEE International Symposium on Information Theory (ISIT), pp. 1065–1070. doi:10.1109/ISIT44484.2020.9174138.

[38] Huffman, W.C., Pless, V., 2003. Fundamentals of error-correcting codes. Cambridge University Press, Cambridge. URL: https://doi.org/10.1017/CBO9780511807077, doi:10.1017/CBO9780511807077.

[39] Kale, S., Agarwal, K., Krishnan, P., 2023. t-pir schemes with flexible parameters via star products of berman codes, in: 2023 IEEE International Symposium on Information Theory (ISIT), pp. 1348–1353. doi:10.1109/ISIT54713.2023.10206639.

[40] Ketkar, A., Klappenecker, A., Kumar, S., Sarvepalli, P.K., 2006. Nonbinary stabilizer codes over finite fields. IEEE Transactions on Information Theory 52, 4892–4914. doi:10.1109/TIT.2006.884430.

[41] Kushilevitz, E., Ostrovsky, R., 1997. Replication is not needed: single database, computationally-private information retrieval, in: Proceedings 38th Annual Symposium on Foundations of Computer Science, pp. 364–373. doi:10.1109/SFCS.1997.646125.

[42] van Lint, J.H., 1999. Introduction to coding theory. volume 86 of *Graduate Texts in Mathematics*. Third ed., Springer-Verlag, Berlin. URL: https://doi.org/10.1007/978-3-642-58575-3, doi:10.1007/978-3-642-58575-3.

[43] López, H.H., Matthews, G.L., Soprunov, I., 2020. Monomial-Cartesian codes and their duals, with applications to LCD codes, quantum codes, and locally recoverable codes. Des. Codes Cryptogr. 88, 1673–1685. URL: https://doi.org/10.1007/s10623-020-00726-x, doi:10.1007/s10623-020-00726-x.

[44] López, H.H., Rentería-Márquez, C., Villarreal, R.H., 2014. Affine Cartesian codes. Des. Codes Cryptogr. 71, 5–19. URL: https://doi.org/10.1007/s10623-012-9714-2, doi:10.1007/s10623-012-9714-2.

[45] MacWilliams, F.J., Sloane, N.J.A., 1977. The theory of error-correcting codes. I. volume Vol. 16 of *North-Holland Mathematical Library*. North-Holland Publishing Co., Amsterdam-New York-Oxford. URL: https://www.sciencedirect.com/bookseries/north-holland-mathematical-library/vol/16/suppl/C.

[46] Martínez-Moro, E., Piñera Nicolás, A., Rúa, I.F., 2018. Codes over affine algebras with a finite commutative chain coefficient ring. Finite Fields Appl. 49, 94–107. URL: https://doi.org/10.1016/j.ffa.2017.09.008, doi:10.1016/j.ffa.2017.09.008.

[47] Martínez-Moro, E., Ruano, D., 2008. Toric codes, in: Advances in algebraic geometry codes. World Sci. Publ., Hackensack, NJ. volume 5 of *Ser. Coding Theory Cryptol.*, pp. 295–322. URL: https://doi.org/10.1142/9789812794017_0008, doi:10.1142/9789812794017_0008.

[48] Melchor, C.A., Barrier, J., Fousse, L., Killijian, M.O., 2016. Xpir: Private information retrieval for everyone. Proceedings on Privacy Enhancing Technologies , 155–174URL: https://doi.org/10.1515/popets-2016-0010, doi:doi.org/10.1515/popets-2016-0010.

[49] Natarajan, L.P., Krishnan, P., 2022. Berman codes: A generalization of reed-muller codes that achieve bec capacity, in: 2022 IEEE International Symposium on Information Theory (ISIT), pp. 1761–1766. doi:10.1109/ISIT50566.2022.9834598.

[50] Nielsen, M.A., Chuang, I.L., 2010. Quantum Computation and Quantum Information. 10th anniversary ed., Cambridge University Press. URL: https://doi.org/10.1017/CBO9780511976667.

[51] Norton, G.H., Sălăgean, A., 2000. On the structure of linear and cyclic codes over a finite chain ring. Appl. Algebra Engrg. Comm. Comput. 10, 489–506. URL: https://doi.org/10.1007/PL00012382, doi:10.1007/PL00012382.

[52] Randriambololona, H., 2015. On products and powers of linear codes under componentwise multiplication, in: Algorithmic arithmetic, geometry, and coding theory. Amer. Math. Soc., Providence, RI. volume 637 of *Contemp. Math.*, pp. 3–78. URL: https://doi.org/10.1090/conm/637/12749, doi:10.1090/conm/637/12749.

[53] Rengaswamy, N., Calderbank, R., Newman, M., Pfister, H.D., 2020a. Classical coding problem from transversal t gates, in: 2020 IEEE International Symposium on Information Theory (ISIT), pp. 1891–1896. doi:10.1109/ISIT44484.2020.9174408.

[54] Rengaswamy, N., Calderbank, R., Newman, M., Pfister, H.D., 2020b. On optimality of css codes for transversal t. IEEE Journal on Selected Areas in Information Theory 1, 499–514. doi:10.1109/JSAIT.2020.3012914.

[55] Rieffel, E.G., Polak, W.H., 2011. Quantum Computing: A Gentle Introduction. Scientific and Engineering Computation, The MIT Press, Cambridge, MA. URL: https://mitpress.mit.edu/9780262015066/quantum-computing/.

[56] Shah, N.B., Rashmi, K.V., Ramchandran, K., 2014. One extra bit of download ensures perfectly private information retrieval, in: 2014 IEEE International Symposium on Information Theory, pp. 856–860. doi:10.1109/ISIT.2014.6874954.

[57] Sion, R., Carbunar, B., 2007. On the computational practicality of private information retrieval, in: Proceedings of the Network and Distributed Systems Security Symposium, Internet Society Geneva, Switzerland. pp. 2006–06. URL: https://users.cs.fiu.edu/~carbunar/pir.pdf.

[58] Soprunov, I., Soprunova, J., 2010. Bringing toric codes to the next dimension. SIAM J. Discrete Math. 24, 655–665. URL: https://doi.org/10.1137/090762592, doi:10.1137/090762592.

[59] Sorensen, A., 1992. Weighted reed-muller codes and algebraic-geometric codes. IEEE Transactions on Information Theory 38, 1821–1826. doi:10.1109/18.165459.

[60] Steane, A., 1996a. Multiple-particle interference and quantum error correction. Proc. Roy. Soc. London Ser. A 452, 2551–2577. doi:10.1098/rspa.1996.0136.

[61] Steane, A.M., 1996b. Error correcting codes in quantum theory. Physical Review Letters 77, 793–797. doi:10.1103/PhysRevLett.77.793.

[62] Ullah, I., Hassan, N., Gill, S.S., Suleiman, B., Ahanger, T.A., Shah, Z., Qadir, J., Kanhere, S.S., 2024. Privacy preserving large language models: Chatgpt case study based vision and framework. IET Blockchain 4, 706–724. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/blc2.12091, doi:10.1049/blc2.12091.

[63] Verma, N., Hollanti, C., 2024. Code-based single-server private information retrieval: Circumventing the sub-query attack. doi:10.1109/ISIT57864.2024.10619469.

[64] Wilde, M.M., 2017. Quantum Information Theory. 2nd ed., Cambridge University Press. URL: https://doi.org/10.1017/9781316809976.

[65] Yardi, A., Pellikaan, R., 2017. On shortened and punctured cyclic codes URL: https://arxiv.org/abs/1705.09859, arXiv:1705.09859.

Appendix: MAGMA code

In this appendix, we provide the MAGMA code used throughout the thesis. Each section corresponds to a specific computation or construction discussed in the main chapters.

A.1 PIR for Several Servers

This MAGMA code constructs Private Information Retrieval Schemes for multiple servers based on cyclic codes (Chapter 2.1). You can produce the code that gives the first row's parameters of Table 2.1.

```
K := GF(2);  // Binary field
n := 127;    // Code length

// Factorize the polynomial x^127 - 1 over GF(2)
Q<y> := PolynomialRing(K);
F := Factorization(y^127 - 1);

// Generate cyclic storage code C
C := CyclicCode(127, F[2][1]*F[3][1]*F[4][1]*F[5][1]*F[6][1]*F[7][1]*F
    [8][1]*F[9][1]*F[10][1]*F[11][1]*F[12][1]*F[13][1]*F[15][1]*F
    [16][1]*F[17][1]*F[18][1]*F[19][1]);

// Compute code parameters for C
d1 := Dimension(C);
Min1 := MinimumDistance(C);

// Generate cyclic retrieval code D
F11 := F[2][1]*F[3][1]*F[5][1]*F[6][1]*F[8][1]*F[9][1]*F[10][1]*F
    [11][1]*F[12][1]*F[13][1]*F[15][1]*F[16][1]*F[17][1]*F[19][1];
D := CyclicCode(127, F11);

// Compute code parameters for D and its dual
d2 := Dimension(D);
dual_d2 := Dimension(Dual(D));
Min2 := MinimumDistance(D);
dual_Min2 := MinimumDistance(Dual(D));

// Get generator matrices
G1 := GeneratorMatrix(C);
G2 := GeneratorMatrix(D);
```

```
28
29 // Construct star product (Schur product) of C and D
30 M := Matrix(GF(2), d1*d2, n, [0: i in [1..d1*d2*n]]);
31 for i in [0..d1-1] do
32     for k in [0..d2-1] do
33         for j in [1..n] do
34             M[k+1+i*d2][j] := G2[k+1][j] * G1[i+1][j];  // Component-
                wise multiplication
35         end for;
36     end for;
37 end for;
38
39 // Star product of C and D
40 C3 := LinearCode(M);
41 d3 := Dimension(Dual(C3));  // Dimension of dual(C \star D)
42 // Print results
43 printf "\n";
44 printf "Code␣C␣Dimension␣=␣%o,␣\n", d1;
45 printf "Code␣D␣Dual␣Dimension␣=␣%o,␣Code␣D␣Dual␣Minimum␣Weight␣=␣%o,␣\n
    ",
46     dual_d2, dual_Min2;
47 printf "Code␣C␣\star␣D␣Dual␣Dimension␣=␣%o,␣\n", d3;
48 printf "\n";
```

A.2 PIR for Single Server

This MAGMA implementation corresponds to the single-server PIR scheme described in Example 55 of Chapter 3. While the same linear codes are used, both the database and the random values in the query matrix are chosen randomly. Consequently, , the queries and responses may differ from those presented in the example.

```
1  s:=2; // number of component
2  t:=3; // number of file
3  L:=1; // number of row of files
4  d:=1; // desired file index
5  // Following part generates inner code: Cin is inner code
6  //1. Generate first component for Inner Code over Z_9
7  P1<x> := PolynomialRing(IntegerRing(9));
8  n:= 13; //code length x^13-1
9  f:=CyclotomicFactors(IntegerRing(9),n); // Factors of x^13-1 over Z_9
10 C1:= CyclicCode(n, f[1]*f[2]*f[3]+3*f[2]*f[1]); //First Component for
       Inner Code
11 D1:=Dual(C1); // Dual of the first Component
12 C3:= CyclicCode(n, 3*f[2]*f[1]); // non-free part of C1
13 D3:=Dual(C3); // Dual of Code C3
14 GG1:=GeneratorMatrix(C1);
15 G1:=Matrix(IntegerRing(),GG1); // Generator Matrix of C1
16 //2. Generate first component for Inner Code over Z_25
17 P2<a> := PolynomialRing(IntegerRing(25));
```

```
f2:=CyclotomicFactors(IntegerRing(25),n); // Factors of x^13-1 over Z_25
C2:= CyclicCode(n, f2[1]*f2[2]*f2[3]+5*f2[2]*f2[1]); //Second Component for Inner Code
D2:=Dual(C2); // Dual Code of the second Component
C4:= CyclicCode(n, 5*f2[2]* f2[1]);  // non-free part of C2
D4:=Dual(C4); // Dual of Code C4
GG2:=GeneratorMatrix(C2);
G2:=Matrix(IntegerRing(),GG2);  // Generator Matrix of C2
Rankc1:= Rank(GeneratorMatrix(C1)); // Number of Rows of Generator Matrix of C1
Rankc2:= Rank(GeneratorMatrix(C2)); // Number of Rows of Generator Matrix of C2
GG3:=GeneratorMatrix(C3);
G3:=Matrix(IntegerRing(),GG3);  // Generator Matrix of C3
GG4:=GeneratorMatrix(C4);
G4:=Matrix(IntegerRing(),GG4); // Generator Matrix of C4
Rankc3:= Rank(GeneratorMatrix(C3)); // Number of Rows of Generator Matrix of C3
Rankc4:= Rank(GeneratorMatrix(C4)); // Number of Rows of Generator Matrix of C4
//Following part Generate Inner Code Cin = (C1,C2)
M1:=ZeroMatrix(IntegerRing(225), Rankc1* Rankc2, 13);
for j in [1..13] do
for i in [1.. Rankc1] do
for k in [1.. Rankc2] do
M1[k+((i-1)* Rankc2)][j]:= CRT([G1[i][j],G2[k][j]],[9,25]);
end for;
end for;
end for;
Cin:=LinearCode(M1);  // Cin= CRT(Nf_Cin , Cin2)
Din:=Dual(Cin);   //Dual Code of Cin
Gin:=GeneratorMatrix(Cin); //Generator Matrix of Cin
Hin:=GeneratorMatrix(Din);  //Parity Check Matrix of Cin
//Following part Generate Inner Code nf_Cin = (C1,C2)
M2:=ZeroMatrix(IntegerRing(225), Rankc3* Rankc4, 13);
for j in [1..13] do
for i in [1.. Rankc3] do
for k in [1.. Rankc4] do
M2[k+((i-1)* Rankc4)][j]:= CRT([G3[i][j],G4[k][j]],[9,25]);
end for;
end for;
end for;
nf_Cin:=LinearCode(M2); //non-free part of Cin
nf_Din:=Dual(nf_Cin ); // Dual of nf_Cin
G_nf_Cin:=GeneratorMatrix(nf_Cin ); // Generator Matrix of nf_Cin
G_nf_Din:=GeneratorMatrix(nf_Din); // Parity Check Matrix of nf_Cin
// Following part generates outer code: s x s : 2x2 Cout= [Tilde(C_1), Tilde(C_2)]M, M is 2x2 full rank Matrix
P3<xx>:=PolynomialRing(IntegerRing(225));
ff:=xx^13-1;
```

```
Q11<yy>:=quo< P3|ff>;
QC2:= ZeroMatrix(Q11,1,s);
for j in [1..s] do
QC2[1][j]:=Random(Q11);
end for;
//1. Generate Tilde(C_1)
t1:=  f[2]*f[3]*f[4]+3*f[3]*f[2];
t3:= f2[3]*f2[2]*f2[4]+5*f2[2]*f2[3];
v1:=Vector(IntegerRing(),Coefficients(t1));
v2:=ZeroMatrix(IntegerRing(),1,13);
InsertBlock(~v2,v1,1,1);
v3:=Vector(IntegerRing(),Coefficients(t3));
v4:=ZeroMatrix(IntegerRing(),1,13);
InsertBlock(~v4,v3,1,1);
r1:=RandomMatrix(IntegerRing(225),1,13);
for ii in [1..13] do
r1[1][ii]:=CRT([v2[1][ii],v4[1][ii]],[9,25]);
end for;
QC1:=Q11! Eltseq(r1[1]);
QC2[1][1]:=QC1;
//2. Generate Tilde(C_2)
t3:=  f[2]*f[3]*f[4]*f[5]+3*f[3]*f[2]*f[4];
t4:= f2[3]*f2[2]*f2[4]+5*f2[2]*f2[3];
v5:=Vector(IntegerRing(),Coefficients(t3));
v6:=ZeroMatrix(IntegerRing(),1,13);
InsertBlock(~v6,v5,1,1);
v7:=Vector(IntegerRing(),Coefficients(t4));
v8:=ZeroMatrix(IntegerRing(),1,13);
InsertBlock(~v8,v7,1,1);
r2:=RandomMatrix(IntegerRing(225),1,13);
for ii in [1..13] do
r2[1][ii]:=CRT([v6[1][ii],v8[1][ii]],[9,25]);
end for;
QC1:=Q11! Eltseq(r2[1]);
QC2[1][2]:=QC1;
//3. Generate Matrix Product Code Cout=(Tilde(C_1),Tilde(C_2))M over R
QC:= ZeroMatrix(Q11,s,s);
M:=Matrix(Q11,s,s,[ 1,1,0,1]);
for i in [1..s] do
for j in [1..s] do
QC[i][j]:=M[i][j]*QC2[1][i]; // QC is the generator matrix of Outer
    Code in R
end for;
end for;
// Following part generate a code which we will use for selecting
    element for matrix U where elements of U in nf(Tilde(C_2)
    intersection Dual(Cin)\ Cin)
C5:= CyclicCode(n, 3*f[3]*f[4]);
D5:=Dual(C5);
C6:= CyclicCode(n, 5*f2[3]);
D6:=Dual(C6);
```

```
GG5:=GeneratorMatrix(C5);
G5:=Matrix(IntegerRing(),GG5);
GG6:=GeneratorMatrix(C6);
G6:=Matrix(IntegerRing(),GG6);
Rankc5:= Rank(GeneratorMatrix(C5));
Rankc6:= Rank(GeneratorMatrix(C6));
M3:=ZeroMatrix(IntegerRing(225), Rankc5* Rankc6, 13);
for j in [1..13] do
for i in [1.. Rankc5] do
for k in [1.. Rankc6] do
M3[k+((i-1)* Rankc6)][j]:= CRT([G5[i][j],G6[k][j]],[9,25]);
end for;
end for;
end for;
Cout1:=LinearCode(M3);
Dout1:=Dual(Cout1);
// Following part write the outer code Cout in Z_m which is named Cout
Cout_Zm:=ZeroMatrix(IntegerRing(225),13*(s),13*s);  // Cout_Zm is a
    projection of QC in Z_m
for i in [1..s] do
for j in [1..s] do
v9:=Vector(IntegerRing(225),Coefficients(QC[i][j]));
v10:=ZeroMatrix(IntegerRing(225),1,13);
InsertBlock(~v10,v9,1,1);
M4:=ZeroMatrix(IntegerRing(225),13,13);
InsertBlock(~M4, v10,1,1);
Y:=Vector(IntegerRing(225),M4[1]);
for i in [2..13] do
Rotate(~Y,1);
InsertBlock(~M4,Y,i,1);
end for;
InsertBlock(~Cout_Zm,M4,13*(i-1)+1, 13*(j-1)+1);
end for;
end for;
Cout:=LinearCode(Cout_Zm);   // The Outer Code Cout
//Database Setup
DB:= ZeroMatrix(Q11,1,L*t);
for i in [1..t*L] do
DB[1][i]:=Random(IntegerRing(15));
end for;
Database:=ChangeRing(DB,Q11);
//Query Generation Process
A:= ZeroMatrix(Q11,L*t,s);  //Random elements in R
a1:=ZeroMatrix(Q11,1,s);
W:= ZeroMatrix(Q11, L*t,s); // Encoding matrix of A, W= A*QC , where QC
    is a generator matrix of outer code in R
E:= ZeroMatrix(Q11, L*t,s); // Random Elements in nf(Cin)
e:=ZeroMatrix(Q11,1,s);
e1:=ZeroMatrix(Q11,1,1);
U:= ZeroMatrix(Q11,L*t,s);  // Random Elements in nf(Tilde(C_2)
    intersection Dual(Cin)\ Cin)
```

```
158 Delta:= ZeroMatrix(Q11,L*t,s);  // Delta=W^i+E^i+U^i
159 //Folllowing Part select random matrix A and encode with outer code and
        get W
160 for i in [1..L*t] do
161 for j in [1..s] do
162 a1[1][j]:=15*Random(Q11);
163 A[i][j]:= a1[1][j];
164 end for;
165 c:=a1*QC;
166 InsertBlock(~W,c,i,1);
167 end for;
168 // Following Part Generate matrix E which has elements in nf(Cin)
169 for i in [1..L*t] do
170 for j in [1..s] do
171 e[1][j]:= Q11 ! Eltseq(Random(nf_Cin ));
172 end for;
173 InsertBlock(~E,e,i,1);
174 end for;
175 // Following Part Generate matrix U which has elements in nf(Tilde(C_2)
        intersection Dual(Cin)\ Cin)
176 for i in [0..L-1] do
177 u:=Random(Cout1);
178 U[((d-1)*L)+1+i][1+i]:= Q11 ! Eltseq(u);
179 end for;
180 //Query=[A || Delta]
181 Delta:=E+W+U;
182 // Server Response R=[R1||R2]=[Database*A|| Database*Delta]
183 R1:=Database*A;  //R1
184 R2:= Database *Delta; //R2
185 //Recovering Process
186 R1_QC:= R1*QC;  // User Calculate: R1*Cout= Database*A*QC
187 Database_EU:= R2- R1_QC; // User Calculate R2-R1*Cout= Databese*(E+U)
188 // Following part Solve the system of Equations in Z_9
189 h1:=ZeroMatrix(IntegerRing(9),L,s*13); // Calculate [Databese*(E+U)]*
        Dual(C1)] C1 is the first component code of Cin in Z_9
190 h2:=ZeroMatrix(IntegerRing(9),1,s*13); // Calculate U*Dual(C1) in Z_9
        then solve equation h1 and h2 to find the desired file
191 for i in [1..1] do
192 for j in [1..s] do
193 h3:=Vector(IntegerRing(9),Coefficients(Database_EU [i][1+j-1]));
194 h4:=ZeroMatrix(IntegerRing(9),1,13);
195 InsertBlock(~h4,h3,1,1);
196 h5:= h4*Transpose(GeneratorMatrix(Dual(C1)));
197 InsertBlock(~h1, h5,i,(j-1)*13+1 );
198 end for;
199 end for;
200 for j in [1..s] do
201 for i in [1..1] do
202 h6:=Vector(IntegerRing(9),Coefficients(U[((d-1)*s)+i][1+(j-1)]));
203 h7:=ZeroMatrix(IntegerRing(9),1,13);
204 InsertBlock(~h7,h6,1,1);
```

```
h8:=h7*Transpose(GeneratorMatrix(Dual(C1)));
InsertBlock(~h2, h8,i,(j-1)*13+1 );
end for;
end for;
s1,s2:=Solution(h2,h1);
// Following part Solve the system of Equations in Z_15
h9:=ZeroMatrix(IntegerRing(25),L,s*13); // Calculate [Databese*(E+U)]*
    Dual(C2)] C2 is the second component code of Cin in Z_15
h10:=ZeroMatrix(IntegerRing(25),1,s*13); // Calculate U*Dual(C1) in
    Z_15 then solve equation h1 and h2 to find the desired file
for i in [1..1] do
for j in [1..1] do
h11:=Vector(IntegerRing(25),Coefficients(Database_EU [i][1+j-1]));
h12:=ZeroMatrix(IntegerRing(25),1,13);
InsertBlock(~h12,h11,1,1);
h13:=h12* Transpose(GeneratorMatrix(Dual(C2)));
InsertBlock(~h9, h13,i,(j-1)*13+1 );
end for;
end for;
for j in [1..1] do
for i in [1..1] do
h14:=Vector(IntegerRing(25),Coefficients(U[((d-1)*s)+i][1+(j-1)]));
h15:=ZeroMatrix(IntegerRing(25),1,13);
InsertBlock(~h15,h14,1,1);
h16:=h15*Transpose(GeneratorMatrix(Dual(C2)));
InsertBlock(~h10, h16,i,(j-1)*13+1 );
end for;
end for;
s3,s4:=Solution(h10,h9);
Desired_File:=CRT([Matrix(IntegerRing(),s1),Matrix(IntegerRing(),s3)
    ],[9,25])[1][1] mod 15;   // Apply CRT to solution in Z_9 and Z_15
Desired_File;
Database[1][1];
```

A.3 Quantum CSS-T Code Construction from WRM Code

The following MAGMA implementation corresponds to the construction of a quantum CSS-T code based on weighted Reed-Muller codes, as described in Section 4.1.1. In particular, it generates the code given in the first row of Table 4.1.

```
q := 2; // Field GF(q)
m := 7; // Number of variables
s := 5;
weights := [1,2,2,2,2,2,2]; // Weights

// Constructing the monomial set
function GenerateMonomials(q, s, m, weights)
    B := []; // Monomial set
    for I in CartesianProduct([[0..q-1] : i in [1..m]]) do
        if &+[weights[j] * I[j] : j in [1..m]] le s then
```

```
            Append(~B, [I[k] : k in [1..m]]);
        end if;
    end for;
    return B;
end function;

// Constructing the evaluation points
function GenerateEvaluationPoints(q, m)
    P := []; // Set of points
    for P_set in CartesianProduct([[0..q-1] : i in [1..m]]) do
        Append(~P, [P_set[j] : j in [1..m]]);
    end for;
    return P;
end function;

// Constructing the generator matrix
function GenerateCodeMatrix(B, P, q)
    M := ZeroMatrix(GF(q), #B, #P);
    for i in [1..#B] do
        for j in [1..#P] do
            value := &*[P[j][k]^B[i][k] : k in [1..#B[i]]];
            M[i][j] := value;
        end for;
    end for;
    return M;
end function;

// Generating the codes
B := GenerateMonomials(q, s, m, weights);
P := GenerateEvaluationPoints(q, m);
M := GenerateCodeMatrix(B, P, q);
C1 := LinearCode(M); //WRM(s,m,weights)
D1:=Dual(C1);
G1:=GeneratorMatrix(C1);
R1:= Rank(G1);

// Computing the star product
M1:=ZeroMatrix(GF(q), R1* R1, q^m);
for j in [1.. q^m] do
for i in [1.. R1] do
 for k in [1.. R1] do
M1[k+((i-1)* R1)][j]:= G1[i][j]*G1[k][j];
end for;
end for;
end for;

sqC1:=LinearCode(M1); //Square of C1: C1\star C1
sqD1:=Dual(sqC1); // Dual(C1*C2)
G11:=GeneratorMatrix(sqC1); //Generator Matrix of : C1\star C1

RM1:=ReedMullerCode(1,m);  // First-order binary Reed-Muller code of
```

```
    length 2^m
62
63 (RM1 meet sqD1) eq RM1; //Check the condition C2 \subseteq Dual( C1\
    star C1)
64 Dimension(RM1); //Dimension of RM(1,m)
65 Dimension(C1)- Dimension(RM1); //Dimension of quantum CSS-T code
66 MinimumDistance(Dual(RM1)); //Minimum Distance of quantum CSS-T code
```

A.4 Quantum CSS-T Code Construction from J-Affine Code

The following MAGMA code is used for the construction of CSS-T codes from the subfield subcode of a J-affine variety code, as explained in Section 4.1.2. Specifically, it produces the code parameters corresponding to the third row of Table 4.2.

```
1  q := 128; // q = 2^7
2  F := GF(q); // Finite field GF(q)
3  s1 := 128;
4  s2 := 2;
5
6  // Define polynomial ring for bivariate affine variety
7  R<x, y> := PolynomialRing(F, 2);
8
9  // Define affine variety using ideals
10 I := ideal<R | x^s1 - x, y^s2 - y>;
11 V := Variety(I); // Affine variety
12
13 // Collect points for the code
14 points := {@ [p[1], p[2]] : p in V @}; // Gather affine variety points
15
16 // Function to convert Galois elements to integers
17 ToInteger := function(el, maxValue, isX)
18     if el eq 0 then
19         return 0; // Directly return 0 for zero
20     elif isX then
21         return Log(F.1, el) mod (maxValue-1); // Transformation for x (
               mod s1-1)
22     else
23         return (Log(F.1, el) mod (maxValue)); // Transformation for y (
               mod s2-1)
24     end if;
25 end function;
26
27 // Function to convert a point to integer representation
28 ConvertPoint := function(point)
29     x_val := ToInteger(point[1], s1, true);
30     y_val := ToInteger(point[2], s2, false);
31     if x_val eq 0 and point[1] ne 0 then
32         x_val := s1-1; // Handle special case for x
33     end if;
34     if y_val eq 0 and point[2] ne 0 then
```

```
        y_val := s2-1; // Handle special case for y
    end if;
    return [x_val, y_val];
end function;

// Function to generate  cyclotomic cosets
CyclotomicCosets := function(points)
    cosets := [];
    seen := [];
    for p in points do
        if p notin seen then
            // Compute cyclotomic coset
            coset := {@
                [p[1]^(2^k), p[2]^(2^k)] : k in [0..#points-1]
            @};
            // Add converted coset
            transformed_coset := {@ ConvertPoint(c) : c in coset @};
            Append(~cosets, transformed_coset);
            seen cat:= SetToSequence(coset);
        end if;
    end for;
    return cosets;
end function;

// Function to compute union of cyclotomic cosets
UnionOfCosets := function(cosets, indices)
    union_set := {@ @}; // Create empty set
    for idx in indices do
        union_set join:= cosets[idx]; // Merge specified cosets
    end for;
    return union_set;
end function;

// Compute cyclotomic cosets
cosets := CyclotomicCosets(points);

// Print cyclotomic cosets
printf "\nTwo-dimensional␣cyclotomic␣cosets␣(as␣integers):\n";
for idx in [1..#cosets] do
    printf "Coset␣%o:\n", idx;
    for p in cosets[idx] do
        printf "%o\n", p;
    end for;
end for;

// Example: Take union of Cosets
union_result1 := UnionOfCosets(cosets, [1,2,5,7,6,9]);
printf "\nUnion␣of␣Coset␣1␣and␣Coset␣2:\n";
for p in union_result1 do
    printf "%o\n", p;
end for;
```

```
86
87  // Find roots of polynomials in splitting fields
88  P1<x> := PolynomialRing(GF(q));
89  F1 := x^s1 - x;
90  r, S<w> := RootsInSplittingField(F1);
91  P2<y> := PolynomialRing(GF(q));
92  F2 := y^s2 - y;
93  r2, S1<v> := RootsInSplittingField(F2);
94
95  // Create evaluation points
96  P1 := [[r[i][1], r2[j][1]] : i in [1..#r], j in [1..#r2]];
97  F := union_result1;
98
99  // Construct evaluation matrix for first code
100 M1 := ZeroMatrix(GF(q), #F, #P1);
101 for i in [1..#F] do
102     for j in [1..#P1] do
103         M1[i][j] := (P1[j][1]^F[i][1]) * (P1[j][2]^F[i][2]);
104     end for;
105 end for;
106 C2 := SubfieldSubcode(LinearCode(M1)); // Create subfield subcode
107 D2 := Dual(C2); // Compute dual code
108 G2 := GeneratorMatrix(C2); // Get generator matrix
109 Rankc2 := Rank(G2); // Compute rank
110
111 // Example: Take union of more cosets
112 union_result2 := UnionOfCosets(cosets, [1,2,5,7,6,8,9,10,13,14]);
113 printf "\nUnion␣of␣Coset␣1␣and␣Coset␣2:\n";
114 for p in union_result2 do
115     printf "%o\n", p;
116 end for;
117 F1 := union_result2;
118
119 // Construct evaluation matrix for second code
120 M2 := ZeroMatrix(GF(q), #F1, #P1);
121 for i in [1..#F1] do
122     for j in [1..#P1] do
123         M2[i][j] := (P1[j][1]^F1[i][1]) * (P1[j][2]^F1[i][2]);
124     end for;
125 end for;
126 C1 := SubfieldSubcode(LinearCode(M2)); // Create subfield subcode
127 D1 := Dual(C1); // Compute dual code
128 G1 := GeneratorMatrix(C1); // Get generator matrix
129 Rankc1 := Rank(G1); // Compute rank
130
131 // Construct matrix for star product code (C1 \star C1)
132 M3 := ZeroMatrix(GF(2), Rankc1 * Rankc1, #P1);
133 for j in [1..#P1] do
134     for i in [1..Rankc1] do
135         for k in [1..Rankc1] do
136             M3[k + ((i-1)*Rankc1)][j] := G1[i][j] * G1[k][j];
```

```
        end for;
    end for;
end for;
C1sq := LinearCode(M3); // (C1 \star C1)
D1sq := Dual(C1sq); // Compute dual
G3 := GeneratorMatrix(C1sq); // Get generator matrix
Rankc3 := Rank(G3); // Compute rank

// Test quantum css-t code relationships
(C2 meet C1) eq C2;
(C2 meet (C1 meet D1sq)) eq C2;


// Compute dimension of quantum css-t code
Dimension(C1) - Dimension(C2);
// Compute minimum distance of quantum css-t code
MinimumDistance(D2);
```

A.5 PIR from Hyperbolic Codes

For the construction presented in Section 4.2.1, the corresponding MAGMA code is given below. It generates a PIR scheme from hyperbolic codes and produces the code parameters shown in the second row of Table 4.4.

```
q := 7;  // Field size (GF(q))
s1 := 1;
s2 := 5;
k := 2;

// Function to generate a set of points based on a given condition
function GenerateSet(q, d, k, condition)
    B := [];
    a := CartesianPower([0..q-1], k); // All k-tuples over [0..q-1]
    for b in a do
        if condition(b, d, q) then
            Include(~B, b); // Include point if condition is satisfied
        end if;
    end for;
    return B;
end function;

// Condition for hyperbolic code
function ConditionB1(b, d, q)
    return &*[q - b[i] : i in [1..#b]] ge d; // Product condition
end function;

// Condition for dual hyperbolic code
function ConditionB2(b, d, q)
    return &*[(b[i] + 1) : i in [1..#b]] lt d; // Product condition
end function;
```

```
27
28 // Condition for  Reed-Muller code
29 function ConditionB3(b, d, q)
30     return &+[b[i] : i in [1..#b]] le d; // Sum condition
31 end function;
32
33 // Function to generate all evaluation points in the space
34 function GeneratePoints(q, k)
35     P := [];
36     a := CartesianPower([0..q-1], k); // All k-tuples over [0..q-1]
37     for point in a do
38         Include(~P, point); // Include all points
39     end for;
40     return P;
41 end function;
42
43 // Function to generate evaluation matrix for a code
44 function GenerateMatrix(q, B, P)
45     numRows := #B;    // Number of monomials
46     numCols := #P;    // Number of evaluation points
47     M := ZeroMatrix(GF(q), numRows, numCols);
48     for i in [1..numRows] do
49         for j in [1..numCols] do
50             product := 1;
51             for dim in [1..#B[1]] do  // For each dimension
52                 product *:= (P[j][dim] ^ B[i][dim]); // Evaluate
                       monomial
53             end for;
54             M[i][j] := product; // Store evaluation
55         end for;
56     end for;
57     return M;
58 end function;
59
60 // Function to generate star product (Schur product) of two codes
61 function GenerateStarProduct(G1, G2, q)
62     R1 := Nrows(G1); // Rows of first matrix
63     R2 := Nrows(G2); // Rows of second matrix
64     M3 := ZeroMatrix(GF(q), R1 * R2, Ncols(G1));
65     for j in [1..Ncols(G1)] do
66         for i in [1..R1] do
67             for kk in [1..R2] do
68                 // Compute product of corresponding entries
69                 M3[kk + ((i - 1) * R2)][j] := G1[i][j] * G2[kk][j];
70             end for;
71         end for;
72     end for;
73     return M3;
74 end function;
75
76 // Function to compute footprint bound (for code parameters)
```

```
function FootprintBound(B, q)
    FB := Minimum({&*[(q - B[i][dim]) : dim in [1..#B[1]]] : i in [1..#
        B]});
    return FB;
end function;

// Generate sets using different conditions
B := GenerateSet(q, s1, k, ConditionB3);  // Reed-Muller set
B2 := GenerateSet(q, s2, k, ConditionB2); // Dual hyperbolic set
B1 := GenerateSet(q, s2, k, ConditionB1); // Hyperbolic set

// Generate evaluation points
P := GeneratePoints(q, k);

// Generate evaluation matrices
M1 := GenerateMatrix(q, B, P);  // For Reed-Muller code
M2 := GenerateMatrix(q, B2, P); // For dual hyperbolic code

// Construct codes
C1 := LinearCode(M1); // Reed-Muller code RM_q(s1, 2)
C2 := LinearCode(M2); // Dual hyperbolic code Dual(Hyp_q(s2,2))

// Compute dual codes
D1 := Dual(C1);
D2 := Dual(C2);

// Get generator matrices
G1 := GeneratorMatrix(C1);
G2 := GeneratorMatrix(C2);

// Compute star product code
M3 := GenerateStarProduct(G1, G2, q);
C1C2 := LinearCode(M3);  // Product code C1\star C2
D1D2 := Dual(C1C2);      // Dual of star product code

// Compute footprint bound
FB1 := FootprintBound(B1, q);

// Output important parameters
Dimension(C2);
Dimension(Dual(C1C2));
FB1;
```

A.6 PIR from Subfield Subcode of J-affine Code

The following MAGMA code, based on Section 4.2.2.1, generates the code listed in the second row of Table 4.6 using subfield subcodes of J-affine variety codes in one variable.

```
q := 49; // Field size GF(49) = 7^2
// Define polynomial ring and cyclotomic polynomial
```

```
P1<x> := PolynomialRing(GF(q));
F1 := x^48 - 1; // Cyclotomic polynomial
r := Roots(F1); // Find  roots

P1 := [[r[i]] : i in [1..#r]]; // Evaluation Points

// First exponent set for code C1
F := [24, 25, 31];

// Construct evaluation matrix for C1
M1 := ZeroMatrix(GF(q), #F, #P1);
for i in [1..#F] do
    for j in [1..#P1] do
        M1[i][j] := (P1[j][1][1]^F[i]); // Evaluate monomial at root
    end for;
end for;

// Create code C1 and its dual
C1 := SubfieldSubcode(LinearCode(M1));  // Subfield Subcode of
   evaluation code using exponents F
D1 := Dual(C1);         // Dual code
G1 := GeneratorMatrix(C1);
Rankc1 := Rank(G1);     // Dimension of C1

// Second exponent set for code C2
F2 := [24, 25, 31, 32];

// Construct evaluation matrix for C2
M2 := ZeroMatrix(GF(q), #F2, #P1);
for i in [1..#F2] do
    for j in [1..#P1] do
        M2[i][j] := (P1[j][1][1]^F2[i]); // Evaluate monomials
    end for;
end for;

// Create code C2 and its dual
C2 := SubfieldSubcode(LinearCode(M2));  // Subfield Subcode of
   evaluation code using exponents F2
D2 := Dual(C2);         // Dual code
G2 := GeneratorMatrix(C2);
Rankc2 := Rank(G2);     // Dimension of C2

// Construct star product code Cin = C1 \star C2 (Schur product)
M3 := ZeroMatrix(GF(7), Rankc1 * Rankc2, 48);
for j in [1..48] do
    for i in [1..Rankc1] do
        for k in [1..Rankc2] do
            // Compute product of generator matrix elements
            M3[k + ((i-1)*Rankc2)][j] := G1[i][j] * G2[k][j];
        end for;
    end for;
```

```
end for;

// Star product code and its dual
Cin := LinearCode(M3);  // Schur product code C1 \star C2
Din := Dual(Cin);       // Dual of star product code

// Code analysis
Cin; // Star product code
BCHBound(D2); // Compute BCH bound for dual code D2
```