Fault tolerance in cryptographic applications using cover-free families

by

Thais Bardini Idalino

Thesis submitted in partial fulfillment of the requirements for the Doctorate in Philosophy degree in Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Cryptography is of fundamental importance to guarantee the security of communications, from the point-to-point transmission of messages and documents to the storage of big sets of data on cloud providers. As an example, we can use encryption, message authentication codes, and digital signatures to help us guarantee the privacy, integrity, and authenticity of data that is stored and/or transmitted. Many cryptographical solutions have a Boolean outcome, for example, the message is either authentic and accepted as it is, or it is not and so it needs to be rejected/re-transmitted. This outcome may be acceptable in scenarios where we can easily re-transmit the message, but it can pose a challenge when we deal with a large amount of data or a more sensitive content in which changes need to be further explored. In this context, this thesis proposes solutions to provide fault tolerance to certain cryptographic problems that traditionally have an all-or-nothing outcome.

Fault tolerance is application dependent. In the case of a digital signature of a document that has been later modified, a fault-tolerant scheme can ensure authenticity and further identify which parts of the document were modified. This approach can be used in data forensics to investigate cybercrime, or to create redactable signatures for the purpose of privacy. In the case of aggregation of signatures, we consider an aggregation of a set of signatures containing a few invalid signatures (in the traditional sense). A fault-tolerant scheme is able to identify which signatures are valid and which ones are invalid, instead of rejecting the whole set. Digital signatures and aggregation of digital signatures require fault tolerance to be ensured at the origin (signer algorithm and aggregation algorithm, respectively), rather than just at the destination (verification algorithm). For this reason, we focus on techniques from combinatorial group testing that are nonadaptive rather than adaptive. This is in contrast with other applications of group testing, such as batch verification of signatures, employed at the verifier's end which allow both adaptive and nonadaptive solutions.

In this work, we explore solutions for fault tolerance using techniques of identification of defective elements used in nonadaptive combinatorial group testing. More specifically, we use the well studied cover-free families (CFFs). A d-cover-free family d-CFF(t, n) is a set system with n subsets of a t-set, where the union of any d subsets does not contain any other. A d-CFF(t, n) allows for the identification of up to d defective elements in a set of n elements by performing only t tests (typically $t \ll n$). In the literature, CFFs are used to solve many problems in cryptography. In this work, we explore different aspects of cover-free families in order to better approach fault tolerance problems.

The problems we investigate can be divided into two categories: *static problems* (fixed size) and *dynamic problems* (increasing size). In the context of static problems, we consider *modification-tolerant digital signature schemes*, which allow the identification of modifications in signed data using a d-CFF, and in some cases the correction of such modifications in order to retrieve the originally signed data. We also propose a generalization of the classical definition of a d-CFF to support *variable* cover-free property, followed by some constructions and potential applications in cryptography. For dynamic problems, we consider the application of *fault-tolerant aggregation of signatures*. This problem requires an

infinite sequence of CFFs with increasing n, consequently increasing t, and potentially increasing d. In this context, we investigate *monotone*, *nested*, and *embedding* sequences of CFFs, and propose constructions using techniques from combinatorial design theory and finite fields. In constructing these families, we are concerned with their *compression ratio*. The compression ratio of a sequence of CFFs measures how slowly the number of tests grow with respect to the number of elements to be tested, which affects the overall efficiency of the method. We provide constructions of CFF sequences with different compression ratios, which depend on relations among the CFF parameters in the sequence and the type of sequence. Some of these sequences achieve the best possible compression ratio, which meets the best known upper bound. Monotone, nested and embedding sequences of CFFs can be used in any group testing problem that is dynamic in nature. We discuss various cryptographical applications that can benefit from these infinite sequences of CFFs.

Acknowledgements

For making this journey an enjoyable one, I need to thank many people. Here is my attempt to put into words what they meant to me.

I cannot thank my supervisor Lucia Moura enough. I was lucky to meet her during my master's and to be introduced to such fascinating research topics. She has dedicated so many hours into this research and spent some late nights and vacations reviewing papers. I am forever grateful for her guidance, encouragement, enthusiasm, and patience, and for sharing her knowledge and great ideas with me. I sure learned much more than Computer Science and Mathematics, and feel very fortunate to have such a great supervisor.

The constant support and company of my partner Felipe were fundamental for the success of this journey. He made the hard times much easier, made my days much more fun, and I am forever grateful for sharing this adventure with him. I am not sure I would succeed without you by my side, so in some way, this diploma is also yours.

For transforming these four years of Canada into such an enjoyable time, I thank the friends I met here, which became my family far from home. I will always cherish our moments together, and I hope to repeat them many times more in the future.

Preciso agradecer também a todos que torceram por mim à distância, entenderam minha ausência e sempre acreditaram na minha capacidade. Primeiramente, agradeço meus pais, Antenor e Sandra, por me incentivarem desde criança a dar o melhor de mim em tudo o que faço, por serem fonte de inspiração e exemplo, e por terem proporcionado todo o apoio necessário para que eu chegasse até aqui. Agradeço também aos meus avós, que são exemplo de determinação, trabalho duro e honestidade, além de uma fonte inesgotável de comida deliciosa e histórias incríveis. Queria extender o agradecimento também ao restante da família, que sempre me apoiou e torceu por mim, me considero uma pessoa de muita sorte por ter vocês.

Também gostaria de agradecer minhas amigas, Bárbara, Carla, Karen, Marílis e Renata. Que sorte a minha ter vocês ao meu lado, mesmo estando fisicamente tão longe. É maravilhoso saber que posso contar com essa amizade, que vem desde a infância, pra qualquer coisa. O apoio de vocês sempre foi fundamental, obrigada por tudo.

For the financial support that allowed me to conduct my studies under ideal conditions, I thank the Brazilian National Council for Scientific and Technological Development (CNPq), the Ontario Graduate Scholarship (OGS) program, my supervisor for supporting my attendance in conferences, and the University of Ottawa for the invaluable opportunity of being a teaching assistant.

I would also like to thank the professors that I met at the University of Ottawa and at Carleton University. I was fortunate to take courses with amazing and inspiring professors, specially Daniel Panario, Carlisle Adams, and Lucia Moura. I am constantly blown away by how much I have learned in these past four years, thank you so much for sharing your time and knowledge with me, and for being a source of inspiration in my career. Finally, for their time and insights, I need to thank my thesis examiners: Douglas Stinson, Brett Stevens, Carlisle Adams, and Vida Dujmović. This thesis was very much improved due to their careful read and feedback.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cryptography is widely studied with the objective of guaranteeing the security of digital information and communications in the presence of adversaries. Digital signatures are among the most powerful techniques to guarantee that. When sending a digitally signed message, by verifying the signature we can, with very high probability, guarantee the origin of the message (authenticity), guarantee that the message has not been modified after the signature (integrity), and also prevent the signer from denying authorship (non-repudiation). They require *asymmetric* cryptography, where a *private key* is used to generate the digital signature, and the corresponding *public key* is used to verify the signature [18]. In applications that handle a large set of data and digital signatures, an *aggregate signature scheme* can be used to combine all signatures together into one aggregate signature, creating one short proof of integrity and authenticity for the entire set of corresponding data [1]. In this way, we can save on communication, storage and verification time.

In a traditional digital signature scheme, a verification algorithm has a Boolean outcome: the message is either authentic and not modified, or it is not and so needs to be rejected/re-transmitted. This is an acceptable outcome when the data can be easily retransmitted, but it poses a challenge for the scenarios where we deal with large amounts of data; for the cases where sensitive contents were modified and should be further explored in an investigation; or even when the modification was expected. For traditional schemes of aggregation of signatures, the verification outputs a positive result only when the entire set of signatures is valid; an invalid signature invalidates the entire aggregate, because when a set of signatures is aggregated into one, this operation does not preserve enough information in order to identify the exact set of invalid signatures. In this thesis, we are interested in providing efficient fault tolerance to certain problems in cryptography that have an "all-or-nothing" flavor. We explore solutions to provide fault tolerance to these problems using techniques of identification of invalid elements.

Identifying invalid elements from a set of n elements is an important problem found in many areas, from biology and chemistry to engineering and computer science. Instead of testing each element individually, we can use *combinatorial group testing* (CGT) as a more efficient way of identifying up to d invalid elements [2, 3, 16, 20]. CGT was proposed as an efficient solution for identifying people infected with syphilis during the second World War

[2, 3]. Given n elements and a number d of invalid elements that the scheme can tolerate, we group the elements into t groups ($t \ll n$), and test the groups instead of each element individually. A group that passes the test contains only valid elements and a group that fails contains at least one invalid element. From the test results of various groups, we can identify which elements are invalid and which ones are valid. There are two types of CGT algorithms: *adaptive* and *nonadaptive*. With adaptive algorithms, the next groups/tests are decided according to the results of previous tests; nonadaptive algorithms decide all groups/tests at once. In this work, we consider problems that require the groups to be decided a priori, therefore we use nonadaptive CGT algorithms [3].

We can represent the elements and groups of a nonadaptive CGT scheme using *d-cover-free families* (d-CFFs). A d-CFF(t, n) can be defined as a set system $(X, \mathcal{B})$, where X is a set of cardinality t, and $\mathcal{B}$ is a collection of n subsets of X such that no subset is contained in the union of any other d subsets. For the case $d = 1$, a set system with this property is equivalent to a Sperner system [21]. Sperner's theorem [21] establishes the upper bound $|\mathcal{B}| \leq \binom{|X|}{\lfloor |X|/2 \rfloor}$, which is achieved when we take all $\lfloor \frac{|X|}{2} \rfloor$-subsets of X. For example, let $d = 1$, $X = \{1, 2, 3, 4, 5\}$, and $\mathcal{B} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$ be all unique 2-subsets of X; since no subset contains any other, this is a 1-CFF$(5, 10)$. We can easily solve a CGT problem using a d-CFF(t, n) incidence matrix $\mathcal{M}$: the incidence vector of each subset forms a column of $\mathcal{M}$, the columns represent the elements to be tested, and each one of the t rows indicates which elements are grouped together. Figure 1.1 shows the incidence matrix of the example above. Therefore, we are interested in minimizing t in order to perform fewer tests. When $d = 1$, $t \approx \log n$ is achieved using Sperner's theorem [21]. For $d \geq 2$, the lower bound on t is known to be $t \geq c\frac{d^2}{\log d} \log n$, for a constant c [5]. Several problems in cryptography can be solved using d-CFFs, such as to construct one-time and multiple-times digital signature schemes [17, 22], to construct broadcast communication schemes [6, 19], to provide fault tolerance in aggregation of signatures [8, 9], and to provide localization of modifications in signed documents [7, 10].

In this thesis, the problems we investigate can be divided into two categories: *dynamic problems*, where the number of elements increases over time; and *static problems*, that have a fixed number of elements. Our main research focus is to create new definitions and constructions of CFFs in order to better approach these problems. We now present the details of our main contributions.

1.1 Research contributions

1.1.1 Dynamic problems

Our major contributions in this thesis are related to dynamic problems. For this scenario, we consider the problem of *fault-tolerant aggregation of signatures*. Solutions to this problem using d-CFFs have been proposed independently in [8, 9]. Instead of aggregating all signatures into one, the idea is to use a d-CFF matrix to compute a more expressive aggregate, which provides enough information to locate up to d invalid signatures. The columns

$$\mathcal{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \rightarrow \begin{array}{l} \tau[1] = Agg(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \\ \tau[2] = Agg(\sigma_1, \sigma_5, \sigma_6, \sigma_7) \\ \tau[3] = Agg(\sigma_2, \sigma_5, \sigma_8, \sigma_9) \\ \tau[4] = Agg(\sigma_3, \sigma_6, \sigma_8, \sigma_{10}) \\ \tau[5] = Agg(\sigma_4, \sigma_7, \sigma_9, \sigma_{10}) \end{array}$$

Figure 1.1: Aggregation of signatures based on a 1-CFF$(5, 10)$.

of the d-CFF represent each one of the digital signatures to be aggregated, and the rows identify each of the aggregates. So we have a total of t aggregations, instead of just one, but we are able to identify up to d invalid signatures with them. For example, consider $n = 10$ signatures $\sigma_1, \ldots, \sigma_{10}$ and the 1-CFF$(5, 10)$ matrix $\mathcal{M}$ in Figure 1.1. The columns of $\mathcal{M}$ represent the individual signatures, the aggregations are done according to the rows of $\mathcal{M}$, and we create the aggregate signature τ of size $t = 5$. The use of a 1-CFF allows us to identify all valid signatures as long as there is at most one invalid signature. For example, if σ_1 is invalid, $\tau[1]$ and $\tau[2]$ will fail the verification, but $\tau[3], \tau[4], \tau[5]$ prove the validity of $\sigma_2, \sigma_3, \ldots, \sigma_{10}$.

The use of d-CFFs is a practical solution for the identification of up to d invalid signatures in an aggregation scheme, but by fixing a d-CFF(t, n), the number of signatures that can be aggregated is bounded by the number n of columns. This number is not always known in advance; for instance, a traditional aggregate signature scheme [1] does not set such a bound, as a new signature can always be combined into an existing aggregation, but also does not allow the identification of invalid signatures. This application requires an *unbounded fault-tolerant aggregation scheme*, which relies on d-CFFs that grow in size as the number of signatures grows, and we represent this growth as a special sequence of d-CFF matrices. In order to provide unbounded aggregation with fault tolerance, we have been exploring infinite sequences of CFFs with increasing n, consequently increasing t, and potentially increasing d. We represent the sequence of incidence matrices of these CFFs as $\mathcal{M}^{(1)}, \mathcal{M}^{(2)}, \mathcal{M}^{(3)}, \ldots$, where we use a CFF until the number of columns is reached, then we "jump" to the next one in the sequence. To solve the problem, these sequences should have the following characteristics:

1. The sequence is formed by CFFs of increasing size, and the previous matrix $\mathcal{M}^{(l)}$ in the sequence must be a submatrix of the next one, $\mathcal{M}^{(l+1)}$, in order to reuse aggregations already computed in $\mathcal{M}^{(l)}$.

2. These sequences should take into consideration that, after aggregation, individual signatures may be discarded to save storage.

In constructing these sequences, we are concerned with their *compression ratio* as n grows; we say the compression ratio is $\rho(n)$ if and only if $\frac{n}{t}$ is $\Theta(\rho(n))$. The compression ratio of a sequence of CFFs measures how slowly the number of tests grows with respect to the number of elements to be tested, which affects the overall efficiency of the method. Hartung et al. [8] proposed a *monotone* sequence of d-CFFs to provide an unbounded

fault-tolerant aggregation scheme, but the construction they proposed has a constant compression ratio. One of the contributions of this thesis is the definition of a more flexible sequence of d-CFFs that can be used to solve this problem, which we call *nested cover-free family*. We show that with nested families we can achieve all the requirements above, and the constructions we proposed have very good compression ratio, much better than the one in [8]. We propose explicit constructions of nested families, and for the specific case of $d = 1$ our construction gives the best compression ratio possible, meeting the information theoretical upper bound of $\frac{n}{\log n}$. For $d \geq 2$ we propose two constructions, the first with ratio $\rho(n) = n^{1-1/c}$, for some constant $c \geq 2$, and the second with $\rho(n) = \frac{n}{(b \log_2 n)^{\log_2 \log_2 n + D}}$, for constants b and D. In order to construct nested families, we propose new recursive constructions of d-CFFs [11, 14].

Now suppose that communication errors happened and some of the t pieces of aggregate signature were lost/corrupted during transmission. This loss would probably compromise the verification process, where some good digital signatures may be considered invalid. In order to approach this problem, we can ensure there are enough valid pieces of the aggregate signature to identify all valid individual signatures. We use a known generalization of d-CFFs called $(d; \lambda)$-CFFs to provide an extra level of fault tolerance. With $(d; \lambda)$-CFFs, every valid individual signature is aggregated in at least λ valid aggregates, so we can handle the losses or corruptions of up to $\lambda - 1$ aggregates without compromising the identification of any valid signatures. We also propose recursive constructions of $(d; \lambda)$-CFFs and use them to obtain nested $(d; \lambda)$-cover free family constructions [14].

Because of the second requirement of this problem, where previous individual signatures can be discarded after aggregation, we are restricted to use monotone or nested families, which can only be constructed for fixed d. However, for any application where the number n of elements grows, it is natural to imagine that the number of invalid elements d will also grow. Therefore, we propose a generalization of monotone and nested families, which we call *embedding cover-free families*. Embedding families allow for the increase of both d and n, and can be applied in any dynamic problems that do not pose such a requirement, for example, general group testing problems, broadcast encryption and broadcast authentication [6, 19]. We propose constructions of embedding families using a known d-CFF construction via polynomials over finite fields [4], where our sequences are created via extension fields [12]. We also show how different parameter combinations in this construction can provide different results, such as to prioritize increases of d, or prioritize the compression ratio as n grows. Some of these sequences achieve the best possible compression ratio, which meets the information theoretical upper bound [12].

An important step to create these constructions was to identify a special structure of the polynomial construction of d-CFFs by Erdös, Frankl and Füredi [4]. This structure allows us to discard specific blocks of rows of the d-CFF and obtain a d'-CFF with a smaller number of rows, $d' < d$. With this structure of blocks, we were able to provide a construction of monotone families with improved compression ratio, improving the results from [8]. In addition, we look at embedding CFF constructions in terms of packing arrays and separating hash families, and identify that similar characteristics of blocks of rows occur in these cases as well [12].

It is important to notice that monotone, nested and embedding sequences of CFFs can not only be used for cryptographical applications, but also for any group testing problem that is dynamic in nature.

1.1.2 Static problems

In the context of static problems, we consider *modification-tolerant signature schemes* (MTSS). They are a more expressive digital signature that goes beyond the "all-or-nothing", and provide enough information to allow the location and correction of modifications in signed data by using d-CFFs. The use of d-CFFs to locate modifications in digital data is presented in [7], where the authors investigate modifications in data stored in several data structures and use this information for forensics investigation. A similar approach is applied for digital signature schemes in [10], where d-CFFs are used to provide the location of modifications for digital signatures. In this scheme, the data to be signed is divided into n blocks, and the structure of the d-CFF will indicate how we should combine the hashes of these blocks in order to form a more expressive digital signature. On the verification side, the extra hashes are used to locate up to d modified blocks.

In this thesis, we define a general framework for modification-tolerant signature schemes. We also extend the solution from [10] using d-CFFs to provide correction of modifications for the cases where the blocks are small enough. In addition, we propose a specific MTSS for redactable signatures using d-CFFs, where pieces of the message are modified on purpose with the intention of hiding private information. In this sense, we guarantee privacy for redactable signatures by proposing a redaction algorithm that prevents leakage of redacted information, while allowing the verification of integrity and authenticity of the rest of the data [13].

In the context of static problems, we also identify the need for a new type of CFF that we call $\mathcal{S}$-CFF, or *variable CFF*. With $\mathcal{S}$-CFF, the cover-free property is explicitly defined by the set $\mathcal{S}$ instead of required for every combination of $d+1$ subsets/columns. One possible application is the MTSS problem, where we may have a concentration of modifications in a few regions of the document, and in each of these regions we want to perform a more detailed verification. The use of a variable CFF may provide a smaller signature than using a traditional d-CFF with large enough d. We define these new families, propose constructions, and mention possible applications of $\mathcal{S}$-CFFs [15].

1.1.3 Articles produced in this thesis

This thesis is in a "thesis by articles" format, and the chapters consist of papers published in journals and conference proceedings, or papers being prepared for submission.

- Chapter 2 consists of a survey on cover-free families that includes the new generalization of d-CFFs to variable CFFs, which is being prepared for submission.

  IDALINO, T. B.; MOURA, L., *Cover-free families and generalizations*. In preparation for submission, 2019.

- A shorter version of Chapter 3 appeared in the proceedings of the 29th International Workshop on Combinatorial Algorithms, IWOCA 2018. This work was invited for submission to the special issue of the journal *Theoretical Computer Science* dedicated to this conference, and the complete paper was submitted in October 2018. The literal content of this paper is in Chapter 3.

  IDALINO, T. B.; MOURA, L., *Efficient Unbounded Fault-Tolerant Aggregate Signatures Using Nested Cover-Free Families*. In: International Workshop on Combinatorial Algorithms, IWOCA 2018. Lecture Notes in Computer Science, vol 10979, pages 52–64. Springer, Cham.

  IDALINO, T. B.; MOURA, L., *Nested Cover-Free Families for Unbounded Fault-Tolerant Aggregate Signatures*. Submitted to Theoretical Computer Science, October 2018.

- Chapter 4 was published in a special issue of the journal *Advances in Mathematics of Communications*, dedicated to *Applications of Discrete Mathematics in Secure Communication* in honour of Prof. Bimal Kumar Roy.

  IDALINO, T. B.; MOURA, L., *Embedding cover-free families and cryptographical applications*. Advances in Mathematics of Communications, **13** (2019), 629–643.

- Chapter 5 consists of a paper being prepared for submission.

  IDALINO, T. B.; MOURA, L., ADAMS, C., *Modification tolerant signature schemes: location and correction*. In preparation for submission, 2019.

1.2 Thesis outline

- **Chapter 2: Cover-free families and generalizations**. In this chapter, we present a review of the literature on cover-free families. We discuss different ways of defining CFFs, from representations with set systems to incidence matrices and corresponding properties. We review related and equivalent objects and constructions, and highlight the relationships between them. In this chapter, we also define variable CFFs and present some constructions. We also present generalizations of recursive constructions of CFFs (originally proposed in Chapters 3 and 4) by considering them a special case of variable CFFs. Finally, we review some applications of d-CFFs in cryptography.

- **Chapter 3: Efficient Unbounded Fault-Tolerant Aggregate Signatures Using Nested Cover-Free Families**. In this chapter, we present a solution for the problem of unbounded fault-tolerant aggregation of signatures. We define nested cover-free families and propose constructions with good compression ratio. We also propose algorithms for aggregation of signatures using nested families, which allows for the dynamic growth on the number n of signatures to be aggregated, while keeping a good compression ratio between n and the size of the aggregate signature. Finally, we present a generalization on nested family constructions that allows for the correct identification of invalid signatures even if some pieces of the aggregate are lost during transmission, adding an extra layer of fault-tolerance.

- **Chapter 4: Embedding cover-free families and cryptographical applications.** In this chapter, we define embedding cover-free families as a generalization of monotone and nested families, which allows for the increase of both n and d. We present constructions of such families with good compression ratio, and for specific parameters, a construction with ratio that achieves the information theoretical upper bound. We also mention some applications in cryptography that may benefit from such CFFs.

- **Chapter 5: Modification tolerant signature schemes: location and correction.** Here we define modification-tolerant signature schemes that allow location and correction of modifications. We explore existing schemes that use d-CFFs to provide the location of modifications, and show how they can be extended to allow correction as well. We prove the security of both schemes, namely that the scheme is unforgeable under a chosen message attack. We also present a variation of MTSS to provide redactable signatures and prove that total privacy of redacted information is achieved in this case.

- **Chapter 6: General conclusion and open questions.** Finally, we present a summary of the contributions of this thesis and discuss some open questions that can lead to future research in this area.

Bibliography

[1] D. Boneh, C. Gentry, B. Lynn, H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques - EUROCRYPT, 2003*. Lecture Notes in Computer Science, vol 2656. Springer, Berlin, Heidelberg, 2003, 416–432.

[2] R. Dorfman. The Detection of Defective Members of Large Populations. *Annals of Mathematical Statistics*, 14:436–440, 1943.

[3] D-Z. Du and F. K. Hwang. *Combinatorial group testing and its applications*. World Scientific, 2000.

[4] P. Erdös, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51:79 – 89, 1985.

[5] Z. Füredi, On r-Cover-free Families. *Journal of Combinatorial Theory, Series A*, **73** (1996), 172–173.

[6] E. Gafni, J. Staddon and Y. L. Yin, Efficient Methods for Integrating Traceability and Broadcast Encryption, In: Wiener M. (eds) *Advances in Cryptology – CRYPTO 1999*. Lecture Notes in Computer Science, vol 1666, Springer, Berlin, Heidelberg, 1999, 372–387.

[7] M. T. Goodrich, M. J. Atallah, and R. Tamassia. Indexing information for data forensics. In *ACNS*, 2005.

[8] G, Hartung, B. Kaidel, A. Koch, J. Koch, and A. Rupp. Fault-tolerant aggregate signatures. In *Public-Key Cryptography – PKC 2016*, pages 331–356, 2016.

[9] T. B. Idalino. Using combinatorial group testing to solve integrity issues. Master's thesis, Universidade Federal de Santa Catarina, Brazil, 2015.

[10] T. B. Idalino, L. Moura, R. F. Custódio, and D. Panario. Locating modifications in signed data for partial data integrity. *Information Processing Letters*, 115(10):731 – 737, 2015.

[11] T.B. Idalino, L. Moura. Efficient Unbounded Fault-Tolerant Aggregate Signatures Using Nested Cover-Free Families In *International Workshop on Combinatorial Algorithms, IWOCA 2018*. Lecture Notes in Computer Science, vol 10979, pages 52–64. Springer, Cham.

[12] T.B. Idalino, L. Moura. Embedding cover-free families and cryptographical applications. *Advances in Mathematics of Communications*, **13** (2019), 629–643.

[13] T.B. Idalino, L. Moura, C. Adams. Modification tolerant signature schemes: location and correction. In preparation for submission, 2019.

[14] T.B. Idalino, L. Moura. Nested Cover-Free Families for Unbounded Fault-Tolerant Aggregate Signatures. Manuscript submitted for publication, 2018.

[15] T.B. Idalino, L. Moura. Cover-free families and generalizations. In preparation for submission, 2019.

[16] C. H. Li. A sequential method for screening experimental variables. *Journal of the American Statistical Association*, 57:455–477, 1962.

[17] J. Pieprzyk, H. Wang, and C. Xing. Multiple-time signature schemes against adaptive chosen message attacks. In *Selected Areas in Cryptography*, pages 88–100. Springer Berlin Heidelberg, 2003.

[18] R. L. Rivest, A. Shamir; L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, v. 21, n. 2, p. 120–126, 1978.

[19] R. Safavi-Naini and H. Wang, New results on multi-receiver authentication codes, In: Nyberg K. (eds) *Advances in Cryptology – EUROCRYPT 1998*. Lecture Notes in Comput. Sci., vol 1403, Springer, Berlin, Heidelberg, 1998, 527–541.

[20] M. Sobel and P. A. Groll. Group testing to eliminate efficiently all defectives in a binomial sample. *Bell System Technical Journal*, 28:1179–1252, 1959.

[21] E. Sperner. Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift*, **27** (1928), 544–548.

[22] G.M. Zaverucha and D.R. Stinson. Short one-time signatures. *Advances in Mathematics of Communications*, 5:473–488, 2011.

Chapter 2

Cover-free families and generalizations

The following chapter consists of a survey of constructions of cover-free families and preliminary results about variable CFFs, which are generalizations of CFFs introduced in the paper.

IDALINO, T. B.; MOURA, L., *Cover-free families and generalizations*. In preparation for submission, 2019.

2.1 Abstract

Cover-free families have been investigated under different names and as a solution to many problems related to combinatorial group testing. Here we review several definitions, related objects and constructions found in the literature, as well as some applications in cryptography. We present generalizations of recursive constructions and introduce the notion of *variable cover-free families*, which presents variable coverage properties.

2.2 Introduction

Cover-free families (CFFs) are usually studied in the context of *combinatorial group testing* (CGT). The origin of CGT is credited to Robert Dorfman, who proposed it as a solution to efficiently identify a few cases of syphilis among millions of blood samples during the second World War [8]. The idea consists of pooling the blood samples together in a few pools and testing the pools instead of each sample individually. The pools that have at least one infected sample will have a positive result, and the ones with no infected samples will be negative. A pool with a negative result ends up saving many tests and identifying several people that are not infected, while the ones with a positive result must be explored further to identify the infected sample(s) in them [8]. The idea ended up not being used for the blood sampling problem because the result was not accurate enough when a few blood samples were pooled together, but nowadays it is used as a solution to many interesting problems [8].

The pools should be chosen in a way that minimizes the number of tests while keeping the ability to identify the infected samples. This results in two important types of combinatorial group testing: *adaptive* and *nonadaptive* algorithms. With adaptive algorithms, the next tests are determined according to the results of the previous ones and they usually require fewer tests; while nonadaptive algorithms decide all tests at once, which allows tests to be executed in parallel [8]. In this work, we are going to focus on problems that require grouping the data a priori and executing the tests/computations later, so we explore nonadaptive algorithms.

In this context, we study cover-free families. They have interesting properties that can help us to determine which blood samples need to be pooled together in order to find all (up to d) infected people, or, in the case of our applications, to determine which elements should interact or be grouped together to solve the problems we will describe in this work. Cover-free families that can tolerate up to d invalid elements are called d-cover-free families (d-CFFs). They were first introduced by Kautz and Singleton as superimposed codes in 1964 [30], then studied in terms of set systems in 1985 by Erdös, Frankl and Füredi [9], and in 1987 they were introduced as nonadaptive combinatorial group testing by Hwang and Sós [21].

We present the definition of d-CFFs in terms of set systems. A set system $\mathcal{F} = (X, \mathcal{B})$ consists of a set X and a collection $\mathcal{B}$ of subsets of X.

Definition 2.2.1. *Let d be a positive integer. A d-cover-free family, denoted d-CFF(t, n), is a set system $\mathcal{F} = (X, \mathcal{B})$ with $|X| = t$ and $|\mathcal{B}| = n$ such that for any $d + 1$ subsets $B_{i_0}, B_{i_1}, \ldots, B_{i_d} \in \mathcal{B}$, we have*

$$|B_{i_0} \setminus \bigcup_{j=1}^{d} B_{i_j}| \geq 1. \tag{2.1}$$

For a given n and d, we are interested in constructing d-CFFs with the smallest possible t, so we define $t(d, n) = \min\{t : \exists\ d\text{-CFF}(n, t)\}$.

For $d = 1$, the subsets of a 1-CFF are equivalent to the ones from a Sperner family.

Theorem 2.2.2. *(Sperner (1928) [51]) Let $\mathcal{B}$ be a collection of subsets of $\{1, \ldots, t\}$ such that $B_1 \not\subseteq B_2$ for all distinct $B_1, B_2 \in \mathcal{B}$. Then $|\mathcal{B}| \leq \binom{t}{\lfloor t/2 \rfloor}$. Moreover, equality holds when $\mathcal{B}$ is the collection of all the $\lfloor t/2 \rfloor$-subsets of $\{1, \ldots, t\}$.*

As an example, for $n = 6$ we have $t = 4$, and every subset has size $t/2 = 2$. Therefore, $X = \{1, 2, 3, 4\}$, and $\mathcal{B} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$ is a Sperner family with maximum cardinality.

Corollary 2.2.3. *Let $n \geq 1$. Then, $t(1, n) = \min\{t : \binom{t}{\lfloor t/2 \rfloor} \geq n\}$.*

Sperner's theorem gives an optimal construction for 1-CFFs. The value t grows as $\log_2 n$ as $n \to \infty$, which meets the information theoretical lower bound on the number of bits necessary to uniquely distinguish the n inputs. In the sense of information theory, for general d, we need at least $d \log(n/d)$ tests in order to locate up to d invalid elements from a set of n elements. The reasoning behind this lower bound comes from the amount of bits needed to distinguish the n inputs and identify the up to d invalid elements. We have $\sum_{i=0}^{d} \binom{n}{i}$ possible sets of invalid elements, and for a total of t tests we have t bits to represent the results, which implies we have 2^t possibilities of test results. We need $\sum_{i=0}^{d} \binom{n}{i} \leq 2^t$ in order to distinguish the sets of invalid elements, which gives us the lower bound $t(d, n) \geq d \log(n/d)$. For $d \geq 2$, the best known lower bound on t for d-CFF(t,n) is given by $t(d, n) \geq c \frac{d^2}{\log d} \log n$ for some constant c [13, 47, 58].

For $d \geq 2$, there are several approaches to construct d-CFFs, for example, we can construct them directly from codes and combinatorial designs. Probabilistic methods usually provide the best existence results known, since they provide upper bounds close to the best lower bound for $t(d, n)$. Furthermore, derandomization techniques can be used to yield efficient algorithms to construct CFFs, such as the ones presented in [15, 16, 45]. Using this approach, Porat and Rotschild [45] gave the first polynomial time algorithm to construct a d-CFF(n, t) with $t = \Theta(d^2 \log n)$. Gargano, Rescigno and Vaccaro [16, 17] gave a polynomial time algorithm based on derandomization of the Lovász local lemma, also with $\Theta(d^2 \log n)$. In a more recent paper [15], they improve the running time to $O(n)$ re-sampling steps. More details related to constructions of d-CFFs are given in Section 2.4.

An important open question concerns closing the gap between the best known lower bound $\Omega(\frac{d^2}{\log d} \log n)$ and the upper bound given by constructions, in particular, the best

general bound of $O(d^2 \log n)$ obtained by some of the algorithms based on the probabilistic method. Shangguan and Ge [49] show that if $d \geq c\sqrt{n}$, for a constant $c \approx 1$, then $t(d, n) = n$, which means the CFF with smallest t is an identity matrix. So, for $d \geq c\sqrt{n}$ there is no benefit in combinatorial group testing over individual tests. Gargano, Rescigno and Vaccaro [15] have recently proven that if $d \leq \sqrt[3]{n}$ then $t(d, n) = \Theta(\frac{d^2 \log n}{\log d})$. Therefore, the region where there remains a gap between asymptotic lower and upper bounds is for $d \in [\sqrt[3]{n}, c\sqrt{n}]$ with $c \approx 1$.

This paper is organized as follows. In Section 2.3, we review definitions, and discuss equivalent and related structures to d-CFFs. In Section 2.4, we review several constructions of CFFs and discuss the relationships between them. In Section 2.5, we define and propose a construction of a new generalization of CFFs, called *variable CFFs*, and we generalize known constructions of d-CFFs to the case of variable CFFs. In Section 2.6, we review some applications of d-CFFs in cryptography and show how a variable CFF could be applied in one of them.

2.3 Background on cover-free families and related structures

Cover-free families are seen in the literature under different names and variations. In this section, we discuss some definitions, generalizations and related structures.

Sometimes, it is easier to study CFFs by considering their matrix representation. We can represent the set system $\mathcal{F} = (X, \mathcal{B})$ as a $t \times n$ binary incidence matrix $\mathcal{M}$ by considering the characteristic vectors of each subset $B \in \mathcal{B}$ as a column of $\mathcal{M}$. A *characteristic vector* (or incidence vector) of a subset B of a set $X = \{x_1, \ldots, x_t\}$ is a vector $e = (e_1, \ldots, e_t)$, with $e_i = 1$ if $x_i \in B$ and $e_i = 0$ otherwise. Throughout this paper, we refer to the *weight* of a binary vector as its number of 1's.

More precisely, the incidence matrix $\mathcal{M}$ of $\mathcal{F}$ is represented as follows:

$$\mathcal{M}_{i,j} = \begin{cases} 1, & \text{if } x_i \in B_j, \\ 0, & \text{otherwise.} \end{cases}$$

As an example, consider $t = 9, n = 12, X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $\mathcal{B} = \{B_1, \ldots, B_{12}\}$, where $B_1 = \{1, 2, 3\}, B_2 = \{4, 5, 6\}, B_3 = \{7, 8, 9\}, B_4 = \{1, 4, 7\}, B_5 = \{2, 5, 8\}, B_6 = \{3, 6, 9\}, B_7 = \{1, 5, 9\}, B_8 = \{2, 6, 7\}, B_9 = \{3, 4, 8\}, B_{10} = \{1, 6, 8\}, B_{11} = \{2, 4, 9\}, B_{12} = \{3, 5, 7\}$. After some inspection we can see that no subset is contained in the union of any 2 others, which gives a 2-CFF$(9, 12)$. The incidence matrix of this 2-CFF$(9, 12)$ is depicted in Figure 2.1.

For simplicity, we say that $\mathcal{M}$ is d-CFF when its corresponding set system $\mathcal{F}$ is d-CFF. We may also say that a column indexed by i_0 is not covered by the union of columns indexed by $i_1, \ldots, i_d$, when their corresponding subsets $B_{i_0}, B_{i_1}, \ldots, B_{i_d} \in \mathcal{B}$ satisfy the

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 2.1: A 2-CFF$(9, 12)$ incidence matrix.

property given in (2.1). We note that if we permute rows and columns of a d-CFF $\mathcal{M}$, we obtain another matrix that is also d-CFF.

We can define d-CFFs in terms of permutation sub-matrices [24, 27, 30, 37]. A permutation matrix is an $n \times n$ binary matrix obtained by permuting the rows of the identity matrix.

Proposition 2.3.1. *A matrix $\mathcal{M}$ is d-CFF if and only if any set of $d+1$ columns contains a permutation sub-matrix of dimension $d+1$.*

Proof. Take $d+1$ columns $\{j_0, j_1, \ldots, j_d\}$ of $\mathcal{M}$. Property (2.1) is true for j_l in place of i_0 and $J = \{j_0, j_1, \ldots, j_d\} \setminus j_l$ in place of $\{i_1, \ldots, i_d\}$ if and only if there exists a row i where $\mathcal{M}_{i,j_l} = 1$ and $\mathcal{M}_{i,j_k} = 0$ for all $j_k \in J$. This is equivalent to a permutation sub-matrix of dimension $d+1$ in columns $\{j_0, j_1, \ldots, j_d\}$. □

We can also define d-CFFs in terms of a *coverage property* of certain binary vectors that form the rows of $\mathcal{M}$. This next proposition is directly related to the one above.

Proposition 2.3.2. *A $t \times n$ binary matrix $\mathcal{M}$ is d-CFF if and only if for any set of $d+1$ columns, the $t \times (d+1)$ sub-array indexed by these columns contains every vector $e \in \{0,1\}^{d+1}$ of weight 1 appearing as one of its rows.*

Proof. The existence of such vectors is equivalent to the existence of a permutation sub-matrix of dimension $d+1$. □

The property of coverage of vectors from Proposition 2.3.2 is directly related to combinatorial objects known as *covering arrays* (CAs) (see Colbourn and Dinitz [4]).

Definition 2.3.3. *A* covering array *$CA_\lambda(N; t, k, v)$ is an $N \times k$ array such that in every $N \times t$ sub-array, each t-tuple appears at least λ times as a row of the sub-array. The value t is called the* strength*, and v is the number of symbols (alphabet).*

Because of Proposition 2.3.2, we notice that a d-CFF can be considered a generalization of a $\mathrm{CA}_1(N; d+1, k, 2)$ over a binary alphabet, where only binary vectors of weight 1 need to be covered.

So far we have discussed the incidence matrix representation of d-CFFs. We can easily solve a combinatorial group testing problem with a d-CFF(t, n) incidence matrix $\mathcal{M}$: the columns of the incidence matrix correspond to the n elements that need to be tested, and each of the t rows indicates how the elements will be grouped together in each test. Since for every column indexed by i_0 there is a position k such that $\mathcal{M}_{k,i_0} = 1$ and $\mathcal{M}_{k,i_1} = \mathcal{M}_{k,i_2} = \ldots = \mathcal{M}_{k,i_d} = 0$, we know that any non-defective element represented by column c_{i_0} is in at least one group that contains only non-defective elements (i.e. none of those defective $c_{i_1}, \ldots, c_{i_d}$). This group will "pass" the test, which indicates that all elements in there are non-defective. Since this is true for every column, after testing all groups we identify all non-defective elements, and the remaining elements are the defective ones.

Generalizations of d-CFFs were later studied, and they are very important for applications that have some extra requirements. These generalizations include (w, d)-CFFs, $(d; \lambda)$-CFFs, and $(w, d; \lambda)$-CFFs. Below we present the general definition of $(w, d; \lambda)$-CFFs from [56].

Definition 2.3.4 (Stinson and Wei (2004) [56])**.** *Let w, r and d be positive integers. A set system $\mathcal{F} = (X, \mathcal{B})$ is a $(w, d; \lambda)$-cover free family, denoted $(w, d; \lambda)$-CFF(t, n), if for any $w + d$ subsets $B_{i_1}, \ldots, B_{i_w}, A_{i_1}, \ldots, A_{i_d} \in \mathcal{B}$, we have*

$$\left| \left(\bigcap_{j=1}^{w} B_{i_j} \right) \setminus \left(\bigcup_{j=1}^{d} A_{i_j} \right) \right| \geq \lambda. \tag{2.2}$$

When we set $w = \lambda = 1$ we get the d-CFFs from Definition 2.2.1. For $\lambda = 1$ we obtain (w, d)-CFFs, which were first presented in [38] for applications in key distribution. For $w = 1$ we get $(d; \lambda)$-CFFs, first proposed in [7] when they investigated superimposed distance codes. We note that some authors, such as the ones in [58], present this definition with a "$> \lambda$" instead of "$\geq \lambda$", so the reader should be aware of this kind of discrepancy when comparing results.

We can also define $(w, d; \lambda)$-CFFs in terms of a coverage property of binary vectors of weight w. In this case, a $(w, d; \lambda)$-CFF can be considered a generalization of a $\mathrm{CA}_\lambda(N; w + d, k, 2)$, where only binary vectors of weight w need to be covered.

Proposition 2.3.5. *A $t \times n$ binary matrix $\mathcal{M}$ is $(w, d; \lambda)$-CFF if and only if for any set of $w + d$ columns, the $t \times (w + d)$ sub-array indexed by these columns contains every vector $e \in \{0, 1\}^{d+w}$ of weight w appearing in at least λ rows.*

Proof. Take $w + d$ column indices $j_1, \ldots, j_w, k_1, \ldots, k_d$ of $\mathcal{M}$. The property above implies that we have λ rows i in which $\mathcal{M}_{i,j_1} = \mathcal{M}_{i,j_2} = \ldots = \mathcal{M}_{i,j_w} = 1$ while $\mathcal{M}_{i,k_1} = \mathcal{M}_{i,k_2} = \ldots = \mathcal{M}_{i,k_d} = 0$. This directly implies the Property (2.2). □

The next propositions state relationships between sub-matrices of CFF matrices. They are presented in [24, 27][1] for d-CFFs, and here we generalize them for $(w, d; \lambda)$-CFFs. Their proofs follow directly from Definition 2.3.4 and Proposition 2.3.5.

Proposition 2.3.6. *Let $\mathcal{M}$ be a matrix and let $\mathcal{M}'$ be a sub-matrix of $\mathcal{M}$ formed by at least $w + d$ of its columns. If $\mathcal{M}$ is $(w, d; \lambda)$-CFF, then $\mathcal{M}'$ is also $(w, d; \lambda)$-CFF.*

Proof. Let $C_{\mathcal{M}}$ and $C_{\mathcal{M}'}$ be the set of columns of $\mathcal{M}$ and $\mathcal{M}'$, respectively. Since $\mathcal{M}$ is $(w, d; \lambda)$-CFF, the property from Proposition 2.3.5 holds for any set of $w + d$ columns in $C_{\mathcal{M}}$, in particular when we restrict to columns in $C_{\mathcal{M}'} \subseteq C_{\mathcal{M}}$. □

Proposition 2.3.7. *Let $\mathcal{M}$ be a matrix and let $\mathcal{M}'$ be a sub-matrix of $\mathcal{M}$ formed by some of its rows. If $\mathcal{M}'$ is $(w, d; \lambda)$-CFF, then $\mathcal{M}$ is $(w, d; \lambda)$-CFF.*

Proof. Let $R_{\mathcal{M}}$ and $R_{\mathcal{M}'}$ be the set of row indices of $\mathcal{M}$ and $\mathcal{M}'$, respectively. So, $R_{\mathcal{M}'} \subseteq R_{\mathcal{M}}$. Since $\mathcal{M}'$ is $(w, d; \lambda)$-CFF, the property from Proposition 2.3.5 holds for $R_{\mathcal{M}'}$ and any set of $w + d$ columns, and since $R_{\mathcal{M}'} \subseteq R_{\mathcal{M}}$, this is also true for $\mathcal{M}$. □

So far, we have seen CFFs as set systems in Definitions 2.2.1 and 2.3.4, and how to go from set systems $\mathcal{F} = (X, \mathcal{B})$ to binary matrices $\mathcal{M}$ using the characteristic vectors of the subsets as *columns* of $\mathcal{M}$. We can also study CFFs by considering subsets that represent the *rows* of $\mathcal{M}$, where different properties are required in order to obtain CFFs. Next we present definitions of subsets on columns and rows, and discuss the necessary properties for CFFs.

Definition 2.3.8. *A* column set system $\mathcal{F}_c = (X, \mathcal{B})$ *of a $t \times n$ binary matrix $\mathcal{M}$ has $|X| = t, |\mathcal{B}| = n$, and the characteristic vectors of $B \in \mathcal{B}$ are the columns in $\mathcal{M}$. A* row set system $\mathcal{F}_r = (X, \mathcal{B})$ *of a $t \times n$ binary matrix $\mathcal{M}$ has $|X| = n, |\mathcal{B}| = t$, and the characteristic vectors of $B \in \mathcal{B}$ are the rows in $\mathcal{M}$.*

From Definition 2.3.4, we can rewrite the CFF property in terms of the column set system.

Proposition 2.3.9. *Let $\mathcal{M}$ be a binary matrix and $\mathcal{F}_c = (X, \mathcal{B})$ be its column set system, with $X = \{x_1, \ldots, x_t\}$ and $\mathcal{B} = \{B_1, \ldots, B_n\}$. Then, $\mathcal{M}$ is $(w, d; \lambda)$-CFF if and only if Property (2.2) holds.*

Similarly, for the row set system $(X, \mathcal{B})$ of $\mathcal{M}$, X now represents the column indices of $\mathcal{M}$, and subsets in $\mathcal{B}$ are subsets of column indices of $\mathcal{M}$. Note that when we consider the same $t \times n$ binary matrix $\mathcal{M}$ and its corresponding $\mathcal{F}_c$ and $\mathcal{F}_r$, the incidence matrix of $\mathcal{F}_c$ is equal to $\mathcal{M}$, and the incidence matrix of $\mathcal{F}_r$ is $\mathcal{M}^T$.

Proposition 2.3.10. *Let $\mathcal{M}$ be a binary matrix and $\mathcal{F}_r = (X, \mathcal{B})$ be its row set system, with $X = \{x_1, \ldots, x_n\}$ and $\mathcal{B} = \{B_1, \ldots, B_t\}$. Then, $\mathcal{M}$ is $(w, d; \lambda)$-CFF if and only if for any two subsets $C, D \subseteq X$, with $|C| \leq w$, $|D| \leq d$, and $C \cap D = \emptyset$, there is at least λ subsets $B \in \mathcal{B}$ such that $C \subseteq B$ and $D \cap B = \emptyset$.*

[1] Given in this thesis in Chapter 3.

Proof. The subset C represents any w columns of $\mathcal{M}$, and subset D represents any other d columns. The fact that there is at least λ subsets $B \in \mathcal{B}$ with $C \subseteq B$ and $D \cap B = \emptyset$ implies that by restricting $\mathcal{M}$ to these $w + d$ columns indexed by $C \cup D$, we will find λ rows with 1's in positions indexed by C and 0's in positions indexed by D. Since this is true for any choice of w and d, Proposition 2.3.5 holds. □

This last proposition is directly related to *disjunct systems* [56], presented next.

2.3.1 Related combinatorial structures

Cover-free families were first introduced by Kautz and Singleton [30] in the context of *superimposed codes*. In the literature, we find them under different names, such as d-disjunct matrices, strongly selective families, and disjunct systems. Here we show the relationship between these structures and the definitions we presented above.

We start with two well-known structures in combinatorial group testing called d-disjunct and d-separable matrices. A *d-disjunct matrix* is a binary matrix in which its column set system has Property (2.1). Therefore, for a column set system $\mathcal{F}_c$ with d-CFF property, a d-disjunct matrix is equivalent to a d-CFF incidence matrix. Moreover, a binary matrix is called *d-separable* if its column set system has the property where the union of (up to) d subsets (columns) are all distinct [8].

Pastuszak et al. [43] define a related structure called *OR-checking matrix*, which can be seen as one type of d-CFF. The authors propose the use of OR-checking matrices to identify bad signatures in a batch of digital signatures.

Definition 2.3.11 (Pastuszak et al. (2000) [43])**.** *A $(d+1)\sqrt{n} \times n$ binary matrix is an* OR-checking matrix *if it has $d+1$ ones per column, $\sqrt{n}$ ones per row, and the inner product of any two columns is either zero or one.*

The last requirement implies that two columns of the binary matrix intersect in at most one position, so the following proposition can be proved [23].

Proposition 2.3.12. *A $(d+1)\sqrt{n} \times n$ OR-checking matrix $\mathcal{C}$ is a d-CFF$((d+1)\sqrt{n}, n)$ incidence matrix.*

Proof. Consider the column set system $\mathcal{F}_c = (X, \mathcal{B})$ of $\mathcal{C}$, with $\mathcal{B} = \{B_1, \ldots, B_n\}$. Pick any $d+1$ columns of $\mathcal{C}$ indexed by $c_0, c_1, \ldots, c_d$. From its definition, we have that $|B_{c_i} \cap B_{c_j}| \leq 1$, which implies that $|B_{c_0} \cap (B_{c_1} \cup \ldots \cup B_{c_d})| \leq \sum_{i=1}^{d} |B_{c_0} \cap B_{c_i}| \leq d$. Since each column has $d+1$ ones, this implies that B_{c_0} has at least one extra position not covered by this union, and Property (2.1) holds. □

Disjunct systems are also closely related to CFFs; in fact, they are dual incidence structures. In terms of incidence matrix, if A is the incidence matrix of a $(w, d; \lambda)$-CFF(t, n), then A^T is the incidence matrix of a $(w, d; \lambda)$-disjunct system [56]. Or in other words, a disjunct system can be defined as the row set system of A with the property presented in

Proposition 2.3.10. They are denoted $(w, d; \lambda)$-DS(n, t) for $|X| = n$, $|\mathcal{B}| = t$, and from this relationship we obtain the following result.

Proposition 2.3.13 (Stinson and Wei (2004), Theorem 1.1 [56])**.** *There exists a $(w, d; \lambda)$-CFF(t, n) if and only if there exists a $(w, d; \lambda)$-DS(n, t).*

There is also a relationship between cover-free families and *strongly selective families* (SSFs).

Definition 2.3.14 (Porat and Rotschild (2011) [45])**.** *A family of subsets $\mathcal{B}$ of a set $X = \{1, \ldots, n\}$ is an $(n, d+1)$-strongly selective family, denoted $(n, d+1)$-SSF, if for every subset $A \subset X$ with $|A| = d + 1$, and for every element $x \in A$, there exists a subset $B \in \mathcal{B}$ such that $A \cap B = \{x\}$.*

We can directly see that a $(n, d+1)$-SSF is a row set system $F_r = \{X_r, \mathcal{B}_r\}$ of a d-CFF. In Proposition 2.3.10, for any two subsets $C, D \subseteq X_r$, and $B \in \mathcal{B}_r$, we need $C \subseteq B$ and $D \cap B = \emptyset$. Now consider the $(n, d+1)$-SSF $\{X, \mathcal{B}\}$, we notice that for any two subsets $C, D \subseteq X$, with $C = \{x\}$ and $D = A \setminus \{x\}$ for every $x \in A$, and any $B \in \mathcal{B}$, the SSF property implies $C \subseteq B$ and $D \cap B = \emptyset$.

2.4 Direct and algorithmic constructions of cover-free families

There exists a large variety of constructions for CFFs in the literature, in particular, the ones based on codes, finite fields, and combinatorial objects. Here we present a non-exhaustive list of constructions.

2.4.1 Direct constructions

2.4.1.1 Construction from codes

Consider an alphabet Q of size q. An (N, n, q)-*code* $\mathcal{C}$ of length N over alphabet Q consists of n codewords $c = (c_1, \ldots, c_N)$, with $c_i \in Q, 1 \leq i \leq N$. We say $\mathcal{C}$ has minimum distance D if any two codewords in $\mathcal{C}$ differ in at least D positions.

The following construction is presented in [42, 58], and builds a set system $(X, \mathcal{B})$ from an (N, n, q)-code $\mathcal{C}$.

$$X = \{1, \ldots, N\} \times Q,$$
$$B_c = \{(i, c_i), 1 \leq i \leq N\},$$
$$\mathcal{B} = \{B_c : c \in \mathcal{C}\}.$$

We obtain an uniform set system with $|X| = Nq, |\mathcal{B}| = n, |B_c| = N$, for all $c \in \mathcal{C}$. For specific values of d, the authors of [42] prove $(X, \mathcal{B})$ is a d-CFF, and the authors of [58] generalize it to (d, λ)-CFFs.

Theorem 2.4.1 (Wei (2006) [58]). *If there exists an (N, n, q)-code $\mathcal{C}$ with minimum distance D, then there exists a $(d; \lambda)$-CFF(Nq, n), for any $d \leq \lfloor \frac{N-\lambda}{N-D} \rfloor$.*

Proof. Because $\mathcal{C}$ has minimum distance D, we know any two codewords have at most $N - D$ positions in common, which implies that any two subsets $B_i, B_j, 1 \leq i, j \leq n, i \neq j$, from the construction above have $|B_i \cap B_j| \leq N - D$. For any $d + 1$ subsets $B_0, \ldots, B_d$ we have $|B_0 \setminus \bigcup_{i=1}^{d} B_i| \geq N - d(N - D)$, and since $N - d(N - D) \geq \lambda$, we conclude that $(X, \mathcal{B})$ is $(d; \lambda)$-CFF(Nq, n) for any $d \leq \lfloor \frac{N-\lambda}{N-D} \rfloor$. □

Note that in [58] the authors define a $(d; \lambda)$-CFF as having $|B_0 \setminus \bigcup_{i=1}^{d} B_i| > \lambda$, instead of $\geq \lambda$ as we do here. Thus, we have adjusted their statement to match our definition.

Figure 2.2 shows an example of a $(3, 9, 3)$-code $\mathcal{C}$ with minimum distance $D = 2$, used to construct the incidence matrix of a 2-CFF$(9, 9)$, with $d = (N-1)/(N-D) = 3-1/3-2 = 2$, according to the construction above. The codewords of $\mathcal{C}$ are listed as columns of C.

$$C = \begin{pmatrix} 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 \\ 1 & 2 & 3 & 2 & 3 & 1 & 3 & 1 & 2 \\ 1 & 2 & 3 & 3 & 1 & 2 & 2 & 3 & 1 \end{pmatrix}$$

$$B_{111} = \{(1,1), (2,1), (3,1)\},$$
$$\vdots$$
$$B_{231} = \{(1,2), (2,3), (3,1)\},$$
$$\vdots$$
$$B_{321} = \{(1,3), (2,2), (3,1)\}.$$

$$\mathcal{M} = \left(\begin{array}{ccccccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right)$$

Figure 2.2: A 2-CFF$(9, 9)$.

It is important to note that due to the relationship of codes with many other structures, similar results are presented in many other works. Moreover, different types of codes give different existence results of CFFs, as can be seen in [42, 58]. Next we present several relationships of constructions using combinatorial designs.

2.4.1.2 Construction from combinatorial designs

Here we present CFF constructions from packing arrays, orthogonal arrays, separating hash families, perfect hash families, mutually orthogonal Latin squares, and relationships among themselves and among code constructions.

Definition 2.4.2 (Stevens and Mendelsohn (2004) [52])**.** *A* packing array *is an $n \times k$ array with values from an alphabet with v symbols, such that in any t columns, every t-tuple of elements is contained in at most one row. We denote it as PA($n; t, k, v$), and the value t is called the* strength *of the PA.*

In fact, codes and packing arrays are equivalent.

Theorem 2.4.3. *There exists a PA($n; t, k, v$) if and only if there exists a (k, n, v)-code with minimum distance $D = k - t + 1$.*

Proof. Suppose we have a PA$(n; t, k, v)$. Every two rows of the PA have at most $t - 1$ elements in common, so they differ in at least $k - (t - 1) = k - t + 1$ positions. Therefore, the rows of this PA are codewords of a code with $D = k - t + 1$. Now suppose we have a (k, n, v)-code, with a total of n codewords of length k and minimum distance D. Any two codewords have at most $k - D$ positions in common, therefore, these codewords are rows of a PA$(n; t, k, v)$ with $t - 1 = k - D$ positions in common, and therefore strength $t = k - D + 1$. □

Next we present an existing construction that was rediscovered by Idalino and Moura [25][2], which uses a PA to build a d-CFF. This construction is very similar to the one from Stinson and Wei [56] that we present at the end of this section.

Construction 2.4.4 (Idalino and Moura (2018) [25])**.** *Let A be a PA($n; t, k, v$), and let $B = A^T$. We build a $kv \times n$ matrix $\mathcal{M}$ as follows. Index each row of $\mathcal{M}$ with tuples (i, x), for $x = 1, \ldots, v$, $i = 1, \ldots, k$, and each column $j = 1, \ldots, n$, then*

$$\mathcal{M}_{(i,x),j} = \begin{cases} 1 & \text{if } B_{i,j} = x, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that the CFF construction from codes presented in Section 2.4.1.1 is equivalent to the one from PAs using Construction 2.4.4. Thus, using Theorem 2.4.1 and Theorem 2.4.3 we get the following result, which generalizes the results from Idalino and Moura [25].

Proposition 2.4.5. *If there is a PA($n; t, k, v$), then there is a $(d; \lambda)$-CFF(kv, n) for any $d \leq \frac{k-\lambda}{t-1}$.*

Orthogonal arrays are a special case of packing arrays (see Colbourn and Dinitz [4]).

[2] Chapter 5 in this thesis.

Definition 2.4.6. *An* orthogonal array *$OA(v^t; t, k, v)$ is a $v^t \times k$ array with elements from an alphabet of v symbols, such that in any t columns, every t-tuple of elements is contained in exactly one row.*

Note that an OA is a PA with maximum number of rows $n = v^t$. Moreover, $\mathrm{OA}(v^t; t, k, v)$ are also known as *maximum distance separable* (MDS) codes [4]. Therefore, constructions of CFFs from OAs are frequently seen in the literature, and are usually equivalent to the CFFs from the corresponding PAs and codes.

For the particular case where we consider $v = q$ a prime power, it is possible to build an $\mathrm{OA}(q^t; t, q+1, q)$ using polynomials over a finite fields via Bush's construction of orthogonal arrays [3], which corresponds to a Reed-Solomon code [4]. This OA gives a $(d; \lambda)$-CFF$(q^2 + q, q^t)$ for $d \leq \frac{q+1-\lambda}{t-1}$ [58] (value of d adjusted to match our definition of $(d; \lambda)$-CFF).

If we remove the very last column of the OA from Bush's construction and consider the resulting $\mathrm{OA}(q^t; t, q, q)$, we can use it to build a d-CFF(q^2, q^t) for $d \leq \frac{q-1}{t-1}$. This CFF is equivalent to the one built directly from polynomials over finite fields given by Erdös, Frankl and Füredi [9]. As an example, for $q = 3$ and $t = 1$ Figure 2.3 shows a transposed $\mathrm{OA}(3^2; 2, 3, 3)$ (which is a PA and a code), and Figure 2.4 shows the corresponding 2-CFF$(3^2, 3^2)$. This CFF can be obtained via [9] with polynomials with coefficients in $\mathbb{F}_3$ and degree at most 2.

0	1	2	0	1	2	0	1	2
0	1	2	1	2	0	2	0	1
0	1	2	2	0	1	1	2	0

Figure 2.3: Transposed $\mathrm{OA}(3^2; 2, 3, 3)/\mathrm{PA}(3^2; 2, 3, 3)/(3, 9, 3)$-code with $D = 2$.

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 2.4: 2-CFF$(9, 9)$ from $\mathrm{OA}(3^2; 2, 3, 3)/\mathrm{PA}(3^2; 2, 3, 3)/(3, 9, 3)$-code with $D = 2$.

Finally, for $t = 2$ an $\mathrm{OA}(v^2; 2, k = s+2, v)$ is equivalent to s mutually orthogonal Latin squares MOLS of order v [4], which implies that MOLS can also be used to construct CFFs. A particular construction of CFF from s MOLS is given in [43], which results in a d-CFF$((d+1)\sqrt{n}, n)$, for $d \leq s$. For $v = q$ a prime power, there exist $q - 1$ MOLS of order q, and therefore the construction achieves $d \leq q - 1$. There is a discussion in [25] about restricting the PA/OA to fewer columns in order to build CFFs with a smaller number of

rows, and consequently a smaller d as well. This property was first observed in [43], and naturally happens in constructions from OAs/PAs and directly from polynomials because of the relationships we presented above.

Separating hash families and perfect hash families can also be used to construct CFFs, and they have relationships with OAs and PAs.

Definition 2.4.7 (Stinson et al. (2008) [57]). *An $(n, m, \{w_1, w_2\})$-λ-separating hash family, denoted λ-SHF$(N; n, m, \{w_1, w_2\})$, is an $N \times n$ matrix A with entries from $\{1, 2, \ldots, m\}$, such that for any two disjoint sets of columns C_1, C_2 such that $|C_1| = w_1$ and $|C_2| = w_2$, there exists at least λ rows r in A with the following property:*

$$\{A_{r,x} : x \in C_1\} \cap \{A_{r,y} : y \in C_2\} = \emptyset.$$

The next relationship between SHFs and PAs is presented in [25].

Proposition 2.4.8 (Idalino and Moura (2018) [25]). *If A is a PA$(n; t, k, v)$, and $w \leq \frac{k-1}{t-1}$, then A^T is a SHF$(N = k; n, m = v, \{1, w\})$.*

This implies that given a PA$(n; t, k, v)$ A, we can use the corresponding SHF$(N = k; n, m = v, \{1, w\})$ in Construction 2.4.4 in place of the transposed PA A^T to build a d-CFF(Nm, n), with $d \leq w \leq \frac{k-1}{t-1}$ [25].

Definition 2.4.9 (Stinson et al. (2000) [54]). *A (n, m, w)-λ-perfect hash family, denoted λ-PHF$(N; n, m, w)$, is an $N \times n$ matrix with entries from $\{1, 2, \ldots, m\}$, having the property that in any w columns there exists at least λ rows such that the w entries in the given w columns are all distinct.*

Corollary 2.4.10 (Stinson and Wei (2004) [56]). *A PHF$(N; n, m, w + r)$ is a SHF$(N, n, m, \{w, r\})$.*

In particular, a PHF$(N; n, m, w+1)$ is a SHF$(N, n, m, \{1, w\})$, so it can also be used to construct a d-CFF(Nm, n) with $d \leq w$ using Construction 2.4.4. This construction of d-CFFs from PHFs is equivalent to the ones given by Kim [31, Theorem 5.3] and Niederreiter et al. [42, Theorem 2.41].

There are also more general constructions of CFFs from SHFs and PHFs. In particular, Stinson and Wei [56] present a recursive construction, which uses SHFs and smaller CFFs to construct larger CFFs. Let A be the $N \times n$ matrix that represents a λ_1-SHF$(N; n; m; \{w, d\})$, let $\mathcal{M}$ be a $(w, d; \lambda_2)$-CFF(t, m) and denote its columns as $c_1, \ldots, c_m$. We can construct the $tN \times n$ incidence matrix of a $(w, d; \lambda_1\lambda_2)$-CFF$(tN, n)$ by replacing every element i in A by column c_i from $\mathcal{M}$ [56]. Then, the following theorem is obtained.

Theorem 2.4.11 (Stinson and Wei (2004) [56]). *If there exists a λ_1-SHF$(N; n; m; \{w, d\})$ and a $(w, d; \lambda_2)$-CFF(t, m), then there exists a $(w, d; \lambda_1\lambda_2)$-CFF$(tN, n)$.*

Moreover, since a PHF$(N; n, m, w + d)$ is an SHF$(N, n, m, \{w, d\})$, we can apply PHFs to this construction as well. Niederreiter et al. [42, Theorem 2.43] present an specific case of this construction, using a d-CFF(t, m) and a PHF$(N; n, m, d+1)$ to build a d-CFF(tN, n).

2.4.2 Algorithmic constructions based on the probabilistic method

Existence of d-CFFs based on the probabilistic method [15, 16, 17, 45] yield the best known upper bounds on $t(n,d)$, in the sense of being close to the lower bound $t(n,d) = \Omega(\frac{d^2}{\log d}\log n)$ [13, 47].

The first explicit construction of a d-CFF(t,n) with $t = O(d^2 \log n)$ was given by Porat and Rothschild [45]. In the breakthrough paper, the authors give a construction of a linear code that meets the Gilbert-Varshamov bound (GV-bound), which by means of the construction in Section 2.4.1.1, yields the desired CFF. In more detail, lets first recall what is a linear code. In Section 2.4.1.1, we consider direct constructions of CFFs from error-correcting codes with minimum distance D. Here, we consider *linear codes* (LC), which are error-correcting codes over the alphabet $\mathbb{F}_q$, in which a linear combination of codewords is also a codeword. An $(m,k,D)_q$-LC $\mathcal{C}$ allows us to send messages of k symbols using a noisy channel by encoding them into codewords of m symbols with $m > k$. We say k is the *dimension*, m is the *length*, D is the *minimum distance* of the code, and $\mathcal{C}$ consists of $n = q^k$ codewords $\mathcal{C} = \{c_1, \ldots, c_n\}$ over an alphabet with q symbols. The code $\mathcal{C}$ is generated by an $m \times k$ generator matrix $\mathcal{G}$, i.e. $\mathcal{C} = \{\mathcal{G}y | y \in \mathcal{F}_q^k\}$.

Porat and Rothschild [45] build $(m, \log_q n, \delta m)_q$ codes over $\mathbb{F}_q$, $0 \leq \delta \leq 1$, with large distance δm to construct d-CFFs with $t = \Theta(d^2 \log n)$. They use these codes to build d-CFF(mq,n) using a construction equivalent to the one presented in Section 2.4.1.1. The authors prove that if we use a code that meets the GV-bound, we have $mq = \Theta(d^2 \log n)$, which leads to a d-CFF with $t = \Theta(d^2 \log n)$ rows [45], which is close to the lower bound $t = \Omega(\frac{d^2}{\log d}\log n)$.

The authors in [45] propose an explicit construction of the generator matrix $\mathcal{G}$, which generates a linear code $\mathcal{C}$ that meets the GV-bound with specific parameters and minimum distance δm. They first propose a probabilistic algorithm that picks values for each position $\mathcal{G}[i,j]$, one-by-one uniformly and independently at random. They define a *goal* function, that receives $\mathcal{G}$ as input and returns $goal(\mathcal{G}) \geq 1$ if $\mathcal{G}$ generates a code with minimum distance smaller than the required δm, or in other words, if the randomized algorithm fails on generating the desired $\mathcal{G}$. They also prove that for the appropriate choice of parameters, the expected value E($goal$)< 1, and therefore there is a positive probability that $\mathcal{G}$ generates the required code $\mathcal{C}$. Finally, the authors derandomize this algorithm by setting values for $\mathcal{G}[i,j]$ in order to minimize the expected value of $goal(\mathcal{G})$, considering the values of $\mathcal{G}$ already chosen far. This yields a deterministic polynomial time algorithm that takes $\Theta(dn \log n)$ time to construct a d-CFF(n,t) with $t = \Theta(d^2 \log n)$.

Gargano, Rescigno and Vaccaro gave polynomial time algorithms to construct CFFs with a constant number w of 1's per column and with $t = \Theta(d^2 \log n)$ [15, 16, 17]. They present a randomized construction based on the Lovász Local Lemma [10], and propose its de-randomization based on the method proposed by Moser and Tardos [39]. In their more recent manuscript, Gargano et al. [15] reduce the running time of the algorithm to $\Theta(n)$. This is accomplished by imposing an upper limit on the intersection between the sets corresponding to any pair of columns, which is enough to guarantee the d-CFF property, and reduces the dependence between columns to pairwise relations other than $(d+1)$-wise.

This reduction allows the reduction in running time.

2.5 Variable CFFs and recursive constructions

Let $\mathcal{M}$ be a binary matrix with n columns indexed by $C = \{c_1, \ldots, c_n\}$. If $\mathcal{M}$ is the incidence matrix of a $(w, d; \lambda)$-CFF, the coverage property from Proposition 2.3.5 must hold for all combinations of $d + w$ columns, or in other words, for all $(d + w)$-subsets of C.

In this section, we define a new type of cover-free family denoted $(w, \mathcal{S}; \lambda)$-CFF$(t, n)$, which we call *variable* CFFs. In this new family, we explicitly specify in $\mathcal{S}$ a collection of subsets of C that we want to "cover". Moreover, d is not a fixed value anymore, it varies with the size of each subset in $\mathcal{S}$. We use hypergraphs to define such CFFs, more specifically, $H = (C, \mathcal{S})$ is a hypergraph, where the vertices are the column indices C of $\mathcal{M}$, and the hyperedges in $\mathcal{S}$ are subsets of C.

Definition 2.5.1 (Variable CFFs)**.** *Let $\mathcal{M}$ be a $t \times n$ binary matrix with columns indexed by $C = \{c_1, \ldots, c_n\}$, and let $H = (C, \mathcal{S})$ be a hypergraph, with $\mathcal{S} = \{S_1, \ldots, S_m\}, S_i \subseteq C$. We say $\mathcal{M}$ is $(w, \mathcal{S}; \lambda)$-CFF$(t, n)$ if for any hyperedge $S \in \mathcal{S}$, the $t \times |S|$ sub-matrix of $\mathcal{M}$ indexed by S has each vector in $\{0, 1\}^{|S|}$ of weight w appearing at least λ times as its rows. A $(1, \mathcal{S}; 1)$-CFF(t, n) is simply denoted $\mathcal{S}$-CFF(t, n).*

An application of variable CFFs is introduced in Section 2.6.

Let F be a set, and let n be an integer such that $|F| \geq n$. We denote $\binom{F}{n}$ to be the set of all n-subsets of F. We notice that a $(w, d; \lambda)$-CFF is a special case of a $(w, \mathcal{S}; \lambda)$-CFF, where hyperedges in $\mathcal{S}$ consist of all sets of columns of cardinality $w + d$.

Proposition 2.5.2. *A $(w, d; \lambda)$-CFF(t, n) is equivalent to an $(w, \mathcal{S}; \lambda)$-CFF$(t, n)$ with column indices $C = \{c_1, \ldots, c_n\}$, and $\mathcal{S} = \binom{C}{w+d}$.*

Proof. It is easy to see that Proposition 2.3.5 guarantees the desired coverage for each $S \in \mathcal{S}$. □

We remark that, by picking all subsets of cardinality $w + d$ in $\mathcal{S}$, all subsets of smaller size $w' + d'$ are also covered, for $w' \leq w$ and $d' \leq d$, which implies that a $(w, d; \lambda)$-CFF is also a $(w', d'; \lambda)$-CFF.

The following proposition can be easily derived from Definition 2.5.1.

Proposition 2.5.3. *Let $\mathcal{M}_1$ be a $(w, \mathcal{S}_1; \lambda)$-CFF$(t_1, n)$ and $\mathcal{M}_2$ be a $(w, \mathcal{S}_2; \lambda)$-CFF$(t_2, n)$, then $\mathcal{M} = \begin{pmatrix} \mathcal{M}_1 \\ \mathcal{M}_2 \end{pmatrix}$ is a $(w, \mathcal{S}_1 \cup \mathcal{S}_2; \lambda)$-CFF$(t_1 + t_2, n)$.*

2.5.1 A construction of variable CFF

We first introduce some notation for multisets. Let $M = \{a_1^{m(a_1)}, a_2^{m(a_2)}, \ldots, a_n^{m(a_n)}\}$ be a multiset composed of n distinct elements a_i with multiplicity $m(a_i) > 0$, for $1 \leq i \leq n$. The *cardinality* of a multiset M is $|M| = \sum_{i=1}^{n} m(a_i)$. Define the *width* of M, denoted $w(M)$, to be the cardinality of M when considered as a set; that is, $w(M) = |\{a_1, a_2, \ldots, a_n\}|$. Define the *height* of M, denoted $h(M)$, to be the largest of its multiplicities, that is, $h(M) = \max\{m(a_1), m(a_2), \ldots, m(a_n)\}$. These names are motivated by representing each element a_i of the multiset as a pile of bricks of height $m(a_i)$, so that M is a building of width $w(M)$ and height $h(M)$. For example, the multiset $M = \{1, 1, 1, 1, 2, 2, 4, 4, 4\}$, also denoted as $M = \{1^4, 2^2, 4^3\}$, has width $w(M) = 3$ and height $h(M) = 4$, and can be depicted as in Figure 2.5.

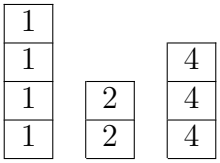


Figure 2.5: A representation of a multiset $M = \{1^4, 2^2, 4^3\}$.

For a set of pairs $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$, we denote by $S|_1 = \{a_1, a_2, \ldots, a_n\}$ and by $S|_2 = \{b_1, b_2, \ldots, b_n\}$, considered as multisets. For example, if $S = \{(2, 1), (2, 3), (4, 2), (4, 3), (5, 1), (5, 4)\}$, then $S|_1 = \{2^2, 4^2, 5^2\}$ and $S|_2 = \{1^2, 2^1, 3^2, 4^1\}$, with $|S| = |S|_1| = |S|_2| = 6$, $w(S|_1) = 3, h(S|_1) = 2, w(S|_2) = 4, h(S|_2) = 2$.

Theorem 2.5.4. *Let A^1 be a d_1-CFF(t_1, n_1) and A^2 be a d_2-CFF(t_2, n_2). Let $A = A^1 \otimes A^2$ with columns indexed by $C = \{(j, q) : 1 \leq j \leq n_1, 1 \leq q \leq n_2\}$ and rows indexed by $R = \{(i, p) : 1 \leq i \leq t_1, 1 \leq p \leq t_2\}$, such that $A_{(i,p),(j,q)} = A^1_{i,j} \cdot A^2_{p,q}$. Let $\mathcal{S}$ be a set containing each subset $S \subseteq C$, $|S| \geq 2$, such that one of the following holds:*

(i) $w(S|_1) \leq d_1 + 1$ and $h(S|_1) \leq d_2 + 1$; or

(ii) $w(S|_2) \leq d_2 + 1$ and $h(S|_2) \leq d_1 + 1$.

Then, A is a $\mathcal{S}$-CFF($t_1 t_2, n_1 n_2$).

Proof. Let $S \in \mathcal{S}$, and let $(j, q) \in S$. We must show that there exists a row $(i, p) \in R$ such that $A_{(i,p),(j,q)} = 1$ and $A_{(i,p),(x,y)} = 0$, for all $(x, y) \in S \setminus \{(j, q)\}$. Consider first the case where S satisfies condition (i), i.e., $w(S|_1) \leq d_1 + 1$ and $h(S|_1) \leq d_2 + 1$. Let i be a row of A^1 such that $A^1_{i,j} = 1$, and $A^1_{i,\ell} = 0$ for all $\ell \in S|_1, \ell \neq j$; such row must exist since $w(S|_1) \leq d_1 + 1$, and A^1 is d_1-CFF. Let $S^2(j) = \{x \in \{1, \ldots, n_2\} : (j, x) \in S\}$, and note that $|S^2(j)| \leq h(S|_1) \leq d_2 + 1$ and $q \in S^2(j)$. Let p be a row of A^2 such that $A^2_{p,q} = 1$ and $A^2_{p,f} = 0$ for all $f \in S^2(j) \setminus \{q\}$, which exists since A^2 is d_2-CFF. Thus, $A_{(i,p),(j,q)} = A^1_{i,j} \cdot A^2_{p,q} = 1 \cdot 1 = 1$. Now let $(x, y) \in S \setminus \{(j, q)\}$. If $x = j$ then $y \in S^2(j) \setminus q$, which implies $A^2_{p,y} = 0$; otherwise, $x \in S|_1, x \neq j$, which implies $A^1_{i,x} = 0$. In either case, we obtain $A_{(i,p),(x,y)} = A^1_{i,x} \cdot A^2_{p,y} = 0$. This concludes the first case. The second case, where S satisfies condition (ii), follows similar steps by switching the roles of A^1 and A^2 and using $S|_2$ in place of $S|_1$. □

We notice that an $S \in \mathcal{S}$ from $A = A^1 \otimes A^2$ can be as big as $|S| = (d_1 + 1)(d_2 + 1)$ while satisfying condition (i) or (ii). This implies that, when applying an $\mathcal{S}$-CFF in a combinatorial group testing problem, we would be able to identify up to $d = (d_1 + 1)(d_2 + 1) - 1$ errors, if they are "well behaved" (according to (i) or (ii)). In more detail, the combination of indices of errors that such $\mathcal{S}$-CFF can identify is defined by $E = \{S \setminus \{x\} : S \in \mathcal{S}, x \in S\}$.

In Figure 2.6 we have a very small example with A^1 a 1-CFF$(4, 6)$ and A^2 a 1-CFF$(2, 2)$. Therefore, $A = A^1 \otimes A^2$ has columns $C = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2), (4, 1), (4, 2), (5, 1), (5, 2), (6, 1), (6, 2)\}$. A is a $\mathcal{S}$-CFF, where any subset $S = \{(j, q), 1 \leq j \leq 6, 1 \leq q \leq 2\}$ of $\mathcal{S}$ has size $|S| \leq 4$ and has one of the two properties:

(i) $w(S|_1) \leq 2$ and $h(S|_1) \leq 2$, or in other words, there is at most two different values for j and each value of j appears at most 2 times; or

(ii) $w(S|_2) \leq 2$ and $h(S|_2) \leq 2$, or in other words, the same requirements, but now for values of q.

For example, $S = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$ fits requirement (i). If we restrict A to the first 4 columns, every binary vector $1000, 0100, 0010, 0001$ appears in at least one row, for instance, in rows $3, 4, 5, 6$, respectively. The same is true for $S = \{(3, 1), (4, 1), (5, 2)\}$, which fits requirement (ii). If we restrict to the columns indexed by S, we can find every vector $100, 010, 001$ appearing in at least one row. For this example, rows $1, 5, 8$, respectively.

$$A^1 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \quad A^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$A = \left(\begin{array}{cccccccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right)$$

Figure 2.6: A $\mathcal{S}$-CFF$(8, 12)$ incidence matrix.

We notice that the previous construction of $\mathcal{S}$-CFFs can be generalized to $(w, \mathcal{S}; \lambda)$-CFFs.

Theorem 2.5.5. *Let A^1 be a $(w_1, d_1; \lambda_1)$-CFF(t_1, n_1) and A^2 be a $(w_2, d_2; \lambda_2)$-CFF(t_2, n_2). Let $A = A^1 \otimes A^2$ with columns indexed by $C = \{(j, q) : 1 \leq j \leq n_1, 1 \leq q \leq n_2\}$ and rows*

indexed by $R = \{(i,p) : 1 \leq i \leq t_1, 1 \leq p \leq t_2\}$*, so that* $A_{(i,p),(j,q)} = A^1_{i,j} \cdot A^2_{p,q}$*. Let* $\mathcal{S}$ *be a set containing each subset* $S \subseteq C$*,* $|S| \geq w_1 w_2 + 1$*, such that one of the following holds:*

(i) $w(S|_1) \leq w_1 + d_1$ *and* $h(S|_1) \leq w_2 + d_2$*; or*

(ii) $w(S|_2) \leq w_2 + d_2$ *and* $h(S|_2) \leq w_1 + d_1$*.*

Then, A is a $(w_1 w_2, \mathcal{S}; \lambda_1 \lambda_2)$*-CFF*$(t_1 t_2, n_1 n_2)$*.*

Proof. Let $S \in \mathcal{S}$, $w = w_1 w_2, \lambda = \lambda_1 \lambda_2$. Let $W_1 = \{j_1, j_2, \ldots, j_{w_1}\} \subset S|_1, W_2 = \{q_1, q_2, \ldots, q_{w_2}\} \subset S|_2$, and $W = \{(j,q) : \text{ for all } j \in W_1, \text{ and for all } q \in W_2\} \subset S$. We must show that there exist λ rows (i,p) such that $A_{(i,p),(j,q)} = 1$ for all $(j,q) \in W$, and $A_{(i,p),(x,y)} = 0$, for all $(x,y) \in S \setminus W$. Consider first the case where S satisfies condition (i), i.e., $w(S|_1) \leq w_1 + d_1$ and $h(S|_1) \leq w_2 + d_2$. Let $L_1 = \{i_1, i_2, \ldots, i_{\lambda_1}\} \subset R|_1$ index the λ_1 rows of A^1 such that $A^1_{i,j} = 1$, for all $i \in L_1, j \in W_1$, and $A^1_{i,\ell} = 0$ for all $i \in L_1, \ell \in S|_1 \setminus W_1$; such rows must exist since $w(S|_1) \leq w_1 + d_1$, and A^1 is $(w_1, d_1; \lambda_1)$-CFF. Let $S^2(W_1) = \{x \in \{1, \ldots, n_2\} : (j,x) \in S, \text{ for all } j \in W_1\}$, and note that $|S^2(W_1)| \leq h(S|_1) \leq w_2 + d_2$ and $W_2 \subset S^2(W_1)$. Let $L_2 = \{p_1, p_2, \ldots, p_{\lambda_2}\} \subset R|_2$ index the λ_2 rows of A^2 such that $A^2_{p,q} = 1$, for all $p \in L_2$, and all $q \in W_2$, and $A^2_{p,f} = 0$, for all $p \in L_2$ and for all $f \in S^2(W_1) \setminus W_2$, these rows exist since $|S^2(W_1)| \leq w_2 + d_2$ and A^2 is $(w_2, d_2; \lambda_2)$-CFF. Now, let $L = \{(i,p) : i \in L_1, p \in L_2\} \subset R$. Thus, $A_{(i,p),(j,q)} = A^1_{i,j} \cdot A^2_{p,q} = 1 \cdot 1 = 1$, for all λ rows indexed by $(i,p) \in L$ and all w columns indexed by $(j,q) \in W$. Now let $(x,y) \in S \setminus W$. If $x \in W_1$ then $y \in S^2(W_1) \setminus W_2$ (otherwise (x,y) would be in W), which implies $A^2_{p,y} = 0$ for all $p \in L_2$; otherwise, $x \in S|_1 \setminus W_1$, which implies $A^1_{i,x} = 0$ for all $i \in L_1$. In either case, we obtain $A_{(i,p),(x,y)} = A^1_{i,x} \cdot A^2_{p,y} = 0$, for all λ rows $(i,p) \in L$. This concludes the first case. The second case, where S satisfies condition (ii), follows similar steps by switching the roles of A^1 and A^2 and using $S|_2$ in place of $S|_1$. □

2.5.2 Generalizations of recursive d-CFF constructions

Now we go back to traditional CFFs and discuss recursive constructions for them. A recursive construction was already presented in Theorem 2.4.11 using smaller CFFs and SHFs to build larger CFFs. Here, we discuss recursive constructions that use only CFFs as ingredients. These constructions are useful, for example, for dynamic applications where the parameter n can grow over time, but we want to reuse all computations that were done with the smaller matrices [24, 27][3].

We start with a straightforward construction by Li et al. [33].

Lemma 2.5.6 (Li et al. (2006) [33])**.** *If there exists a* $(d;\lambda)$*-CFF*(t_1, n_1) *and a* $(d;\lambda)$*-CFF*(t_2, n_2)*, then there exists a* $(d;\lambda)$*-CFF*$(t_1 + t_2, n_1 + n_2)$*.*

[3] Given in this thesis as Chapter 3

Proof. Let $\mathcal{M}_1$ be a $(d;\lambda)$-CFF(t_1, n_1) and $\mathcal{M}_2$ be a $(d;\lambda)$-CFF(t_2, n_2), we construct $(d;\lambda)$-CFF$(t_1 + t_2, n_1 + n_2)$ $\mathcal{M}$ as follows.

$$\mathcal{M} = \begin{pmatrix} \mathcal{M}_1 & 0 \\ 0 & \mathcal{M}_2 \end{pmatrix}.$$

□

The next two constructions were first presented in [33] for $(2;\lambda)$-CFFs, then generalized by Idalino and Moura [27] for $(d;\lambda)$-CFFs. Here we generalize it for $(w, d;\lambda)$-CFFs, and the proofs are an application of Theorem 2.5.5.

Theorem 2.5.7. *Let A_1 be a $(w, d;\lambda_1)$-CFF(t_1, n_1) and A_2 be a $(w, d;\lambda_2)$-CFF(t_2, n_2), then $A = A_1 \otimes A_2$ is a $(w, d;\lambda_1\lambda_2)$-CFF$(t_1t_2, n_1n_2)$.*

Proof. This is a particular case of Theorem 2.5.5. Let $C = \{(j, q) : 1 \leq j \leq n_1, 1 \leq q \leq n_2\}$ be the set of all the indices of columns of A, and let $\mathcal{S}' = \binom{C}{w+d}$. For any $S \in \mathcal{S}'$ we know that $|S| = w + d$, therefore it holds that

(i) $w(S|_1) \leq w + d$ and $h(S|_1) \leq w + d$; or

(ii) $w(S|_2) \leq w + d$ and $h(S|_2) \leq w + d$.

Therefore, $\mathcal{S}' \subseteq \mathcal{S}$ from Theorem 2.5.5, and since $w \leq w^2$, A is a $(w, \mathcal{S}';\lambda_1\lambda_2)$-CFF$(t_1t_2, n_1n_2)$. Since $\mathcal{S}'$ contains all the combinations of $w + d$ columns, by Proposition 2.5.2, A is $(w, d;\lambda_1\lambda_2)$-CFF. □

For $w = 1, \lambda_1 = \lambda_2 = 1$, this recursive construction is used in [24, 27] as a solution for unbounded fault-tolerant aggregation of signatures. The case where $w = 1$ and $\lambda \geq 1$ was also explored in [27] as a way of allowing errors in the transmission of aggregate signatures without losing the ability to do verification.

The next construction was also introduced by Li et al. [33] for $(2;\lambda)$-CFFs, then generalized by Idalino and Moura [27] for $(d;\lambda)$-CFFs. Here we generalize it for $(w, d;\lambda)$-CFFs in Theorem 2.5.9.

Construction 2.5.8. *Let A_1 be a $t_1 \times n_1$ binary matrix, A_2 be a $t_2 \times n_2$ binary matrix, and B be an $s \times n_2$ binary matrix. Create a matrix $P = B \otimes A_1$. This results in P with n_2 "blocks" of n_1 columns each. For each column in block i, append the i^{th} column of A_2, for $1 \leq i \leq n_2$.*

Theorem 2.5.9. *Let A_1 be a $(w, d;\lambda_1)$-CFF(t_1, n_1), A_2 be a $(w, d;\lambda_2)$-CFF(t_2, n_2), B be a $(w, d-1;\lambda_3)$-CFF(s, n_2), and $\lambda = min(\lambda_1\lambda_3, \lambda_2)$. Then there is a $(w, d;\lambda)$-CFF$(st_1 + t_2, n_1n_2)$.*

Proof. Use A_1, A_2, and B to construct a matrix A via Construction 2.5.8. We prove that A is a $(w, d; \lambda)$-CFF by showing this is a variation of the construction from Theorem 2.5.5 for different sub-matrices. Let $C = \{(j, q) : 1 \leq j \leq n_1, 1 \leq q \leq n_2\}$ be the indices of columns of A, and let $\mathcal{S}' = \binom{C}{w+d}$. We divide A in two submatrices, according to Construction 2.5.8:
$\mathcal{M}_1$: submatrix formed by $B \otimes A_1$;
$\mathcal{M}_2$: submatrix formed by replicating each column of A_2 consecutively n_2 times.

We will show that $\mathcal{S}' = \mathcal{S}_1 \cup \mathcal{S}_2$, for $\mathcal{S}_1$ covered by $\mathcal{M}_1$ and $\mathcal{S}_2$ covered by $\mathcal{M}_2$.

The submatrix $\mathcal{M}_1$ is a special case of Theorem 2.5.5, so we can take $\mathcal{S}_1$ consisting of all subsets $S \in \mathcal{S}'$ such that $w(S|_1) \leq w + (d-1)$, and $h(S|_1) \leq w + d$. In other words, $\mathcal{S}_1 = \mathcal{S}' \setminus \mathcal{S}_2$, for $\mathcal{S}_2$ being the collection of all subsets $S \in \mathcal{S}_2$ such that $w(S|_1) = w + d$, and because $|S| = w + d$, $h(S|_1) = 1$. Therefore, the coverage of binary $(w+d)$-tuples of weight w is guaranteed for every subset of columns in $\mathcal{S}_1$ using Theorem 2.5.5, and consequently $\mathcal{M}_1$ is a $(w, \mathcal{S}_1; \lambda_1\lambda_3)$-CFF$(st_1, n_1n_2)$.

Now recall that the set $\mathcal{S}_2$ consists of all subsets $S \in \mathcal{S}'$, with $|S| = w + d$, such that $w(S|_1) = w + d$, and $h(S|_1) = 1$. We claim these sets of columns are covered in $\mathcal{M}_2$. Note that this is equivalent to picking every $w + d$ combinations of unique columns of A_2, since A_2 is $(w, d; \lambda_2)$-CFF. The coverage of binary $(w + d)$-tuples of weight w is guaranteed for every subset of columns in $\mathcal{S}_2$, and consequently $\mathcal{M}_2$ is a $(w, \mathcal{S}_2; \lambda_2)$-CFF$(t_2, n_1n_2)$.

Since $A = \begin{pmatrix} \mathcal{M}_1 \\ \mathcal{M}_2 \end{pmatrix}$, we get from Proposition 2.5.3 that A is $(w, \mathcal{S}_1 \cup \mathcal{S}_2; \lambda)$-CFF$(st_1 + t_2, n_1n_2)$. Note that $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}'$, so we cover all the combinations of $w + d$ columns. By Proposition 2.5.2, this implies that A is $(w, d; \lambda)$-CFF.

□

2.6 Applications of cover-free families in cryptography

In this section, we discuss some cryptographic problems and how they were addressed in the literature using CFFs. Most problems presented here have a "fault tolerance" flavor, where the d-CFF help us to, for example, identify d invalid digital signatures, d modified blocks, or even tolerate d malicious users.

It is interesting to see how the incidence matrix of a CFF can be seen from different perspectives depending on the problem we are trying to solve. For aggregation of signatures or batch verification, we use the rows of the matrix to indicate which signatures we should group or test together, but for one-time or multiple times signature these rows now represent pieces of the secret key that will be used to create the signature, while in broadcast communication they represent the pieces of keys that will be shared among participants.

2.6.1 Fault tolerance in digital signature problems

Digital signatures were idealized by Diffie and Hellman [6] in 1976, and a practical implementation was later presented by Rivest, Shamir and Adleman [46], known as the RSA algorithm. Today they are a very popular mechanism that, with very high probability, guarantee integrity and authenticity, and to support non-repudiation of digital data. They are based on public key cryptography, which uses a pair of keys, a public key and a secret key. The secret key is used to generate the signature of a specific message/data, and the corresponding public key is used to verify the signature.

A digital signature scheme usually has three algorithms: the key generation KEYGEN(ℓ) that receives a security parameter ℓ as input and outputs a secret key SK and a public key PK; the signature generation SIGN(m, SK) that receives a message and secret key as input and outputs a signature σ; and signature verification algorithm VERIFY(m, σ, PK), that receives the signature, message, and public key, and outputs 1 if the signature is valid for that m and PK, and 0 otherwise.

As an example, lets consider the RSA signature scheme. Algorithm KEYGEN(ℓ) picks two large prime numbers p, q and computes $n = pq$, n with ℓ bits. Compute $\phi(n) = (p-1)(q-1)$ (Euler's totient function), choose a value e such that $gcd(e, \phi(n)) = 1$, and compute d such that $de \equiv 1 \mod \phi(n)$. The public key pk consists of (e, n) and the secret key sk consists of (d, p, q). For a message m, algorithm SIGN(m, SK) outputs $\sigma = m^d \pmod n$, and algorithm VERIFY(m, σ, PK) returns 1 if and only if $\sigma^e = m \pmod n$ [46].

Now we present how the d-CFFs can be used to provide fault tolerance for some digital signature applications.

2.6.1.1 Modification tolerant signature scheme

When a message m was modified after the signature process, the outcome of the verification algorithm is VERIFY(m, σ, PK) = 0. There are several possible causes for this modification: a simple failure in the transmission process, a malicious modification, an expected modification due to collaborative work, or even redaction of pieces of the message to provide privacy. The original data can be very large or even difficult/impossible to retrieve. Consequently, it may be useful to distinguish the modified portion of the data from the rest. Unfortunately, traditional digital signatures are designed to give a Boolean result, that is, whether the data matches or not its digital signature, and no extra information about the modification can be obtained.

Locating modifications was considered in [22] for digital signatures and in [19] for message authentication codes, both with solutions using d-CFFs. The signature scheme proposed in [22] assumes a message $m = (m[1], \ldots, m[n])$ divided into n blocks, uses a collision resistant hash function h, and a d-CFF(t, n) incidence matrix $\mathcal{M}$ to generate and verify the digital signature.

Signature generation: receives $m = (m[1], \ldots, m[n])$, SK, and proceeds as follows:

1. Compute $h_1 = h(m[1]), \ldots, h_n = h(m[n])$;

2. For each row i in $\mathcal{M}$, compute c_i, which is the concatenation of all h_j such that $\mathcal{M}_{i,j} = 1$, and set $T[i] = h(c_i)$.

3. Compute one extra hash of the entire message $h^* = h(m)$, set $T = (T[1], T[2], \ldots, T[t], h^*)$.

4. Compute the signature $\sigma' = \text{SIGN}(T, SK)$, and output $\sigma = (\sigma', T)$.

The verification process has three possible outputs: 0 if the signature was modified; 1 if signature and message were not modified; a set I with indices of the modified blocks if signature was not modified and message was modified.

Signature verification: receives the message m, $\sigma = (\sigma', T)$, and PK, and proceeds as follows:

1. Ensures $\text{VERIFY}(T, \sigma', PK) = 1$, otherwise output 0;

2. Checks if $h^* = h(m)$. If it matches, return 1, if not, proceeds with the location of modifications.

3. Uses $\mathcal{M}$ and m to compute the concatenations and generate $T[1]', \ldots, T[t]'$ just as the signer did.

4. For all $1 \leq i \leq t$, checks if $T[i] = T[i]'$; the ones that match determine all the block indices $B = \{j : \mathcal{M}_{i,j} = 1\}$ that are intact, and the remaining blocks $I = \{1, \ldots, n\} \setminus B$ are the modified ones. If $|I| \leq d$ return I, otherwise return 0.

This solution proposes a digital signature that carries extra information ($t+1$ hashes), and because they are formed based on a d-CFF, the verification algorithm is able to locate modifications if they occurred. Based on this solution, the authors in [26] propose a scheme that allows for the correction of the modified blocks, when picking small enough block sizes. They also proposed a digital signature scheme that allows for redaction of portions of the message to guarantee privacy. For more details on these two application, see [26][4].

Solution using variable CFFs

We notice we can model this problem using $\mathcal{S}$-CFFs. These variable CFFs can be especially interesting here when modifications are more concentrated in a few regions of the document, and we are interested in performing a more detailed verification inside each one of these regions. We can use $\mathcal{S}$-CFFs to concentrate the location capability on blocks that are closer together, i.e. inside these regions, and relax the property among blocks that are far from each other.

Blocks are divided in sub-blocks and indexed by the sub-block level. Suppose a message divided in blocks of ℓ levels. At level i we have $n_0 n_1 \ldots n_i$ blocks, each divided into n_{i+1} blocks at level $i+1$, for $0 \leq i \leq \ell - 1$, for a total of $n = n_1 n_2 \ldots n_\ell$ blocks. The whole message is considered to be a single block at level 0, so $n_0 = 1$. Let us assume modifications

[4]Included as Chapter 4 in this thesis.

can happen in at most d_i sub-blocks of each block at level $i-1$, $i = 1, \ldots, \ell$. It may make sense to use $d_1 \leq d_2 \leq \ldots \leq d_\ell$ in some applications, but this is not a requirement in general.

We model the error/modification distribution according to $\mathcal{S}$ as follows, and require the use of an $\mathcal{S}$-CFF. Let $C = \{(j_1, j_2, \ldots, j_\ell) : 1 \leq j_i \leq n_i, 1 \leq i \leq \ell\}$ be the index of all sub-blocks, where $(j_1, \ldots, j_i)$ identifies the sub-block at level i. We want an $\mathcal{S}$-CFF that allows the identification of up to $d = d_1 d_2 \ldots d_\ell$ modifications, in at most d_i sub-blocks inside each block at level i. Let X be a set of ℓ-tuples of the form $(x_1, x_2, \ldots, x_\ell)$, for $1 \leq x_i \leq n_i, 1 \leq i \leq \ell$. Define the multi-set $X|_i = \{(x_1, x_2, \ldots, x_i) : (x_1, x_2, \ldots, x_\ell) \in X\}$, $X|_0 = \{()\}$, and $M_i(X) = \{(x_1, x_2, \ldots, x_i) : (x_1, x_2, \ldots, x_{i+1}) \in X|_{i+1}\}$. We define $\mathcal{S}$ as the set containing every subset $S \subseteq C$ such that $h(M_i(S)) \leq d_{i+1} + 1$, for all $0 \leq i \leq \ell - 1$.

If there exists a construction of this $\mathcal{S}$-CFF such that the minimum number of rows $t(\mathcal{S}, n_1 n_2 \ldots n_\ell)$ is smaller than the minimum number of rows $t(d, n_1 n_2 \ldots n_\ell)$ of a traditional d-CFF, then variable CFFs present an improvement for this application. Obviously, $t(\mathcal{S}, n_1 n_2 \ldots n_\ell) \leq t(d, n_1 n_2 \ldots n_\ell)$ is true because with $\mathcal{S}$-CFF we do not need to cover every single combination of $d+1$ columns. Finding constructions that give such $\mathcal{S}$-CFF is a subject for further investigation. Applying the construction from Theorem 2.5.4 repeated times gives a $\mathcal{S}$-CFF $A_1 \otimes A_2 \otimes \ldots \otimes A_\ell$, where A_i is a d_i-CFF, but it seems this construction would not be more efficient than building a $(d_1 d_2 \ldots d_\ell)$-CFF.

2.6.1.2 Batch verification

Batch verification is a method used to reduce the number of operations when we need to verify several digital signatures simultaneously. It was introduced by Fiat [11, 12] for RSA with the main idea that we can perform several modular exponentiations at the cost of one. It has since then been improved and proposed as a solution for many applications, such as vehicular communications [61, 62], wireless sensor networks [50], among others.

A batch verification algorithm receives as an input a list (or batch) B of n signature and message pairs (σ_i, m_i) from potentially different signers, and outputs $\textsc{batch}(B) = 1$ if $\textsc{verify}(m_i, \sigma_i, PK_i) = 1$ for all i, $\textsc{batch}(B) = 0$ otherwise [59]. There are many ways of implementing $\textsc{batch}(B)$, which are closely related to the signature algorithm considered, but the common idea is that it should be much more efficient to run $\textsc{batch}(B)$ than verify each individual signature independently. For the case where $\textsc{batch}(B) = 0$ we know there is at least one invalid signature in B, and it is necessary to determine these invalid signatures. We can do that by verifying each signature individually, which would require a total of n verifications.

Zaverucha and Stinson [59] suggest that applying $\textsc{batch}$ on subsets of B may be a more efficient approach. They propose the use of combinatorial group testing (CGT) to verify the batch of digital signatures, in order to reduce the number of verifications in the identification of invalid signatures. They discuss several approaches using both adaptive and non-adaptive schemes. An adaptive scheme decides the next tests according to the results of the previous ones, while non-adaptive schemes decide all tests at the beginning. For non-adaptive CGT, they suggest the use of d-CFFs. The n columns of the incidence

matrix $\mathcal{M}$ represent the individual signatures σ_j, and each row represents a sub-batch $B_i = \{(m_j, \sigma_j) : \mathcal{M}_{i,j} = 1\}$. For each row i, if $\text{BATCH}(B_i) = 1$, all $\sigma_j \in B_i$ are marked as valid, and after verifying all sub-batches, the remaining signatures are returned as invalid. The authors show that usually, adaptive CGT schemes require a smaller number of tests [59], and all methods presented have a big improvement when compared to individually verifying each signature.

2.6.1.3 Aggregation of signatures

An aggregate signature scheme is a digital signature scheme that allows us to compress the number of digital signatures that need to be stored, transmitted, and verified. Given a set of n digital signatures $\{\sigma_1, \ldots, \sigma_n\}$, an aggregation algorithm combines them into one single aggregate signature $AGG(\sigma_1, \ldots, \sigma_n) = \sigma$. By verifying σ, the verifier can be convinced that the n signatures are valid [1], and if at least one of the n signatures is invalid the entire aggregate σ is compromised. This is similar to the case where $\text{BATCH}(B) = 0$ for batch verification, but in the case of aggregation of signatures, the verifier only has access to σ, which does not carry enough information to determine the invalid signatures [59].

The concept of aggregate signature was first introduced by Boneh et al. [1]. They propose aggregate signatures based on a digital signature scheme using bilinear maps. A bilinear map is a map $e : G \times G \to G_T$, with G, G_T being multiplicative cyclic groups of prime order p, with the bilinear property which states that for all $u, v \in G$ and positive integers a, b, we have that $e(u^a, v^b) = e(u, v)^{ab}$ [1].

A regular signature generation and verification based on bilinear maps consists of the following steps. The secret key is a random $x \in \mathbb{Z}_p$, for p a prime number, and the public key is computed as $v = g^x$, where g is a generator of G. Given a message m, we compute the hash $h = h(m)$ and signature $\sigma = h^x$, with $h, \sigma \in G$. The verification algorithm receives v, m, σ, computes $h = h(m)$ and accepts m as valid if $e(\sigma, g) = e(h, v)$ [1]. This comes from the bilinear property, where $e(\sigma, g) = e(h^x, g) = e(h, g)^x = e(h, g^x) = e(h, v)$. The aggregate signature scheme works as follows. Given n users $u_1, \ldots, u_n$, and their corresponding message and signature pairs (m_i, σ_i) generated as above, the aggregate signature is computed by $\sigma = \prod_{i=1}^{n} \sigma_i$. The verification of σ consists of computing $h_i = h(m_i)$, for all $1 \leq i \leq n$, and ensuring that $e(\sigma, g) = \prod_{i=1}^{n} e(h_i, v_i)$. Other aggregation schemes were later proposed based on a variety of digital signature schemes, such as [29, 35, 36, 40].

Solutions that allow identification of invalid signatures in an aggregate scheme were proposed using d-CFFs [20, 23]. The d-CFFs are used during the aggregation process to increase the expressiveness of σ and allow the identification of up to d invalid signatures. The columns of $\mathcal{M}$ represent the individual signatures σ_j, and each row identifies which signatures are aggregated together: $\sigma[i] = AGG(\sigma_j : \mathcal{M}_{i,j} = 1), 1 \leq i \leq t$. The identification of invalid signatures is similar to the one of batch verification: if $\text{VERIFY}(\sigma[i], \{(m_j, PK_j) : \mathcal{M}_{i,j} = 1\}) = 1$ all respective σ_j are marked as valid, and after running this for all aggregates $\sigma[1], \ldots, \sigma[t]$, the verifier outputs the remaining individual signatures as the invalid ones. The aggregate signature is as big as t individual signatures,

with the additional characteristic of being able to identify up to d invalid signatures. For more information on this application, see [20] [24, 27][5].

2.6.2 One-time and multiple-times signature

So far we presented how d-CFFs are used to identify invalid/modified data in digital signature schemes, which directly matches the purpose of combinatorial group testing. Now we show applications that take advantage of the d-CFF property to create one-time and multiple-times signature schemes. One-time signatures (OTS) were first proposed by Lamport in 1979 [32], and usually have very fast signature and verification algorithms. We may use a key pair to sign only one message, instead of using the same key for several signatures. Multiple-time signatures are a variant of OTS and allow us to sign a predetermined (multiple) number of messages. Usually, the scheme is secure as long as we do not exceed the number of signatures we are expected to generate.

There are several methods in the literature that propose the use of d-CFFs for the construction of these signature schemes. Zaverucha and Stinson [60] point out that each one of these schemes is a variation of the general construction presented next. For the general construction, let $(X, \mathcal{B})$ be a d-CFF with $|X| = t$ and $|\mathcal{B}| = 2^m$, where the subsets $B_M \in \mathcal{B}$ correspond to all possible messages $M \in \{0,1\}^m$ to be signed, and let f be a way-one function.

Key generation: Choose t random ℓ-bit numbers s_i for a security parameter ℓ, and compute $v_i = f(s_i), 1 \leq i \leq t$. Output the secret key $SK = \{s_1, \ldots, s_t\}$ and the public key $PK = \{v_1, \ldots, v_t\}$.

Signature generation: Given a message $M \in \{0,1\}^m$, compute the subset $B_M = \{i_1, \ldots, i_k\} \in \mathcal{B}$ corresponding to M. The signature consists of revealing $\sigma = \{(s_{i_1}, i_1), \ldots, (s_{i_k}, i_k)\}$.

Signature verification: Given a signature $\sigma = \{(s_{i_1}, i_1), \ldots, (s_{i_k}, i_k)\}$ and public key $\{v_1, \ldots, v_t\}$ on M, compute $B_M = \{i_1, \ldots, i_k\}$ and check that $f(s_{i_1}) = v_{i_1}, \ldots, f(s_{i_k}) = v_{i_k}$. If that is the case, return 1; otherwise return 0.

We can use a d-CFF to sign at most d messages. The one-wayness of f together with the d-CFF property guarantee the security of the scheme: it is infeasible to retrieve s_i given v_i, and since $|B_{i_0} \setminus \cup_{j=1}^{d} B_{i_j}| \geq 1$, even after revealing d subsets of secret keys, we need at least one extra s_i to forge a signature on a new message [60].

Variations of this scheme were presented by several researchers, which propose the use of different one-way functions f and a variety of approaches to compute σ from B_M and SK. Bos and Chaum [2] propose an improvement on Lamport's scheme [32] by considering subsets of keys of size $t/2$ where no subset contains any other, which form an optimal 1-CFF with t rows. The scheme given in [60] proposes an OTS based on 1-CFFs and achieves very small signature size by combining the subset of secret keys with modular sums. The scheme and its security is based on the discrete logarithm problem and the 1-CFF to

[5] Given in this thesis as Chapter 3.

compute the signature. Kalach and Safavi-Naini [28] also use 1-CFFs for OTS schemes, and propose the use a compact knapsack function as the one-way function f, so they are able to prove that the scheme is resistant against attacks by quantum computers [28]. For multiple-times signature schemes, different approaches can be used. The most obvious one consists of using a d-CFF, which allows for the signature of at most d messages. Pieprzyk et al. [44] are the first ones to explicitly propose the use of d-CFFs for such a scheme, which works exactly as described above. However, the authors in [60] show that the size of the public and private keys is smaller when we use d instances of 1-CFF schemes rather than one d-CFF. This is related to the lower bound on the number of rows for the cases $d = 1$ and $d \geq 2$. We have $t \approx \log n$ for the first and $t \geq c\frac{d^2}{\log d} \log n$ for the second one, with constant c. So, using d instances of a 1-CFF would give smaller key sizes than using a single d-CFF. Variable CFFs can be explored as a solution here, where columns inside each instance of the 1-CFF only need the 1-CFF property, and any $d + 1$ columns across different instances need the d-CFF property. It remains to be investigated whether it is possible to construct such a variable CFF with less rows than d times the number of rows in a 1-CFF.

2.6.3 Broadcast communication

When a sender wants to transmit a message to a receiver over an insecure channel, usually encryption and authentication are used to ensure it is highly unlikely that somebody else can read and/or modify the message. In this section, we describe schemes where a sender wants to transmit a message to multiple receivers, and not all of them can be trusted. The use of d-CFFs provide a solution to the problem when the number of untrusted receivers does not exceed d.

2.6.3.1 Broadcast authentication

In this scenario, a sender and various receivers agree on secret keys, and these keys are used to guarantee the authenticity of broadcasted messages. The message is broadcast with a *tag*, which is generated using the sender's key and contains information about the integrity and authenticity of the message. Here the tag can be a digital signature or a message authentication code. The receivers can independently verify the authenticity of the received message by verifying the tag using their own keys. A scheme where the sender shares n keys with n receivers may be infeasible, due to the large number of keys that need to be maintained and tags that need to be generated and sent with the message. On the other hand, if users share common keys, there may be malicious users who can get together and use their secret keys and previous communication to create fraudulent messages, which may be accepted by some other users as authentic [34, 48].

Safavi-Naini and Wang [48] propose the use of d-CFFs to manage the distribution of keys, and in this way we avoid that a set of d malicious users can forge a message. Let $(X, \mathcal{B})$ be a d-CFF(t, n), with $X = \{x_1, \ldots, x_t\}$, and n distinct subsets $B_j \subseteq X$. Assume a total of n receivers represented by each one of the subsets B_j, and a set K of t random keys corresponding to X. The scheme works as follows:

Key generation: Generate t random keys $K = \{k_1, \ldots, k_t\}$. Through a secure channel, send K to the sender and send k_i to each receiver R_j such that $x_i \in B_j$, $1 \leq i \leq t$. Each receiver R_j receives a set of $|B_j|$ keys.

Broadcast: For a message m, the sender calculates authentication tags $a_i = f(m, k_i), 1 \leq i \leq t$ and broadcasts $(m, a_1, \ldots, a_t)$.

Verification: R_j accepts $(m, a_1, \ldots, a_t)$ as authentic only if for all his keys $\{k_i : x_i \in B_j\}$, $a_i = f(m, k_i)$.

Because of the d-CFF property, the union of the keys of up to d malicious users is not enough to create a fraudulent message, since every other user has at least one extra key [34, 48].

2.6.3.2 Broadcast encryption

A broadcast encryption scheme allows a server to broadcast encrypted data to a set of n users over an insecure channel. Here we consider the context where the server wants to prevent some of these users from recovering these messages. An example of such application is paid television, where only some privileged users can have access to paid channels.

We can again use d-CFFs to distribute the keys and solve this problem [5, 14]. Let $(X, \mathcal{B})$ be a d-CFF(t, n), with $X = \{x_1, \ldots, x_t\}$, and n subsets $B_j \subseteq X$, and let $K = \{k_1, \ldots, k_t\}$ be the set of secret keys used in the scheme. Each user R_j is represented by a subset $B_j \in \mathcal{B}$ and receives the keys k_i such that $x_i \in B_j$, $1 \leq j \leq n, 1 \leq i \leq t$. When the server wants to send encrypted messages to all but (up to) d users $R_{i_1}, \ldots, R_{i_d}$, it simply needs to encrypt the message using all keys but the ones that belong to those excluded users. Therefore, users $R_{i_1}, \ldots, R_{i_d}$ can not decrypt the message because they do not have the necessary keys, and because of the d-CFF property, this does not compromise the ability of the remaining users to decrypt the content [5, 14].

2.7 Conclusion

In this article, we review several variations of d-CFF definitions, related objects, several constructions from combinatorial designs and codes, and point out their relationships and equivalences. We also define a generalization of CFFs called variable $\mathcal{S}$-CFFs, where instead of a value d, we receive a set $\mathcal{S}$ specifying the coverage we are interested in providing, which corresponds to a more general distribution of defective items. We propose a construction for $\mathcal{S}$-CFFs and show that recursive constructions seen in the literature can be generalized, and the generalization is proved as a special case of the $\mathcal{S}$-CFF construction. It would be an interesting future research to explore applications of $\mathcal{S}$-CFF, and constructions that lead to a smaller set of rows for a given $\mathcal{S}$ than the one obtained with d-CFFs that "contain" $\mathcal{S}$.

Bibliography

[1] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03, 2003, 416–432.

[2] J.N.E. Bos and D. Chaum. Provably Unforgeable Signatures. In: Brickell E.F. (eds) *Advances in Cryptology – CRYPTO 1992*. Lecture Notes in Comput. Sci., vol 740. Springer, Berlin, Heidelberg, 1993, 1–14.

[3] K. A. Bush, Orthogonal arrays of index unity. *The Annals of Mathematical Statistics*, **23** (1952), 426–434.

[4] C. J. Colbourn and J. H. Dinitz, *Handbook of Combinatorial Designs*. Second Edition, Chapman & Hall/CRC, 2007.

[5] P. D'Arco and D. R. Stinson. Fault tolerant and distributed broadcast encryption. In *Proceedings of the 2003 RSA Conference on The Cryptographers' Track*, 2003, 263–280.

[6] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, **22** (1976), 644–654.

[7] A.G. Dyachkov, V.V. Rykov, A.M. Rashad. Superimposed distance codes. *Problems of Control and Information Theory*, **18** (1989), 237–250.

[8] D-Z. Du and F. K. Hwang. *Combinatorial group testing and its applications*. World Scientific, 2000.

[9] P. Erdös, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, **51** (1985), 79 – 89.

[10] P. Erdös and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hanjal, R. Rado, and V. T. Sós (eds), *Infinite and Finite Sets,* volume 11, North-Holland, 1975, 609–627.

[11] A. Fiat. Batch RSA. In *Proceedings on Advances in Cryptology*, CRYPTO '89, 1989, 175–185.

[12] A. Fiat. Batch RSA. *Journal of Cryptology*, **10** (1997), 75–88.

[13] Z. Füredi, On r-Cover-free Families. *Journal of Combinatorial Theory, Series A*, **73** (1996), 172–173.

[14] E. Gafni, J. Staddon and Y. L. Yin, Efficient Methods for Integrating Traceability and Broadcast Encryption, In: Wiener M. (eds) *Advances in Cryptology – CRYPTO 1999*. Lecture Notes in Computer Science, vol 1666, Springer, Berlin, Heidelberg, 1999, 372–387.

[15] L. Gargano, A.A. Rescigno and U. Vaccaro, Efficient Construction of Superimposed Codes, and Related Combinatorial Structures. Manuscript.

[16] L. Gargano, A.A. Rescigno and U. Vaccaro, Low-Weight Superimposed Codes and their Applications,, In: *Proceedings of the 12th International Frontiers of Algorithmics Workshop (FAW'18)*. Lectures Notes in Computer Science vol. 10823, 2018, 197–211.

[17] L. Gargano, A.A. Rescigno and U. Vaccaro, On k-Strong Conflict-Free Multicoloring. In: *Proceedings of the 11th Annual International Conference on Combinatorial Optimization and Applications (COCOA'17)*. Lectures Notes in Computer Science vol. 10628, 2017, 276–290.

[18] C. Gentry and Z. Ramzan. Identity-based aggregate signatures. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006*, 2006, 257–273.

[19] M. T. Goodrich, M. J. Atallah, and R. Tamassia. Indexing information for data forensics. In *International Conference on Applied Cryptography and Network Security*, 2005, 206–221.

[20] G, Hartung, B. Kaidel, A. Koch, J. Koch, and A. Rupp. Fault-tolerant aggregate signatures. In *Public-Key Cryptography – PKC 2016*, 2016, 331–356.

[21] F. K. Hwang, and V. T. Sós. Non-adaptive hypergeometric group testing. *Studia Sci. Math. Hungar.*, 22 (1987), 257–263.

[22] T. B. Idalino, L. Moura, R. F. Custódio, and D. Panario. Locating modifications in signed data for partial data integrity. *Information Processing Letters*, **115** (2015), 731 – 737.

[23] T. B. Idalino. Using combinatorial group testing to solve integrity issues. Master's thesis, Universidade Federal de Santa Catarina, Brazil, 2015.

[24] T.B. Idalino, L. Moura. Efficient Unbounded Fault-Tolerant Aggregate Signatures Using Nested Cover-Free Families In *International Workshop on Combinatorial Algorithms, IWOCA 2018*. Lecture Notes in Computer Science, vol 10979, pages 52–64. Springer, Cham.

[25] T.B. Idalino, L. Moura. Embedding cover-free families and cryptographical applications. *Advances in Mathematics of Communications*, **13** (2019), 629–643.

[26] T.B. Idalino, L. Moura, C. Adams. Modification tolerant signature schemes: location and correction. In preparation for submission, 2019.

[27] T.B. Idalino, L. Moura. Nested Cover-Free Families for Unbounded Fault-Tolerant Aggregate Signatures. Manuscript submitted for publication, 2018.

[28] K. Kalach and R. Safavi-Naini. An efficient post-quantum one-time signature scheme. In *Selected Areas in Cryptography – SAC 2015*. Lecture Notes in Computer Science, vol 9566, pages 331–351, 2016.

[29] J. Katz, A.Y. Lindell. Aggregate Message Authentication Codes. In: *Topics in Cryptology – CT-RSA 2008*. Lecture Notes in Computer Science, vol 4964, pages 155–169. Springer, Berlin, Heidelberg

[30] W. Kautz, R. Singleton Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, **10** (1964), 363–377.

[31] K-M. Kim. Perfect hash families: Constructions and applications. Master's thesis, University of Waterloo, Canada, 2003.

[32] L. Lamport. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[33] P. C. Li, G. H. J. van Rees and R. Wei, Constructions of 2-cover-free families and related separating hash families, *Journal of Combinatorial Designs*, **14** (2006), 423–440.

[34] S. Ling, H. Wang and C. Xing, *Cover-Free Families and Their Applications*, In: *Security in Distributed and Networking Systems*, chapter 4, 2007.

[35] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, B. Waters. Sequential Aggregate Signatures and Multisignatures Without Random Oracles. In: *Advances in Cryptology - EUROCRYPT 2006*. Lecture Notes in Computer Science, vol 4004, pages 465–485. Springer, Berlin, Heidelberg

[36] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, 2004, 74–90.

[37] A. J. Macula. A simple construction of d-disjunct matrices with certain constant weights. *Discrete Mathematics*, **162** (1996), 311 – 312.

[38] C.J. Mitchell, F.C. Piper. Key storage in secure networks. *Discrete Applied Mathematics*, **21** (1988), 215–228.

[39] R.A. Moser, G. Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM*, **57** (2010), 1–15.

[40] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. *ACM Transactions on Storage*, **2** (2006), 107–138.

[41] M. Narasimha and G. Tsudik. Authentication of outsourced databases using signature aggregation and chaining. In Mong Li Lee, Kian-Lee Tan, and Vilas Wuwongse, editors, *Database Systems for Advanced Applications*, 2006, 420–436.

[42] H. Niederreiter, H. Wang, and C. Xing. Function Fields Over Finite Fields and Their Applications to Cryptography. In: Garcia A., Stichtenoth H. (eds) *Topics in Geometry, Coding Theory and Cryptography*. Algebra and Applications, 2006, 59–104.

[43] J. Pastuszak, J. Pieprzyk, and J. Seberry. Codes identifying bad signature in batches. In *INDOCRYPT '00 Proceedings of the First International Conference on Progress in Cryptology, 2000*. Lecture Notes in Computer Science, vol 1997, pages 143–154. Springer.

[44] J. Pieprzyk, H. Wang, and C. Xing. Multiple-time signature schemes against adaptive chosen message attacks. In *Selected Areas in Cryptography*, 2003, 88–100.

[45] E. Porat and A. Rothschild. Explicit nonadaptive combinatorial group testing schemes. *IEEE Transactions on Information Theory*, **57** (2011), 7982–7989.

[46] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, **21** (1978), 120–126.

[47] M. Ruszinó. On the upper bound of the size of the r-cover-free families. *Journal of Combinatorial Theory, Series A*, **66** (1994), 302–310.

[48] R. Safavi-Naini and H. Wang, New results on multi-receiver authentication codes, In: Nyberg K. (eds) *Advances in Cryptology – EUROCRYPT 1998*. Lecture Notes in Computer Science, vol 1403, Springer, Berlin, Heidelberg, 1998, 527–541.

[49] C. Shangguan and G. Ge. New bounds on the number of tests for disjunct matrices *IEEE Transactions on information theory*, **62** (2016), 7518–7521.

[50] K. Shim and C. Park. A secure data aggregation scheme based on appropriate cryptographic primitives in heterogeneous wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, **26** (2015), 2128–2139.

[51] E. Sperner. Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift*, **27** (1928), 544–548.

[52] B. Stevens and E. Mendelsohn, Packing arrays, *Theoretical Computer Science*, **321** (2004), 25–148.

[53] D.R. Stinson. *Combinatorial Designs: Constructions and Analysis*. Springer, New York, 2004.

[54] D.R. Stinson, T. van Trung, and R. Wei. Secure frameproof codes, key distribution patterns, group testing algorithms and related structures. *Journal of Statistical Planning and Inference*, **86** (2000), 595 –617.

[55] D. R. Stinson and R. Wei. Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes. *SIAM J. Discret. Math.* **11** (1998), 41–53.

[56] D.R. Stinson and R. Wei. Generalized cover-free families. *Discrete Mathematics*, **279** (2004), 463 – 477.

[57] D.R. Stinson, R. Wei, and K. Chen. On generalized separating hash families. *Journal of Combinatorial Theory, Series A*, **115** (2008),105 – 120.

[58] R. Wei. On cover-free families. Technical report, Lakehead University, 2006.

[59] G. M. Zaverucha and D. R. Stinson. Group testing and batch verification. In *Proceedings of the 4th International Conference on Information Theoretic Security, ICITS'09*, 2009, 140–157.

[60] G.M. Zaverucha and D.R. Stinson. Short one-time signatures. *Advances in Mathematics of Communications*, **5** (2011), 473–488.

[61] C. Zhang, R. Lu, X. Lin, P. . Ho, and X. Shen. An efficient identity-based batch verification scheme for vehicular sensor networks. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 2008, 246–250.

[62] C. Zhang, P.-H. Ho, and J. Tapolcai. On batch verification with group testing for vehicular communications. *Wireless Networks*, **17** (2011), 1851–1865.

Chapter 3

Efficient unbounded fault-tolerant aggregate signatures using nested cover-free families

A short version of this chapter was published in the proceedings of the 29th International Workshop on Combinatorial Algorithms (IWOCA). After that, we were invited to submit an extended version to the *Theoretical Computer Science* journal. This chapter contains the material of the journal paper version.

IDALINO, T. B.; MOURA, L., *Efficient Unbounded Fault-Tolerant Aggregate Signatures Using Nested Cover-Free Families*. In: International Workshop on Combinatorial Algorithms, IWOCA 2018. Lecture Notes in Computer Science, vol 10979, pages 52–64, 2018.

IDALINO, T. B.; MOURA, L., Nested Cover-Free Families for Unbounded Fault-Tolerant Aggregate Signatures. *Submitted to Theoretical Computer Science*, October 2018.

3.1 Abstract

Aggregate signatures are used to create one short proof of authenticity and integrity from a set of digital signatures. However, one invalid signature in the set invalidates the entire aggregate, giving no information on which signatures are valid. Hartung et al. (2016) propose a fault-tolerant aggregate signature scheme based on combinatorial group testing. Given a bound d on the number of invalid signatures among n signatures to be aggregated, this scheme uses d-cover-free families to determine which signatures are invalid. These combinatorial structures guarantee a moderate increase in the size of the aggregate signature that can reach the best possible compression ratio upper bound of $O(\frac{n}{\log n})$, for fixed d, derived from the information theoretical lower bound on the signature size. The case where the total number of signatures grows dynamically (unbounded scheme) was not satisfactorily solved in their original paper, since explicit constructions had constant compression ratios. In the present paper, we propose efficient solutions for the unbounded scheme, relying on sequences of d-cover-free families that we call *nested families*. Some of our constructions yield high compression ratio close to the best known upper bound. We also propose the use of $(d; \lambda)$-cover-free families to support the loss of up to $\lambda - 1$ parts of the aggregate.

3.2 Introduction

In cryptography, aggregate signature schemes allow us to combine a set of digital signatures into a single one, which can be used as proof of integrity and authenticity of a possibly large set of data. This solution is especially useful for applications that manage a large quantity of data and digital signatures, since it can save on communication and storage, as well as improve the signature verification process. A few examples of such applications are outsourced databases [13], sensor networks [9], secure logging [11], certificate chains [1], and vehicular communication [16], among others.

The verification of an aggregate signature outputs a positive result only if the entire set of signatures is valid. If we have at least one faulty signature in the set, the proof of integrity and authenticity of all the data involved is invalidated. This happens because when a set of signatures is aggregated into one, this operation does not preserve enough information in order to identify the exact set of invalid signatures. In order to address this shortcoming, Hartung et al. [4] propose a fault-tolerant scheme using d-cover-free families (d-CFF). This scheme generates a more expressive aggregate signature, that can tolerate up to d invalid signatures and identify all the valid ones.

Combinatorial group testing [2] deals precisely with this type of problem of determining a set of defective items via testing groups of items. A d-CFF is a collection of subsets such that no subset is contained in the union of any d other sets in the family. A 2-CFF is shown in Fig. 3.1, where each column represents the incidence vector of a set; in group testing, the rows of the matrix represent the groups and its columns, the items. By inspection, we can see that if up to 2 items are defective, this set of defectives can be determined from

the outcomes of the group tests. For example, if only items 1 and 2 are defective, tests $1-6$ fail and tests $7-9$ pass, and the defectives can be determined from this outcome.

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 3.1: A 2-cover-free family 2-CFF$(9, 12)$.

What is distinctive in the signature aggregation application is that the groups are formed at a different stage than testing: groups of signatures are combined at the aggregation stage and their testing is done at the verification stage. Another distinctive characteristic of signature aggregation is that we do not have a bound on the number of signatures that need to be aggregated; for example, secure logging applications cannot always predict how many log entries will be saved, and outsourced databases do not always have control over the amount of data that will be inserted. This requires an unbounded fault-tolerant aggregation scheme relying on matrix growth, which, as pointed out by Hartung et al. [4], requires a special sequence of d-CFF matrices rather than a single one. Here, the concept of compression ratio is important. Consider the total number of signatures n (number of columns) and the size of the aggregate signature $s(n)$ (number of rows) at each stage. The compression ratio is equal to $\rho(n)$ iff $\frac{n}{s(n)}$ is $\Theta(\rho(n))$. The best known lower bounds on $s(n)$ [3] show that $\rho(n)$ is $O(\frac{n}{(d^2/\log d)\log n})$.

Monotone families of d-CFFs were defined in [4] to accomplish unbounded aggregation. The explicit constructions of monotone families given in [4] yield aggregate signatures with length linear in n and thus *constant* compression ratios; the authors pose the open problem of finding better monotone families in order to achieve a more efficient unbounded scheme. In [7], the present authors find monotone families with $\rho(n) = n^{1-1/c}$, for every integer $c \geq 2$ and fixed $d \geq 1$. In the present work, we approach unbounded aggregate signatures by defining a more flexible sequence of d-CFFs that we call a *nested family*, and which generalizes monotone families. We then propose various constructions of nested families, some with compression ratio close to the best known upper bound. For $d = 1$, our construction reaches $\rho(n) = \frac{n}{\log n}$, while for $d \geq 2$ we propose two constructions, the first with ratio $\rho(n) = n^{1-1/e}$, for some constant $e \geq 2$, and the second with $\rho(n) = \frac{n}{(b\log_2 n)^{\log_2 \log_2 n + D}}$, for some constants b and D. Although we focus on the specific problem of fault-tolerant digital signature aggregation, our new constructions can be applied to other combinatorial group testing problems where the number of items to be tested grows dynamically. An earlier version of the present work appeared in [6].

In Section 3.3, we present the background on fault-tolerant aggregate signatures intro-

duced in [4] and on d-CFF constructions. In Section 3.4, we define nested families and present unbounded signature aggregation algorithms that use these new families. In Section 3.5 , we provide explicit constructions of such families that yield unbounded aggregate schemes with better compression ratios than previously known. In Section 3.6, we generalize our d-CFF constructions to build $(d; \lambda)$-CFFs, which are defined there, and discuss how they can also support faults on signature transmission. These extensions in Section 3.6 are not mentioned in the early version of this paper [6]. Section 3.7 includes conclusions and future work.

3.3 Background on fault-tolerant schemes

In this section, we present the fault-tolerant aggregate signature scheme by Hartung et al. [4]. We summarize the concepts introduced by the authors [4, Sections 1,3,4], including the necessary formalization to contextualize our construction. Then, we present background and recursive constructions of cover-free families.

3.3.1 Fault-Tolerant Aggregate Signature

Let $\{\sigma_1, \ldots, \sigma_n\}$ be a set of n signatures and let $C = \{(pk_1, m_1), \ldots, (pk_n, m_n)\}$ be their corresponding pairs of public key and message. A traditional aggregate signature scheme consists of combining all n signatures together in one aggregate signature σ, which can be as small as a single digital signature [1]. By verifying only σ we can ensure the integrity and authenticity of the entire set C. If all signatures $\{\sigma_1, \ldots, \sigma_n\}$ are correctly formed from C, the signature verification outputs 1, but if at least one σ_i does not match its corresponding pair (pk_i, m_i), the verification outputs 0. In more detail, Boneh et al. [1] define an aggregate signature scheme with four algorithms:

1. **KeyGen**(1^κ) takes security parameter κ and creates a key pair (pk, sk).
2. **Sign**(sk, m) creates a signature for message m using secret key sk.
3. **Agg**$(C_1, C_2, \sigma_1, \sigma_2)$ takes two multisets C_1 and C_2 of pairs of public key and message and their corresponding signatures σ_1 and σ_2, and outputs an aggregate signature σ that certifies the integrity and authenticity of $C = C_1 \cup C_2$.
4. **Verify**(C, σ) takes a multiset of public key and message pairs and its aggregate signature σ. Outputs 1 if the signature is valid and 0 otherwise.

The security of the scheme is based on the difficulty of an adversary to forge a signature of a chosen message after performing q signature queries to an oracle. More specifically, the aggregate scheme is (t, q, ϵ)-secure if there is no adversary $\mathcal{A}$ capable of winning the game with probability at least ϵ, performing at most q queries to the oracle, and running in time at most t [1, 4].

As an example of an aggregate signature scheme, let us consider the one proposed by Boneh et al. [1], which is based on bilinear maps. A bilinear map is a map $e : G \times G \to G_T$, with G, G_T being multiplicative cyclic groups of prime order p, with the bilinear property which states that for all $u, v \in G$ and positive integers a, b, we have that $e(u^a, v^b) = e(u, v)^{ab}$ [1]. The regular signature generation and verification consist of the following steps. Algorithm $\mathbf{KeyGen}(1^\kappa)$ generates a random secret key $x \in \mathbb{Z}_p$, and the public key is computed as $v = g^x$, where g is a generator of G. Given a message m, $\mathbf{Sign}(x, m)$ computes the hash $h = h(m)$ and signature $\sigma = h^x$, with $h, \sigma \in G$. The verification algorithm receives v, m, σ, computes $h = h(m)$ and accepts m as valid if $e(\sigma, g) = e(h, v)$ [1]. This comes from the bilinear property, where $e(\sigma, g) = e(h^x, g) = e(h, g)^x = e(h, g^x) = e(h, v)$. The algorithm $\mathbf{Agg}(C_1, C_2, \sigma_1, \sigma_2)$ outputs $\sigma = \sigma_1 \times \sigma_2$, which certifies the integrity and authenticity of $C = C_1 \cup C_2 = \{(v_1, m_1), \ldots, (v_n, m_n)\}$. In general, given n signatures $\sigma_1, \ldots, \sigma_n$, the aggregation consists of $\sigma = \prod_{i=1}^{n} \sigma_i$ [1]. The verification algorithm $\mathbf{Verify}(C, \sigma)$ computes $h_i = h(m_i)$ for all messages $m_i, 1 \leq i \leq n$, and ensures that $e(\sigma, g) = \prod_{i=1}^{n} e(h_i, v_i)$. Of course, in this scheme one invalid pair (v_i, m_i) would make σ invalid. In the remaining of this section we discuss fault-tolerant schemes.

In order to provide fault-tolerance, the signature verification needs to output a list of valid signatures instead of just 0 or 1, so the scheme should provide *list verification* instead of boolean verification. To describe this scheme, Hartung et al. [4] use the concepts of "claim" $c = (pk, m)$ as a pair of public key pk and message m, and "claim sequence" C as a sequence of claims. A claim sequence requires an order among the claims, so each position i may contain one claim or a placeholder $\perp$. Two claim sequences C_1, C_2 are defined as *exclusively mergeable* if for all i, $C_1[i] = \perp$ or $C_2[i] = \perp$, and for a C_1 and C_2 of length k and l, with $k \geq l$, $C_1 \sqcup C_2 = (c_1, \ldots, c_k)$ is defined by

$$c_i = \begin{cases} C_1[i], \text{ if } C_2[i] = \perp, C_2[i] = C_1[i] \text{ or } i > l \\ C_2[i], \text{ otherwise.} \end{cases}$$

Let $C = (c_1, \ldots, c_n)$ be a claim sequence and $b \in \{0, 1\}^n$ be a bit sequence that specifies a selection of indices. Then $C[b]$ denotes a subsequence of C having claim c_j in position j whenever $b[j] = 1$, and $\perp$ otherwise. The claim sequence is used as a way to impose an order to the claims, which is required for the process of generating a fault-tolerant aggregation signature and its verification. For more details, see Hartung et al. [4, Section 3]. The definition of aggregate signature with list verification is given below.

Definition 3.3.1. *[Hartung et al. [4]] An aggregate signature scheme with list verification consists of four algorithms Σ:*

1. ***KeyGen**(1^κ) takes security parameter κ and creates a key pair (pk, sk).*
2. ***Sign**(sk, m) creates a signature for message m using secret key sk.*
3. ***Agg**$(C_1, C_2, \tau_1, \tau_2)$ takes two exclusively mergeable claim sequences C_1 and C_2 and their corresponding signatures τ_1 and τ_2, and outputs an aggregate signature τ that certifies the integrity and authenticity of the sequence $C = C_1 \sqcup C_2$.*

4. ***Verify***(C, τ) *takes a claim sequence* C *and its aggregate signature* τ *as input. Outputs the set of valid claims, which can be all the elements in* C *or even the empty set.*

Consider a claim sequence C with n claims, their corresponding signatures $\sigma_1, \ldots, \sigma_n$ with at most d invalid ones, and the aggregate signature τ. The aggregate signature scheme Σ is tolerant against d errors if $\Sigma.\mathbf{Verify}(C, \tau)$ outputs the set of claims that have valid signatures. Therefore, a *d-fault-tolerant* aggregate signature scheme is an aggregate signature scheme with list verification with a tolerance against d errors. From now on we will use σ to represent a standard aggregate signature, and τ for signatures of a fault-tolerant scheme.

The security of the scheme is presented by Hartung et al. [4]. An adversary advantage on forging an aggregate signature scheme with list verification is given by its probability of winning the following game:

- **Setup:** $\mathcal{A}$ receives a random public key pk.
- **Queries:** $\mathcal{A}$ adaptively requests signatures of messages m_i to an oracle using pk and receives $\tau_i = Sign(sk, m_i)$.
- **Response:** $\mathcal{A}$ outputs a claim sequence C^* and an aggregate signature τ^*.

The adversary wins if there is a claim $c^* = (pk, m^*)$ in C^* that belongs to the set of valid claims output by $\mathbf{Verify}(C^*, \tau^*)$, and the signature of m^* was never requested to the signature oracle. The scheme is said to be (t, q, ϵ)-secure if there is no adversary that runs in time t, performs at most q queries and wins the game above with probability at least ϵ.

Hartung et al. [4] use d-CFFs to instantiate a generic fault-tolerant aggregate signature scheme. Let $\mathcal{M}$ be a $t \times n$ binary incidence matrix of a d-CFF, where each column is the characteristic vector of a set in the family. Given $\mathcal{M}$ and a set $\{\sigma_1, \ldots, \sigma_n\}$ of signatures to be aggregated, each column j represents a signature σ_j, and the rows of $\mathcal{M}$ indicate which signatures will be aggregated together. We are able to identify all valid signatures as long as the number of invalid ones does not exceed a bound d.

Hartung et al. [4, Section 4] define a fault-tolerant aggregate signature scheme based on an ordinary aggregate signature scheme Σ that supports claims, claim sequences, and the empty signature λ as input. We denote $\mathcal{M}_i$ as row i of matrix $\mathcal{M}$, so $C[\mathcal{M}_i]$ represents the corresponding subsequence of a claim sequence C. This scheme inherits the security of Σ, and its algorithms are presented below:

1. $\mathbf{KeyGen}(1^\kappa)$ creates a key pair (pk, sk) using **KeyGen** from Σ and security parameter κ.
2. $\mathbf{Sign}(sk, m)$ receives a secret key and message, and outputs the signature given by $\Sigma.\mathbf{Sign}(sk, m)$.
3. $\mathbf{Agg}(C_1, C_2, \tau_1, \tau_2)$ takes two exclusively mergeable claim sequences C_1 and C_2 and corresponding signatures τ_1 and τ_2, and proceeds as follows:

(a) If one or both of the claim sequences C_k ($k \in \{1, 2\}$) contain only one claim c, τ_k is an individual signature σ_k. We initialize σ_k as τ_k and expand it to a vector as follows, with j equal to the index of c in C_k:

$$\tau_k[i] = \begin{cases} \sigma_k, & \text{if } \mathcal{M}[i, j] = 1, \\ \lambda, & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, t$.

(b) Once τ_1 and τ_2 are both vectors, we aggregate them, position by position, according to the incidence matrix $\mathcal{M}$:

$$\tau[i] = \Sigma.\mathbf{Agg}(C_1[\mathcal{M}_i], C_2[\mathcal{M}_i], \tau_1[i], \tau_2[i]), \text{ for } i = 1, \ldots, t.$$

(c) Output τ, which certifies the integrity and authenticity of the sequence $C = C_1 \sqcup C_2$.

4. **Verify**(C, τ) takes a claim sequence C and the aggregate signature τ, and outputs the set of valid claims. Computes $b_i = \Sigma.\mathbf{Verify}(C[\mathcal{M}_i], \tau[i])$ for each $1 \leq i \leq t$ and outputs the set of valid claims consisting of the union of each $C[\mathcal{M}_i]$ such that $b_i = 1$.

The following theorems are from Hartung et al. [4, Section 4] and address the security and correctness of the scheme. For details regarding their proofs, see [4].

Theorem 3.3.2 (Hartung et al. [4])**.** *Let Σ' be the aggregate signature scheme with list verification presented above. If Σ' is based on a d-CFF, then it is correct and tolerant against up to d errors.*

Sketch of Proof. Considering the aggregation algorithm, we need to ensure that the verification algorithm computes the correct set of valid claims, assuming there are at most d invalid ones. The d-CFF property guarantees that if a claim is valid, it will be included in an aggregation $\tau[i]$ that results in $b_i = 1$, since there will be a row i that avoids all invalid claims and contains this valid claim. Each aggregation $\tau[i]$ that contains an invalid claim will result in $b_i = 0$. Therefore, $\Sigma'.\mathbf{Verify}(C, \tau)$ is correct. □

Theorem 3.3.3 (Hartung et al. [4])**.** *If Σ is a (t, q, ϵ)-secure aggregate signature scheme, then the aggregate signature scheme with list verification above is (t', q, ϵ)-secure, with t' approximately the same as t.*

The following example consists of $n = 10$ signatures aggregated according to the 1-CFF$(5, 10)$ matrix $\mathcal{M}$, which allows us to identify all valid signatures as long as we have at most one invalid signature. For instance, if σ_1 is invalid, $\tau[1]$ and $\tau[2]$ will fail, but $\tau[3], \tau[4], \tau[5]$ prove the validity of $\sigma_i, 2 \leq i \leq 10$.

$$\mathcal{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \rightarrow \begin{array}{l} \tau[1] = Agg(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \\ \tau[2] = Agg(\sigma_1, \sigma_5, \sigma_6, \sigma_7) \\ \tau[3] = Agg(\sigma_2, \sigma_5, \sigma_8, \sigma_9) \\ \tau[4] = Agg(\sigma_3, \sigma_6, \sigma_8, \sigma_{10}) \\ \tau[5] = Agg(\sigma_4, \sigma_7, \sigma_9, \sigma_{10}) \end{array}$$

Now we exemplify aggregation using the claim sequences. Assume the d-CFF above, assume $n = 10$ digital signatures and the corresponding claims $c_i = (pk_i, m_i)$, for $1 \leq i \leq n$. Consider two claim sequences and their corresponding aggregate signatures

$$C_1 = (\perp, c_2, \perp, \perp, c_5, \perp, \perp, \perp, \perp, \perp), C_2 = (\perp, \perp, \perp, c_4, \perp, \perp, \perp, \perp, c_9, c_{10})),$$

$$\tau_1 = [\sigma_2, \sigma_5, Agg(\sigma_2, \sigma_5), \lambda, \lambda], \tau_2 = [\sigma_4, \lambda, \sigma_9, \sigma_{10}, Agg(\sigma_4, \sigma_9, \sigma_{10})],$$

where $Agg(\sigma_{i_1}, \ldots, \sigma_{i_k})$ represents the output of a call to algorithm $\Sigma.Agg$. We compute τ and C as follows.

$$\tau[1] = \Sigma.Agg((\perp, c_2, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp), (\perp, \perp, \perp, c_4, \perp, \perp, \perp, \perp, \perp, \perp), \tau_1[1], \tau_2[1]),$$

$$\tau[2] = \Sigma.Agg((\perp, \perp, \perp, \perp, c_5, \perp, \perp, \perp, \perp, \perp), (\perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp), \tau_1[2], \tau_2[2]),$$

$$\tau[3] = \Sigma.Agg((\perp, c_2, \perp, \perp, c_5, \perp, \perp, \perp, \perp, \perp), (\perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, c_9, \perp), \tau_1[3], \tau_2[3]),$$

$$\tau[4] = \Sigma.Agg((\perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp), (\perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, c_{10}), \tau_1[4], \tau_2[4]),$$

$$\tau[5] = \Sigma.Agg((\perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp), (\perp, \perp, \perp, c_4, \perp, \perp, \perp, \perp, c_9, c_{10}), \tau_1[5], \tau_2[5]),$$

$$C = C_1 \sqcup C_2 = (\perp, c_2, \perp, c_4, c_5, \perp, \perp, \perp, c_9, c_{10}).$$

The idea of fault tolerance in signature aggregation using CFFs appeared independently in the master's thesis of the first author as *level-d signature aggregation* [5, Chapter 5]. A related application of CFFs to modification tolerant digital signatures can be found in [8].

3.3.2 Cover-free family constructions

Cover-free families (CFFs) are combinatorial structures studied in the context of combinatorial group testing, and frequently used in scenarios where we need to test a set of n elements to identify up to d invalid ones. We use them to combine these elements into a few groups, and test the groups instead of each element.

Definition 3.3.4. *A set system $\mathcal{F} = (X, \mathcal{B})$ consists of a set $X = \{x_1, \ldots, x_t\}$ with $|X| = t$, and a collection $\mathcal{B} = \{B_1, \ldots, B_n\}$ with $B_i \subseteq X, 1 \leq i \leq n$, and $|\mathcal{B}| = n$. A d-cover-free family, denoted d-CFF(t, n), is a set system such that for any subset $B_{i_0} \in \mathcal{B}$ and any other d subsets $B_{i_1}, \ldots, B_{i_d} \in \mathcal{B}$, we have*

$$B_{i_0} \nsubseteq \bigcup_{j=1}^{d} B_{i_j} \tag{3.1}$$

We can represent $\mathcal{F}$ as a $t \times n$ binary incidence matrix $\mathcal{M}$ by considering the characteristic vectors of subsets in $\mathcal{B}$ as columns of $\mathcal{M}$. More precisely, $\mathcal{M}_{i,j} = 1$ if $x_i \in B_j$, and $\mathcal{M}_{i,j} = 0$ otherwise. We will interchangeably say that $\mathcal{M}$ is d-CFF when its corresponding set system is d-CFF. Note that if $\mathcal{M}$ is d-CFF, then a matrix obtained by row and column permutations is also d-CFF.

An equivalent definition of d-CFF is based on the existence of permutation submatrices of dimension $d + 1$ [12]. A permutation matrix is an $n \times n$ binary matrix with exactly one "1" per row and per column, or in other words, it is obtained by permuting the rows of the identity matrix.

Proposition 3.3.5. *A matrix $\mathcal{M}$ is d-CFF if and only if any set of $d+1$ columns contains a permutation sub-matrix of dimension $d+1$.*

Proof. Take $d+1$ columns $\{j_0, j_1, \ldots, j_d\}$ of $\mathcal{M}$. Property (3.1) is true for j_l in place of i_0 and $J = \{j_0, j_1, \ldots, j_d\} \setminus j_l$ in place of $\{i_1, \ldots, i_d\}$ if and only if there exists a row i where $\mathcal{M}_{i,j_l} = 1$ and $\mathcal{M}_{i,j_k} = 0$ for $j_k \in J$. This is equivalent to a permutation sub-matrix of dimension $d+1$ in columns $\{j_0, j_1, \ldots, j_d\}$. □

The next propositions state relationships between sub-matrices with respect to d-CFF properties. Their proofs follow directly from Definition 3.3.4.

Proposition 3.3.6. *Let $\mathcal{M}$ be a matrix and let $\mathcal{M}'$ be a sub-matrix of $\mathcal{M}$ formed by some of its columns. If $\mathcal{M}$ is d-CFF, then $\mathcal{M}'$ is also d-CFF.*

Proposition 3.3.7. *Let $\mathcal{M}$ be a matrix and let $\mathcal{M}'$ be a sub-matrix of $\mathcal{M}$ formed by some of its rows. If $\mathcal{M}'$ is d-CFF, then $\mathcal{M}$ is d-CFF.*

For $d \geq 2$, a lower bound on t is given by $t \geq c\frac{d^2}{\log d}\log n$, for a constant c [3]. For $d = 1$ the best possible 1-CFF can be constructed using Sperner's theorem.

Theorem 3.3.8. *(Sperner [15]) Let $\mathcal{B}$ be a collection of subsets of $\{1, \ldots, t\}$ such that $B_1 \not\subseteq B_2$ for all $B_1, B_2 \in \mathcal{B}$. Then $|\mathcal{B}| \leq \binom{t}{\lfloor t/2 \rfloor}$. Moreover, equality holds when $\mathcal{B}$ is the collection of all the $\lfloor t/2 \rfloor$-subsets of $\{1, \ldots, t\}$.*

Corollary 3.3.9. *Given n and $d = 1$, let $t(n)$ be the smallest integer such that 1-CFF($t(n), n$) exists. Then, $t(n) = \min\{s : \binom{s}{\lfloor s/2 \rfloor} \geq n\}$.*

Proof. Property (3.1) is equivalent to the family of sets having the Sperner property. Build each column of the matrix from the characteristic vector of a distinct $\lfloor t/2 \rfloor$-subset of a t-set, with $t = \min\{s : \binom{s}{\lfloor s/2 \rfloor} \geq n\}$. Theorem 3.3.8 guarantees $t(n) = t$. □

The value t grows as $\log_2 n$ as $n \to \infty$, which meets the information theoretical lower bound on the amount of bits necessary to uniquely distinguish the n inputs, yielding an optimal construction.

Other constructions of CFF exist for larger d; see Zaverucha and Stinson [17, Section 3.2] for a discussion on how other combinatorial objects yield good CFF methods depending on the relation of d and n. In particular, Porat and Rothschild [14] give a construction that yields $t = c(d+1)^2 \log n$ for a constant c, which for fixed d is optimal in terms of meeting a lower bound $\Theta(\log n)$ (see [17, Theorem 3]). Next we give generalizations of two constructions by Li et al. [10, Theorems 3.4 and 3.5] that allows us to build larger d-CFFs from smaller ones. To the best of our knowledge, these results have only been stated for $d = 2$.

Definition 3.3.10 (Kronecker product)**.** *Let A_k be an $m_k \times n_k$ binary matrix, for $k = 1, 2$, and $\mathbf{0}$ be the matrix of all zeroes with the same dimension as A_2. The product $P = A_1 \otimes A_2$ is a binary matrix such that*

$$P = \begin{pmatrix} P_{1,1} & \dots P_{1,n_1} \\ \vdots & \vdots \\ P_{m_1,1} & \dots P_{m_1,n_1} \end{pmatrix} \text{ where } P_{i,j} = \begin{cases} A_2, & \text{if } A_{1_{i,j}} = 1, \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

The following theorem generalizes a construction by Li et al. [10] given for $d = 2$.

Theorem 3.3.11. *Let A_1 be a d-CFF(t_1, n_1) and A_2 be a d-CFF(t_2, n_2), then $C = A_1 \otimes A_2$ is a d-CFF$(t_1 t_2, n_1 n_2)$.*

Proof. It is enough to show that every sub-matrix of C formed by $d+1$ of its columns is d-CFF. We first establish the existence of a special sub-matrix in C. We define a *block* of C as any set of n_2 consecutive columns that were created by a single column of A_1 in the Kronecker product (Definition 3.3.10). Since A_1 is d-CFF, Proposition 3.3.5 implies that any l blocks of C, $l \leq d+1$, contain a set of rows that after permutations are of the form

$$\overline{C} = \begin{pmatrix} A_2 & \dots & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & \dots & A_2 \end{pmatrix}$$

In other words, $\overline{C}$ is a "generalized permutation sub-matrix". Let $\mathcal{C} = \{c_0, c_1, \dots, c_d\}$ be $d+1$ columns indices of C and let B be the sub-matrix of C restricted to these columns. These columns belong to l blocks of C, where $l \leq d+1$. Therefore, we can find a sub-matrix $\overline{C}$ of C of the form presented above with exactly l blocks. Let D_1, D_2 be such that $D_1 \cup D_2 = \{c_1, \dots, c_d\}$, where D_1 consists of the columns that belong to the same block as c_0 and D_2 consists of all the other columns. Let i be a row such that $B_{i,c_0} = 1$ and $B_{i,c_k} = 0$, for all $c_k \in D_1$; since A_2 is d-CFF, such i exists. Because $B_{i,c_0} = 1$, we know that $B_{i,c_l} = 0$ for all $c_l \in D_2$ since $\overline{C}$ is a generalized permutation sub-matrix. Since the same is true for every column in $\mathcal{C}$, we obtain a permutation matrix of order $d+1$. By Proposition 3.3.5, B is d-CFF. Since this is true for every choice of B, C is d-CFF. □

Next we present a construction of d-CFFs based on another result by Li et al. [10] for $d = 2$. This construction gives a better result than the one from Theorem 3.3.11 for the cases where $s < \frac{t_1 t_2 - t_2}{t_1}$.

Construction 3.3.12. *Let A_1 be a $t_1 \times n_1$ binary matrix, A_2 be a $t_2 \times n_2$ binary matrix, and B be a $s \times n_2$ binary matrix. Create a matrix $P = B \otimes A_1$ as in Definition 3.3.10. This results in P with n_2 "blocks" of n_1 columns each. For each column in block i, append the ith column of A_2, for $1 \leq i \leq n_2$. Call* Const1(A_1, A_2, B) *the matrix obtained.*

Theorem 3.3.13. *Let $d \geq 2$, A_1 be a d-CFF(t_1, n_1), A_2 be a d-CFF(t_2, n_2), and B be a $(d-1)$-CFF(s, n_2). Then* C $:=$ **Const1**(A_1, A_2, B) *is a d-CFF$(st_1 + t_2, n_1 n_2)$.*

Proof. We need to prove that C is a d-CFF. Let $\mathcal{C} = \{c_0, c_1, \dots, c_d\}$ be $d+1$ column indices of C. Consider the n_2 blocks of n_1 consecutive columns of C originating each from a single

column of B. We divide this proof into 2 cases.
Case 1: The $d+1$ columns belong to $d+1$ different blocks. By the construction, each column considered contains in its lower part one distinct column from A_2. Since A_2 is d-CFF, Proposition 1 and 2 guarantees that C is d-CFF.
Case 2: The $d+1$ columns belong to l different blocks, $1 \leq l \leq d$. By considering the first part of Construction 3.3.12, where we compute $B \otimes A_1$, we note that since B is a $(d-1)$-CFF, Proposition 3.3.5 implies we can find a generalized permutation submatrix of C with a total of l blocks, covering $\{c_0, c_1, \ldots, c_d\}$, which after permutations has the following form

$$\overline{C} = \begin{pmatrix} A_1 & \ldots & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & \ldots & A_1 \end{pmatrix}.$$

Now let $\overline{C}_1, \overline{C}_2, \ldots, \overline{C}_l$ be the set of column indices corresponding to each block of $\overline{C}$, and $\overline{R}_1, \overline{R}_2, \ldots, \overline{R}_l$ be sets with t_1 consecutive row indices each.

Since A_1 is d-CFF, and $|\overline{C}_i \cap \mathcal{C}| \leq d+1$, Proposition 3.3.5 implies that the matrix formed by column indices $\overline{C}_i \cap \mathcal{C}$ and row indices $\overline{R}_i$ contains a permutation submatrix which after permutations gives an identity matrix I_i of dimension $|\overline{C}_i \cap \mathcal{C}|$. Now given the form of $\overline{C}$ we know $\overline{C}_{i_1,j_1} = 0$, for all $i_1 \in \overline{R}_i$, for all $j_1 \in \overline{C}_j \cap \mathcal{C}$, where $1 \leq j \leq l, i \neq j$.

As a consequence, there exists a submatrix of $\overline{C}$ that after permutation is of the form

$$\begin{pmatrix} I_1 & \ldots & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & \ldots & I_l \end{pmatrix},$$

which is an identity matrix of dimension $d+1$. Therefore, C is d-CFF$(st_1 + t_2, n_1 n_2)$. □

As a corollary, it is possible to obtain a previous result by Li et al. [10] for the specific case of $d = 2$.

Corollary 3.3.14. *[10, Theorem 3.5] Suppose there exists a* 2*-CFF*(t_1, n_1) *and a* 2*-CFF*(t_2, n_2)*, then there exists a* 2*-CFF*$(st_1 + t_2, n_1 n_2)$ *for any* s *satisfying* $\binom{s}{\lfloor \frac{s}{2} \rfloor} \geq n_2$*.*

Proof. Let A_1 be a 2-CFF(t_1, n_1), A_2 be a 2-CFF(t_2, n_2). Let s satisfy $\binom{s}{\lfloor \frac{s}{2} \rfloor} \geq n_2$, and B consist of n_2 columns of the 1-CFF given in the proof of Corollary 3.3.9. By Theorem 3.3.13, Const1(A_1, A_2, B) is a 2-CFF$(st_1 + t_2, n_1 n_2)$. □

3.4 Our general unbounded scheme based on nested families

If we fix a d-CFF in a fault-tolerant aggregate signature scheme, we set a bound on the number of signatures n that can be aggregated, which may not be known in advance (eg.

applications in secure logging and dynamic databases). These unbounded applications require a sequence of d-CFFs that allows the increase of n as necessary. Several of these applications also deal with a large number of signatures, and it may not be possible to save each one of them individually. So besides requiring increasing size, the d-CFF should also take into consideration that once aggregated, the individual signatures may not be available anymore. This raises the need for a sequence of d-CFFs to support all these requirements.

In order to address this problem, Hartung et al. [4] propose the notion of a fault-tolerant unbounded scheme based on what they call a *monotone* family of d-CFFs. It consists of using a CFF incidence matrix $\mathcal{M}^{(1)}$ until its maximum n is achieved, and then jumping to the next matrix. Each new matrix $\mathcal{M}^{(l+1)}$ contains $\mathcal{M}^{(l)}$ in its upper left corner, containing a matrix of zeroes below it, in a sequence that presents a monotonicity property.

In this section, we extend the notion of unbounded aggregation and suggest a more flexible sequence of d-CFFs, called *nested* family. We show that the nested property is enough to be able to discard individual signatures after they are aggregated. This allows us to construct an infinite sequence of d-cover-free families with a better compression ratio than the ones currently known for monotone families.

In the remainder of the paper, an infinite sequence $a_1, a_2, \ldots$ is compactly denoted as $(a_l)_l$ for sets and $(a^{(l)})_l$ for matrices.

Definition 3.4.1. *Let $(\mathcal{M}^{(l)})_l$ be a sequence of incidence matrices of d-cover-free families $(\mathcal{F}_l)_l = (X_l, \mathcal{B}_l)_l$, where the number of rows and columns of $\mathcal{M}^{(l)}$ are denoted by rows(l) and cols(l), respectively. $(\mathcal{M}^{(l)})_l$ is a* nested family *of incidence matrices of d-CFFs, if $X_l \subseteq X_{l+1}$, rows(l) $\leq$ rows(l+1), and cols(l) $\leq$ cols(l+1), and*

$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y \\ Z & W \end{pmatrix}$$

where W, Y, Z are 0-1 matrices of appropriate dimensions, and each row of Z is one of the rows of $\mathcal{M}^{(l)}$, or a row of all zeros, or a row of all ones.

Note that a monotone family defined in [4] is a special case of nested family, where $Z = 0$. The authors in [4] use monotone families to achieve unbounded aggregation in the following way. For each $1 \leq i \leq cols(l)$, if $B_i \in \mathcal{B}_l$ and $D_i \in \mathcal{B}_{l+1}$, then $B_i = D_i$. In the case of nested families, instead of $B_i = D_i$ we get $B_i \subseteq D_i$ for all $1 \leq i \leq cols(l)$. The additional property requiring that the rows of Z must repeat rows of $\mathcal{M}^{(l)}$, or be trivial, is what allows us to be able to only need previous aggregations and not original signatures. We observe that subsequences of nested families are also nested families. As an example, Figure 3.2 shows an infinite sequence d-CFFs, where the dashed portion under each $\mathcal{M}^{(l)}$ represents Z.

Our unbounded fault-tolerant aggregate signature scheme with nested families is defined by the following algorithms. Note that **KeyGen** and **Sign** are equal to the algorithms given on page 47. For **Agg** we also create a new position $\tau[0]$ in the aggregate signature τ, which holds a full aggregation of all signatures considered up to that point. Let $(\mathcal{M}^{(l)})_l$ be

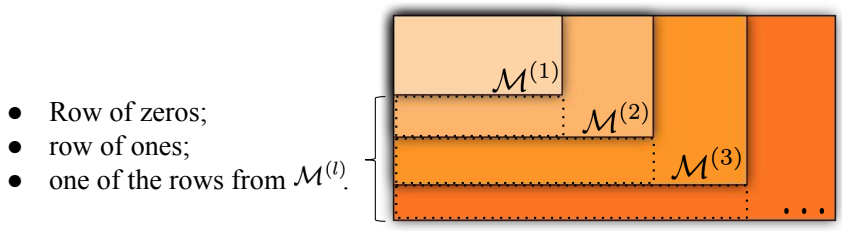


Figure 3.2: Example of a sequence of d-CFFs.

a nested family of incidence matrices of d-CFFs and let Σ be a simple aggregate signature scheme that supports claim sequences, claim placeholders, and the empty signature λ.

Unbounded fault tolerant aggregation with nested families

1. **KeyGen**(1^κ) creates a key pair (pk, sk) using **KeyGen** from Σ and security parameter κ.

2. **Sign**(sk, m) receives a secret key and message, and outputs the signature given by Σ.**Sign**(sk, m).

3. **Agg**$(C_1, C_2, \tau_1, \tau_2)$ takes two exclusively mergeable claim sequences C_1 and C_2 and corresponding signatures τ_1 and τ_2, and outputs the aggregate signature τ, where $|\tau| = \max\{|\tau_1|, |\tau_2|\}$.

   (a) Let n_k be the dimension of C_k for $k = 1, 2$, and assume w.l.o.g. that $n_1 \leq n_2$. Determine l_k such that $\text{cols}(\mathcal{M}^{(l_k-1)}) < n_k \leq \text{cols}(\mathcal{M}^{(l_k)})$, and denote by $t_k = \text{rows}(\mathcal{M}^{(l_k)})$, $k = 1, 2$. Note that $l_1 \leq l_2$ and take $\mathcal{M}$ as the submatrix of $\mathcal{M}^{(l_2)}$ consisting of the first n_2 columns. Note that $\mathcal{M} = \left(\begin{smallmatrix} \mathcal{M}^{(l_1)} & Y \\ Z & W \end{smallmatrix}\right)$, for some matrices Z, Y, W satisfying the "nesting" properties of Definition 3.4.1. For this aggregation, $\mathcal{M}$ will be the d-CFF matrix that plays the same role as the fixed matrix used in the bounded scheme on page 47.

   (b) If one or both of the claim sequences C_k ($k \in \{1, 2\}$) contain only one claim c, τ_k is an individual signature σ_k. We expand τ_k to a vector as follows, with j equal to the index of c in C_k:

$$\tau_k[i] = \begin{cases} \sigma_k, & \text{if } i = 0 \text{ or } (\mathcal{M}[i, j] = 1 \text{ and } 1 \leq i \leq t_k), \\ \lambda, & \text{otherwise.} \end{cases}$$

   (c) Once τ_1 and τ_2 are both vectors, we aggregate them position by position according to $\mathcal{M}$. Note that by the nested family definition we have three types of row index i depending on the row type of Z: a row of zeros, where $\mathcal{M}[i, 1] = \ldots = \mathcal{M}[i, n_1] = 0$ (Type 0); a row of ones, where $\mathcal{M}[i, 1] = \ldots = \mathcal{M}[i, n_1] = 1$ (Type 1); and a repeated row r of $\mathcal{M}^{(l_1)}$, where $\mathcal{M}[i, 1] = \mathcal{M}^{(l_1)}[r, 1], \ldots, \mathcal{M}[i, n_1] = \mathcal{M}^{(l_1)}[r, n_1]$ (Type 2 (r)). First we expand C_1 to $\overline{C}_1$ having the same dimension as C_2, i.e. $\overline{C}_1[i] = C_1[i]$ for $1 \leq i \leq n_1$, and $\overline{C}_1[i] = \perp$ for $n_1 + 1 \leq i \leq n_2$, then we proceed as follows.

$$\tau[0] = \Sigma.\mathbf{Agg}(\overline{C}_1, C_2, \tau_1[0], \tau_2[0])$$

For $i = 1, \ldots t_1$:

$$\tau[i] = \Sigma.\mathbf{Agg}(\overline{C}_1[\mathcal{M}_i], C_2[\mathcal{M}_i], \tau_1[i], \tau_2[i])$$

For $i = t_1 + 1, \ldots, t_2$:

$$\tau[i] = \begin{cases} \tau_2[i], & \text{if } i \text{ is Type 0,} \\ \Sigma.\mathbf{Agg}(\overline{C}_1[\mathcal{M}_i], C_2[\mathcal{M}_i], \tau_1[0], \tau_2[i]), & \text{if } i \text{ is Type 1,} \\ \Sigma.\mathbf{Agg}(\overline{C}_1[\mathcal{M}_i], C_2[\mathcal{M}_i], \tau_1[r], \tau_2[i]), & \text{if } i \text{ is Type 2 (r).} \end{cases}$$

Output τ.

4. $\mathbf{Verify}(C, \tau)$ takes a set of public key and message pairs and the aggregate signature τ and outputs the valid claims. If $\Sigma.\mathbf{Verify}(C, \tau[0]) = 1$, output all claims, otherwise compute $b_i = \Sigma.\mathbf{Verify}(C[\mathcal{M}_i], \tau[i])$ for each $1 \leq i \leq t_2$ and output the set of valid claims consisting of the union of each $C[\mathcal{M}_i]$ such that $b_i = 1$.

The correctness of the aggregation and verification algorithms comes from the fact that the matrices used are d-CFF. For the aggregation algorithm we just need to verify that the aggregated signature computed in step (c) yields the same results as if $\mathcal{M}^{(l_2)}$ was used directly on the original signatures and apply Theorem 3.3.2. The security of the scheme comes from Theorem 3.3.3, which relies on the security of the underlying aggregate scheme Σ.

Figure 3.3 depicts how algorithm $\mathbf{Agg}(C_1, C_2, \tau_1, \tau_2)$ above deals with each type of row in part (c). Assume $C_1 = (\perp, c_2, \perp, \perp, c_5), C_2 = (\perp, \perp, \perp, c_4, \perp, \perp, \perp, \perp, c_9, c_{10}))$, $n_1 = 5, n_2 = 10$, and corresponding τ_1 and τ_2, with τ_1 a vector of $t_1 + 1$ positions created according to $\mathcal{M}^{(l_1)}$ and τ_2 a vector of $t_2 + 1$ positions created according to $\mathcal{M}^{(l_2)}$. In this example, we have $\mathcal{M} = \mathcal{M}^{(l_2)}$. According to step (c), we start by expanding C_1 to $\overline{C}_1 = (\perp, c_2, \perp, \perp, c_5, \perp, \perp, \perp, \perp, \perp)$. Then construct τ as follows. We start by computing $\tau[0]$, which holds the aggregation of all signatures considered so far.

$\tau[0] = \Sigma.Agg((\perp, c_2, \perp, \perp, c_5, \perp, \perp, \perp, \perp, \perp), (\perp, \perp, \perp, c_4, \perp, \perp, \perp, \perp, c_9, c_{10})), \tau_1[0], \tau_2[0])$.

For $i = 1, \ldots, t_1$ we perform regular aggregations according to $\mathcal{M}$. For $i = t_1 + 1, \ldots, t_2$ we perform aggregations according to $\mathcal{M}$ and the type of row under $\mathcal{M}^{(l_1)}$. For instance, we compute $\tau[a], \tau[b], \tau[c]$ using rows $\mathcal{M}_a, \mathcal{M}_b, \mathcal{M}_c$ of $\mathcal{M}$, respectively, as follows.

$\tau[a] = \tau_2[a]$,
$\tau[b] = \Sigma.Agg((\perp, c_2, \perp, \perp, c_5, \perp, \perp, \perp, \perp, \perp), (\perp, \perp, \perp, c_4, \perp, \perp, \perp, \perp, c_9, \perp)), \tau_1[0], \tau_2[b])$,
$\tau[c] = \Sigma.Agg((\perp, c_2, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp), (\perp, \perp, \perp, c_4, \perp, \perp, \perp, \perp, \perp, c_{10})), \tau_1[r], \tau_2[c])$.

In the next section, we give three explicit constructions of nested families that allow us to achieve unbounded aggregation with optimal compression ratio for $d = 1$ and very good compression ratios for general d.

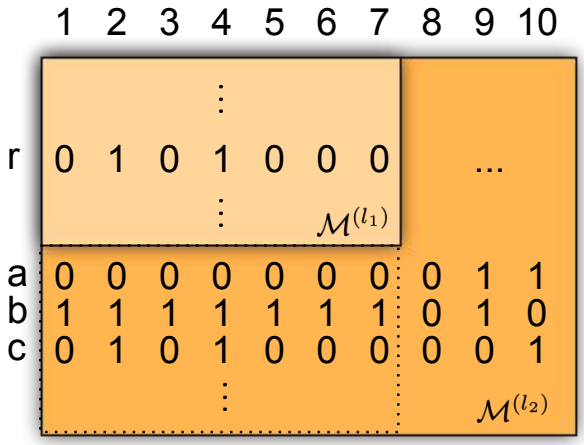


Figure 3.3: Example of aggregation using nested d-CFFs.

3.5 Construction of nested families with near optimal compression ratios

Now we aim to construct a nested family of incidence matrices (Definition 3.4.1) where we can increase n as necessary while avoiding to save every individual signature for further use. In this section, we propose explicit constructions of nested families for the cases where $d = 1$, $d = 2$, and for general values of d. We note that all sequences of CFF given are constructive, as they rely on ingredients that can be constructed explicitly by known methods.

3.5.1 Nested family for d=1

In Corollary 3.3.9, optimal 1-CFFs are given based on optimal Sperner families of subsets of a t-set, giving a 1-CFF$(t, \binom{t}{\lfloor t/2 \rfloor})$ for every t. Since the order of elements is important for nested families, we represent the family as a tuple of sets and all we need is to order the subsets in a way to guarantee the nested properties. Let $n_l = \binom{l}{\lfloor \frac{l}{2} \rfloor}$, and $\mathcal{C}_k^n$ be the list of all k-subsets from $\{1, \ldots, n\}$ in lexicographical order. Define $\mathcal{B}_t$ as follows:

$$\mathcal{B}_2 = [\{1\}, \{2\}],$$
$$\mathcal{B}_t = \begin{cases} [\mathcal{B}_{t-1}[1], \ldots, \mathcal{B}_{t-1}[n_{t-1}], \mathcal{C}^{t-1}_{\lfloor t/2 \rfloor - 1}[1] \cup \{t\}, \ldots, \mathcal{C}^{t-1}_{\lfloor t/2 \rfloor - 1}[n_t] \cup \{t\}], & t \text{ odd}, \\ [\mathcal{B}_{t-1}[1] \cup \{t\}, \ldots, \mathcal{B}_{t-1}[n_{t-1}] \cup \{t\}, \mathcal{C}^{t-1}_{t/2}[1], \ldots, \mathcal{C}^{t-1}_{t/2}[n_t]], & t \text{ even}, \end{cases}$$

for $t > 2$.

Theorem 3.5.1. *The sequence $(X_t, \mathcal{B}_t)_t$ defined above is a nested family of 1-CFF$(t, \binom{t}{\lfloor t/2 \rfloor})$.*

Proof. The proof that the construction gives 1-cover-free families comes from Corollary 3.3.9, and the fact that each $\mathcal{B}_t$ has all distinct subsets of size $\binom{t}{\lfloor \frac{t}{2} \rfloor}$.

Next we need to show that the recursive construction of $(X_t, \mathcal{B}_t)_t$ gives a nested family. For the case of t odd, consider $\mathcal{M}^{(t-1)}$ the incidence matrix of $\mathcal{F}_{t-1}$, and A as the characteristic vectors of $\mathcal{C}^{t-1}_{\lfloor t/2 \rfloor - 1}$. According to the recursive construction, the incidence matrix $\mathcal{M}^{(t)}$ for $\mathcal{F}_t$ has the following form:

$$\mathcal{M}^{(t)} = \begin{pmatrix} \mathcal{M}^{(t-1)} & A \\ 0 \ldots 0 & 1 \ldots 1 \end{pmatrix}$$

For the case of t even, with incidence matrix $\mathcal{M}^{(t-1)}$ from $\mathcal{F}_{(t-1)}$ and A being the characteristic vectors of $\mathcal{C}^{t-1}_{t/2}$, the incidence matrix $\mathcal{M}^{(t)}$ of $\mathcal{F}_t$ has the following form:

$$\mathcal{M}^{(t)} = \begin{pmatrix} \mathcal{M}^{(t-1)} & A \\ 1 \ldots 1 & 0 \ldots 0 \end{pmatrix}$$

We note that both cases follow the form of matrix Z prescribed by Definition 3.4.1 of nested family. We further observe that subsequences of $(X_t, \mathcal{B}_t)_t$ ("jumping" by more than one row) also satisfy the properties required for the corresponding Z matrix in Definition 3.4.1. □

It is easy to see that the property holds for larger increases of t as well, where each row of Z in this construction consists of all zeros or all ones. In other words, taking sub-sequences of $(X_t, \mathcal{B}_t)_t$ also gives a nested family.

Theorem 3.5.2. *Let $(\mathcal{M}^{(l)})_l$ be the nested family defined in Theorem 3.5.1. This sequence has a compression ratio $\rho(n) = \frac{n}{\log_2 n}$.*

Proof. The ratio is given by $\frac{n}{t} = \frac{\binom{t}{\lfloor \frac{t}{2} \rfloor}}{t}$. Using the approximation by the central binomial coefficient $\binom{t}{\lfloor \frac{t}{2} \rfloor} \sim \frac{2^t}{\sqrt{\pi \lfloor \frac{t}{2} \rfloor}}$, we note that $t \sim \log_2 n$. Thus, $\rho(n) = \frac{n}{\log_2 n}$. □

Note that this compression ratio meets the information theoretical upper bound, where $\log_2 n$ is a lower bound on the number of bits that are necessary to uniquely distinguish the result of the verification algorithm.

3.5.2 Nested family for $d \geq 2$

In this section, we give three classes of constructions of nested families. Theorems 3.5.4 and 3.5.6 give specific compression ratios obtained for the cases of $d \geq 2$. The one from Theorem 3.5.6 is asymptotically better, but for different constants and ranges of n, either one may be more suitable. Theorem 3.5.7 gives yet a better asymptotic ratio for the case $d = 2$.

Theorem 3.5.3 gives nested families via Theorem 3.3.11, and Theorem 3.5.4 provides its corresponding ratio.

Theorem 3.5.3. *Let $\mathcal{M}$ be the incidence matrix of a d-CFF(t, n) (wlog we require $\mathcal{M}_{1,1} = 1$), and set $\mathcal{M}^{(1)} = \mathcal{M}$. We define $\mathcal{M}^{(l)} = \mathcal{M} \otimes \mathcal{M}^{(l-1)}$ for $l \geq 2$. Then $(\mathcal{M}^{(l)})_l$ is a nested family of incidence matrices of d-CFFs.*

Proof. The fact that each $\mathcal{M}^{(l)}$ is d-CFF comes from Theorem 3.3.11. We need to show that $\mathcal{M}^{(l)}$ follows the structure of nested families given in Definition 3.4.1. First we verify that the upper-left corner of $\mathcal{M}^{(l)}$ is equal to $\mathcal{M}^{(l-1)}$; this is true due to the requirement of $\mathcal{M}_{1,1} = 1$. Then we check that every row under $\mathcal{M}^{(l-1)}$ is either a row of $\mathcal{M}^{(l-1)}$ itself or a row of zeros, which is true due to the Kronecker product operation. □

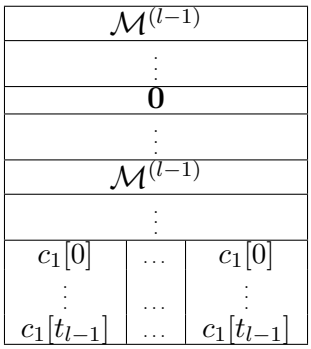
Theorem 3.5.4. *Let $d \geq 2$. Let $(\mathcal{M}^{(l)})_l$ be the nested family defined in Theorem 3.5.3 using a d-CFF$(\bar{t}, \overline{n})$ matrix $\mathcal{M}$ with $\overline{n} > \bar{t} > 1$. Then, the sequence has increasing compression ratio $\rho(n) = \frac{n}{n^{1/c}} = n^{1-1/c}$, for $c = \log_{\bar{t}} \overline{n} > 1$ (c depending on d).*

Proof. Note that $\mathcal{M}^{(l)} = \mathcal{M} \times \mathcal{M}^{(l-1)}$ has $n_l = \overline{n} \times \overline{n}^{l-1} = \overline{n}^l$ columns and $t_l = (\bar{t}) \times (\bar{t})^{l-1} = (\bar{t})^l$ rows. Thus, $t_l = (\bar{t})^l = (\bar{t})^{\log_{\overline{n}} n_l} = (\bar{t})^{\log_{\bar{t}} n_l / \log_{\bar{t}} \overline{n}} = n_l^{1/\log_{\bar{t}} \overline{n}} = n_l^{1/c}$ for $c = \log_{\bar{t}} \overline{n} > 1$. Consequently, the ratio is given by $\rho(n) = \frac{n}{n^{1/c}} = n^{1-1/c}$, where c depends on d. □

In Theorem 3.5.5, we show how to use the construction from Theorem 3.3.13 to build a sequence of nested families for general d. Then, in Theorems 3.5.6 and 3.5.7 we give compression ratios by using specific d-CFF matrices.

Theorem 3.5.5. *Let $\mathcal{M}$ be a d-CFF(t_1, n_1) matrix and set $\mathcal{M}^{(1)} = \mathcal{M}$. Let B_i be a $(d-1)$-CFF(s_i, n_i) matrix (wlog we require $B_{i_{1,1}} = 1$), for each $i \geq 1$. We recursively define $\mathcal{M}^{(l)} =$ **Const1**$(\mathcal{M}^{(l-1)}, \mathcal{M}^{(l-1)}, B_{l-1})$, for $l \geq 2$, using Construction 3.3.12. The sequence of matrices $(\mathcal{M}^{(l)})_l$ is a nested family of incidence matrices of d-CFFs.*

Proof. The fact that $\mathcal{M}^{(l)}$ is d-CFF comes from Theorem 3.3.13. Now we need to show that $\mathcal{M}^{(l)}$ follows the definition of nested family given in Definition 3.4.1. First, we verify that the upper left corner of $\mathcal{M}^{(l)}$ is equal to $\mathcal{M}^{(l-1)}$ due to fixing $B_{l-1_{1,1}} = 1$. Then we need to check that every partial row of $\mathcal{M}^{(l)}$ under $\mathcal{M}^{(l-1)}$ is either a row of $\mathcal{M}^{(l-1)}$, a row of zeros, or a row of ones. We note that the first n_{l-1} columns on the left side of $\mathcal{M}^{(l)}$ will be of the following form, with c_1 equal to the first column of $\mathcal{M}^{(l-1)}$ repeated n_{l-1} times, and $\mathbf{0}$ equal to an all zero matrix:



Note that the rows of the constructed matrix are divided into two parts; in the top part we apply the Kronecker product and in the bottom part we append the columns of $\mathcal{M}^{(l-1)}$. Therefore, partial rows under the $\mathcal{M}^{(l-1)}$ in the left corner in the second part are a repetition of the same column c_1, the first column of $\mathcal{M}^{(l-1)}$. In the top part, the rows are either equal to the rows of $\mathcal{M}^{(l-1)}$ or of zero matrices; in the bottom part of the construction we clearly obtain rows of all zeroes and/or all ones. This matches the requirements of matrix Z in Definition 3.4.1. □

Theorem 3.5.6. *Let $d \geq 2$. Let $(\mathcal{M}^{(l)})_l$ be the nested family defined in Theorem 3.5.5 using a d-CFF($\bar{t}, \overline{n}$) matrix $\mathcal{M}$ with $\overline{n} > \bar{t}$. Then, there exists a $(d-1)$-CFF B_l such that the sequence $(\mathcal{M}^{(l)})_l$ has ratio $\rho(n) = \frac{n}{(b \log_2 n)^{\log_2 \log_2 n + D}}$ for constants $b = \frac{2d^2 \ln \overline{n}}{\log_2 \overline{n}}$, and $D = 1 - \log_2 \log_2 \overline{n}$.*

Proof. Take $\mathcal{M}^{(1)} = \mathcal{M}$. Take B_l as a $(d-1)$-CFF(s_l, n_l) constructed using Porat and Rothschild [14] for every l, which gives an s_l that is $\Theta(d^2 2^{l-1} \ln \overline{n})$. We observe that $\mathcal{M}^{(l)}$ has $n_l = \overline{n}^{2^{l-1}}$ columns and $t_l = \bar{t} \prod_{i=1}^{l-1}(s_i + 1)$ rows. Therefore, t_l is $\Theta(2^{l^2} d^{2(l-1)} (\ln \overline{n})^{l-1})$ and by using the fact that $2^{l-1} = \log_{\overline{n}} n_l$, $l - 1 = \log_2 \log_{\overline{n}} n_l$, and therefore $l = \log_2 \frac{\log_2 n_l}{\log_2 \overline{n}} + 1$, we see that t_l is $\Theta((2^l d^2 \ln \overline{n})^l) = \Theta(((\frac{\log_2 n_l}{\log_2 \overline{n}} \times 2) d^2 \ln \overline{n})^l) = \Theta((b \log_2 n_l)^{\log_2 \log_2 n_l + D})$ for constants $b = \frac{2d^2 \ln \overline{n}}{\log_2 \overline{n}}$, and $D = 1 - \log_2 \log_2 \overline{n}$. Therefore, $\rho(n) = \frac{n}{(b \log_2 n)^{\log_2 \log_2 n + D}}$. □

The next theorem improves the ratio from Theorem 3.5.6 for $d = 2$ by using optimal 1-CFFs given by Corollary 3.3.9.

Theorem 3.5.7. *Let $(\mathcal{M}^{(l)})_l$ be the nested family defined in Theorem 3.5.5 using a 2-CFF($\bar{t}, \overline{n}$) matrix $\mathcal{M}$ with $\overline{n} > \bar{t}$, and each B_i is a 1-CFF given by Corollary 3.3.9 for $n = \overline{n}^{2^{i-1}}$, $i \geq 1$. Then, the sequence $(\mathcal{M}^{(l)})_l$ has increasing ratio $\rho(n) = \frac{n}{(2 \log_2 n)^{\log_2 \log_2 n + D}}$, for constant $D = 1 - \log_2 \log_2 \overline{n}$.*

Proof. Take B_i as a 1-CFF(s_i, n_i) constructed using Corollary 3.3.9, which gives a $s_i \simeq 2^{i-1} s_1 \simeq 2^{i-1} \log_2 \overline{n}$. We observe that $\mathcal{M}^{(l)}$ has $n_l = \overline{n}^{2^{l-1}}$ columns and $t_l = \bar{t} \prod_{i=1}^{l-1}(s_i + 1)$ rows. Then $t_l \simeq \bar{t} \prod_{i=1}^{l-1}(2^{i-1} \log_2 \overline{n} + 1)$, which is $\Theta(2^{l^2} (\log_2 \overline{n})^{l-1})$. Now we use the fact that $2^{l-1} = \log_{\overline{n}} n_l$, $l - 1 = \log_2 \log_{\overline{n}} n_l$, and therefore $l = \log_2 \frac{\log_2 n_l}{\log_2 \overline{n}} + 1$. Consequently, t_l is $\Theta((2^l \log_2 \overline{n})^l) = \Theta(((\frac{\log_2 n_l}{\log_2 \overline{n}} \times 2) \log_2 \overline{n})^l) = \Theta((2 \log_2 n_l)^{\log_2 \log_2 n_l + D})$, and the compression ratio $\rho(n) = \frac{n}{(2 \log_2 n)^{\log_2 \log_2 n + D}}$ for constant $D = 1 - \log_2 \log_2 \overline{n}$. □

3.6 Generalized CFFs to support corrupted aggregation tuple

A $(d; \lambda)$-CFF generalizes d-CFFs requiring that λ elements in a column are not covered by the union of any other d columns [10].

Definition 3.6.1. *A set system $\mathcal{F} = (X, \mathcal{B})$ is said to be a $(d; \lambda)$-cover free family ($(d; \lambda)$-CFF) if for any subset $B_{i_0} \in \mathcal{B}$ and any other d subsets $B_{i_1}, \ldots, B_{i_d} \in \mathcal{B}$, we have*

$$\left| B_{i_0} \setminus \left(\bigcup_{j=1}^{d} B_{i_j} \right) \right| \geq \lambda. \tag{3.2}$$

Note that a d-CFF is a $(d; 1)$-CFF.

Suppose that we use a nested family of $(d; \lambda)$-CFFs in Scheme 1. For the verification algorithm, suppose that up to $\lambda - 1$ signatures got corrupted. Definition 3.6.1 guarantees that removing $\lambda - 1$ rows of the matrix gives a $(d; 1)$-CFF matrix. So the verification algorithm can still identify the invalid signatures, even without $\lambda - 1$ of the aggregations.

We note we can generalize the d-CFF construction in Theorem 3.3.11 to generate $(d; \lambda)$-CFFs, which can be used later to generate $(d; \lambda)$ nested families.

Theorem 3.6.2. *Let A_1 be a $(d; \lambda_1)$-CFF(t_1, n_1) and A_2 be a $(d; \lambda_2)$-CFF(t_2, n_2), then $C = A_1 \otimes A_2$ is a $(d; \lambda_1\lambda_2)$-CFF(t_1t_2, n_1n_2).*

Proof. Due to the fact that A_1 is a $(d; \lambda_1)$-CFF, we know that for any of its $d+1$ columns there will be at least λ_1 positions i where $B_{i,c_{j_0}} = 1$ and $B_{i,c_{j_k}} = 0$, for all $1 \leq k \leq d$. The same is true for A_2 with λ_2 such positions. So, if we follow a similar proof from Theorem 3.3.11 we will obtain a "generalized permutation sub-matrix" of the form

$$\overline{C} = \begin{pmatrix} A_2 & \ldots & \mathbf{0} \\ \vdots & \ldots & \vdots \\ A_2 & \ldots & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & \ldots & A_2 \\ \vdots & \ldots & \vdots \\ \mathbf{0} & \ldots & A_2 \end{pmatrix},$$

where every A_2 appears λ_1 times in each column, and since each A_2 is also a $(d; \lambda_2)$-CFF, C is a $(d; \lambda_1\lambda_2)$-CFF(t_1t_2, n_1n_2). □

We can also generalize Theorem 3.3.13 to generate $(d; \lambda)$-CFFs as follows.

Theorem 3.6.3. *Let A_1 be a $(d; \lambda_1)$-CFF(t_1, n_1), A_2 be a $(d; \lambda_2)$-CFF(t_2, n_2), B be a $(d-1; \lambda_3)$-CFF(s, n_2), and $\lambda = min(\lambda_1\lambda_3, \lambda_2)$. Then* C := **Const1**$(A_1, A_2, B)$ *is a $(d; \lambda)$-CFF$(st_1 + t_2, n_1n_2)$.*

Proof. Follow a similar proof as in Theorem 3.3.13. *Case 1* holds because A_2 is a $(d; \lambda_2)$-CFF. *Case 2* holds when we consider a "generalized permutation sub-matrix" of the form

$$\overline{C} = \begin{pmatrix} A_1 & \dots & \mathbf{0} \\ \vdots & \dots & \vdots \\ A_1 & \dots & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & \dots & A_1 \\ \vdots & \dots & \vdots \\ \mathbf{0} & \dots & A_1 \end{pmatrix},$$

where every A_1 appears λ_3 times in each column, with A_1 a $(d;\lambda_1)$-CFF. □

Now we can use these two constructions to build $(d;\lambda)$ nested families, which can be applied, for example, to the problem of fault-tolerant aggregation of signatures with transmission errors that was discussed above.

Theorem 3.6.4. *Let $\mathcal{M}$ be the incidence matrix of a $(d;\lambda)$-CFF(t,n) (wlog we require $\mathcal{M}_{1,1}=1$), and set $\mathcal{M}^{(1)}=\mathcal{M}$. We define $\mathcal{M}^{(l)}=\mathcal{M}\otimes\mathcal{M}^{(l-1)}$ for $l\geq 2$. Then $(\mathcal{M}^{(l)})_l$ is a nested family of incidence matrices of $(d;\lambda^l)$-CFFs, with the same ratio of Theorem 3.5.4.*

Proof. The proof $\mathcal{M}^{(l)}$ is $(d;\lambda^l)$-CFF comes from Theorem 3.6.2, and the fact that in each step we are multiplying the λs. The proof it is a nested family follows a similar idea as the one from Theorem 3.5.3. □

Theorem 3.6.5. *Let $\mathcal{M}$ be a $(d;\lambda)$-CFF(t_1,n_1) matrix and set $\mathcal{M}^{(1)}=\mathcal{M}$. Let B_i be a $(d-1;1)$-CFF(s_i,n_i) matrix (wlog we require $B_{i_{1,1}}=1$), for each $i\geq 1$. We recursively define $\mathcal{M}^{(l)}=$ **Const1**$(\mathcal{M}^{(l-1)},\mathcal{M}^{(l-1)},B_{l-1})$, for $l\geq 2$ and **Const1** as defined in Construction 3.3.12. The sequence of matrices $(\mathcal{M}^{(l)})_l$ is a nested family of incidence matrices of $(d;\lambda)$-CFFs, with the same ratio of Theorem 3.5.6. The same applies for $(2;\lambda)$-CFFs with B_i a $(1;1)$-CFF from Corollary 3.3.9; the resulting nested family has the same ratio of Theorem 3.5.7.*

Proof. The proof $\mathcal{M}^{(l)}$ is $(d;\lambda)$-CFF comes from Theorem 3.6.3. The proof it is a nested family follows a similar idea as the one from Theorem 3.5.5. □

3.7 Final remarks and open problems

In this work, we focus on the problem of unbounded fault-tolerant aggregation of signatures proposed by Hartung et al. [4]. Here we define a sequence of cover-free families, called *nested* families, which generalizes monotone families introduced in [4]. We give unbounded schemes using nested families with better compression ratios than the ones currently known using monotone families.

In Section 3.4, we detail the algorithms that achieve unbounded aggregation using nested families (algorithms in page 54). In Section 3.5, we give explicit constructions of

such families for all positive integers d with compression ratios relatively close to the best known upper bound, and for $d = 1$ the compression ratio meets the information theoretical upper bound.

In Section 6, we show how an extra level of fault tolerance can be achieved by using well known $(d; \lambda)$-CFFs. This gives the ability to handle loss or corruption of up to $\lambda - 1$ aggregates of the aggregation tuple τ. Our constructions and compression ratios generalize naturally to nested families of $(d; \lambda)$-CFFs.

We observe that as n increases in our constructions, it would be desirable to increase the threshold value d as well. However, it is not hard to show that there cannot be nested families with increasing d. Indeed, in the definition of nested families, if $\mathcal{M}_{l+1}$ is d-CFF, then so is the first block of columns (Proposition 3.3.5); by the structure of Z we know it does not add to the cover-free Property (3.1), and we conclude $\mathcal{M}_l$ must be d-CFF. Therefore, it appears to us that there is not much hope of increasing d in fault tolerant aggregate signature schemes.

In [7][1], we investigate embedding sequences of CFFs that allows us to increase d and n, which can be useful for other cryptographical applications of CFFs. There, we propose some constructions of embedding families based on polynomials over finite fields. Some proposed variations of these constructions give monotone families (fixed d) with compression ratio $\rho(n) = n^{1-1/c}$, for every integer $c \geq 2$. We believe further studies of monotone, nested and embedding families of CFFs are important directions for modern applications of group testing where n is naturally unbounded, in particular applications in cryptography.

[1]Given in this thesis as Chapter 4.

Bibliography

[1] D. Boneh, C. Gentry, B. Lynn, H. Shacham. *Aggregate and verifiably encrypted signatures from bilinear maps*. In EUROCRYPT, 2003. Lecture Notes in Computer Science, vol 2656. Springer, Berlin, Heidelberg, 2003, 416–432.

[2] D.Z. Du, F.K. Hwang. *Combinatorial group testing and its applications*. World Scientific, 1999, 2nd edition.

[3] Z. Füredi. *On r-Cover-free Families*. Journal of Combinatorial Theory, Series A 73 (1996), 172–173.

[4] G. Hartung, B. Kaidel, A. Koch, J. Koch, A. Rupp. *Fault-tolerant aggregate signatures*. In Public-Key Cryptography–PKC 2016. Lecture Notes in Computer Science, vol 9614, Springer, Cham, 2016, 331–356.

[5] T. B. Idalino, *Using combinatorial group testing to solve integrity issues*. Master's thesis, Universidade Federal de Santa Catarina, Brazil (2015).

[6] T. B. Idalino, L. Moura, *Efficient unbounded fault-tolerant aggregate signatures using nested cover-free families*. In Iliopoulos C., Leong H., Sung WK. (eds) Combinatorial Algorithms – IWOCA 2018. Lecture Notes in Computer Science, vol 10979, Springer, Cham, 2018, 52–64.

[7] T. B. Idalino, L. Moura, *Embedding cover-free families and cryptographical applications*. Advances in Mathematics of Communications, 13 (2019), 629–643.

[8] T. B. Idalino, L. Moura, R. F. Custódio, D. Panario, *Locating modifications in signed data for partial data integrity*. Information Processing Letters, 115 (2015), 731 – 737.

[9] Z. Li, G. Gong, *Data Aggregation Integrity Based on Homomorphic Primitives in Sensor Networks*. In Nikolaidis I., Wu K. (eds) Ad-Hoc, Mobile and Wireless Networks. ADHOC-NOW 2010. Lecture Notes in Computer Science, vol 6288. Springer, Berlin, Heidelberg, 2010, 149–162.

[10] P. C. Li, G. H. J. van Rees, R. Wei, *Constructions of 2-cover-free families and related separating hash families*. Journal of Combinatorial Designs, 14 (2006), 423–440.

[11] D. Ma, *Practical forward secure sequential aggregate signatures*. In Proceedings of the 2008 ACM symposium on Information, computer and communications security – ASIACCS, ACM, 2008, 341–352.

[12] A. J. Macula, *A simple construction of d-disjunct matrices with certain constant weights*. Discrete Mathematics, 162 (1996), 311–312.

[13] E. Mykletun, M. Narasimha, G. Tsudik, *Authentication and integrity in outsourced databases*. ACM Transactions on Storage, 2 (2006), 107–138.

[14] E. Porat, A. Rothschild, *Explicit nonadaptive combinatorial group testing schemes*. IEEE Transactions on Information Theory, 57 (2011), 7982–7989.

[15] E. Sperner, *Ein Satz über Untermengen einer endlichen Menge*. Mathematische Zeitschrift, 27 (1928), 544–548.

[16] A. Wasef, X. Shen, *ASIC: Aggregate signatures and certificates verification scheme for vehicular networks*. In IEEE Global Telecommunications Conference – GLOBECOM, 2009, 1–6.

[17] G. M. Zaverucha, D. R. Stinson, *Group testing and batch verification*. In Kurosawa K. (eds) Information Theoretic Security – ICITS 2009. Lecture Notes in Computer Science, vol 5973, Springer, Berlin, Heidelberg, 2009, 140–157.

Chapter 4

Embedding cover-free families and cryptographical applications

This chapter contains the literal contents of the following paper (except for some typos which have been corrected), published in a special issue of the journal *Advances in Mathematics of Communications*, dedicated to *Applications of Discrete Mathematics in Secure Communication* in honour of Prof. Bimal Kumar Roy.

IDALINO, T. B.; MOURA, L., Embedding cover-free families and cryptographical applications. *Advances in Mathematics of Communications*, 13 (2019), 629–643.

4.1 Abstract

Cover-free families are set systems used as solutions for a large variety of problems, and in particular, problems where we deal with n elements and want to identify d defective ones among them by performing only t tests ($t \leq n$). We are especially interested in cryptographic problems, and we note that some of these problems need cover-free families with an increasing size n. Solutions that propose the increase of n, such as *monotone families* and *nested families*, have been recently considered in the literature. In this paper, we propose a generalization that we call *embedding families*, which allows us to increase both n and d. We propose constructions of *embedding families* using polynomials over finite fields embedded via extension fields; we study how different parameter combinations can be used to prioritize increase of d or of the compression ratio as n grows. We also provide new constructions for monotone families with improved compression ratio. Finally, we show how to use embedded sequences of orthogonal arrays and packing arrays to build embedding families.

4.2 Introduction

A cover-free family (CFF) is a set system usually studied in the context of group testing applications. In this scenario, we are given a set of n elements and want to identify up to d defective ones in a more efficient way than testing each one of them individually. A d-cover-free family (or d-CFF) indicates how to group the n elements into t groups ($t \leq n$), and by performing only t tests we are able to identify up to d defective elements. These families are used to solve several problems in cryptography, such as one-time and multiple-times digital signature schemes [15, 25], fault-tolerant aggregation of signatures [8, 9, 24], localization of modifications on signed documents and redactable signatures [10], broadcast authentication [17], broadcast encryption [7], frameproof and traceability codes [18, 20], among others [13].

We can represent d-CFFs as set systems or their corresponding incidence matrices. A set system $\mathcal{F} = (X, \mathcal{B})$ consists of a set $X = \{x_1, \ldots, x_t\}$ with $|X| = t$, and a collection $\mathcal{B} = \{B_1, \ldots, B_n\}$ with $B_i \subseteq X, 1 \leq i \leq n$, and $|\mathcal{B}| = n$. A d-cover-free family, denoted $d-\mathrm{CFF}(t, n)$, is a set system such that for any subset $B_{i_0} \in \mathcal{B}$ and any other d subsets $B_{i_1}, \ldots, B_{i_d} \in \mathcal{B}$, we have

$$B_{i_0} \nsubseteq \bigcup_{j=1}^{d} B_{i_j}. \tag{4.1}$$

The family $\mathcal{F}$ can be represented by its $t \times n$ binary incidence matrix $\mathcal{M}$:

$$\mathcal{M}_{i,j} = \begin{cases} 1, & \text{if } x_i \in B_j, \\ 0 & \text{otherwise.} \end{cases}$$

In the remainder of this paper, we may use the term d-CFFs to refer to their incidence matrices. For a basic reference in combinatorial group testing see [4], and for more information about combinatorial designs see [3]. Figure 4.1 shows how a 2-CFF$(9, 12)$ can be used to test $n = 12$ elements and identify up to $d = 2$ defective ones using $t = 9$ tests.

	1	2	3	4	5	6	7	8	9	10	11	12	result
$test_1$	1	0	0	1	0	0	1	0	0	1	0	0	0
$test_2$	1	0	0	0	1	0	0	1	0	0	1	0	0
$test_3$	1	0	0	0	0	1	0	0	1	0	0	1	**1**
$test_4$	0	1	0	1	0	0	0	0	1	0	1	0	0
$test_5$	0	1	0	0	1	0	1	0	0	0	0	1	**1**
$test_6$	0	1	0	0	0	1	0	1	0	1	0	0	0
$test_7$	0	0	1	1	0	0	0	1	0	0	0	1	**1**
$test_8$	0	0	1	0	1	0	0	0	1	1	0	0	**1**
$test_9$	0	0	1	0	0	1	1	0	0	0	1	0	**1**

Figure 4.1: Example of a 2-CFF$(9, 12)$ used in group testing.

The columns of the matrix represent the elements to be tested, and the rows indicate which elements we are testing together. After performing the 9 testes we obtain the last column with some results. If all the elements in a test are valid, the test *passes* (represented as 0), but if there is at least one defective element in a test, it *fails* (represented as 1). By the tests that pass we can identify all the valid elements, which in the example are $1, 2, 4, 5, 6, 7, 8, 9, 10$, and 11. Since the remaining set of elements $S = \{3, 12\}$ has $|S| \leq d$, by the definition of d-CFF we can conclude 3 and 12 are the defective elements (since each of them are the only possible cause for failure in one of the tests $test_3, test_5, test_7, test_8, test_9$).

CFFs provide a practical solution for problems where the number of elements n is known a priori. For applications where n is not known or can dynamically increase over time, we need a scheme that provides matrix growth. This can be done with a special sequence of d-CFFs, where the previous matrix is a sub-matrix of the next ones, so that we can reuse the groups and computations we already performed for smaller values of n. *Monotone families* [8] and *nested families* [9] of d-CFFs are examples of such special sequences that are used to achieve unbounded fault-tolerant aggregate digital signatures. One drawback of these families is that d must be constant, so we need more general sequences of families if we wish d to grow with n. For this purpose, in this paper, we define a generalization of both monotone and nested families called *embedding families*.

To compare the efficiency of different families of CFF, we consider the *compression ratio*, which is given by $\rho(n)$ when $\frac{n}{t(n)}$ is $O(\rho(n))$. The compression ratio measures the efficiency gained from group testing, which performs $t(n)$ tests rather than n. We look for constructions with $\rho(n)$ as large as possible, for example, the ones that meet or are close to the upper bound $\rho(n) = \frac{n}{(d^2/\log d)\log n}$ [6]. In the literature, monotone families with constant compression ratio have been given in [8], while several constructions of (the more general) nested families with compression ratio closer to the upper bound were presented in [9]. In the context of frameproof codes (special types of CFFs [18]), monotone families

have been implicitly given in [20, Theorem 4.6] with compression ratio $\sqrt{n}$. Considering the limitation of constant d for monotone and nested families, the present paper gives constructions of embedding families with good compression ratios that allow d to grow with n.

Our contributions in this paper are as follows. We revisit a construction of d-CFF by Erdös et al. [5] based on polynomials over finite fields (Theorem 4.4.2) and highlight some useful properties related to progressive d (Theorem 4.4.4). This property can be observed in the example in Table 4.2, where a matrix has submatrices with smaller d inside it, which allows us to early abort the testing after enough tests are done for the actual level of defective elements. We then give a general construction of embedding families of CFF, each CFF based on this polynomial construction, and stacked together using extension fields (Theorem 4.4.6). Specific applications of this general construction give embedding families with sublinear $d = d(n)$ and $\rho(n) = n^{1-\frac{2}{k+1}}$ (Corollary 4.4.7) as well as with constant d and $\rho(n) = \frac{n}{d \log n}$, achieving the information theoretical upper bound (Corollary 4.4.8). Moreover, we show it is possible to adapt this construction to build monotone families with compression ratio $\rho(n) = n^{1-\frac{1}{k+1}}$, for any arbitrary constant $k \geq 1$ (Theorem 4.4.9), which improves over known compression ratios [8], [20, Theorem 4.6]. Finally, we show that families of orthogonal arrays and packing arrays with some specific properties generalize the polynomial construction of embedding families (Proposition 4.6.11), which can open the door for new constructions in the future.

In Section 4.3, we define embedding families and discuss cryptographical applications. In Section 4.4, we give constructions of embedding families based on polynomials over finite fields. In Section 4.5, we discuss the use of these constructions in applications, and challenges related to the drop of actual compression ratios when columns are not used. In Section 4.6, we generalize the polynomial construction by using other combinatorial designs, and in Section 4.7, we give conclusions.

4.3 Embedding Sequences and their Applications

In this section, we present CFF constructions for unbounded applications, which are applications where n may not be known a priori or can grow over time. We introduce the notion of *embedding families* to be a sequence of CFFs that allows for the increase of n and d, and we also show how they are a generalization of nested families [9] and monotone families [8].

Definition 4.3.1 (Embedding family)**.** *Let $d(l)$ be a positive integer and let $(\mathcal{M}^{(l)})_l$ be a sequence of incidence matrices of cover-free families $(\mathcal{F}_l)_l = (X_l, \mathcal{B}_l)_l$, where $\mathcal{M}^{(l)}$ is a $d(l)$-CFF with number of rows and columns denoted by rows(l) and cols(l), respectively. Then $(\mathcal{M}^{(l)})_l$ is an* embedding family *of incidence matrices of CFFs, if $X_l \subseteq X_{l+1}$, rows(l) $\leq$ rows(l + 1), cols(l) $\leq$ cols(l + 1), $d(l) \leq d(l+1)$ and*

$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & Y^{(l)} \\ Z^{(l)} & W^{(l)} \end{pmatrix},$$

where $Y^{(l)}, Z^{(l)}, W^{(l)}$ are 0-1 matrices of appropriate dimensions.

We have that monotone and nested families are a special case of embedding families. While in embedding families we allow the increase of both n and d, monotone and nested families allow us to increase n for fixed d, due to the fact that $Z^{(l)}$ has a special format. The definitions are shown below.

Definition 4.3.2 (Nested family)**.** *A nested family of incidence matrices of d-CFFs is an embedding family of incidence matrices with fixed d such that each row of $Z^{(l)}$ is one of the rows of $\mathcal{M}^{(l)}$, a row of all zeros, or a row of all ones.*

Nested families were defined in [9] to solve a problem in unbounded aggregation of digital signatures, where three different constructions with increasing compression ratio are presented.

Definition 4.3.3 (Monotone family)**.** *A monotone family of incidence matrices of d-CFFs is an embedding family of incidence matrices with fixed d such that $Z^{(l)}$ is a matrix of zeros.*

Monotone families were introduced by Hartung et al. [8] to solve the problem of unbounded aggregation of signatures. They showed a concrete instantiation of monotone families with a constant compression ratio. A result in [20, Theorem 4.6] can be used to build monotone families with $\rho(n) = \sqrt{n}$ using subgeometries of affine geometries. Note that in [20] the term "embedding" is used for the specific case of monotone families, since it imposes $Z^{(l)} = 0$. We show in Theorem 4.4.9 a construction for monotone families with $\rho(n) = n^{1-\frac{1}{c}}$, for any arbitrary constant c.

The following proposition justifies why the definition of nested and monotone families can not be generalized to increase d.

Proposition 4.3.4. *Let $d' \geq d$. Consider $\mathcal{M}' = \begin{pmatrix} \mathcal{M} & Y \\ Z & W \end{pmatrix}$ where $\mathcal{M}'$ is a d'-CFF, $\mathcal{M}$ is a d-CFF with at least $d+1$ columns that is not a $(d+1)$-CFF, and Z, W, Y are 0-1 submatrices of appropriate dimensions. If Z is constrained as in Definition 4.3.2 or Definition 4.3.3, then $d = d'$.*

Proof. It is easy to see that if $\mathcal{M}$ is not $(d+1)$-CFF then $\begin{pmatrix} \mathcal{M} \\ Z \end{pmatrix}$ is not $(d+1)$-CFF, given the structure of Z. Thus $\mathcal{M}'$ is not $(d+1)$-CFF which implies $d' = d$. □

4.3.1 Cryptographical Applications

We can think of a variety of applications for embedding families. General group testing applications, for example, may take advantage of this family for cases where increasing n is necessary, together with the possibility of larger d's. Here we are most interested in applications related to cryptography.

Aggregation of signatures: The purpose of aggregation of signatures is to save on storage, communication and verification time by combining several signatures together [1], and d-CFFs are known to provide this while allowing the identification of up to d invalid signatures [8, 9, 24]. Since the number of signatures may not be known a priori, it is important to have a d-CFF that allows the increase of n. However, after signatures are aggregated together using a smaller matrix, the individual signatures are discarded and we only keep the aggregated ones, which implies that larger matrices should not require the knowledge of those signatures that were discarded. A solution for this was first proposed by Hartung et al. [8] using monotone families, where the zero matrix below $\mathcal{M}^{(l)}$ address this requirement directly. Nested families can also be used as a solution to this problem, since its submatrix $Z^{(l)}$ also addresses this requirement by requiring only one extra aggregation of past signatures [9]. The advantage of nested families is that the known constructions present a much better compression ratio, which is closer to the upper bound, and consequently gives smaller aggregate signature size [9]. For this application, it seems unlikely that we can apply embedding families that are not nested, so there is little hope for increasing d.

Broadcast authentication: In this scheme, sender and receivers agree on secret keys, and these keys are used to guarantee the authenticity of broadcasted messages. However, there may be malicious users who can get together and use their secret keys and previous communication to create fraudulent messages, which may be accepted by some users as authentic [13, 17]. Safavi-Naini and Wang [17] propose the use of d-CFFs to manage the distribution of keys. Again, the columns of the CFF represent the users, the rows represent the keys, and each user receives a subset of these keys corresponding to their column of the matrix. Because of the d-CFF property, the union of the keys of up to d malicious users is not enough to create a fraudulent message [13, 17]. In this scenario, we could again think of embedding families as a way to support an increasing number of malicious users as the number of users grow. Due to the fact that $Z^{(l)} \neq 0$ for embedding families with increasing d, this requires old users to receive new keys when we increase n and d, which can be considered a reasonable requirement given the benefit that increasing d brings to the application.

Broadcast encryption: In this scheme, a sender broadcasts encrypted messages to a set of n users, but wants to prevent some of them from recovering these messages. An example of such application is paid television, where only some users can have access to certain paid channels [7]. Gafni et al. [7] discuss the use of d-CFF for distributing the keys that are used to encrypt and decrypt the message. In this scenario, the columns of the d-CFF represent the users, and the rows represent a set of t keys. Each user receives a subset of the keys according to their column, and the d-CFF property guarantees that we can exclude up to d users from broadcasts and their respective keys without compromising the ability of the remaining users to decrypt the content. In this scheme, an embedding family would provide a *scalable* scheme [7], where we can add new users by increasing n, and additionally handle a larger number d of users that may be excluded. In light of Proposition 4.3.4, in order to increase d, we must have $Z^{(l)} \neq 0$, so some *rekeying* is necessary, where old users receive some of the new keys. This is a reasonable requirement when considering that this scheme tolerates an increasing number of excluded users as we increase the number of users.

4.4 Embedding Sequences Using Polynomials Over Finite Fields

In this section, we present a construction for embedding sequences based on a known construction of d-CFFs. We start by presenting a construction proposed by Erdös, Frankl and Füredi [5], that uses polynomials of degree up to k over a finite field $\mathbb{F}_q$, denoted as $f \in \mathbb{F}_q[x]_{\leq k}$, and generates a d-CFF$(t = q^2, n = q^{k+1})$ for $d \leq \frac{q-1}{k}$. We also note that this known construction presents some interesting properties that allow us to ignore a few rows of the d-CFF if we need smaller values of d (see Theorem 4.4.4). We finally show how to use this polynomial construction to obtain embedding families, and how we can focus on prioritizing increases of d, n, and obtain monotone families with increasing compression ratio from it.

Construction 4.4.1 (Erdös et al. [5], Example 3.2)**.** *Let q be a prime power, k a positive integer, and consider the elements of the finite field as $\mathbb{F}_q = \{x_1, \ldots, x_q\}$. We define $(X, \mathcal{B})$ as follows, for each polynomial $f \in \mathbb{F}_q[x]_{\leq k}$:*

$$X = \mathbb{F}_q \times \mathbb{F}_q = \{(x_i, x_j) : i, j = 1, \ldots, q\},$$
$$B_f = \{(x_1, f(x_1)), \ldots, (x_q, f(x_q))\} \subset X,$$
$$\mathcal{B} = \{B_f : f \in \mathbb{F}_q[x]_{\leq k}\}.$$

Call $C_{q,k}$ the incidence matrix obtained.

The argument in the following proof was observed in [8, 15].

Theorem 4.4.2. *[Erdös et al. [5], Example 3.2] Let q be a prime power, $k \geq 1$, and $d \leq \frac{q-1}{k}$. Then $C_{q,k}$ from Construction 4.4.1 is a d-CFF(q^2, q^{k+1}).*

Proof. It is easy to see that $|X| = q^2, |B_f| = q$, and $|\mathcal{B}| = q^{k+1}$. Moreover, for $f_i \neq f_j$ we have $|B_{f_i} \cap B_{f_j}| \leq k$. So when we consider any $d+1$ subsets $B_{f_0}, B_{f_1}, \ldots, B_{f_d}$, we have that

$$\left| B_{f_0} \setminus \bigcup_{i=1}^{d} B_{f_i} \right| \geq q - dk \geq 1,$$

the last inequality due to hypothesis $dk \leq q - 1$. Thus, $B_{f_0} \not\subseteq \bigcup_{i=1}^{d} B_{f_i}$.

Therefore, $(X, \mathcal{B})$ represents a d-CFF(q^2, q^{k+1}) with $d \leq \frac{q-1}{k}$. □

As an example, for $q = 3, d = 2, k = 1$, we have $X = \mathbb{F}_3 \times \mathbb{F}_3 = \mathbb{Z}_3 \times \mathbb{Z}_3$, polynomials with coefficients in $\mathbb{F}_3$ of degree up to $k = 1$, and obtain the 2-CFF$(9, 9)$ in Figure 4.2.

We note that the d-CFF construction presented above has a special structure that can be exploited to guarantee some interesting properties. Here we focus on the property of being able to discard some rows of the d-CFF incidence matrix when smaller values of d are enough, and the ability to increase d as necessary (up to a maximum) by considering

	0	1	2	x	$x+1$	$x+2$	$2x$	$2x+1$	$2x+2$
$(0,0)$	1	0	0	1	0	0	1	0	0
$(0,1)$	0	1	0	0	1	0	0	1	0
$(0,2)$	0	0	1	0	0	1	0	0	1
$(1,0)$	1	0	0	0	0	1	0	1	0
$(1,1)$	0	1	0	1	0	0	0	0	1
$(1,2)$	0	0	1	0	1	0	1	0	0
$(2,0)$	1	0	0	0	1	0	0	0	1
$(2,1)$	0	1	0	0	0	1	1	0	0
$(2,2)$	0	0	1	1	0	0	0	1	0

Figure 4.2: Example of a 0-CFF$(3,9)$, 1-CFF$(6,9)$ and a 2-CFF$(9,9)$.

extra rows. This property is important because it allows the testing algorithm to do an early abort doing only enough tests to detect the actual defective elements d' if $d' < d$, where d is the maximum $\lfloor \frac{q-1}{k} \rfloor$ allowed by the d-CFF matrix. In the example above, for instance, if we discard the last three rows we obtain a 1-CFF$(6,9)$.

For the remainder of this paper we consider a *block of rows* in this construction as the set of q rows $\{(x_i, x_1), (x_i, x_1), \ldots, (x_i, x_q)\}$ for every $x_i \in \mathbb{F}_q$. When we restrict our matrix to i blocks of rows, we are considering $X = \{x_1, \ldots, x_i\} \times \mathbb{F}_q$, $B_f(i) = \{(x_1, f(x_1)), \ldots, (x_i, f(x_i))\}$ and $\mathcal{B}(i) = \{B_f(i) : f \in \mathbb{F}_q[x]_{\leq k}\}$.

Construction 4.4.3. *Let q be a prime power, $k \geq 1$, and $q \geq dk+1$. Let $C_{q,k,d}$ be the matrix corresponding to $\mathcal{B}(dk+1)$, or in other words, the matrix $C_{q,k}$ from Construction 4.4.1 restricted to the first $(dk+1)$ blocks of rows.*

Theorem 4.4.4. *Let q be a prime power, $k \geq 1$, and $q \geq dk+1$. Then $C_{q,k,d}$ from Construction 4.4.3 is a d-CFF$((dk+1)q, q^{k+1})$.*

Proof. The proof follows a similar argument as for Theorem 4.4.2. We have $|B_{f_i}(a) \cap B_{f_j}(a)| \leq k$ for all $f_i, f_j \in \mathbb{F}_q[x]_{\leq k}$ and $1 \leq a \leq q$. Taking any $d+1$ distinct sets $B_{f_0}(dk+1), B_{f_1}(dk+1), \ldots, B_{f_d}(dk+1)$, we have $|B_{f_0}(dk+1) \setminus \bigcup_{i=1}^{d} B_{f_i}(dk+1)| \geq dk+1-dk \geq 1$. So $\mathcal{B}(dk+1)$ has the d-cover-free property and $C_{q,k,d}$ is a d-CFF$((dk+1)q, q^{k+1})$. □

For the case of $k = 1$, we observe that this incremental d property was given in [14]. This is because the constructions presented in [14] are based on Mutually Orthogonal Latin Squares (MOLS), which can be constructed with polynomials of degree $k = 1$. For general k, Construction 4.4.1 is equivalent to a construction of traceability codes using Reed-Solomon codes [18, Theorem 4.4].

4.4.1 Embedding Sequence Construction

In this section, we give constructions of embedding sequences of CFFs using the previous construction and extension fields. We start with a prime power q and consider the increase

as q^{2^i} for $i \geq 0$, which gives a direct increase of n and t. Since we are increasing q, we may also consider increasing k and/or d as long as we respect the inequality $q \geq dk + 1$. By increasing k to some k' we make n grow faster and consequently improve the compression ratio. By increasing d to d' we can allow the identification of more defective elements, which may be necessary as the number of elements n grows.

The following theorem is the basic step to be used in the embedding sequence construction, which is related to embedding Reed-Solomon codes [16] into larger Reed-Solomon codes via extension fields.

Theorem 4.4.5. *Let $q \geq dk+1, k' \geq k, d' \geq d$ and $q^2 \geq d'k'+1$. Let $C_{q,k,d}$ and $C_{q^2,k',d'}$ be the CFF matrices obtained from the polynomial construction (Construction 4.4.3). Then, there exists $\mathbf{C}_{q^2,k',d'}$ obtained from $C_{q^2,k',d'}$ by a column and row permutation that has the form*

$$\mathbf{C}_{q^2,k',d'} = \begin{pmatrix} C_{q,k,d} & Y \\ Z & W \end{pmatrix}.$$

Moreover, $\mathbf{C}_{q^2,k',d'}$ is a d'-CFF$((d'k'+1)q^2, q^{2(k'+1)})$ and $C_{q,k,d}$ is a d-CFF$((dk+1)q, q^{(k+1)})$.

Proof. Let $\mathbb{F}_q = \{x_1, \ldots, x_q\}$. To form $\mathbf{C}_{q^2,k',d'}$ we first list the rows of $C_{q^2,k',d'}$ that are of the form (x_i, x_j) for all $1 \leq i \leq dk+1$, $1 \leq j \leq q$ and its columns indexed by B_f for all $f \in \mathbb{F}_q[x]_{\leq k}$, followed by the remaining rows and columns in some order. Since $\mathbb{F}_q$ is a subfield of $\mathbb{F}_{q^2}$, $\mathbb{F}_q[x]_{\leq k} \subseteq \mathbb{F}_{q^2}[x]_{\leq k'}$, so we can list the columns starting by all $f \in \mathbb{F}_q[x]_{\leq k}$ followed by $f \in \mathbb{F}_{q^2}[x]_{\leq k'} \setminus \mathbb{F}_q[x]_{\leq k}$. Thus, the $(dk+1)q \times q^{k+1}$ submatrix of $\mathbf{C}_{q^2,k',d'}$ in the upper left corner coincides precisely with $C_{q,k,d}$. The fact they are d-CFF and d'-CFF comes from Theorem 4.4.4. □

As an example, consider $q = 9, d = 2$ and $k = 1$, and increase the last two parameters to $d' = 4, k' = 2$. We are able to increase $C_{3,1,2}$ from Figure 4.2 and obtain a 4-CFF$(81, 729)$ $\mathbf{C}_{9,2,4}$, as shown in Figure 4.3.

	0	1	2	x	$x+1$	$x+2$	$2x$	$2x+1$	$2x+2$	α	$\alpha+1$	$\ldots$	$(2\alpha+2)x^2+(2\alpha+2)x+2\alpha+2$
$(0,0)$	1	0	0	1	0	0	1	0	0				
$(0,1)$	0	1	0	0	1	0	0	1	0				
$(0,2)$	0	0	1	0	0	1	0	0	1				
$(1,0)$	1	0	0	0	0	1	0	1	0				
$(1,1)$	0	1	0	1	0	0	0	0	1				
$(1,2)$	0	0	1	0	1	0	1	0	0				
$(2,0)$	1	0	0	0	1	0	0	0	1				
$(2,1)$	0	1	0	0	0	1	1	0	0			$\ldots$	
$(2,2)$	0	0	1	1	0	0	0	1	0				
$(\mathbb{F}_3, \mathbb{F}_9 \setminus \mathbb{F}_3)$						0							
$\vdots$						$\ldots$							
$(2\alpha+2, 2\alpha+2)$	0	0	0	1	0			$\ldots$					

Figure 4.3: Example of a 4-CFF$(81, 729)$.

Theorem 4.4.5 allows the construction of infinite families of CFF with special properties, namely monotone families, and embedding families with increasing d and/or k which have specific advantages.

Table 4.1: Example of prioritizing d increases with fixed $k = 2$.

i	q	k	d	n	t	n/t
0	4	2	1	64	12	5.33
1	16	2	7	4096	240	17.06
2	256	2	127	16777216	65280	257.00
3	65536	2	32767	281474976710656	4294901760	65537.00

Table 4.2: Example of prioritizing d increases with fixed $k = 3$.

i	q	k	d	n	t	n/t
0	4	3	1	256	16	16
1	16	3	5	65536	256	256
2	256	3	85	4294967296	65536	65536
3	65536	3	21845	65536^4	4294967296	4294967296

Theorem 4.4.6. *Let q be a prime power, $q \geq d_0k_0 + 1$. Let $k_i \leq k_{i+1}, d_i \leq d_{i+1}$, $q^{2^i} \geq d_{i+1}k_{i+1} + 1$, for all $i \geq 0$. Then, the sequence $\{\mathbf{C}_{q^{2^i},k_i,d_i}\}_{i\geq 0}$ is an embedding family of CFFs.*

Proof. For $q \geq d_0k_0 + 1$ we build a d_0-CFF C_{q,k_0,d_0} using the polynomial construction (Construction 4.4.1, Theorem 4.4.4). Since $k_i \leq k_{i+1}, d_i \leq d_{i+1}, q^{2^i} \geq d_ik_i + 1$, for each $i \geq 0$, we apply Theorem 4.4.5 to embed $\mathbf{C}_{q^{2^i},k_i,d_i}$ in $\mathbf{C}_{q^{2^{i+1}},k_{i+1},d_{i+1}}$. □

The following corollaries show useful applications of Theorem 4.4.6.

Corollary 4.4.7 (Prioritizing d increase)**.** *Let $d_0 \geq 1$, $k \geq 1$, and let q be a prime power such that $q > d_0k$. Let $d_i = \lceil \frac{q^{2^i}}{k} \rceil - 1$, for $i \geq 1$. Then $\{\mathbf{C}_{q^{2^i},k,d_i}\}_{i\geq 0}$ is an embedding family of CFFs. Moreover, its compression ratio is $\rho(n) = n^{1-\frac{2}{k+1}}$ and $d \sim \frac{n^{1/(k+1)}}{k}$.*

Proof. We have that $d_i = \lceil \frac{q^{2^i}}{k} \rceil - 1 < \frac{q^{2^i}}{k}$ and therefore $d_ik < q^{2^i}$, which satisfies the hypothesis of Theorem 4.4.6. Finally, for fixed k and assuming the use of all rows of the matrix, we easily calculate the compression ratio $\frac{n}{t} = \frac{(q^{2^i})^{k+1}}{(q^{2^i})^2} = \frac{n}{n^{2/k+1}} = n^{1-\frac{2}{k+1}}$, which is increasing when $k \geq 2$. □

We show a few examples in Table 4.1 and Table 4.2 for $q = 4, 16, 256, 65536$ and for fixed values of k. For each q and k we compute $d = \lceil \frac{q}{k} \rceil - 1$ and $n = q^{k+1}$. We note that as k increases, the maximum value of d decreases but we get constructions with a better ratio.

Corollary 4.4.8 (Prioritizing ratio increase)**.** *Let $d \geq 1$ and $k_0 \geq 1$, q a prime power such that $q > dk_0$. Let $k_i = \lceil \frac{q^{2^i}}{d} \rceil - 1$. Then $\{\mathbf{C}_{q^{2^i},k_i,d}\}_{i\geq 0}$ is an embedding family of CFFs. Moreover, the compression ratio is $\rho(n) = \frac{n}{\log n}$.*

Proof. We have that $k_i = \lceil \frac{q^{2^i}}{d} \rceil - 1 < \frac{q^{2^i}}{d}$, and therefore $k_i d < q^{2^i}$, which satisfies the hypothesis of Theorem 4.4.6. Finally, for fixed d and assuming the use of all rows of the matrix, we easily obtain the compression ratio $\frac{n}{t} = \frac{(q^{2^i})^{k_i+1}}{(q^{2^i})^2} \leq \frac{(q^{2^i})^{\frac{q^{2^i}}{d}}}{(q^{2^i})^2} \leq \frac{n}{d \log n}$, since $d \log n = q^{2^i} \log q^{2^i} < (q^{2^i})^2$. Thus, $\frac{n}{t}$ is $\mathrm{O}(\frac{n}{\log n})$. □

We show some examples in Table 4.3 and Table 4.4 for $q = 4, 16, 256, 65536$, fixed values of d, and increasing k. For each q and d we compute $k = \lceil \frac{q}{d} \rceil - 1$ and $n = q^{k+1}$. We note that the ratio grows very quickly as k increases to its maximum.

Table 4.3: Example of prioritizing ratio increase with fixed $d = 2$.

i	q	k	d	n	t	n/t
0	4	1	2	16	12	1.33
1	16	7	2	4294967296	240	17895697.07
2	256	127	2	256^{128}	65280	2.75×10^{303}
3	65536	32767	2	65536^{32768}	4294901760	6.04×10^{157816}

Table 4.4: Example of prioritizing ratio increase with fixed $d = 3$.

i	q	k	d	n	t	n/t
0	4	1	3	16	16	1
1	16	5	3	16777216	256	65536
2	256	85	3	256^{86}	65536	1.95×10^{202}
3	65536	21845	3	65536^{21846}	4294967296	1.54×10^{105211}

Monotone families are desirable for some applications due to their flexibility since the new tests involve only new items. By selecting specific blocks of rows for the embedding family, we are able to achieve monotone families with better compression ratio than previously known.

Theorem 4.4.9. *Let $d \geq 1$ and $k \geq 1$, q a prime power such that $q \geq dk + 1$. Let $C_{q,k,d}$ be a d-CFF obtained from Construction 4.4.3 and $\{\mathbf{C}_{q^{2^i},k,d}\}_{i \geq 0}$ be an embedding family of d-CFFs for fixed k and d, obtained from recursively applying Theorem 4.4.5, where $\mathbf{C}_{q,k,d} = C_{q,k,d}$ and $\mathbf{C}_{q^{2^i},k,d}$ be the reordered matrix as shown in Theorem 4.4.5. Then $\{\mathbf{C}_{q^{2^i},k,d}\}_{i \geq 0}$ is a monotone family of d-CFF$(t = (dk+1)q^{2^i}, n = q^{2^i(k+1)})$. Moreover, the compression ratio is $\rho(n) = n^{1-\frac{1}{k+1}}$.*

Proof. By fixing $x_l \in \mathbb{F}_q, l = 1, \ldots, dk + 1$ we obtain a matrix with $dk + 1$ blocks of rows and consequently $|B_f| = dk + 1$, and by the same argument as in Theorem 4.4.4 we know that each matrix $\mathbf{C}_{q^{2^i},k,d}$ is a d-CFF with $d \leq \frac{q-1}{k}$. Moreover, if we consider the columns of $\mathbf{C}_{q^{2^i},k,d}$ that are represented by polynomials $f \in \mathbb{F}_{q^{2^{i-1}}}[x]_{\leq k}$, we know $f(x_l) = x_j \in \mathbb{F}_{q^{2^{i-1}}}$, and consequently $\mathbf{C}_{(x_l,x_j),f} = 0$ for all the cases where $x_j \in \mathbb{F}_{q^{2^i}} \backslash \mathbb{F}_{q^{2^{i-1}}}, f \in \mathbb{F}_{q^{2^{i-1}}}[x]_{\leq k}$. It is easy to see that this matches the definition of monotone family of d-CFFs. Finally, if we use the maximum $d = \lfloor \frac{q-1}{k} \rfloor$ we obtain a sequence of d-CFF$(t = q \times q^{2^i}, n = (q^{2^i})^{k+1})$, which has compression ratio $\frac{n}{t} = \frac{(q^{2^i})^{k+1}}{q \times q^{2^i}} = \frac{n}{q n^{1/(k+1)}}$, which is $O(n^{1-\frac{1}{k+1}})$. □

For an example of applying Theorem 4.4.9 with $q = 3, d = 2, k = 1$, we refer to Table 4.3 to obtain the first two matrices in the sequence. Indeed, $M_{3,1,2}$ is the top left sub-matrix, and $M_{9,1,2}$ is the given matrix restricted to the first $dk + 1 = 3$ blocks of rows (first two groups of rows in Table 4.3), and the columns corresponding to polynomials of degree up to $k = 1$. The ratio of this monotone family is $\rho(n) = \sqrt{n}/3$.

Table 4.5 summarizes the results of Corollaries 4.4.7, 4.4.8, and Theorem 4.4.9.

Table 4.5: Summary of results for $k \geq 2$.

	k	d	$\rho(n)$	Feature
Corollary 4.4.7	fixed	$d \sim \frac{n^{1/(k+1)}}{k}$	$n^{1-\frac{2}{k+1}}$	increasing d
Corollary 4.4.8	increasing	fixed	$\frac{n}{\log n}$	optimal ratio
Theorem 4.4.9	fixed	fixed	$n^{1-\frac{1}{k+1}}$	monotone

4.5 Using embedding families in applications

The use of embedding families given in Theorem 4.4.6 requires some caution. While compression ratios are excellent when each full matrix is used, as seen in Corollary 4.4.7, Corollary 4.4.8, and Theorem 4.4.9, bad ratios can be found when we need to add many fewer items than the maximum n for a matrix. Note that when the number of columns of $\mathcal{M}^{(l)}$ is exceeded we need to use $\mathcal{M}^{(l+1)}$ and remove unused columns. For example, with $q = 4, d = 1, k = 2$ we get a maximum $n = 64$, $t = (dk+1)q = 12$ and ratio $\rho(n) = 5.33$. If we decide to use the extension field to get larger values of n, the next value will be $q = 16$ which gives $t = (dk + 1)q = 48$. This new matrix can handle up to $n = 4096$, but for the case where we only need $n = 65$ we will have a very small ratio of $\rho(n) = 1.35$. For this reason it would be desirable to develop techniques for "smoothing out" the transition in compression ratio when we move from one matrix to the next in the embedding family.

One strategy to reduce these sharp transitions is to use values of d and k much smaller than the maximum allowed by the construction. In Figure 4.4 and Table 4.6 we show a choice of $q = 16, 256$, $d = \log_4 n$, and necessary increases of $k = 1, 2, 3$ to achieve the desired values of n. We note that as we change from one field ($q = 16$) to the next one ($q = 256$) we have a drop on the compression ratio, and this is due to the increase in the number of rows $(dk + 1)q$ as we increase q. As n grows, the increasing ratio is restored.

4.6 Generalized Construction of Embedding Families

In this section, we present a generalization of the results presented in Section 4.4. We start by defining a few objects that will be used, such as Orthogonal Arrays (OAs), Packing Arrays (PAs), and their relationships with separating hash families (SHF). Then we discuss how they can be used to construct embedding families.

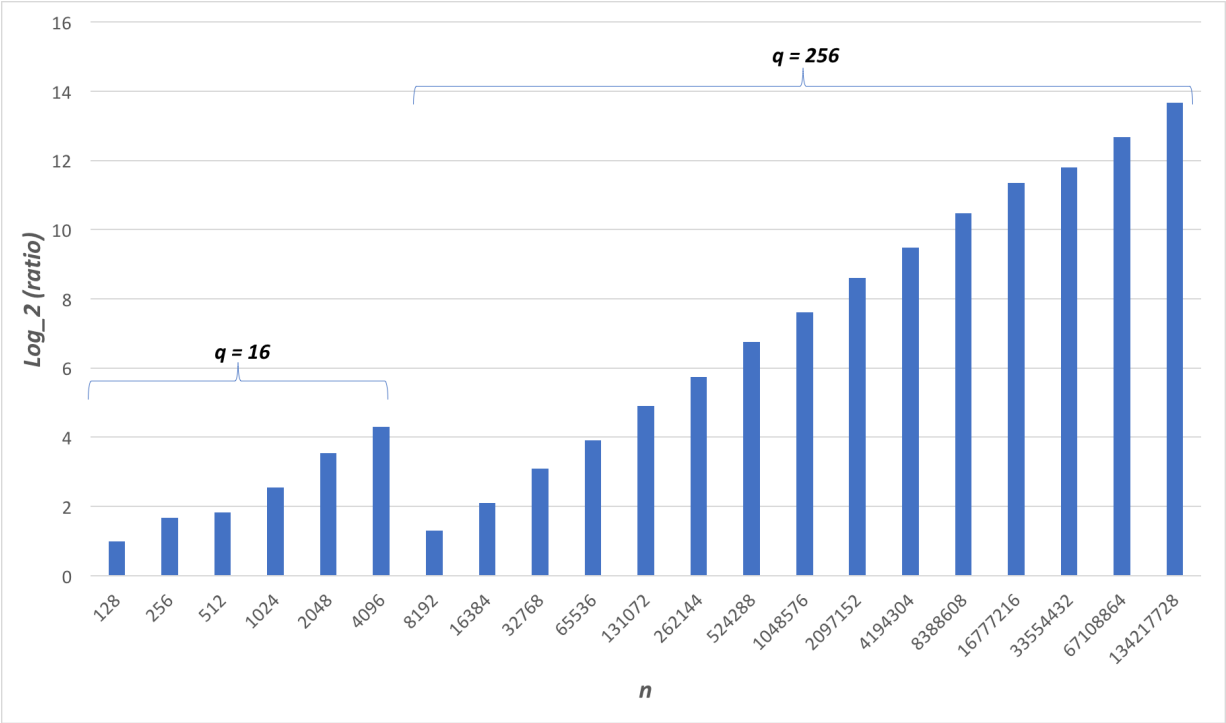


Figure 4.4: Compression ratio for $q = 16, 256; 1 \leq k \leq 3, ; d = \log_4 n$.

Definition 4.6.1. *An* orthogonal array *$OA(v^t; t, k, v)$ is an $v^t \times k$ array with elements from an alphabet of v symbols, such that in any t columns, every t-tuple of elements is contained in exactly one row.*

Definition 4.6.2 (Stevens and Mendelsohn [19])**.** *A* packing array*, $PA(n; t, k, v)$, is an $n \times k$ array with values from an alphabet with v symbols, such that in any t columns, every t-tuple of elements is contained in at most one row.*

We note that for a $\mathrm{PA}(n; t, k, v)$ we must have $n \leq v^t$, and that an $\mathrm{OA}(v^t; t, k, v)$ is a packing array with maximum number of rows.

Definition 4.6.3 (Stinson et al. [22])**.** *An $(n, m, \{w_1, \ldots, w_t\})$-separating hash family is a set of functions $\mathcal{F}$, such that $|Y| = n, |X| = m$, $f : Y \to X$ for each $f \in \mathcal{F}$, and for any sets $C_1, \ldots, C_t \subseteq \{1, 2, \ldots, n\}$ such that $|C_i| = w_i, 1 \leq i \leq t$ and $C_i \cap C_j = \emptyset$ for $i \neq j$, there exist at least one function $f \in \mathcal{F}$ such that*

$$\{f(y) : y \in C_i\} \cap \{f(y) : y \in C_j\} = \emptyset.$$

For $|\mathcal{F}| = N$, the $(n, m, \{w_1, \ldots, w_t\})$-separating hash family is denoted $SHF(N; n, m, \{w_1, \ldots, w_t\})$.

Remark 4.6.4. *A $SHF(N; n, m, \{w_1, \ldots, w_t\})$ can be depicted as an $N \times n$ matrix A with entries from $\{1, 2, \ldots, m\}$, where the rows represent the functions f, the columns*

Table 4.6: Compression ratio for $q = 16, 256; 1 \leq k \leq 3; d = \log_4 n$.

q	k	d	n	t	$\rho(n) = n/t$
16	1	3	128	64	2.00
16	1	4	256	80	3.20
16	2	4	512	144	3.55
16	2	5	1024	176	5.81
16	2	5	2048	176	11.63
16	2	6	4096	208	19.69
256	2	6	8192	3328	2.46
256	2	7	16384	3840	4.26
256	2	7	32768	3840	8.53
256	2	8	65536	4352	15.05
256	2	8	131072	4352	30.11
256	2	9	262144	4864	53.89
256	2	9	524288	4864	107.78
256	2	10	1048576	5376	195.04
256	2	10	2097152	5376	390.09
256	2	11	4194304	5888	712.34
256	2	11	8388608	5888	1424.69
256	2	12	16777216	6400	2621.44
256	3	12	33554432	9472	3542.48
256	3	13	67108864	10240	6553.60
256	3	13	134217728	10240	13107.20

represent the elements in Y, and the entry in $A(f, y) = f(y)$. Given disjoint sets of columns $C_1, \ldots, C_t$, there exists at least one row r in A with the following property:

$$\{A(r, y) : y \in C_i\} \cap \{A(r, y) : y \in C_j\} = \emptyset,$$

for all $i \neq j$.

The next figure shows an example of an SHF$(2; 6, 4, \{1, 2\})$, based on a construction given by Li, Van Rees and Wei [12].

1	2	3	4	4	4
4	4	4	1	2	3

Figure 4.5: An SHF$(2; 6, 4, \{1, 2\})$.

Now we present some relationships between packing arrays and separating hash families, and how we can use them to construct CFFs and embedding families. In the following propositions, we consider a PA$(n; t, k, v)$ and the fact that any two rows have at most $t - 1$ positions in common. A similar result is presented by Stinson et al. [23], where they propose the use of orthogonal arrays to construct perfect hash families, and mention that similar results can be achieved for SHFs.

Proposition 4.6.5. *If A is a PA$(n; t, k, v)$, and $w \leq \frac{k-1}{t-1}$, then A^T is a SHF$(N = k; n, m = v, \{1, w\})$.*

Proof. Take $w+1$ rows $r, r_1, r_2, \ldots, r_w$ of A. We need to find a column j such that

$$A[r_l, j] \neq A[r, j] \text{ for all } l \in \{1, \ldots, w\}. \tag{4.2}$$

Let $B_i = \{A[r, l] = A[r_i, l] : l \in \{1, \ldots, k\}\}$, $i \in \{1, \ldots, w\}$; we know $|B_i| \leq t-1$ since A is a PA. We claim there exists a column $j \in T = \{1, \ldots, k\} \backslash \bigcup_{i=1}^{w} B_i$, which is the desired column in (4.2). In fact,

$$|T| = |\{1, \ldots, k\} \backslash \bigcup_{i=1}^{w} B_i| \geq k - \sum_{i=1}^{w} |B_i| \geq k - w(t-1) \geq k - \frac{(k-1)}{(t-1)}(t-1) = 1$$

so column j exists, which becomes row j when using A^T, and therefore guarantees the necessary and sufficient property for a SHF of type $\{1, w\}$. □

Now we show we can construct d-CFF incidence matrices from SHFs. Similar results can be found in [11] from perfect hash families, and a more general result can be found in [21] for the construction of (w, d)-CFFs from SHFs of type $\{w, d\}$.

Construction 4.6.6. *Let A be a $SHF(N; n, m, \{1, w\})$, we build an $mN \times n$ matrix $\mathcal{M}$ as follows. Index each row of $\mathcal{M}$ with tuples (i, x), for $x = 1, \ldots, m$, $i = 1, \ldots, N$, and each column $j = 1, \ldots, n$, then*

$$\mathcal{M}_{(i,x),j} = \begin{cases} 1 & \text{if } A_{i,j} = x, \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 4.6.7. *Let A be a SHF$(N; n, m, \{1, w\})$, then $\mathcal{M}$ built via Construction 4.6.6 gives a d-CFF(Nm, n), with $d \leq w$.*

Proof. If we take $w+1$ columns $c_0, c_1, \ldots, c_w$ of a SHF A of type $\{1, w\}$, we know there is a row i such that $A[i, c_0] \neq A[i, c_j]$ for $j \in \{1, \ldots, w\}$. In Construction 4.6.6 we convert each element x in A into a vector of size m with one "1" in position x and "0" in the remaining positions. Due to the SHF property, we note that there is at least one row (i, x) such that $\mathcal{M}_{(i,x),c_0} = 1$ while $\mathcal{M}_{(i,x),c_j} = 0$, $(1 \leq j \leq w)$, for any columns $c_0, c_1, \ldots, c_w$ in $\mathcal{M}$, which matches the requirements for a w-CFF. Moreover, since we expand each row of A into an array of size m, $\mathcal{M}$ is w-CFF(mN, n). □

Remark 4.6.8. *If we use a PA$(n; t, k, v)$ to build a SHF$(N = k; n, m = v, \{1, w\})$ with $w \leq \frac{k-1}{t-1}$ as in Proposition 4.6.5 and then apply Construction 4.6.6, we obtain a d-CFF(Nm, n) with $d \leq \frac{k-1}{t-1}$.*

Remark 4.6.9. *For the special case where we use an OA($q^t; t, q, q$) constructed using polynomials over finite fields using Bush's construction [2], q a prime power, to build a SHF($q; q^t, q, \{1, w\}$) and then apply Construction 4.6.6, we have a d-CFF(q^2, q^t) for $d \leq \frac{q-1}{t-1}$, which is equivalent to Construction 4.4.1 using polynomials. This OA is another way of presenting Reed-Solomon codes.*

When we construct a d-CFF from a SHF that came from a PA, we observe a similar property of blocks of rows giving increasing values of d as in Theorem 4.4.4.

Proposition 4.6.10. *Let P be a PA($n; t, k, v$), $A = P^T$ be a SHF($N = k; n, m = v, \{1, w\}$) with $w \leq \frac{k-1}{t-1}$, and $k_i = i(t-1)+1$, for some $1 \leq i \leq w$. Consider $\mathcal{M}$ the w-CFF($k \times v, n$) obtained from Construction 4.6.6 using A and let a "block" of rows be any consecutive v rows of $\mathcal{M}$ indexed by $(b, 1), \ldots, (b, v)$, $1 \leq b \leq k_i$. When we restrict $\mathcal{M}$ to k_i blocks of rows we obtain an i-CFF($k_i \times v, n$).*

Proof. From Proposition 4.6.5 we know that a SHF of type $\{1, w\}$ can be created from a PA P of strength t as long as the number of columns of the PA is at least $k \geq w(t-1)+1$. We can restrict the packing array P to $k_i = i(t-1)+1$ columns, $1 \leq i \leq w$, without compromising its properties and therefore obtain a SHF A_i of type $\{1, i\}$. By applying Construction 4.6.6 with A_i we obtain an i-CFF($k_i \times v, n$). □

The next proposition shows that a special sequence of PAs generalizes the polynomial construction of embedding family given in Theorem 4.4.6.

Proposition 4.6.11. *Let $(\mathcal{P}^{(l)})_l$ be a sequence of PAs, where $\mathcal{P}^{(l)}$ is a PA($n_l; t_l, k_l, v_l$), $n_l \leq n_{l+1}, t_l \leq t_{l+1}, k_l \leq k_{l+1}, v_l \leq v_{l+1}$,*

$$\mathcal{P}^{(l+1)} = \begin{pmatrix} \mathcal{P}^{(l)} & Y \\ Z & W \end{pmatrix},$$

where Y, Z, W are 0-1 matrices of appropriate dimensions, and in addition $d_l \leq \frac{k_l-1}{t_l-1}$, $d_l \leq d_{l+1}$, for all $l \geq 1$. Then there exists an embedding family of d_l-CFF($k_l \times v_l, n_l$).

Proof. Let $\mathcal{A}^{(l)}$ be obtained by transposing $\mathcal{P}^{(l)}$. By Proposition 4.6.5, since $d_l \leq \frac{k_l-1}{t_l-1}$, $\mathcal{A}^{(l)}$ is a SHF($N_l = k_l; n_l, m_l = v_l, \{1, d_l\}$), and consequently $(\mathcal{A}^{(l)})_l$ is a sequence of SHFs with the following form

$$\mathcal{A}^{(l+1)} = \begin{pmatrix} \mathcal{A}^{(l)} & Z^T \\ Y^T & W^T \end{pmatrix}.$$

Then, for each $\mathcal{A}^{(l)}$ we apply Construction 4.6.6 and we get $\mathcal{M}^{(l)}$, a d_l-CFF($k_l \times v_l, n_l$) by Proposition 4.6.7. It is easy to see that the smaller CFF $\mathcal{M}^{(l)}$ is in the top corner of $\mathcal{M}^{(l+1)}$, and all the requirements for being an embedding family are satisfied. □

4.7 Conclusion

This paper introduces the idea of embedding families of CFFs as a general framework to look at how CFF constructions can be leveraged to optimize parameters of interest to applications. The infinite families obtained in Section 4.4 have good asymptotic compression ratios, some matching the information theoretical upper bound, and some permitting increase of d. However, these constructions present abrupt increases of t and n (when moving to the next q) that need to be "smoothed out" for improved use in applications, as discussed in Section 4.5. An important direction for future work is the study of adequate growth for d and k to yield smoother instances of these families.

We remark once more that while it is convenient for some applications (eg. [7]) that we use monotone families ($Z^{(l)} = 0$), Proposition 4.3.4 shows they cannot be used when we need to increase d. We hope our definition of embedding families gives a general framework for research on new constructions for increasing d.

Acknowledgments

The authors would like to thank the referees for valuable suggestions that helped improving this work.

Bibliography

[1] D. Boneh, C. Gentry, B. Lynn and H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, In: Biham E. (eds) *Advances in Cryptology – EUROCRYPT 2003*. Lecture Notes in Comput. Sci., vol 2656, Springer, Berlin, Heidelberg, 2003, 416–432.

[2] K. A. Bush, Orthogonal arrays of index unity, *Ann. Math. Statistics*, **23** (1952), 426–434.

[3] C. J. Colbourn and J. H. Dinitz, *Handbook of Combinatorial Designs*, Second Edition, Chapman & Hall/CRC, 2007.

[4] D. Z. Du and F. K. Hwang, *Combinatorial Group Testing and Its Applications*, World Scientific, 2000.

[5] P. Erdös, P. Frankl and Z. Füredi, Families of finite sets in which no set is covered by the union of r others, *Israel J. Math.*, **51** (1985), 79–89.

[6] Z. Füredi, On r-Cover-free Families, *J. Combin. Theory Ser. A*, **73** (1996), 172–173.

[7] E. Gafni, J. Staddon and Y. L. Yin, Efficient Methods for Integrating Traceability and Broadcast Encryption, In: Wiener M. (eds) *Advances in Cryptology – CRYPTO 1999*. Lecture Notes in Comput. Sci., vol 1666, Springer, Berlin, Heidelberg, 1999, 372–387.

[8] G. Hartung, B. Kaidel, A. Koch, J. Koch and A. Rupp, Fault-tolerant aggregate signatures. In *Public-Key Cryptography – PKC 2016*. Lecture Notes in Comput. Sci., vol 9614, Springer, Cham, 2016, 331–356.

[9] T. B. Idalino and L. Moura, Efficient unbounded fault-tolerant aggregate signatures using nested cover-free families, In: Iliopoulos C., Leong H., Sung WK. (eds) *Combinatorial Algorithms – IWOCA 2018*. Lecture Notes in Comput. Sci., vol 10979, Springer, Cham, 2018, 52–64.

[10] T. B. Idalino, L. Moura, R. F. Custódio and D. Panario, Locating modifications in signed data for partial data integrity, *Inform. Process. Lett.*, **115** (2015), 731–737.

[11] K. M. Kim, Perfect Hash Families: Constructions and Applications, Master's thesis, University of Waterloo, 2003.

[12] P. C. Li, G. H. J. van Rees and R. Wei, Constructions of 2-cover-free families and related separating hash families, *J. Combin. Des.* , **14** (2006), 423–440.

[13] S. Ling, H. Wang and C. Xing, *Cover-Free Families and Their Applications*, In: *Security in Distributed and Networking Systems*, chapter 4, 2007.

[14] J. Pastuszak, J. Pieprzyk and J. Seberry, Codes identifying bad signature in batches, In: Roy B., Okamoto E. (eds) *Progress in Cryptology – INDOCRYPT 2000*. Lecture Notes in Comput. Sci., vol 1977, Springer, Berlin, Heidelberg, 2000, 143–154.

[15] J. Pieprzyk, H. Wang and C. Xing, Multiple-time signature schemes against adaptive chosen message attacks, In: Matsui M., Zuccherato R.J. (eds) *Selected Areas in Cryptography – SAC 2003*. Lecture Notes in Comput. Sci., vol 3006, Springer, Berlin, Heidelberg, 2004, 88–100.

[16] I. S. Reed and G. Solomon, Polynomial codes over certain finite fields, *J. Soc. Indust. Appl. Math.*, **8** (1960), 300–304.

[17] R. Safavi-Naini and H. Wang, New results on multi-receiver authentication codes, In: Nyberg K. (eds) *Advances in Cryptology – EUROCRYPT 1998*. Lecture Notes in Comput. Sci., vol 1403, Springer, Berlin, Heidelberg, 1998, 527–541.

[18] J. N. Staddon, D. R. Stinson and R. Wei, Combinatorial properties of frameproof and traceability codes, *IEEE Trans. Inform. Theory*, **47** (2001), 1042–1049.

[19] B. Stevens and E. Mendelsohn, Packing arrays, *Theoret. Comput. Sci.*, **321** (2004), 25–148.

[20] D. R. Stinson and R. Wei, Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM J. Discrete Math.* , **11** (1998), 41–53.

[21] D. R. Stinson and R. Wei, Generalized cover-free families, *Discrete Math.*, **279** (2004), 463–477.

[22] D. R. Stinson, R. Wei and K. Chen, On generalized separating hash families, *J. Combin. Theory Ser. A*, **115** (2008), 105–120.

[23] D. R. Stinson, R. Wei and L. Zhu, New constructions for perfect hash families and related structures using combinatorial designs and codes, *J. Combin. Des.*, **8** (2000), 189–200.

[24] G. M. Zaverucha and D. R. Stinson, Group testing and batch verification, In: Kurosawa K. (eds) *Information Theoretic Security – ICITS 2009*. Lecture Notes in Comput. Sci., vol 5973, Springer, Berlin, Heidelberg, 2009, 140–157.

[25] G. M. Zaverucha and D. R. Stinson, Short one-time signatures, *Adv. Math. Commun.*, **5** (2011), 473–488.

Chapter 5

Modification tolerant signature schemes: location and correction

This chapter contains the following paper in preparation to be submitted for a conference.

IDALINO, T. B.; MOURA, L., ADAMS, C., *Modification tolerant signature schemes: location and correction*. In preparation for submission, 2019.

5.1 Abstract

This paper considers malleable digital signatures, for situations where data is modified after it is signed. They can be used in applications where either the data can be modified (collaborative work), or the data must be modified (redactable and content extraction signatures) or we need to know which parts of the data have been modified (data forensics). A classical digital signature is valid for a message only if the signature is authentic and not even one bit of the message has been modified. We propose a general framework of modification tolerant signature schemes (MTSS), which can provide either location only or both location and correction, for modifications in a signed message divided into n blocks. This general scheme uses a set of allowed modifications that must be specified. We present an instantiation of MTSS with a tolerance level of d, indicating modifications can appear in any set of up to d message blocks. This tolerance level d is needed in practice for parametrizing and controlling the growth of the signature size with respect to the number n of blocks; using combinatorial group testing (CGT) the signature has size $O(d^2 \log n)$ which is close to the best known lower bound of $\Omega(\frac{d^2}{\log d}(\log n))$. There has been work in this very same direction using CGT by Goodrich et al. (ACNS 2005) and Idalino et al. (IPL 2015). Our work differs from theirs in that in one scheme we extend these ideas to include corrections of modification with provable security, and in another variation of the scheme we go in the opposite direction and guarantee privacy for redactable signatures, in this case preventing any leakage of redacted information.

5.2 Introduction

Classical digital signature schemes (CDSS) are used to guarantee that a document was created by the sender (authenticity) and has not been modified along the way (integrity); they also help to prevent the signer from claiming s/he did not sign a message (non-repudiation). The verification algorithm has a boolean output: a successful outcome is achieved if and only if both the signature is valid and the document has not been modified. In this paper, we consider more general digital signature schemes which we call *modification-tolerant signature schemes* (MTSS), which go beyond the ability of **detecting** modifications provided by CDSS, and have the ability of **locating** modifications or **locating and correcting** modifications. We discuss two types of modification-tolerant signature schemes: a general MTSS that allows the location of modified blocks of the data, and an MTSS with *correction capability*, that allows the correction of the modified blocks, recovering the original message. We give three instantiations of the scheme for the purpose of location, correction, and redaction.

In which situations can modifications be allowed or even desired in the context of digital signatures? One situation where modifications are desirable is the so-called redactable signatures [13], also called content extraction signatures [21]. In redactable signature schemes [13, 18, 21], some blocks of a document are redacted (blacked out) for privacy purposes, without interacting with the original signer and without compromising the validity of the signature. Related to these are the "sanitizable" signatures introduced by

Ateniese et al. [1] which are also used for the purpose of privacy of (parts of) the original message, but the modifications are done by a semi-trusted third party (the sanitizer) who can modify blocks of the document and sign the modified version, without the need of intervention by the original signer. Thus, in both redactable and sanitizable signature schemes, the privacy of the (redacted or modified parts of the) original message must be preserved. For MTSS, privacy of the original data that has been modified is not required. An MTSS with only location capability that guarantees privacy can be used for implementing redactable signatures, but that is not an intrinsic requirement for MTSS. Indeed, as pointed out in [18] the scheme provided in [12] does not meet standard privacy requirements. In the case of MTSS with correction capability, privacy cannot be guaranteed by definition, since the method permits the recovery of the original document.

A different scenario where a moderate amount of modification is desirable involves collaborative work. The authors of a document can use MTSS to allow further modifications as long as the original parts can be identified as their own. Other collaborators may apply modifications to the original document and append the original document's signature, which provides information about which blocks were modified, as well as a guarantee of integrity of the original blocks. MTSS can separate the original blocks from the modified ones, while correction capability further provides retrieval of the original version of the document.

Locating modifications has also been considered in situations where modifications are not desired, but their localization is used as a mechanism to mitigate damage. Indeed, in the context of message authentication codes for data structures, Goodrich et al. [11] propose a message authentication code (MAC) with modification locating capabilities. They propose their use in data forensics applications since the identification of which information was modified can be used to identify the perpetrator of the crime (for example: the salary of a person or the grade of a student was modified on a database). In [12], in the context of MTSS with only location capability, the authors mention the advantage of being able to guarantee the integrity of part of the data instead of the all-or-nothing situation given by a CDSS boolean outcome. For example, the integrity of 95% of a document or database may contain enough information needed for a specific application, whereas it would have been considered completely corrupted and unusable in the case of CDSS. In the case of MTSS with correction capability, we can go beyond mitigating the damage, and actually recover the originally signed document.

The mechanism behind the MTSS schemes instantiated here, like in [11, 12], is the use of cover-free families (CFF) in the same way as it is traditionally employed in combinatorial group testing. Combinatorial group testing has been used in cryptography in the context of digital signatures [23], broadcast communication [8, 20], and many others. The main idea is to test t groups, which are subsets of the n blocks, together (with enough redundancy and intersections between groups), and each group is used to produce a distinct hash. The tuple of t hashes is then signed and provided that no more than d blocks were modified, it is possible to identify precisely which blocks have been modified. The main efficiency concern is the compression ratio: the order of growth of n/t as n grows, for a fixed modification tolerance d. Using cover-free families, it is possible to achieve a compression ratio of $O(n/(d^2 \log n))$, which is not much worse than the $O(n)$ compression

ratio given by modification intolerant schemes such as CDSS.

Our contributions: In the present paper, we propose a general framework for MTSS, and an specific MTSS scheme for modification correction and another for redacted signatures. Both schemes are based on a modification tolerance d, indicating the maximum number of modifications or redactions. The security of the schemes rely only on an underlying classical digital signature scheme and on collision resistant hash functions; therefore the schemes can be used in the context of postquantum cryptography as long as these underlying building blocks are postquantum secure.

First, we extend methods that use cover-free families for modification location [11, 12] to further provide modification correction (Scheme 2 in Section 5.5.2), provided that the size of the blocks are small enough, say bounded by a constant s where an algorithm with 2^s steps can run in an acceptable amount of time. In short, the localization of the modified blocks is done using CFF similarly to [11, 12] with an extra constraint on blocks of size at most s, and an exhaustive search is used to correct the block to a value that makes the concatenation of a specific group of blocks match its original hash. The assumption that a collision resistant hash function does not cause collisions for messages of small size up to s is not only reasonable, but can be tested before employing it in the method.

Second, we propose a variation of the scheme for modification location to ensure total privacy of the modified blocks in order to extend it for the purpose of redactable signatures (Scheme 3 in Section 5.7). In this case, a block modification can be a redaction to hide private information. Unlike the modification correction scheme, this scheme does not need a restriction on block size.

Paper Organization: In Section 5.3, we discuss related work. In Section 5.4, we give a general framework for modification tolerant signature schemes with and without modification correction. In Section 5.5, we instantiate the schemes for location and/or correction that allows any modifications in up to d blocks using cover-free families. In Section 5.6, we prove the unforgeability of the schemes under the adaptive chosen message attack. In Section 5.7, we extend and instantiate the modification location scheme for redactable signatures, and prove it guarantees total privacy. In Section 5.8, we discuss implementation issues of the schemes proposed in this paper. In Section 5.9, we consider the relationship of parameters and the impact on the size of the signature and the ability to locate and correct modifications. A conclusion is given in Section 5.10.

5.3 Related work

The idea of error detection and correction of corrupted data is well established in coding theory. Combinatorial group testing can help locating where errors (modification of data) occurred, which can be considered an intermediate goal, which is stronger than error detection and more efficient than error correction. In the context of cryptography, localization of modified portions of data has been studied in the context of hash functions [4, 5], digital signatures [3, 12], and message authentication codes [7, 11]. Correction of modifications

is proposed by some schemes, but they have severe limitations on the correction capability [3, 7]. Schemes such as the ones in [4, 5, 11, 12] use techniques related to cover-free families to generate redundant information, which is used later to locate the modified portions of the data. The benefit of cover-free families is that they require a small amount of redundant data for a fixed threshold d in location capability.

The methods that provide location at the hash level [4, 5] are based on superimposed codes, which are binary codes equivalent to d-cover-free families (d-CFF). These codes are used in two steps of the process: to generate a small set of hash tags from an input message, and to verify the integrity of the message using these tags. Because of the d-CFF property, the scheme allows the identification of up to d corrupted data segments [4, 5]. A digital signature scheme with location of modifications is presented in [12]. Their scheme uses d-CFF matrices to generate a digital signature scheme that carries extra information for location of modifications. In this case, data is divided into n blocks which are concatenated and hashed according to the rows of the d-CFF. This set of hash values is signed using any classical digital signature algorithm, and both the signature and the set of hashes form the new signature of the message. The verification algorithm uses this information to precisely locate up to d blocks that contain modified content [12]. This is presented in detail in Section 5.5.2 as Scheme 1. In [11], the authors propose a solution for locating the items that were modified inside a data structure. They compute a small set of message authentication tags based on the rows of a d-disjunct matrix, which is equivalent to d-CFFs. These tags are stored within the topology of the data structure, and an auditor can later use these tags to determine exactly which pieces of data were modified and perform forensics investigation.

The methods described above have a common approach: they all compute extra integrity information based on combinatorial group testing techniques and use this information to locate up to d modified pieces of data. In our work, we show that if these pieces are small enough, it is possible to actually correct them to the originally-signed values. In the literature, we find signature and message authentication schemes that provide the correction of modifications, such as in [3, 7]. These schemes use different approaches than the ones previously discussed, and the capacity of location and correction is very limited. In [3] only a single error can be corrected, while in [7] the data is divided into blocks, and only one bit per block can be corrected.

Error correcting codes (d-ECC) are related to this work, as we could use them to provide authentication and correction of d modifications as we explain next. For a message m formed by blocks of up to $\log_2 q$ bits each, sign it using a CDSS and compute the corresponding codeword c_m using a d-ECC over alphabet q. Send the codeword c_m and the signature σ. The receiver obtains c', which may differ from c_m in at most d blocks, and by decoding c', obtains c_m and thus the original message m. Then, the verifier can check its authenticity and integrity by verifying (m, σ). In Section 5.5.3, we compare the direct use of d-ECC and Scheme 2 using d-CFF, and find that they have very similar compression ratios. Scheme 2 has the advantage of giving more information in the failing case, of being more efficient when we do not need correction, and of being a simple variation of Schemes 1 and 3, facilitating comparison among the different approaches.

There is also a solution proposed in [2] for location of modifications in images with digital signatures, using a different approach than d-CFFs. Their scheme consists of partitioning the image into n blocks, and generating one digital signature per block (with dependency on neighbours to avoid attacks). Although we can use signature schemes with very small signature sizes, the total number of signatures in this scheme is linear with the number of blocks n, while in [12], for example, the amount of extra information produced is logarithmic in n because of the d-CFF.

Redactable or content extraction digital signatures [6, 13, 21] are used when the owner of a document signed by another party, needs to show parts of that document to a third party, but wants to keep parts of the document private. In Section 5.7, we give a variation of an MTSS scheme that implements redactable signatures. More on related work involving redactable signatures is discussed in that section.

5.4 Framework for modification-tolerant digital signatures

5.4.1 Classical digital signature schemes (CDSS)

Classical digital signature schemes are based on public-key cryptography and consist of three algorithms: KeyGeneration, Sign, and Verify. We consider that any document or piece of data to be signed is a sequence of bits called a *message*.

Definition 5.4.1. *A classical digital signature scheme (CDSS) is a tuple Σ of three algorithms:*

- KeyGeneration(ℓ) *generates a pair of keys (secret and public) (SK, PK) for a given security parameter ℓ.*
- Sign(m, SK) *receives the message m to be signed, the secret key SK, and outputs the signature σ.*
- Verify(m, σ, PK) *takes as input a message m, signature σ, and public key PK. Using PK, outputs* 1 *if the pair (m, σ) is valid (as in Definition 5.4.2), and outputs 0 otherwise.*

Definition 5.4.2. *Let Σ be a CDSS as defined above, and let (SK, PK) be a pair of secret and public keys. A pair of message and signature (m, σ) is* valid *if σ is a valid output*[1] *of Σ.Sign(m, SK).*

Generally speaking, we say CDSS is *unforgeable* if a signature that verifies using PK can only be generated by the signer who has SK. In more detail, we consider the model

[1] We use "valid output" instead of Σ.Sign$(m, SK) = \sigma$ because the signing algorithm does not need to be deterministic.

of *adaptive chosen message attack*, where the attacker $\mathcal{A}$ adaptivelly chooses a list of messages $m_1, \ldots, m_q$, and requests the respective signatures $\sigma_1, \ldots, \sigma_q$ from an oracle $\mathcal{O}$. Given this information, we say the attacker performs an *existential forgery* if he is able to produce a valid signature σ on a new message m chosen by him, $m \neq m_i, 1 \leq i \leq q$ [15, 22]. Because with unlimited time an adversary can perform all possible computations, we limit the computational power of the attacker by requiring an efficient algorithm to be one that runs in probabilistic polynomial time (PPT). Moreover, we say a function f is *negligible* if for every polynomial p there exists an N such that for all $n > N$ we have that $f(n) < 1/p(n)$ [14].

Definition 5.4.3 (Unforgeability)**.** *A digital signature scheme is existentially unforgeable under an adaptive chosen message attack if there is no PPT adversary $\mathcal{A}$ that can create an existential forgery with non-negligible probability.*

5.4.2 Description of MTSS

Now we define the algorithms of the modification tolerant signature scheme (MTSS). We assume that the message m to be signed has size $|m|$ in bits, and is split into n *blocks*, not necessarily of the same size. A message split into blocks is represented as a sequence $m = (m[1], \ldots, m[n])$, where $m[i]$ represents the ith block of message m. For two messages m and m', each split into n blocks, we denote $\text{diff}(m, m') = \{i \in \{1, \ldots, n\} : m[i] \neq m'[i]\}$. An *authorized modification structure* is a collection $\mathcal{S} \subseteq P(\{1, \ldots, n\})$, where P is the power set of $\{1, \ldots, n\}$, that contains each set of blocks that the signer allows to be modified. The idea is that if σ is a valid signature for m, then σ is a valid signature for m' if and only if $\text{diff}(m, m') \in \mathcal{S}$. Of course, to prevent inconsistencies we must have $\emptyset \in \mathcal{S}$. Indeed, an MTSS is equivalent to a CDSS if $\mathcal{S} = \{\emptyset\}$. A more general example is $\mathcal{S} = \{\emptyset, \{2\}, \{3\}, \{5\}, \{2, 3\}, \{2, 5\}, \{3, 5\}\}$ for $n = 5$ blocks, which specifies blocks 1 and 4 cannot be changed and any change of at most two other blocks is allowed. The authorized modification structure is used to provide flexibility of modifications while providing control for the signer. In practice, we do not expect $\mathcal{S}$ to be stored explicitly, but instead to be implicitly enforced by the scheme.

Definition 5.4.4. *A modification tolerant signature scheme (MTSS) for authorized modification structure $\mathcal{S} \subseteq P(\{1, \ldots, n\})$ on messages with n blocks is a tuple Σ of three algorithms:*

- MTSS-KeyGeneration(ℓ) *generates a pair of secret and public keys (SK, PK) for a given security parameter ℓ.*
- MTSS-Sign(m, SK) *receives the message m to be signed, the secret key SK, and outputs the signature σ.*
- MTSS-Verify(m, σ, PK) *takes as input a message m, signature σ, and public key PK. Outputs $(1, I)$ if (m, σ) is valid for modification set I (as in Definition 5.4.5), and outputs 0 otherwise.*

Definition 5.4.5. *Let Σ be an MTSS for authorized modification structure $\mathcal{S} \subseteq P(\{1,\ldots,n\})$, and let (SK, PK) be a pair of secret and public keys. A pair (m,σ) of message and signature is* valid *if there exists m' such that σ is a valid output*[2] *of Σ.MTSS-Sign(m', SK) and diff$(m,m') \in \mathcal{S}$. In this case, we say (m,σ) is valid for modification set I (where $I =$ diff(m,m')).*

The definition of unforgeability of an MTSS scheme is exactly like Definition 5.4.3, but the existential forgery now needs to produce a valid signature as given in Definition 5.4.5.

Definition 5.4.6. *An MTSS Σ for authorized modification structure $\mathcal{S} \subseteq P(\{1,\ldots,n\})$ has* correction capability *if it is a tuple Σ of four algorithms, which, in addition to the algorithms in Definition 5.4.4, has the following algorithm:*

- MTSS-Verify&Correct(m,σ,PK)*: takes as input a message m, signature σ, and public key PK. Outputs a pair (ver, m') where:*
  1. *$ver = \Sigma$.MTSS-Verify(m,σ,PK).*
  2. *m' is a message with $m' \neq \lambda$ (the corrected message) if $ver = (1, I)$, $I =$ diff(m,m'), and (m',σ) is a valid pair for modification set $I' = \emptyset$; in all other cases $m' = \lambda$, which indicates failure to correct.*

Location of modifications with MTSS would be trivial for any $\mathcal{S} \subseteq P(\{1,\ldots,n\})$ if the signer simply produced σ as a tuple of n signatures, one for each block. However, this would be extremely inefficient for a large number of blocks n. We must reconcile the objectives of having a large $\mathcal{S}$ and having a compact signature. Of course, the signature size depends on the security parameter, but once this is fixed, we would like the signature to grow moderately as a function of n. This motivates the following definition of compression ratio.

Definition 5.4.7. *An MTSS Σ^n for messages with n blocks and signature σ with $|\sigma| \leq s(n)$ has compression ratio $\rho(n)$ if $\frac{n}{s(n)}$ is $O(\rho(n))$.*

The compression ratio measures how efficient our signature is with respect to the trivial scheme of keeping one signature per block, with $\rho(n) = O(1)$, supporting $\mathcal{S} = P(\{1,\ldots,n\})$. Classical signatures have $\rho(n) = n$ (best possible), but $\mathcal{S} = \{\emptyset\}$. In the next section we present a tradeoff, where $\rho(n) = \frac{n}{\log n}$ and $\mathcal{S}$ is the set of all sets of up to d modifications, for fixed d. Indeed, when using cover-free families it is possible to have a compression ratio of $O(\frac{n}{d^2 \log n})$ using known CFF constructions [19], while a lower bound on $s(n)$ [10] tells us we cannot have $\rho(n)$ larger than $\Theta(\frac{n}{(d^2/\log d)\log n})$.

[2]We use "valid output" instead of Σ.MTSS-Sign$(m', SK) = \sigma$ because the signing algorithm does not need to be deterministic.

5.5 d-modification tolerant MTSS based on combinatorial group testing

Here we propose an MTSS that allows the modification of any set of up to d blocks in the message, for a constant d which we call a tolerance level. In other words, the authorized modification structure is $\mathcal{S} = \{T \subseteq \{1, \ldots, n\} : |T| \leq d\}$. To obtain a compact signature size, we rely on combinatorial group testing, which we summarize in Section 5.5.1 before we describe the scheme in Section 5.5.2. Similar modification location methods based on combinatorial group testing have been proposed in [11, 12], and the instantiation we propose for the first three algorithms of MTSS (Section 5.5.2) are based on [12]. Our new contributions in this section include proof of security for the MTSS scheme based on cover-free families and the addition of error correction capability by proposing algorithm MTSS-Verify&Correct that corrects modifications in this context.

5.5.1 Cover-free families and group testing

Combinatorial group testing deals with discovering d defective items in a set of n items, via testing various groups of items for the presence of defects in each group. In nonadaptive combinatorial group testing, the groups are determined before the testing process starts. For the problems considered in this paper, a modified block is a defective item, and the groups are sets of blocks combined and hashed together. In our case, we must use nonadaptive combinatorial group testing, as the tests are generated at time of signing, while verification is done later. Cover-free families allow for a small number of groups that help to identify the modified blocks.

Recall that a permutation matrix of dimension l is an $l \times l$ matrix that is obtained from permuting rows of the identity matrix.

Definition 5.5.1. *A d-cover-free family (d-CFF) is a $t \times n$ binary matrix $\mathcal{M}$ such that any set of $d+1$ columns contains a permutation submatrix of dimension $d+1$.*

Each column of $\mathcal{M}$ corresponds to a block of the message, and each row corresponds to a group of blocks that will be tested together. A test fails if a group contains a modified block and passes if every block in a group is unchanged. The definition of d-CFF guarantees that for any column index j and any other set of d column indices $j_1, \ldots, j_d$, there will be a row i s.t. $\mathcal{M}_{i,j} = 1$, and $\mathcal{M}_{i,j_1} = \ldots = \mathcal{M}_{i,j_d} = 0$. In other words, for any unchanged block, there exists a test that contains that block but none of the up to d modified blocks.

Figure 5.1 shows an example of how to use a 2-CFF$(9, 12)$ to test a message with 12 blocks (represented by the columns) by testing 9 groups of blocks (the rows). Every unchanged block is in at least one group that passes the test (with result 0), and every group that failed the test (result **1**) contains at least one modified block, which in this example are blocks 3 and 12. We note that column "result" is the bitwise-or of the columns corresponding to modified blocks 3 and 12.

blocks	1	2	3	4	5	6	7	8	9	10	11	12	
	✓	✓	X	✓	✓	✓	✓	✓	✓	✓	✓	X	result:
t_1	1	0	0	1	0	0	1	0	0	1	0	0	0
t_2	1	0	0	0	1	0	0	1	0	0	1	0	0
t_3	1	0	0	0	0	1	0	0	1	0	0	1	**1**
t_4	0	1	0	1	0	0	0	0	1	0	1	0	0
t_5	0	1	0	0	1	0	1	0	0	0	0	1	**1**
t_6	0	1	0	0	0	1	0	1	0	1	0	0	0
t_7	0	0	1	1	0	0	0	1	0	0	0	1	**1**
t_8	0	0	1	0	1	0	0	0	1	1	0	0	**1**
t_9	0	0	1	0	0	1	1	0	0	0	1	0	**1**

Figure 5.1: Example of a 2-CFF$(9, 12)$.

The next theorem is important for the efficiency of MTSS-Verify&Correct. Indeed, it ensures that for each modified block, we can find a test that contains it together with only other unchanged blocks. Therefore, an exhaustive trial-and-error can be used to guess this block until the hash of this group of blocks matches the original hash.

Theorem 5.5.2. *Let $j_1, \ldots, j_d$ be the column indices that represent invalid elements. There is a row i such that $\mathcal{M}_{i,j_1} = 1, \mathcal{M}_{i,j_2} = \ldots = \mathcal{M}_{i,j_d} = 0$.*

Proof. Since $\mathcal{M}$ is a d-CFF, it is also a $(d-1)$-CFF, therefore one of the rows in the permutation submatrix indexed by $j_1, \ldots, j_d$ is as stated. □

5.5.2 Description of d-modification tolerant signature scheme

The main idea of a d-modification tolerant signature scheme is to sign a message split into blocks by concatenating the hashes of these blocks according to a d-CFF matrix. This allows us to locate up to d modified blocks in the signed message, and correct these modifications. In this context, we represent a concatenation of two strings a and b as $a||b$, and λ represents an empty string.

Definition 5.5.3. *A d-modification tolerant signature scheme (d-MTSS) is an MTSS with authorized modification structure $\mathcal{S} = \{T \subseteq \{1, \ldots, n\} : |T| \leq d\}$.*

We now give an instantiation of d-MTSS using d-cover-free families based on [12].

Scheme 1: A d-Modification Tolerant Signature Scheme

The scheme requires an underlying CDSS Σ, a public hash function h, and a d-CFF(t, n) matrix $\mathcal{M}$. The algorithms are given next:

- MTSS-KeyGeneration(ℓ): generates a key pair (SK, PK) using algorithm Σ.KeyGeneration(ℓ).

- MTSS-SIGN(m, SK): Takes as input a secret key SK and a message $m = (m[1], \ldots, m[n])$, and proceeds as follows.
  1. Calculate $h_j = h(m[j]), 1 \leq j \leq n$.
  2. For each $1 \leq i \leq t$, compute c_i as the concatenation of all h_j such that $\mathcal{M}_{i,j} = 1, 1 \leq j \leq n$. Set $T[i] = h(c_i)$.
  3. Compute $h^* = h(m)$ and set $T = (T[1], T[2], \ldots, T[t], h^*)$.
  4. Calculate $\sigma' = \Sigma.\textsc{Sign}(T, SK)$. Output signature $\sigma = (\sigma', T)$.
- MTSS-VERIFY(m, σ, PK): takes as input a message $m = (m[1], \ldots, m[n])$, signature $\sigma = (\sigma', T)$ for $T = (T[1], T[2], \ldots, T[t], h^*)$, and public key PK, and proceeds as follows.
  1. Ensure that $\Sigma.\textsc{Verify}(T, \sigma', PK) = 1$, otherwise stop and output $(0, -)$.
  2. Check if $h^* = h(m)$. Stop and output $(1, \emptyset)$ if that is the case, continue otherwise.
  3. Use $\mathcal{M}$ and m and do the same process as in steps 1 and 2 of MTSS-SIGN to produce hashes $T'[1], \ldots, T'[t]$.
  4. Start with an empty set V, and for each $1 \leq i \leq t$ such that $T[i] = T'[i]$, compute the set of indices of unmodified blocks $V_i = \{j : \mathcal{M}_{i,j} = 1\}$, and accumulate these values in the set of all indices of unmodified blocks $V = V \cup V_i$. Compute $I = \{1, \ldots, n\} \setminus V$. If $|I| \leq d$ output $(1, I)$, else output $(0, I)$.

The correctness of the scheme is shown next.

Theorem 5.5.4. *Consider a valid signature σ generated by* MTSS-SIGN *for a message m and key pair (SK, PK), and let m' be a possibly modified message with $|\mathit{diff}(m, m')| \leq d$. Then,* MTSS-VERIFY$(m', \sigma, PK) = (1, \mathit{diff}(m, m'))$.

Proof. Since σ is valid, MTSS-VERIFY(m', σ, PK) does not stop at step 1. If $m = m'$, then $h(m) = h(m')$ and the algorithm will stop in step 2, with $(1, \text{diff}(m, m') = \emptyset)$. It remains to check the case it stops in step 4. The d-CFF property guarantees that if a block has not been modified, it is contained in at least one valid concatenation with $T[i] = T'[i]$, since there is a row i that avoids all modified blocks. Therefore, this block is contained in V_i. For each row i that contains a modified block, we have $T[i] \neq T'[i]$, so modified blocks are not contained in any V_i. Therefore I consists of precisely the modified blocks. Thus, $|I| \leq d$, and the algorithm outputs $(1, I = \text{diff}(m, m'))$. □

The next theorem shows that when step 4 outputs $(0, I)$, Scheme 1 may give more information than required in Definition 5.4.4, as it may identify some unmodified blocks, even if not all.

Theorem 5.5.5. *Consider a valid signature σ generated by* MTSS-SIGN *for a message m and key pair (SK, PK), and let m' be a modified message with $|\mathit{diff}(m, m')| > d$. Then,* MTSS-VERIFY$(m', \sigma, PK) = (0, I)$*, and for any $i \in \{1, \ldots, n\} \setminus I$, block $m[i]$ is guaranteed to be unmodified.*

Proof. This case will lead MTSS-VERIFY to step 4, and since $|\text{diff}(m, m')| > d$, the output will be $(0, I)$. Any block in $\{1, \ldots, n\} \setminus I$ is guaranteed to be part of matching row i, and must be unmodified, even though not every unmodified block will necessarily be placed in $\{1, \ldots, n\} \setminus I$. □

Scheme 1 has been proposed in [12]. One of our main contributions here is to add correcting capability to d-MTSS. We require the size of each block to be upper bounded by a value s that is small enough that guessing each of the (up to) d modified blocks is "computationally feasible". Basically, by brute force we compute up to $O(d2^s + n)$ hashes to accomplish the modification correction (see the algorithm under Scheme 2). We use the indices of the modified blocks in I and do an exhaustive search to recover their original values.

Scheme 2: A d-MTSS with correction capability

The scheme requires an underlying CDSS Σ, a public hash function h, and a d-CFF(t, n) matrix $\mathcal{M}$. It further requires that the message is divided into n blocks of size at most s. The scheme has the three algorithms from Scheme 1, and additionally the algorithm below:

- MTSS-VERIFY&CORRECT(m, σ, PK): receives as input a message $m = (m[1], \ldots, m[n])$, a signature $\sigma = (\sigma', T)$ where $T = (T[1], T[2], \ldots, T[t], h^*)$, and a public key PK, and proceeds as follows.
  1. Compute *result* = MTSS-VERIFY(m, σ, PK). If *result* $= (0, X)$, then stop and output $(0, X)$, otherwise *result* $= (1, I)$. If $|I| = 0$ go to step 6, otherwise run steps $2 - 5$ for each $k \in I$.
  2. Identify a row i in $\mathcal{M}$ such that $\mathcal{M}_{i,k} = 1$ and $\mathcal{M}_{i,j} = 0$, for all $j \in I \setminus \{k\}$.
  3. Compute $h_j = h(m[j])$ for all j such that $\mathcal{M}_{i,j} = 1, j \neq k$, i from step 2. Set *corrected[k] = false*.
  4. For every possible binary string b of size $\leq s$, proceed as follows:
     - Compute $h_k = h(b)$.
     - For i obtained in step 2 and $1 \leq j \leq n$, compute c_i as the concatenation of every h_j such that $\mathcal{M}_{i,j} = 1$ and set $T'[i] = h(c_i)$.
     - If $T'[i] = T[i]$ and *corrected[k] = false*, set *corrected[k] = true* and correct the block $m[k] = b$.
     - Else, if $T'[i] = T[i]$ and *corrected[k] = true*, stop and output $(1, I, \lambda)$.
  5. Return to step 2 with the next $k \in I$.
  6. Output $(1, I, m)$.

We note that the flag *corrected*$[k]$ is used to identify a possible collision of two different bit strings b giving the same hash value $h(m[k])$. Since the correct block cannot be determined, we exit with λ indicating failure. The next proposition has details on the correctness of this algorithm.

Proposition 5.5.6. *Let (m, σ) be a valid pair of message and signature produced by* MTSS-SIGN*(m, SK), using a hash function h and with $m = (m[1], \ldots, m[n])$. Let m' be a message and let $I = diff(m, m')$ with $|I| \leq d$. If for every $k \in I$, $h(m[k])$ has no other preimage of size up to s, then*

MTSS-VERIFY&CORRECT*$(m', \sigma, PK) = (1, I, m)$.*

Proof. As seen in Theorem 5.5.4, the set I contains precisely the indices of the modified blocks, and Theorem 5.5.2 guarantees that such a row i in step 2 of the algorithm exists. Finally, if for every $k \in I$ the hash $h(m[k])$ has no second preimage, then step 4 of the algorithm computes a unique replacement for each modified block, and the algorithm outputs the corrected message in step 6. □

Now we prove that when selecting a good hash function, we can always guarantee the correction of any up to d modified blocks.

Theorem 5.5.7. *Consider Scheme 2 restricted to messages with blocks of size at most s, and such that h is a hash function where no two inputs of size up to s have the same hash value. Then,* MTSS-VERIFY&CORRECT*(m', σ, SK) can always correct a message with up to d modified blocks.*

Proof. Easily obtained by Proposition 5.5.6 since no matter what is the value of the block, no other block of size up to s can have the same image under h. □

Next, we show that the assumption of existence of such hash function is realistic, and in fact very easy to find.

Proposition 5.5.8. *Consider a family of hash functions $h : X \to Y$ where $|Y| = 2^l$ and a subset of the inputs $S \subseteq X$ where $|S| \leq 2^{s+1}$. The probability that there is collision in S, i.e. the probability that there exists x,z in S with $h(x) = h(z)$, is approximately $\epsilon = 1 - e^{-2^{2s-l+1}}$.*

Proof. This comes from the fact that the probability of finding at least one collision after Q hash calculations is approximately $1 - e^{-\frac{Q^2}{2^{l+1}}}$, and in our application $Q = |S| = 2^{s+1}$. □

In practice, we will set $2s \ll l$. This ensures via Proposition 5.5.8 that h is very likely to have the desired property of being injective on S, the set of binary strings with size at most s. In the unlikely event that h fails this property, we try another hash function until we succeed. The expected number of attempts will be very close to 1. For example, if we consider SHA-256 as the hash function, $|Y| = 2^{256}$, and for $s = 20$, the probability of a collision within the set of size at most 20 is $\epsilon = 1 - e^{-2^{40-256+1}} \approx 3.70 \times 10^{-68}$. Indeed, we experimentally verified that SHA-256 has no collisions for $s = 20$.

Theorem 5.5.9. *Consider Scheme 2 with tolerance level d, and let m be a message split into n blocks, each block of size at most s. Let $\sigma =$* MTSS-SIGN*(m, SK) and m' be a message with $I = diff(m, m')$ with $|I| \leq d$. Then,* MTSS-VERIFY&COR- *rect(m', σ, PK) returns $(1, I, m)$ and uses $O(n + d^2 \log n + d2^s)$ hash computations.*

Proof. By choosing a suitable hash function, Theorem 5.5.7 guarantees that MTSS-Verify&Correct returns $(1, I, m)$. The algorithm starts with the location of the modified blocks. This step uses a d-CFF for which we can use a construction with $t = O(d^2 \log n)$ rows [19], and therefore a total of $n + t$ hash calculations are required to locate these modifications. After locating up to d modified blocks, we need to perform the following computations for each one of them. We compute every possible block of size up to s and their corresponding hash values (total of $2^{s+1} - 1$), and according to the row of the CFF matrix, we compute a few extra hashes (in total not more than n, if storing h_i instead of recomputing among different runs of of line 3). This gives a total of $O(d2^s + n)$ hash computations for the correction step. □

5.5.3 Comparing Scheme 2 with a scheme using ECC

Let an $(l, n, D)_q$-ECC $\mathcal{C}$ be an error correcting code with minimum distance D, codewords of length l that encode messages of size n over an alphabet of size q. The alphabet must be so that it can distinguish each possible block considered in Scheme 2, so $q \geq 2^s$. For simplification, assume $c_m = (m, b_m)$ where b_m is a tuple of $l - n$ letters ("check bits"). This code can correct $d = \lfloor \frac{D-1}{2} \rfloor$ errors (modifications in the message). Next, we describe signature and verification with ECC. Using the same inputs as MTSS-Sign, algorithm ECC-Sign(m, SK) do the following steps: 1) Compute $\sigma' = \Sigma.\text{Sign}(m, SK)$; 2) Compute $c_m = (m, b_m)$ according to ECC $\mathcal{C}$ and return $\sigma = (b_m, \sigma')$. Then, algorithm ECC-Verify&Correct$(m, \sigma = (b, \sigma'), PK)$ do the following steps: 1) Decode $c = (m, b)$ to m' using $\mathcal{C}$; 2) If $\Sigma.\text{Verify}(m', \sigma, PK) = 1$ then return $(1, \text{diff}(m, m'), m')$, else return 0.

Since $\mathcal{C}$ can correct up to d errors, if the signature σ is valid (σ' is authentic for m' and $|\text{diff}(m, m')| \leq d$), then ECC-Verify&Correct will behave in the same way as MTSS-Verify&Correct and will return $(1, \text{diff}(m, m'), m')$. However, when ECC-Verify&Correct returns 0, it does not distinguish between the two failing cases obtained by MTSS-Verify&Correct, namely: **Case 1)** output $(0, -)$, which means the CDSS signature σ' is not authentic; **Case 2)** output $(0, I)$, which means σ' is authentic and message m differs from the signed m' in more than d blocks, and also if $|I| < n$, then $|\{1, \ldots, n\} \setminus I| > 0$ and we are sure of at least one unmodified block. Therefore, while the scheme based on ECC is an MTSS scheme according to Definition 5.4.4, it provides less information than Scheme 2.

A comparison of Scheme 2 with the ECC scheme shows they have similar compression ratios. Indeed, we first note that given an $(l, n, D)_q$-ECC, it is possible to construct a d-CFF$(l \cdot q, q^n)$, with $d = \lfloor (l-1)/(l-D) \rfloor$ [16]. Then, we consider some families of error correcting codes, and for each family we restrict Scheme 2 to only use CFFs constructed using codes from this family. The comparisons are done by computing the compression ratio of each approach, where $\rho = \frac{n}{t}$ for CFFs and $\rho = \frac{k}{l-k}$ for the ECC scheme. For the Hamming codes, we consider $r \geq 2$, binary Hamming codes with length $l = 2^r - 1$ and $k = 2^r - r - 1$, and the corresponding 1-CFF$(2(2^r - 1), 2^{2^r - r - 1})$. We compare it with an ECC scheme, using codes with $k = 2^{2^r - r - 1} - 2^r + r$, $l = 2^{2^r - r - 1} - 1$, $d = \lfloor (D-1)/2 \rfloor = 1$. For the Reed-Solomon case, we consider codes over an alphabet of size q, $D = l - k + 1$, $k = q - D + 1$,

and the corresponding d-CFF$(q^2, q^{\frac{q-1}{d}+1})$ with $d = \lfloor (q-1)/(q-D) \rfloor$. We compare it with the ECC scheme, using codes with $k = q^{\frac{q-1}{d}+1}$, $l = k+D-1 = k+2d+1 = q^{\frac{q-1}{d}+1}+2d+1$. Finally, using the code constructed by Porat and Rothschild [19], which we call PR-code, we get a d-CFF(t, n) with $t = \Theta(d^2 \log n)$, and compression ratio $\Theta(\frac{n}{d^2 \log n})$. The lower bound found in [10] guarantees that for any CFF, $t = \Omega(\frac{d^2 \log n}{\log d})$. So even though we did not produce an exact comparison of the ratio using PR-code in the ECC scheme, due to the lower bound on t, we know the ratio can be at most $O(\frac{(\log d)n}{(d^2 \log n)})$.

In the next table, we summarize the compression ratios obtained for 3 families of codes.

code:	Hamming, $n = 2^{2^r-1}$	Reed-Solomon, $n = q^{\frac{(q-1)}{d}+1}$	PR-code[19]
Scheme 2	$\approx 2^{2^r-2r-2}$	$q^{\frac{(q-1)}{d}-1}$	$\Theta(\frac{n}{d^2 \log n})$
ECC scheme	$\approx 2^{2^r-2r-1}$	$\frac{q^{\frac{(q-1)}{d}+1}}{2d+1}$	$O(\frac{(\log d)n}{d^2 \log n})$

In conclusion, both schemes serve the same purpose with similar compression ratios, but Scheme 2 has several advantages. First, Scheme 2 gives more information in the failing case where ECC returns 0, as discussed above. Second, Scheme 2 provides a non-correcting version of the verification algorithm (MTSS-Verify) which in the case of unmodified messages is basically as time efficient as Σ.Verify (see step 1 of MTSS-Verify); in this case, the ECC scheme still needs to run a decoding algorithm, with complexity influenced by $q \geq 2^s$, where s is the largest size of a block. Finally, Scheme 2 is part of a family of similar schemes presented here (Schemes 1-3), which can be more easily compared.

5.6 Security

In this section, we present the security proof of d-MTSS for Schemes 1 and 2. In order to do this, we need to check that the security of the hash function h and the unforgeability of the underlying CDSS can together ensure unforgeability of d-MTSS. Note that although Scheme 1 appeared in [12], no security proof has been given in that paper. For the next proof we assume a *collision-resistant* hash function, i.e. a hash function in which a collision cannot be efficiently found [22].

When we consider d-MTSS using d-CFFs, a valid (m, σ) as in Definition 5.4.5 implies that there exists m' such that σ is an output of MTSS-Sign(m', SK) and $|\text{diff}(m, m')| \leq d$. In the next theorem we suppose there is a valid (m, σ) as a forgery to our scheme. We consider two types of forgery: a *strong forgery* consists of (m, σ) such that MTSS-Verify$(m, \sigma, PK) = (1, I), |I| \leq d$; a *weak forgery* consists of (m, σ) such that MTSS-Verify$(m, \sigma, PK) = (1, \emptyset)$.

Theorem 5.6.1. *Let X be a d-MTSS as described in Scheme 1 based on an existentially unforgeable CDSS Σ and on a collision resistant hash function h. Then, X is existentially unforgeable.*

Proof. (By contradiction) Suppose X is not existentially unforgeable. Then, there is an adversary $\mathcal{A}$ that, after q adaptive queries to a signing oracle $\mathcal{O}$, obtains pairs $(m_1, \sigma_1), \ldots, (m_q, \sigma_q)$, with $\sigma_i = (\sigma_i', T_i), 1 \leq i \leq q$, and with non-negligible probability, outputs a valid pair (m, σ), with $\sigma = (\sigma', T)$, $T = (T[1], T[2], \ldots, T[t], h^*)$, and $|\text{diff}(m, m_i)| > d$, for all $1 \leq i \leq q$.

We show that if such $\mathcal{A}$ exists, then we can build a probabilistic polynomial time algorithm $\mathcal{A}'$ which has the following input and output:

Input: an existentially unforgeable CDSS Σ and a collision-resistant hash function h, both with security parameter ℓ.

Output: either a existential forgery (T, σ') of Σ or a collision pair for h.

Next we describe the steps of such $\mathcal{A}'$.

1. Simulate the probabilistic polynomial time adversary $\mathcal{A}$ forging an MTSS based on Σ and h using queries mentioned above. With non-negligible probability, this will produce a forgery $(m, \sigma = (\sigma', T))$ in X, as described above. Otherwise, return "FAIL".

2. If $T \neq T_i$, for all $1 \leq i \leq q$, $\mathcal{A}'$ presents (T, σ') as a forgery in Σ, for σ' the corresponding signature of T.

3. If $T = T_i$, for some $1 \leq i \leq q$, first calculate I by computing MTSS-VERIFY(m, σ, PK). Then, we have:

   - In the case of weak forgery, we must have $I = \emptyset$, and so the final element of T is $h(m)$ and the final element of T_i is $h(m_i)$, and since $m \neq m_i$, $\mathcal{A}'$ presents (m, m_i) as a collision pair for h.
   - Otherwise, we must have a strong forgery with $1 \leq |I| \leq d$. So, there exists m' such that MTSS-VERIFY$(m', \sigma, PK) = (1, \emptyset)$ and $|\text{diff}(m, m')| \leq d$. Since $|\text{diff}(m, m_i)| > d$, there must be a block $m[k], k \in \{1, \ldots, n\} \setminus I$, that is considered valid in m but $m[k] \neq m_i[k]$. Let p be any row of the CFF matrix $\mathcal{M}$ with $\mathcal{M}_{p,k} = 1$. Because $T = T_i$, this implies $T[p] = h(c) = h(c') = T_i[p]$, for $c = h(m[j_1])||h(m[j_2])|| \ldots ||h(m[j_s])$ and $c' = h(m_i[j_1])||h(m_i[j_2])|| \ldots ||h(m_i[j_s])$, where $k \in \{j_1, j_2, \ldots, j_s\}$. If $h(m[k]) = h(m_i[k])$, then $\mathcal{A}'$ presents $(m[k], m_i[k])$ as a collision pair for h. Otherwise, $\mathcal{A}'$ presents (c, c') as a collision pair for h.

The probability $p(\ell)$ that $\mathcal{A}'$ succeeds is the same as the probability that $\mathcal{A}$ succeeds, where ℓ is the security parameter. Whenever $\mathcal{A}'$ succeeds, we have either an existential forgery of Σ or a collision for h, corresponding to steps 2 and 3, respectively. For each ℓ, one of these steps is at least as likely to occur as the other, so it has probability at least $1/2$. So, for any security parameter ℓ, use one of two algorithms (an existential forger for Σ, or a collision finder for h) that runs in probabilistic polynomial time and succeeds with probability at least $p(\ell)/2$. In other words, we can technically design two adversaries $\mathcal{A}_1'$ and $\mathcal{A}_2'$ based on $\mathcal{A}'$. $\mathcal{A}_1'$ forges Σ by proceeding as $\mathcal{A}'$ but returning "FAIL" if it falls in the case of step

3. Similarly, $\mathcal{A}'_2$ finds a collision for h or returns "FAIL" if it falls in the case of step 2. Then, either $\mathcal{A}'_1$ or $\mathcal{A}'_2$ will succeed with probability at least $p(\ell)/2$ for infinitely many ℓ. This contradicts either the unforgeability of Σ or the collision resistance of h. □

5.7 Using MTSS for redactable signatures

Now we turn our attention to using MTSS in general and using similar algorithms to our proposed d-MTSS for redactable signature schemes. Redactable and sanitizable signature schemes allow for parts of a message to be removed or modified while still having a valid signature without the intervention of the signer. Sanitizable signatures usually requires the existence of a semi-trusted party called the *sanitizer* who is entrusted to do the modifications and recomputation of the signature, in some schemes done in a transparent way (see [18]). Our proposed scheme does not deal with sanitizable, but rather with redactable signatures.

Redactable signatures have been proposed in several variants also under the names of content extraction signatures [21] and homomorphic signatures [13]. Redactable signature schemes (RSS) "allow removing parts of a signed message by any party without invalidating the respective signature" [6] and without interaction with the signer. In [21], the authors mention content extraction for privacy protection or bandwidth savings. Suppose Bob is the owner of a document signed by Alice and does not want to send the whole document to a third verifying party Cathy but only some extracted parts of the document; however, Cathy still needs to verify that Alice is the signer of the original document. Alice is agreeable with future redactions when she signed the document. An example given in [21] is that Bob has his university transcripts signed by the issuing university (Alice) and wants to submit the transcripts to a prospect employer (Cathy) without revealing some of his private information such as his date of birth. Cathy must be able to verify that the signature came from the university in spite of the redaction. In addition to the privacy application, content extraction can be used in a similar way when only part of a large document needs to be passed by Bob to Cathy for the purpose of reducing the communication bandwidth [21].

The notion of redactable signatures we consider next is in line with our general definition of MTSS, but differs in that we add another algorithm called MTSS-Redact and we require total privacy of the redacted parts. We give next a d-MTSS version that is redactable based on ideas similar to Scheme 1 but modifying it to guarantee total privacy of redacted blocks. As mentioned in [18], the scheme proposed in [12] which was presented as Scheme 1 does not meet standard privacy requirements and in particular leaks the original message's hash value. The scheme we propose below addresses this issue by individually signing each of the hashes of the groups of blocks, and at the time of redaction of blocks, also redacting from the signature tuple any hashes involving concatenations of the modified blocks. To avoid more complex forms of forgery, that could take advantage of combining individual signatures of concatenations coming from different signed messages, we add the same random string to the various hashes of concatenations to link the individual parts that are signed at the same time; in addition, to avoid reordering of blocks within the same message via reordering the groups, we add a group counter to these hashes before signing.

In the description below, a redaction is represented by the symbol ■.

Scheme 3: A redactable d-MTSS with privacy protection

The scheme requires an underlying CDSS Σ, a public hash function h, a d-CFF(t, n) matrix $\mathcal{M}$ and a random number generator RAND. The algorithms are given next:

- MTSS-KEYGENERATION(ℓ): generates a key pair (SK, PK) using algorithm Σ.KEYGENERATION(ℓ).
- MTSS-SIGN(m, SK): Takes as input SK and a message $m = (m[1], \ldots, m[n])$, and proceeds as follows.
  1. Calculate $h_j = h(m[j]), 1 \leq j \leq n$. Compute a random string $r =$RAND().
  2. For each $1 \leq i \leq t$, compute c_i as the concatenation of all h_j such that $\mathcal{M}_{i,j} = 1, 1 \leq j \leq n$, and set $T[i] = h(c_i)||r||id(i, t+1)$, where $id(i, t+1)$ encodes the numbers i and $t+1$.
  3. Compute $h^* = h(m)$, set $T[t+1] = h^*||r||id(t+1, t+1)$ and set $T = (T[1], \ldots, T[t+1])$.
  4. Calculate $\sigma'[i] = \Sigma$.SIGN$(T[i], SK)$, for each $1 \leq i \leq t+1$ and set $\sigma' = (\sigma'[1], \sigma'[2], \ldots, \sigma'[t+1])$. Output signature $\sigma = (\sigma', r)$.
- MTSS-VERIFY(m, σ, PK): takes as input a message $m = (m[1], \ldots, m[n])$, a signature $\sigma = (\sigma', r)$, and a public key PK, and proceed as follows.
  1. Check if Σ.VERIFY$(h(m)||r||id(t+1, t+1), \sigma'[t+1], PK) = 1$. Stop and output $(1, \emptyset)$ if that is the case; continue otherwise.
  2. Use $\mathcal{M}$, m, r and do the same process as in steps 1-3 of MTSS-SIGN to produce tuple $T' = (T'[1], \ldots, T'[t])$.
  3. For each $1 \leq i \leq t$ such that Σ.VERIFY$(T'[i], \sigma'[i], PK) = 1$, compute the set of indices of intact blocks $V_i = \{j : \mathcal{M}_{i,j} = 1\}$ and do $V = V \cup V_i$. Compute $I = \{1, \ldots, n\} \setminus V$. Output $(1, I)$ if $|I| \leq d$; output 0 otherwise.
- MTSS-REDACT(m, σ, R): takes as input a message $m = (m[1], \ldots, m[n])$, a signature $\sigma = (\sigma', r)$, and a set $R \subseteq \{1, \ldots, n\}$ of blocks to be redacted, with $|R| \leq d$, and proceeds as follows.
  1. If $R = \emptyset$, then stop and output (m, σ).
  2. Create copies of the message and signature: $\overline{m} = m$, $\overline{\sigma} = \sigma$, so that $\overline{\sigma} = (\overline{\sigma}', r)$.
  3. Set $\overline{\sigma}'[t+1] =$ ■.
  4. For each $j \in R$: set $\overline{m}[j] =$ ■ and for each index i such that $\mathcal{M}_{i,j} = 1$ set $\overline{\sigma}'[i] =$ ■
  5. Output $(\overline{m}, \overline{\sigma} = (\overline{\sigma}', r))$.

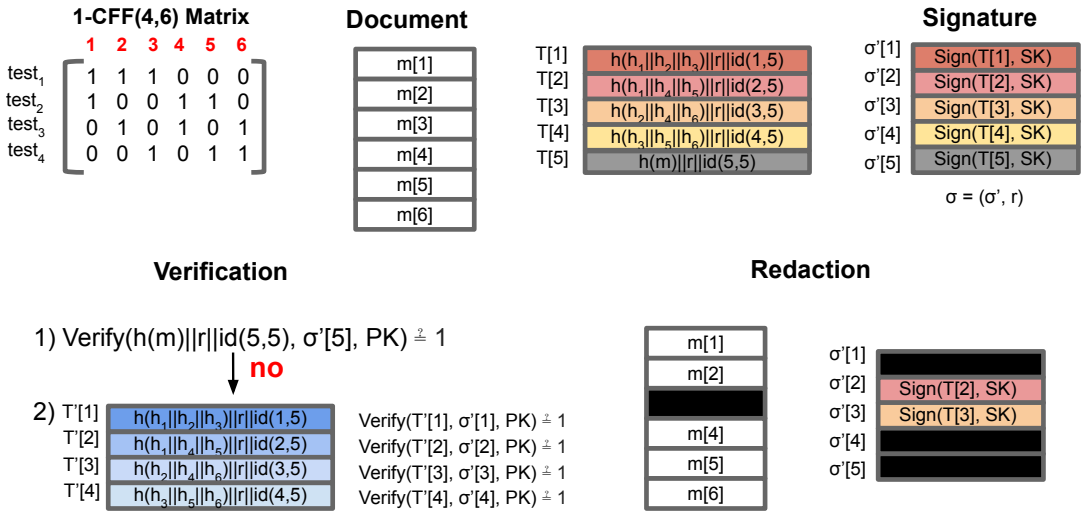


Figure 5.2: Example of Scheme 3 using a 1-CFF$(4, 6)$.

Figure 5.2 depicts the signature generation, verification and redaction based on a 1-CFF$(4, 6)$ and a message divided into $n = 6$ blocks. The redaction uses $R = \{3\}$ and removes block $m[3]$ and all corresponding signatures involving $m[3]$.

The correctness of the first three algorithms follows similar reasoning using the CFF properties as argued in Theorem 5.5.4, taking into account the different approach used here where t CDSS signatures and t CDSS verifications are required. The redaction offers total privacy in the sense of information theory, as no information related to the redacted blocks is kept in the signature, which are erased in steps 3 and 4 of MTSS-REDACT. The redacted blocks and redacted parts of the signature will only affect the verification of the parts of the signatures involving redacted blocks; all other block indices will be indicated as unmodified in the output of line 3 of MTSS-VERIFY, as long as no more than d blocks have been redacted.

We now discuss how the added random string and $id(i, t + 1)$ helps preventing some types of forgery that could happen in Scheme 3 due to individually signing each $T[i]$. The use of different random strings for each signature prevents an adversary from combining blocks and pieces of signature from previously signed messages in order to forge a new pair (m, σ). As an example of such an attack, consider an identity matrix $\mathcal{M}$ of dimension 3 as a 2-CFF$(3, 3)$. Consider an adversary queried two messages $m_1 = ABC$ and $m_2 = DEF$ and received the corresponding signatures $\sigma_1 = (\sigma_A, \sigma_B, \sigma_C, \sigma_{m_1})$ and $\sigma_2 = (\sigma_D, \sigma_E, \sigma_F, \sigma_{m_2})$, where $\sigma_X = \Sigma.Sign(X, SK)$ and suppose no random string or $id(i, t + 1)$ is used. The adversary can present as a forgery the pair $(m = AT_1F, \sigma = (\sigma_A, T_2, \sigma_F, T_3))$, with T_i meaning any "trash" content, $1 \leq i \leq 3$. The pair would produce output as if it was valid, implying that it was an authorized modification on block $m[2]$ of the presented m, which was never signed before.

The use of $id(i, t+1)$ prevents an adversary from removing, adding or reordering blocks

without being noticed. As an example of a reordering attack, assume the 1-CFF above and that the adversary queried $m_1 = ABC$ and received $\sigma_1 = (\sigma_A, \sigma_B, \sigma_C, \sigma_{m_1})$. The adversary can present as a forgery the pair $(m = CAT_1, \sigma = (\sigma_C, \sigma_A, T_2, T_3))$, for T_i a "trash" content, $1 \leq i \leq 3$. The pair is again valid and it looks like an authorized modification on block $m[3]$ of m, but such a message was never signed. Therefore, $id(i, t+1)$ would prevent such reordering due to the use of a counter i, and also prevent removing and adding new blocks in this example due to the use of $t + 1$.

In the next theorem, we establish the unforgeability of this scheme.

Theorem 5.7.1. *Let X be a redactable d-MTSS given in Scheme 3 based on an existentially unforgeable CDSS Σ, on a collision-resistant hash function h, and on a random Oracle. Then, X is existentially unforgeable.*

Sketch of the proof. Similar reasoning as in Theorem 5.6.1 can be used to argue that the unforgeability of Σ and the collision resistance of the hash function are sufficient to protect against forgery at the level of individual parts of the tuple signature. However, we need to rule out more complex forgery attempts that could try to combine individual signatures in the signature tuple in different ways as well as different blocks from different messages. The use of a common random string in the creation of each $T[i]$ in lines 2 and 3 of algorithm MLSS-Sign prevents an adversary from trying to combine message blocks and signature parts of $(m_1, \sigma_1), \ldots, (m_q, \sigma_q)$ coming from different calls to the random Oracle to create a new valid signature since, by definition, two different calls to the oracle will, with high probability, not produce the same random string. Furthermore, the concatenation of the encoding of the test index and the number of tuple elements, $id(i, t+1)$, within each tuple position $T[i]$ in line 2 and 3, makes it highly unlikely that blocks are added, removed or reordered, since this would amount to finding a collision in the hash function. Therefore, by making it very unlikely reordering of blocks within a message, and combinations of blocks from different messages, we fall into the case of forgeries of similar types as dealt with in the proof of Theorem 5.6.1. □

Remark. The random Oracle hypothesis can be weakened, as it is sufficient to draw numbers from a sequence that does not repeat numbers, or repeats with negligible probability.

5.8 Implementation issues

When implementing Schemes 1-3, there are a few details that need to be considered regarding efficiency, security, and even flexibility on the inputs or outputs of the algorithms. For example, we consider a message divided into blocks that we represented as a sequence, but blocks may be represented using different data structures depending on the application and the type of data. Pohls [18] discusses sequence, set and tree data structures for blocks, which could be employed.

When signing a message using MTSS-Sign as described in Schemes 1-3, we could consider not hashing every single block before concatenation, especially if the sizes of the blocks are small enough that a hash function will end up increasing the size of the

data being concatenated. This approach needs to be carefully considered, since a simple concatenation does not insert a delimitation on where the individual blocks start or end, and therefore blocks with a shared suffix or prefix may lead to wrong identifications of valid/invalid. To be safe, we can hash each block before concatenating them as presented in our schemes, or ensure the concatenations use delimiters to separate blocks, or require blocks of fixed size.

The correction algorithm MTSS-Verify&Correct computes the set of indices of modified blocks I and tries to correct these blocks to their original value. If there is at least one position where two different blocks match the hash of the original one (a second preimage on the hash function), the algorithm aborts the correction process and outputs an empty string λ to represent correction error. As seen in Proposition 5.5.8 and the discussion that follows, it is very unlikely for such event to occur, and we could always choose another hash function with no collisions for certain block sizes. However, if some specific application is not flexible in the choice of hash functions and such an event happens, an interesting approach can be to correct as many modified blocks as possible, and return some information regarding the ones with collisions (such as a fail message or even a list of possible values for those specific blocks). This approach allows for partial correction of modified data, that may be interesting for some applications. Moreover, if we already know that the chosen hash function has no collisions for blocks of size up to s, we do not need to run step 4 for every possible block of size $\leq s$, and we could stop this loop as soon as we find the first match, improving the efficiency of the method.

Regarding the multiple hash computations that happen in the correction algorithm, one could consider some improvements. Note that we already compute the hashes of all unmodified blocks when we do the call of MTSS-Verify to obtain the set I, so these hashes of unmodified blocks could be reused in step 2. Moreover, we repeat for every modified block in I the same process of computing all blocks of size $\leq s$ and their corresponding hash values. For the cases where we have a big set of modified blocks (big d), one may consider to pre-compute all these values, in case the application can handle the storage that this will require.

5.9 Parameter relations

For MTSS with modification location only (Scheme 1), a message m is split into n blocks of no specific size, and a location capability parameter d needs to be decided. These parameters are used to construct a d-CFF(t, n) with number of rows $t = \Theta((d+1)^2 \log n)$ if using [19], $t = d\sqrt{n}$ if using [17], and $t = q^2$ when using [9], for any prime power q and positive integer k that satisfy $q \geq dk + 1$. The signature size is directly impacted by d and n, since there are $t + 1$ hashes as part of the signature. Therefore, while smaller sizes of blocks and larger d give a more precise location of modifications, they also increase the size of the signature.

Now consider the exact same message m, but for the case of an MTSS with correction capabilities (Scheme 2). This scheme requires that the blocks have a small size of up to s

bits, which implies that the very same message m now has many more blocks $n' \gg n$. A larger number of blocks directly increases the size of the signature. But now, the d that was enough to locate the modifications before may not be enough anymore, since modifications that were in one big block before now may be spread over several small blocks. When locating the modifications, the algorithm aborts the process if the number of modified blocks is larger than the expected location capability d, which may cause the scheme to fail to correct more often if this value is not increased as n' increases.

The size of the signature σ in Schemes 1 and 2 is $|\sigma| = |h(.)| \times (t+1) + |\sigma'|$, and in Scheme 3 is $|\sigma| = |\sigma'| \times (t+1) + |r|$, for σ' a classical digital signature, r a random bit string, $h(.)$ a traditional hash function, and t the number of rows of a d-CFF(t, n). The input consists of a message of size $|m| = N$ in bits, divided into n blocks, each of size at most s, so $N \approx n \times s$. In Scheme 2, given N, we need to wisely choose s so it is small enough to allow corrections of blocks, while guaranteeing n is small enough to have a reasonable $|\sigma|$. We cannot expect our signature to be as small as the ones from classical digital signature schemes, since we require extra information in order to be able to locate and correct portions of the data. In summary what we want is $n \times s \gg |\sigma|$. The next examples show that even for small $s = 8$, we still have a reasonably small signature.

Example 1: $N = 1$ GB $= 2^{30}$ bytes $= 2^{30}2^3 = 2^{33}$ bits, $h =$ SHA-256, $s = \log|h| = \log 2^8 = 8$ bits, $d = 4$, and we use RSA with a 2048-bit modulus. Then we have $n = N/s = 2^{33}/2^3 = 2^{30}$ blocks, with 8 bits each. Consider the d-CFF(q^2, q^{k+1}) from [9], with $k = 6, q = 25, t = 25^2, n = 25^7$. Now, since $|\sigma| = |h(.)| \times (t+1) + |\sigma'|$ in Schemes 1 and 2, we have $|\sigma| = 2^8 \times (25^2 + 1) + 2048 = 162304$ bits, which is 20288 bytes, or ~ 20 KB.

Example 2: For the same message and parameters as in Example 1, now we use a random bit string r of size 128 bits and create a signature as in Scheme 3. Since $|\sigma| = |\sigma'| \times (t+1) + |r|$, we have $|\sigma| = 2048 \times (25^2 + 1) + 128 = 1282176$ bits, which is 160272 bytes, or ~ 160 KB.

For Scheme 3, small blocks like in Example 2 are not required, so we can get a much smaller signature. The choice of s and n in this case depends on the application, and on whether we wish to correct non-redactable blocks. In the case where we want to correct non-redactable blocks, note that both non-redactable and redactable blocks are all contributions to the same maximum modification tolerance d.

5.10 Conclusion

We introduce modification tolerant signature schemes (MTSS) as a general framework to allow localization and correction of modifications in digitally signed data. We propose a scheme based on d-CFFs that allows correction of modifications, and a variation without correction that gives redactable signatures. The presented schemes are provably secure and present a digital signature of size close to the best known lower bound for d-CFFs.

Interesting future research includes an implementation of the schemes proposed here, with practical analysis of parameter selection for specific applications. In addition, new

solutions for MTSS beyond d-CFFs can also be of interest. The d-CFF treats every block the same and allows for any combination of up to d blocks to be modified. Specific applications can have different requirements about what combinations of blocks can be modified by specifying a different authorized modification structure. Moreover, other hypotheses about more likely distribution of modifications throughout the document can be used for efficiency; for example, modified blocks may be concentrated close to each other. One idea in this direction is to use blocks with sub-blocks for increasing granularity of modification location and to aid correction. This would involve matrices with a smaller d for the bigger blocks and a larger d for sub-blocks of a block, picking these parameters with the aim of decreasing t and consequently signature size. It is out of the scope of this paper to go into detailed studies of other types of modification location matrices, which we leave for future work.

Bibliography

[1] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable Signatures. In *European Symposium on Research in Computer Security (ESORICS 2005)*, Lecture Notes in Computer Science, 3679, pages 159–177.

[2] P. S. L. M. Barreto, H. Y. Kim and V. Rijmen. Toward secure public-key blockwise fragile authentication watermarking. In *IEE Proceedings - Vision, Image and Signal Processing*, 149(2):57–62, 2002.

[3] R. G. Biyashev and S. E. Nyssanbayeva. Algorithm for creating a digital signature with error detection and correction. *Cybernetics and Systems Analysis*, 48(4):489–497, 2012.

[4] A. De Bonis and G. Di Crescenzo. Combinatorial group testing for corruption localizing hashing. In *Computing and Combinatorics*, pages 579–591, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[5] A. De Bonis and G. Di Crescenzo. A group testing approach to improved corruption localizing hashing. Cryptology ePrint Archive, Report 2011/562, 2011. `https://eprint.iacr.org/2011/562`.

[6] D. Derler, H. C. Pöhls, K. Samelin, and D. Slamanig. A general framework for redactable signatures and new constructions. In *International Conference on Information Security and Cryptology (ICISC 2015)*, Lecture Notes in Computer Science, 9558, pages 3–19. Springer, 2015.

[7] G. Di Crescenzo, R. Ge, and G. R. Arce. Design and analysis of dbmac, an error localizing message authentication code. In *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, volume 4, pages 2224–2228, 2004.

[8] P. D'Arco and D. Stinson. Fault tolerant and distributed broadcast encryption. In *Proceedings of the 2003 RSA Conference on The Cryptographers' Track*, pages 263–280. Springer-Verlag, 2003.

[9] P. Erdös, P. Frankl and Z. Füredi, Families of finite sets in which no set is covered by the union of r others, *Israel J. Math.*, **51** (1985), 79–89.

[10] Z. Füredi, On r-Cover-free Families, *Journal of Combinatorial Theory*, **73** (1996), 172–173.

[11] M. T. Goodrich, M. J. Atallah, and R. Tamassia. Indexing information for data forensics. In *ACNS*, 2005.

[12] T. B. Idalino, L. Moura, R. F. Custódio, and D. Panario. Locating modifications in signed data for partial data integrity. *Information Processing Letters*, 115(10):731 – 737, 2015.

[13] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. In *Topics in Cryptology - Cryptographers Track - RSA 2002.*, pages 244–262. Springer, 2002.

[14] J. Katz, Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2007.

[15] A. J. Menezes, S. A. Vanstone, and P. C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[16] H. Niederreiter, H. Wang, and C. Xing. Function Fields Over Finite Fields and Their Applications to Cryptography. In: Garcia A., Stichtenoth H. (eds) *Topics in Geometry, Coding Theory and Cryptography.* Algebra and Applications, 2006, 59–104.

[17] J. Pastuszak, J. Pieprzyk, and J. Seberry. Codes identifying bad signature in batches. In *INDOCRYPT*, volume 1977 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 2000.

[18] H. C. Pöhls. Increasing the Legal Probative Value of Cryptographically Private Malleable Signatures. Ph.D. Thesis, University of Passau, 2018.

[19] E. Porat, A. Rothschild. *Explicit nonadaptive combinatorial group testing schemes.* IEEE Transactions on Information Theory 57 (2011) 7982– 7989.

[20] R. Safavi-Naini and H. Wang. New results on multi-receiver authentication codes. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 527–541. Springer Berlin Heidelberg, 1998.

[21] R. Steinfeld, L. Bull, and Y. Zheng. Content Extraction Signatures. In *Information Security and Cryptology — ICISC 2001.* Lecture Notes in Computer Science, vol 2288, pages 285–304

[22] D. Stinson and M. Paterson. *Cryptography: Theory and Practice, Fourth Edition.* CRC Press, 4th edition, 2018.

[23] G. M. Zaverucha and D. Stinson. Group testing and batch verification. In *Proceedings of the 4th International Conference on Information Theoretic Security*, ICITS'09, pages 140–157, 2009.

Chapter 6

General conclusion and open questions

Now we conclude with a summary of our major contributions and interesting open questions that can lead to future research.

6.1 Summary of major contributions

In this thesis, we proposed several new definitions and constructions of cover-free families, which were inspired by the need for different properties of CFFs in certain fault-tolerance problems in cryptography. Our major contributions are discussed next.

6.1.1 Cover-free family constructions

We proposed three new definitions and respective constructions in this thesis: nested cover-free families, embedding cover-free families, and variable cover-free families.

6.1.1.1 Nested cover-free families

In Chapter 3 (our papers [4, 5]) we propose nested cover-free families. They are infinite sequences of d-CFFs based on a recursive construction of the form $\mathcal{M}^{(l+1)} = \left(\begin{smallmatrix} \mathcal{M}^{(l)} & Y \\ Z & W \end{smallmatrix}\right)$, where $\mathcal{M}^{(l)}$ and $\mathcal{M}^{(l+1)}$ are two consecutive incidence matrices in the sequence, and Z consists of rows of zeros, rows of ones, or repeated rows from $\mathcal{M}^{(l)}$. They are a generalization of monotone families from Hartung et al. [2], where Z is fixed as zero rows. These families are important for applications of a dynamic nature, where an increasing number of columns n is a requirement. We provide three different constructions for nested families.

In Theorem 3.5.1 we propose a nested family construction for $d = 1$, which consists of Sperner set systems with increasing size of subsets listed in a specific order to achieve the nested property. This construction provides Z equal to rows of zeros and rows of

ones, and has compression ratio reaching the information theoretical bound $\rho(n) = \frac{n}{\log n}$ (Theorem 3.5.2).

In order to provide constructions for nested families with $d \geq 2$, we first generalized two known recursive constructions for 2-CFFs in [10] based on the Kronecker product of matrices. We proved in Theorems 3.3.11 and 3.3.13 that similar constructions can be used to provide d-CFFs as well. In Theorem 3.5.3 we proved that we can construct nested families by instantiating the first recursive construction in a sequence, and this construction gives for any chosen integer constant $c \geq 2$ a family with compression ratio $\rho(n) = \frac{n}{n^{1/c}} = n^{1-1/c}$ (Theorem 3.5.4); in Theorem 3.5.5 we prove the same can be done with the second recursive construction, and we obtain a nested families with compression $\rho(n) = \frac{n}{(b \log_2 n)^{\log_2 \log_2 n + D}}$ for constants b, D (Theorem 3.5.6). All our constructions present much better compression ratio than the construction of monotone families given in Hartung et al. [2], which has constant compression ratio.

We also proved that both recursive constructions with $d \geq 2$ can be generalized to create $(d; \lambda)$-CFFs (Theorems 3.6.2 and 3.6.3). From these constructions, we provided nested $(d; \lambda)$-CFFs with the same ratio as before (Theorems 3.6.4 and 3.6.5). This is an important property that offers an extra layer of fault tolerance to certain problems, in particular, the problem of unbounded fault tolerance in aggregation of signatures that we solved using nested families.

6.1.1.2 Embedding cover-free families

We notice that the requirements imposed on sub-matrix Z in the definitions of both nested and monotone families imply that they only exist for fixed d. In Chapter 4 (our paper [6]) we propose embedding cover-free families as a generalization of monotone and nested families, with no requirement for Z, so we can propose constructions of such families with increasing n and d. Note that the term "embedding" has been previously employed in recursive constructions of CFFs [14], but they yield what here we called monotone families, which are more restrictive as discussed before.

In this chapter, we start by studying a known d-CFF construction based on polynomials over finite fields given by Erdös et al. [1] (Construction 4.4.1). We identify a special structure on this construction, which allows us to discard blocks of rows of the d-CFF if we need a d smaller than the maximum the construction provides (Theorem 4.4.4). This property is especially interesting for group testing applications, which allows us to stop the tests earlier if the application has a number of errors $d' < d$. This characteristic is also explored in our next constructions.

We propose a construction of embedding families based on a sequence of CFFs constructed via polynomials, where each CFF in the sequence is built using the extension field of the previous ones, forming a sequence based on a tower of finite fields. Because of the properties of finite fields, we know that the elements and polynomials of the subfield are contained in the extension field, and therefore the corresponding CFFs have the embedding cover-free family properties. For these constructions, we deal with parameters d, n, and k. The first two are parameters of the CFF, and k is the maximum degree of the

polynomials used to create the CFF (Construction 4.4.1). By picking different parameters for this construction, we were able to:

(i) Fix k and focus on d and n increases with compression ratio $\rho(n) = n^{1-\frac{2}{k+1}}$ and $d \sim \frac{n^{1/(k+1)}}{k}$ (Theorem 4.4.7).

(ii) Fix d and focus on k and n increases to achieve the best compression ratio possible $\rho(n) = \frac{n}{\log n}$ (Theorem 4.4.8).

(iii) Fix both d and k and pick specific blocks of rows of the incidence matrix (because of the property we identified earlier), and build monotone families with increasing compression ratio of $\rho(n) = n^{1-\frac{1}{k+1}}$, as opposed to constant compression ratio as given in [2] (Theorem 4.4.9).

Finally, in Section 4.6 we show it is possible to generalize our embedding family construction to rely on the existence of certain sequences of packing arrays and separating hash families, which is more general than the polynomial construction using towers of finite fields.

6.1.1.3 Generalizations of d-CFFs

In Chapter 2 (our paper [7]) we define a new cover-free family generalization that we call $(w, \mathcal{S}; \lambda)$-CFF, or variable CFFs. In this generalization, we explicitly give in $\mathcal{S}$ the combinations of subsets that are required to present a cover-free property, instead of requiring the property for every combination of $d + w$ subsets. In Theorem 2.5.4 we propose a construction of an $\mathcal{S}$-CFF based on the Kronecker product of a d_1-CFF and a d_2-CFF, and in Theorem 2.5.5 we prove that this construction can be generalized to build $(w, \mathcal{S}; \lambda)$-CFFs.

In this same chapter, we identify that the two recursive constructions we proposed for $(d; \lambda)$-CFFs in Theorems 3.6.2 and 3.6.3 can be generalized to build $(w, d; \lambda)$-CFFs. These generalizations are proved in Theorems 2.5.7 and 2.5.9, and their proofs derive from the construction of $(w, \mathcal{S}; \lambda)$-CFFs from Theorem 2.5.5.

6.1.2 Cryptographic application

6.1.2.1 Dynamic problems

In Chapter 3 we provide a solution for unbounded fault-tolerant aggregation of signatures using nested cover-free families. We propose algorithms for aggregation and verification of signatures using nested cover-free families (page 54), which allow for dynamic increases on the number of signatures n. Due to the constructions we proposed for nested families, with good compression ratio, the aggregate signature has a moderate increase as n grows. We also provide an extra layer of fault tolerance to this problem by proposing the use of nested $(d; \lambda)$-CFFs. With these families, we are able to handle up to $\lambda - 1$ corruptions in the aggregate signature without compromising the identification of any valid signature.

In Chapter 4, our embedding cover-free families provide a solution for problems that require not only growth of n, but also the growth of d. We list in this chapter possible cryptographical applications that may benefit from such families, which include broadcast authentication and encryption, which has now the possibility of scalability as the number of participants increases. By using embedding families, old participants in these schemes would be required to receive some extra keys every time we increase n and d, but this is a reasonable requirement given the benefit of increasing d.

In general terms, these problems served as an inspiration to create nested and embedding cover-free families. Our constructions of nested and embedding families with increasing compression ratio have a direct impact on the efficiency of all these schemes. Not only is the dynamic nature of these problems maintained, but we also obtain a reasonable growth on the size of the aggregate signature for the first problem, and a reasonable growth on the number of keys for the broadcast authentication and encryption schemes.

6.1.2.2 Static problems

We define a general framework for modification tolerant signature schemes, where it is possible to not only locate but also correct modifications that occur after a document is signed (Chapter 5 and our paper [8]). We show the scheme from [3] as one possible instantiation of MTSS, where d-CFFs are used to provide a signature that carries enough information to locate up to d modified blocks. We also propose a simple scheme to correct these modified blocks, which is efficient if the blocks are small enough. In Section 2.6.1.1 we discuss that this problem can be seen in terms of variable CFFs, where modifications are concentrated in specific locations of a document.

In Section 5.7 we show that our general definition of MTSS can also be instantiated in a way to provide redactable signature schemes, where the signed document is modified on purpose in order to guarantee the privacy of certain parts. We instantiate a redactable scheme based on d-CFFs, where we propose algorithms for the generation and verification of digital signatures, as well as a redacting algorithm that erases from the document and signature any information related to the private portions of data. Since the constructions are based on d-CFFs, the scheme can redact up to d private portions of the document without compromising the verification of the remaining parts.

6.2 Open questions

Here we describe some open questions that arose from this thesis research.

6.2.1 Questions related to general compression ratio

We achieved the best compression ratio $\rho(n) = \frac{n}{\log n}$ in our nested family construction for $d = 1$, and in our embedding construction when we fix d and grow n to its maximum. All the

other constructions of nested and monotone families (fixed d), and of embedding families (increasing d), present ratios that are much better than the previously known construction for increasing n (monotones by [2]). We wonder if there exist other constructions that can achieve even better compression ratio, i.e. closer to the information theoretical upper bound of $\frac{n}{(d^2/\log d)\log n}$. In more detail, we have the following questions:

Q1. Can we construct embedding cover-free families with increasing d and ratio equal/closer to the information theoretical upper bound?

Q2. Can we construct monotone and nested cover-free families (fixed d) with better compression ratio than the ones presented in this work?

Q3. For $d \geq 2$, is the compression ratio upper bound of the best monotone and nested families equal to the information theoretical upper bound, or are they more limited because of the strict requirements?

One possible approach to create new constructions is to explore the many known constructions of CFFs (for example, the ones we present in Chapter 2) and explore the possibilities of creating recursive constructions that may carry some monotone, nested, or embedding properties. In the same direction, it would be interesting to investigate how algorithms such as the one by Porat and Rothschild [12], or new ones, could be used to construct monotone, nested or embedding families.

6.2.2 Questions related to embedding families

Our construction of embedding cover-free families using extension fields allows for sequences of CFFs with increasing d and n. The extension field approach requires growth via taking specific powers of the initial prime power q, which results in an abrupt growth of n every time we go to the next family in the sequence, and consequently gives very good compression ratios. We discuss in Section 4.5 the challenge of using this construction when increases of n happen on a smaller scale. If the desired application requires smaller increases of n, we can simply discard the unnecessary columns of the embedding incidence matrix, but with this solution we end up compromising the compression ratio, and drops may happen as shown in the example of Figure 4.4. This gives rise to the following question:

Q4. Can we build embedding families with a more gradual increase of n, and consequently a more smooth compression ratio (with no drops)?

One possible path is to consider constructing d-CFFs via rings instead of finite fields, and explore if we could create embedding families with increasing d and n via these constructions. If that is possible, we believe they might lead to embedding family constructions with moderate growth of n, due to the fact that their existence does not rely on prime powers. However, many of the properties we had before came from the fact that we are using finite fields. More study needs to be conducted on how to obtain similar properties with rings, to allow for new constructions of embedding families with increasing n and d. Another approach may be to explore embeddings of combinatorial designs as a possibility to create embedding families, for example, explore constructions for embedding block

designs into larger block designs and the construction of creating embedding cover-free families from them.

6.2.3 Questions related to ranking and unranking

Consider the d-CFF representation as set system $(X, \mathcal{B})$, with $\mathcal{B}$ a collection of subsets. For any $B \in \mathcal{B}$, a *ranking* algorithm receives B and outputs an integer in $\{0, \ldots, n-1\}$ which determines the position of B among all subsets in $\mathcal{B}$. An *unranking* algorithm receives the integer in $\{0, \ldots, n-1\}$ and returns the corresponding subset B [9].

Unranking algorithms can be particularly important for d-CFF applications. For example, one-time and multiple-times signature schemes receive a message M as input, and need to identify the subset B_M in order to create the corresponding digital signature; for applications such as broadcast encryption, if we want to exclude some user from recovering a content, we need to identify this user's keys by identifying the subset that represents the user; it may also be an interesting feature for selective verification in aggregation of signatures, where we are interested in verifying a signature σ_j without verifying all pieces of the aggregate signature, so the knowledge of B_j identifies the aggregates that contain this signature [2].

An efficient unranking algorithm for 1-CFFs was proposed in [15] for a one-time signature scheme. Unranking for the d-CFF construction via polynomials over finite fields was discussed in [11]. It consists of transforming each message into a unique polynomial f, and then evaluating this polynomial at elements in that finite field to recover B_f. An unranking algorithm for our nested 1-CFF construction should be very similar to the one in [15]. We also believe that unranking algorithms applied to our embedding construction via polynomials may have a similar solution to the one from [11], with the extra concern of stopping the evaluation earlier when we restrict the construction to specific blocks of rows. For the constructions of nested families based on Kronecker product, it may be interesting to explore a bit further and propose ranking and unranking algorithms. Therefore, the following remains an open question.

Q5. Propose ranking and unranking algorithms for each construction of monotone, nested and embedding families proposed in this work.

6.2.4 Questions related to variable CFFs

In Chapter 2, we define $(w, \mathcal{S}; \lambda)$-CFFs, or variable CFFs, and present a construction based on the Kronecker product of a $(w, d_1; \lambda_1)$-CFF and a $(w, d_2; \lambda_2)$-CFF. One of the first open questions is related to other constructions for such objects. We know Kronecker product may not be the best approach since it multiplies the number of rows of the two CFFs, so we wonder if there are other (better) constructions.

Q8. Are there other constructions for variable CFFs with smaller t?

One of the possible paths that can be explored for constructions of variable CFFs is using their connection to covering arrays. In Chapter 2, we observed some relationships

between traditional CFFs and covering arrays, in which we define the CFF incidence matrix as a generalization of a covering array with a binary alphabet. Our definition of variable CFFs is also somehow related to variable strength covering arrays (VCA) [13]. One possible path to design new constructions of variable CFFs may be to explore existing constructions of VCA.

Another open problem is related to finding lower bounds on the number of rows of variable CFFs, which depends on the coverage parameter $\mathcal{S}$. One approach to measuring it may be to explore the properties of the hypergraph that defines the coverage in the variable CFF definition (Definition 2.5.1). Details on how this bound can be computed remain an open problem.

Q9. Compute lower and upper bounds on variable CFFs.

If a problem can be solved using both variable CFFs and traditional CFFs, the use of variable CFFs in these scenarios may imply solutions with a smaller t, since it does not require all subsets of the CFF to have the very same property. Results on lower and upper bounds can help us properly measure this. From these bounds we may be able to prove that $t(\mathcal{S}_n, n) < t(d, n)$ for an $\mathcal{S}_n$ that allows us to identify up to d modifications of specific types. In general, we believe that variable CFFs may lead to interesting future research and applications.

Bibliography

[1] P. Erdös, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51:79 – 89, 1985.

[2] G. Hartung, B. Kaidel, A. Koch, J. Koch, and A. Rupp. Fault-tolerant aggregate signatures. In *Public-Key Cryptography – PKC 2016*, pages 331–356, 2016.

[3] T. B. Idalino, L. Moura, R. F. Custódio, and D. Panario. Locating modifications in signed data for partial data integrity. *Information Processing Letters*, 115(10):731 – 737, 2015.

[4] T.B. Idalino, L. Moura. Efficient Unbounded Fault-Tolerant Aggregate Signatures Using Nested Cover-Free Families In *International Workshop on Combinatorial Algorithms, IWOCA 2018*. Lecture Notes in Computer Science, vol 10979, pages 52–64. Springer, Cham.

[5] T.B. Idalino, L. Moura. Nested Cover-Free Families for Unbounded Fault-Tolerant Aggregate Signatures. Manuscript submitted for publication, 2018.

[6] T.B. Idalino, L. Moura. Embedding cover-free families and cryptographical applications. *Advances in Mathematics of Communications*, **13** (2019), 629–643.

[7] T.B. Idalino, L. Moura. Cover-free families and generalizations. In preparation for submission, 2019.

[8] T.B. Idalino, L. Moura, C. Adams. Modification tolerant signature schemes: location and correction. In preparation for submission, 2019.

[9] D. L. Kreher, and D. R. Stinson, *Combinatorial Algorithms: Generation, Enumeration and Search.* CRC Press, 1999.

[10] P. C. Li, G. H. J. van Rees and R. Wei, Constructions of 2-cover-free families and related separating hash families, *Journal of Combinatorial Designs*, 14:423–440, 2006.

[11] J. Pieprzyk, H. Wang, and C. Xing. Multiple-time signature schemes against adaptive chosen message attacks. In *Selected Areas in Cryptography*, pages 88–100. Springer Berlin Heidelberg, 2003.

[12] E. Porat and A. Rothschild. Explicit nonadaptive combinatorial group testing schemes. *IEEE Transactions on Information Theory*, **57** (2011), 7982–7989.

[13] S. Raaphorst, L. Moura, and B. Stevens. Variable strength covering arrays. *Journal of Combinatorial Designs*, 26:417–438, 2018.

[14] D. R. Stinson and R. Wei, Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM J. Discrete Math.* , **11** (1998), 41–53.

[15] G.M. Zaverucha and D.R. Stinson. Short one-time signatures. *Advances in Mathematics of Communications*, 5:473–488, 2011.