Hash Families and Cover-Free Families with Cryptographic Applications

by

Gregory Zaverucha

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis is focused on hash families and cover-free families and their application to problems in cryptography. We present new necessary conditions for generalized separating hash families, and provide new explicit constructions. We then consider three cryptographic applications of hash families and cover-free families. We provide a stronger definition of anonymity in the context of shared symmetric key primitives and give a new scheme with improved anonymity properties. Second, we observe that finding the invalid signatures in a set of digital signatures that fails batch verification is a group testing problem, then apply and compare many group testing algorithms to solve this problem efficiently. In particular, we apply group testing algorithms based on cover-free families. Finally, we construct a one-time signature scheme based on cover-free families with short signatures.

Acknowledgements

I had a lot of help and support during the four years I worked towards this degree. First I'd like to thank my supervisor, Doug Stinson, who consistently provided sound guidance, on matters technical or otherwise. I'm also grateful for help from Ian Goldberg, Urs Hengartner and Alfred Menezes who were always available to hear whatever questions I thought they could answer. My internships at the IBM Zurich Lab were both fun and productive, thanks to Jan Camenisch and members of the Security and Cryptography group who were excellent hosts.

While a Ph.D. student I met many new and interesting people who made the time I spent in Waterloo enjoyable. Overall, my colleagues in the CrySP lab were top notch, and I especially enjoyed working and chatting with Aniket Kate, Joel Reardon and Jeremy Clark. I had a lot of fun times outside the lab (mostly at Kickoffs) with Jaime Ruiz, Christina Boucher, Rob Warren, Mike Patterson, Jeff Pound, Nick Miller, Söeren Bleikertz and Cam Zwarich.

My family has always been very supportive. My parents Patricia and Walter encouraged me to pursue graduate studies, and have always been understanding of my busy schedule. I was happy to have them visit regularly, along with my brother Andrew and sister Rachel while I was in Waterloo. The letters I regularly received from my grandmother Anne (on paper!) were a welcome distraction. I'm fortunate to have such a great family. I had plenty of additional love and support from my fiance Kate. While this chapter of my life is wrapping up, I'm looking forward to starting the next chapter together.

Finally, I'm thankful I did not have to worry about money while completing my degree. Thanks to the Canadian taxpayers, who (indirectly) funded me through the agencies NSERC and MITACS, and to David Cheriton, who helped fund me and many of my peers.

Thank you all.

Contents

List of Tables

Chapter 1

Introduction

This thesis is focused on hash families and cover-free families and their application to problems in cryptography.

A hash family is a collection of functions sharing a domain and co-domain. We say the family separates two inputs, if there is at least one function in the family that gives different outputs. This naturally generalizes to two or more sets of inputs, which are separated by a function when their respective output sets are disjoint. Hash families with such a property are called separating hash families (SHF). Perfect hash families (PHF) are an important special case, where we always have one function that separates a fixed number of single inputs. The number and size of the inputs is called the type of the SHF. Both PHF and SHF are important tools for cryptographers.

Cover-free families are a closely related combinatorial object. Given a set, a cover-free family is a collection of subsets, such that d of these subsets do not cover any other subset. In other words, no subset is contained in the union of any other d sets in the family.

1.1 Summary of Contributions

Our first contribution relates to the existence of SHF; we present bounds on SHF of arbitrary type. Our motivation for studying SHF of general type is that this single structure unifies previous definitions, since it includes many interesting structures previously studied in the literature as special cases. An upper bound gives the largest domain possible, for a fixed size co-domain and number of functions. We give the first upper bound for SHF of arbitrary type, which immediately yields an improved bound for a specific type of SHF called "secure frameproof codes". The latter type has applications in fingerprinting digital data.

Next, we consider ways of explicitly constructing SHF of arbitrary type, by generalizing existing constructions that apply to special types. We provide generalizations of many of the constructions from the literature. One construction that we examine in detail is based on algebraic geometry codes. We compare various instantiations of this construction and give a detailed description and comparison of the resulting parameters. We have implemented this construction and discuss some practical issues. For applications where a small amount of error is tolerable, random hash families can be used to separate a given input type with high probability, but require significantly fewer rows. We give an analysis of random hash families. Random hash families can easily be implemented.

With respect to cryptographic applications, we improve the anonymity of schemes for sharing symmetric key primitives. In this problem, we would like to share a secret key amongst many participants, so that any sufficiently large subset may compute a function of the secret key, and such that none of them learns the secret key. An example of such a function is a message authentication code. We provide a natural, stronger definition of anonymity (for the participants), and show that existing schemes do not provide this level of anonymity. A new scheme is presented, based on a special type of PHF, and we quantify the amount of information the output of the shared operation provides about the group of participants that collaborated to produce it.

The second cryptographic application we consider is a problem arising in batch verification of digital signatures. A batch verification algorithm allows a set of signatures to be verified as a group, and outputs a single bit to indicate that all the signatures are valid, or that the batch contains one or more invalid signatures. We observe that finding the invalid signatures in a failed batch is a group testing problem, then apply and compare many group testing algorithms to solve this problem efficiently. In particular, cover-free families provide a group testing algorithm that parallelizes with linear speedup, allowing all invalid signatures in a batch to be located quickly when multiple processors are available.

The last cryptographic application we consider is one-time signatures based on cover-free families. Five one-time signature schemes in the literature are based, either implicitly or explicitly, on cover-free families. We give a general construction based on cover-free families capturing these schemes, then give a new scheme. Our first goal is to reduce the signature size, as schemes in this family have long signatures. Second, we show how to choose parameters for these schemes to reduce the amount of storage required.

1.2 Combinatorial Structures

In this section we will introduce the main combinatorial structures used throughout this thesis: perfect and separating hash families, and cover-free families. Along with the definition of cover-free families we include some bounds from the literature which will be

relevant in later chapters. For perfect and separating hash families, bounds are discussed in Chapter 2 and constructions are the topic of Chapter 3. A construction of cover-free families is also presented in Chapter 3.

Definition 1.1. Let X and Y be sets with $|X| = n$ and $|Y| = m$. An $(N; n, m)$-hash family is a set $\mathcal{F}$ of N functions from X to Y.

While X can be any any n-set, for simplicity we often choose $X = \{1, \ldots, n\}$. The *matrix representation* of an $(N; n, m)$-hash family is the $N \times n$ matrix A where $A_{i,j} = f_i(j)$ for all i, j. We sometimes refer to the entries of A as *symbols*. A third representation is also useful at times.

Definition 1.2. An $(N; n, m)$-*code* is a set of n vectors (called *codewords*) of length N with coordinates from an alphabet of size m. Let C be an $(N; n, m)$ code. The *distance* between two codewords $x, y \in C$, denoted $\text{dist}(x, y)$, is the number of coordinates in which x and y differ (also called the *Hamming distance* of x and y). The *minimum distance* of a code is defined as the smallest distance between two different codewords, i.e.

$$\text{dist}(C) = \min \{\text{dist}(x, y) : x \in C, y \in C, x \neq y\} \ .$$

If C has minimum distance D, we call C an $(N; n, m, D)$-code.

Let A be the matrix representation of an $(N; n, m)$-hash family with domain and range X and Y, respectively. A can also be viewed as a code of size n and length N over the alphabet Y of cardinality m, where each column of A is a codeword. The code representation of a hash family will be used to give constructions of hash families with a given separating property from codes known to have large distance (§3.1.1).

1.2.1 Perfect Hash Families

A perfect hash family of strength two is a simple but useful hash family. Here we say that the family of functions satisfies the following property: For any pair of inputs, at least one of the functions in the family hashes them to distinct outputs (i.e., does not collide). The natural generalization is to increase the strength from two to some value $t \leq m$.

Definition 1.3. Let $\mathcal{F}$ be an $(N; n, m)$-hash family from $X \to Y$ with $|X| = n$ and $|Y| = m$. $\mathcal{F}$ is called a *perfect hash family* of *strength* t if for any distinct inputs $c_1, \ldots, c_t \in X$, there exists some $f \in \mathcal{F}$ such that $f(c_i) \neq f(c_j)$ for all $i \neq j$, $1 \leq i, j \leq t$. We use the notation $\text{PHF}(N; n, m, t)$.

In the matrix representation of a PHF$(N; n, m, t)$ we are guaranteed that any $N \times t$ subarray contains a row with no repeated symbol.

This well-known structure was introduced by Mehlhorn [120, 121] in the context of hash table data structures; if we must hash t elements of X at the same time, we can choose the function in the family which is one-to-one on these elements. Perfect hashing is a common tool in algorithms and complexity, see Alon and Naor [6] and the references given there. PHF are used as inputs for constructing group testing algorithms [161], cover-free families [161, 155] (see also §1.2.3), and covering arrays [117], a combinatorial object used in software interaction testing [55] (amongst other things).

In terms of cryptographic applications, PHF have been used in threshold secret sharing [23], broadcast encryption [83], shared symmetric key primitives [115, 116], private information retrieval [17], construction of fingerprinting codes [155] and key distribution patterns [161, 158].

Example 1.4. Here is the matrix representation of a PHF$(4; 9, 3, 3)$, where $X = \{1, \ldots, 9\}$ and $Y = \{1, 2, 3\}$, and $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$.

	1	2	3	4	5	6	7	8	9
f_1	1	3	2	2	3	2	3	1	1
f_2	1	3	1	3	1	2	2	2	3
f_3	1	2	2	1	3	3	1	2	3
f_4	3	3	2	1	1	3	2	1	2

1.2.2 Separating Hash Families

A perfect hash family is a special case of a more general type of hash family, a *separating hash family* (SHF). Consider hashing a subset C of X with a function f, where the output is the set $\{f(c) : c \in C\}$. In an SHF, the input is comprised of multiple disjoint subsets of X, and the *separating* property guarantees that the corresponding outputs sets are pairwise disjoint for at least one function in the family.

Definition 1.5. Let $\mathcal{F}$ be an $(N; n, m)$ hash family from $X \to Y$. $\mathcal{F}$ is called a *separating hash family* of type $\{w_1, w_2, \ldots, w_t\}$ if it satisfies the following property: For any disjoint subsets $C_1, \ldots, C_t$ of X with $|C_i| = w_i$ for $1 \leq i \leq t$, there exists at least one function $f \in \mathcal{F}$ such that

$$\{f(a) : a \in C_i\} \cap \{f(b) : b \in C_j\} = \emptyset$$

for every $i \neq j$. We will use the notation SHF$(N; n, m, \{w_1, \ldots, w_t\})$ to denote such a hash family.

If A is the matrix representation of $\mathcal{F}$, then A represents an SHF$(N; n, m, \{w_1, \ldots, w_t\})$ provided that the following condition is satisfied: for disjoint sets of column indices $C_1, \ldots, C_t$ where $|C_i| = w_i$ for $1 \leq i \leq t$, there exists a row r such that

$$\{A_{r,a} : a \in C_i\} \cap \{A_{r,b} : b \in C_i\} = \emptyset$$

for all $i \neq j$. We say that the row r *separates* $C_1, \ldots, C_t$.

We also write $\left\{1^\ell\right\}$ for the SHF type consisting of ℓ ones. By generalizing to t input sets, several previously studied separating properties are unified in a single definition. Here is a list of some structures which may be described as separating hash families.

Perfect hash families are of SHF of type $\{1^t\}$. Perfect hash families are the subject of Section 1.2.1.

Frameproof codes are SHF of type $\{w, 1\}$. These were introduced by Boneh and Shaw in [36] for fingerprinting digital data. This continues to be an active area of research [24, 58, 74, 75, 148, 155, 161, 174, 183]

Secure frameproof codes are SHF of type $\{w, w\}$, introduced in Stinson et al. [161] to provide improved security in fingerprinting applications. See also Cohen et al. [56, 58]. There have been several recent papers giving constructions for 2-secure frameproof codes; see [73, 172, 173]. Explicit constructions for w-secure frameproof codes for arbitrary $w \geq 2$ are found in [109, 161, 165]. Existence results using the probabilistic method are provided in [165]; we will revisit these results in Section 2.5. Finally, a necessary condition for the existence of w-secure frameproof codes was proven in [155] (we will give a significant improvement of this result in Section 2.2).

Strong SHF are SHF of type $\{1^{w_1}, w_2\}$, defined in Sarkar and Stinson [147]. Strong separating hash families were invented independently by Sarkar et al. and Barg et al. while studying w-IPP codes [14, 147]. (In [14] the term *partially hashing code* was used.) We will use the notation SSHF$(N; n, m, \{w_1, w_2\})$ to denote this type,

k-IPP Codes i.e., codes with the k-identifiable parent property, imply separating hash families which are simultaneously of type $\left\{1^{k+1}\right\}$ and $\{k, k\}$ (though the converse is not necessarily true). 2-IPP codes were introduced in Hollmann et al. [96], and generalized by Alon et al. [5]. k-IPP codes are also used in fingerprinting applications. Staddon et al. describe relationships between various fingerprinting codes, PHF, SHF and cover-free families [155]. Further results on k-IPP codes can be found in [7, 15, 27, 147, 155, 174, 178].

(k, u)**-hashing families** are SHF of type $\left\{1^k, u-k\right\}$ (a strong SHF). This type was introduced by Barg et al. [14] while studying k-IPP codes (see also Alon et al. [4]).

Separating hash families in their full generality were first considered by Cohen et al. [57], who generalized a sufficient condition for a code with large distance to be an SHF of a specified type. Stinson et al. did a thorough study (necessary conditions and existence results) of SHF having small type, that is, for $w_1 + \ldots + w_t \leq 4$ [164]. This study was extended to SHF of arbitrary type by Blackburn et al. [28].

To avoid trivialities, we always assume that $t \geq 2$ and that $u \leq n$, where we define $u = \sum_{i=1}^{t} w_i$. Clearly we must have that $t \leq m$ if an SHF$(N; n, m, \{w_1, w_2, \ldots, w_t\})$ exists. Note that when $m \geq n$, an injective function is a separating hash family (with $N = 1$) and the problem becomes trivial. So we may always assume that $m < n$.

Example 1.6. Here is the matrix representation of an SHF$(3; 16, 8, \{2, 2\})$.

0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7
0	1	0	1	2	3	2	3	4	5	4	5	6	7	6	7
0	1	1	0	2	3	3	2	4	5	5	4	6	7	7	6

1.2.3 Cover-Free Families

Cover-free families (CFF) have a long history. They were first defined by Kautz and Singleton [104] in 1964 (under the name superimposed binary codes). They were also defined in other areas such as information theory, combinatorics and group testing (e.g. [44, 69, 76, 77]). CFF also have many applications, especially in cryptography and communications: blacklisting [106], broadcast encryption [50, 85, 158, 157], anti-jamming [64], source authentication in networks [146], group key predistribution [124, 71, 157, 161], and frameproof/traceability codes [155, 162].

In this thesis cover-free families will be used in group testing for batch verification of digital signatures (Chapter 5), and the construction of one-time signatures (Chapter 6). Informally, a CFF is a set of sets, such that the union of w of these sets does not contain any of the other sets. A precise definition follows.

Definition 1.7. A w-cover-free family $(X, \mathcal{B})$ is a set X of m elements, and a set $\mathcal{B}$ of n subsets of X, with the following property. For any w sets $B_{i_1}, \ldots, B_{i_w} \in \mathcal{B}$, and all other $B \in \mathcal{B}$, it holds that

$$B \not\subseteq \bigcup_{j=1}^{w} B_{i_j}$$

We say that $B_{i_1}, \ldots, B_{i_w}$ does not *cover* any other $B \in \mathcal{B}$. We will use the notation w-CFF(m, n) for cover-free families. If $|B_i| = N$, for all i, we call the CFF N-*uniform*.

We now give the relation between CFF and binary SHF of type $\{1, w\}$.

Theorem 1.8. *Suppose there exists a* w-CFF(m, n). *Then there exists an* SHF$(m; n, 2, \{w, 1\})$.

Proof. Let $(X, \mathcal{B})$ be the set of elements and subsets in the w-CFF(m, n) and define the $m \times n$ incidence matrix A as follows: $A_{i,j} = 1$ if element i belongs to subset B_j, and $A_{i,j} = 0$ otherwise. The cover-free property ensures that no column of A is covered by any w others, since B_j contains an element not contained in the union of any w others sets. Therefore column j has a row which is 1 where the w other columns are 0. Hence A is an SHF of type $\{w, 1\}$. □

The following theorem describes a partial converse.

Theorem 1.9. *Suppose there exists an* SHF$(N; n, 2, \{w, 1\})$. *Then there exists an* N*-uniform* w-CFF$(2N, n)$.

Proof. Let A' be an SHF$(N; n, 2, \{w, 1\})$. Let $\overline{A'}$ be the complement of A' (formed by flipping each entry in A'), and let A be the SHF$(2N; n, 2, \{w, 1\})$ consisting of the rows of A' followed by the rows of $\overline{A'}$. The matrix A is the incidence matrix of a w-CFF$(2N, n)$, call it C, and use $(X, \mathcal{B})$ to denote the elements and set of subsets in C. Each row of A corresponds to an element of X, and each column defines a block $B_j \in \mathcal{B}$ as follows: if $A_{i,j} = 1$ then B_j contains the i-th element of X. By the type $\{w, 1\}$ SHF property any $w + 1$ columns of A' contain either the pattern $00\ldots0, 1$ or $11\ldots1, 0$ in some row, and always have the pattern $00\ldots0, 1$ in A. This ensures that each set $B \in \mathcal{B}$ will contain some element which is not present in the union of w other subsets in $\mathcal{B}$. To see that C is N uniform, note that each column of A will have weight[†] N, since if column j of A' has weight w_j, then column j in $\overline{A'}$ will have weight $N - w_j$ and the weight of column j in A is $w_j + N - w_j = N$. □

Example 1.10. We illustrate Theorem 1.9 with an example, where A' is an SHF$(4; 5, 2, \{2, 1\})$.

$$A' = \begin{array}{|c|c|c|c|c|}\hline 0&0&0&1&1\\\hline 1&0&0&0&1\\\hline 0&0&1&0&1\\\hline 0&1&0&0&1\\\hline\end{array} \qquad \overline{A'} = \begin{array}{|c|c|c|c|c|}\hline 1&1&1&0&0\\\hline 0&1&1&1&0\\\hline 1&1&0&1&0\\\hline 1&0&1&1&0\\\hline\end{array} \qquad A = \begin{array}{|c|}\hline A'\\\hline \overline{A'}\\\hline\end{array}$$

The output is a 4-uniform 2-CFF$(8, 5)$, denoted $(X, \mathcal{B})$. A has 8 rows, where row i of A corresponds to element x_i of X. The set $B_j \in \mathcal{B}$ is given by column j of A, therefore we

[†]Here *weight* refers to the number of nonzero entries in the column.

have:

$$B_1 = \{x_2, x_5, x_7, x_8\}$$
$$B_2 = \{x_4, x_5, x_6, x_7\}$$
$$B_3 = \{x_3, x_5, x_6, x_8\}$$
$$B_4 = \{x_1, x_6, x_7, x_8\}$$
$$B_5 = \{x_1, x_2, x_3, x_4\}$$

The 2-CFF property is easily checked. For example $B_1 \cup B_2$ does not contain x_3 and x_1, and so it does not cover B_3, B_4 or B_5.

Construction of 1-CFF(m, n) is optimal and simple, provided $n \leq \binom{m}{\lfloor m/2 \rfloor}$. The set of all $\binom{m}{\lfloor m/2 \rfloor}$ length m binary vectors with weight $\lfloor m/2 \rfloor$ form the $m \times n$ incidence matrix. Since each column is distinct, the 1-CFF property holds. This is Sperner's theorem[1] [154]. In Section 3.1.1 we will give a more general construction for w-CFF.

Bounds for Cover-Free Families

First we present a necessary condition for the existence of CFF, a lower bound on the size of the sets.

Theorem 1.11 (see [166, Theorem 1.1])**.** *For any $d \geq 1$, in a d-CFF(m, n), we have that*

$$m \geq c \left(\frac{d^2}{\log d} \right) \log n \ .$$

The constant c is approximately $1/8$ (shown in [145]).

De Bonis and Vaccaro bound m from the other direction.

Theorem 1.12 ([37, Corollary 1])**.** *There exists a d-CFF(m, n) with*

$$m < 24 d^2 \log(n + 2) \ .$$

Their proof method is constructive, based on a greedy algorithm, and it is efficient for small CFF (i.e., when n is not large).

[1] For a proof in English, see Spencer [153].

1.3 Information Theory

Here we review some basic concepts from information theory that will be used in this thesis.

We start with the definition of entropy of a random variable, (often called the *Shannon entropy*). Let X be a random variable defined on a set $\mathcal{X}$. The *entropy* of X is defined by

$$H(X) = -\sum_{x \in \mathcal{X}} \left(\Pr[X = x] \times \log_2 \Pr[X = x] \right) \ .$$

The quantity $H(X)$ measures the number of bits of uncertainty associated with X. When X is the uniform distribution on $\mathcal{X}$, i.e., X takes all values $x \in \mathcal{X}$ equally often, the entropy of X is a maximal $\log_2 |\mathcal{X}|$ bits. In the other extreme, when X takes only one value from $\mathcal{X}$, $H(X) = 0$. We will also work with the entropy of a conditional probability distribution, which is the entropy of X given that Y, a second random variable defined on a set $\mathcal{Y}$, takes the value y:

$$H(X|y) = -\sum_{x \in \mathcal{X}} \left(\Pr[X = x|Y = y] \times \log_2 \Pr[X = x|Y = y] \right) \ .$$

The final measure of entropy we define is the min-entropy. Again, let X be a random variable defined on a set $\mathcal{X}$. Let $\gamma = \max_{x \in \mathcal{X}} \{\Pr[X = x]\}$. The *min-entropy* of X is defined to be

$$H_\infty(X) = \log_2 \left(\frac{1}{\gamma} \right) = -\log_2 \gamma \ .$$

Since $H(X) \geq H_\infty(X)$, the min-entropy gives a lower bound on the Shannon entropy.

Chapter 2

Bounds

This chapter covers bounds on the size of SHF, i.e., given the number of rows N, the size of the range m, and a type T, how large a value of n, the size of the domain, is possible in an SHF of type T? An upper bound on n is sometimes called a necessary condition. Equivalently, we also consider bounds on the number of rows, i.e., given n, m and T what is the smallest possible value of N such that an SHF$(N; n, m, T)$ exists?

2.1 Previous Results

In this section we present previously known upper bounds on n for PHF, SHF and CFF. First we start with PHF. For strength two, note that any $N \times n$ array with m symbols is a PHF$(N; n, m, 2)$ if all columns are distinct, which can only happen when $n \leq m^N$. This was generalized to arbitrary strength in the following theorem (due to Blackburn and Wild [29]).

Theorem 2.1. *[29] In a* PHF$(N; n, m, t)$*, we have* $n \leq (t-1)(m^{\lceil \frac{N}{t-1} \rceil} - 1)$.

For SHF of type $\{2, 2\}$, Stinson et al. proved the following.

Theorem 2.2. *[164] In an* SHF$(N; n, m, \{2, 2\})$*, we have* $n \leq 4m^{\lceil \frac{N}{3} \rceil} - 3$.

The only previously known general bound for SHF of type $\{w, w\}$ (secure frameproof codes) is stated in Theorem 2.3.

Theorem 2.3. *[155] In an* SHF$(N; n, m, \{w, w\})$*, we have* $n \leq m^{\lceil \frac{N}{w} \rceil} + 2w - 2$.

Observe that Theorem 2.2 provides a considerable improvement to Theorem 2.3 in the case $w = 2$. In Section 2.2, we improve the bound of Theorem 2.3 first for SHF of type $\{3, 2\}$, and then for all types $\{w, w\}$ and $\{w, w-1\}$, by generalizing and extending the proof techniques of Stinson et al. [164].

For SHF of type $\{w, 1\}$ (frameproof codes) the following bound was previously known.

Theorem 2.4. *[24] In an* SHF$(N; n, m, \{w, 1\})$, $n \leq tm^{\lceil \frac{N}{w} \rceil} + O(m^{\lceil \frac{N}{w} \rceil - 1})$, *where* t *is the unique integer such that* $t \in \{1, 2, \ldots, w\}$ *and* $t \equiv N \mod w$.

Staddon et al. prove the following more precise bound for type $\{w, 1\}$. Using our proof techniques, we will give an alternate proof this bound in Section 2.3.

Theorem 2.5. *[155] In an* SHF$(N; n, m, \{w, 1\})$, $n \leq w(m^{\lceil \frac{N}{w} \rceil} - 1)$.

Our new bounds for SHF begin by considering types $\{w, w\}$ and $\{w, w-1\}$, then type $\{w, d\}$, and finally, the most general type $\{w_1, \ldots, w_t\}$. These results were published in a series of papers: Stinson and Zaverucha [168] contains the results for types $\{w, w\}$ and $\{w, w-1\}$, Stinson and Zaverucha [167] generalizes these to type $\{w, d\}$ and the results for type $\{w_1, \ldots, w_t\}$ were published in Blackburn, Etzion, Stinson and Zaverucha [28].

2.2 New Upper Bounds for SHF of Type $\{w, w\}$ and $\{w, w-1\}$

In this section we present an upper bound on SHF of type $\{w, w\}$, which correspond to secure frameproof codes. At the same time, since our proof techniques also apply to type $\{w, w-1\}$, we consider this type as well.

We begin with a couple of well-known elementary lemmas.

Lemma 2.6. *Suppose there exists an* SHF$(N; n, m, \{w_1, w_2, \ldots, w_t\})$, *and let* $c \geq 1$ *be an integer. Then there exists an* SHF$(\lceil \frac{N}{c} \rceil; n, m^c, \{w_1, w_2, \ldots, w_t\})$.

Proof. The case $c = 1$ is trivial. Let A be the matrix representation of an SHF$(N; n, m, \{w_1, \ldots, w_t\})$. For $c \geq 2$, create a new SHF A' as follows. Group the rows of A into blocks of size c, each group will correspond to a row of A' (note that there are $\lceil N/c \rceil$ groups). Replace the c symbols in each column of each group by the concatenation of the c symbols in the group (ordered by row). Note that A' will have no more than m^c distinct symbols, and the number of columns remains n. A' inherits the separating property from A: for any input X of type $\{w_1, \ldots, w_t\}$, some row r of A separates X, which causes the row of A' corresponding to the group containing r to separate X. □

The next lemma is a useful special case of Lemma 2.6.

Lemma 2.7. *Suppose there exists an* $\mathrm{SHF}(N; n, m, \{w_1, w_2, \ldots, w_t\})$*, and let* $d \geq 1$ *be an integer. Then there exists an* $\mathrm{SHF}(d; n, m^{\lceil \frac{N}{d} \rceil}, \{w_1, w_2, \ldots, w_t\})$.

Proof. By setting $c = \lceil \frac{N}{d} \rceil$ in Lemma 2.6, there exists an $\mathrm{SHF}(r; n, m^{\lceil \frac{N}{d} \rceil}, \{w_1, \ldots, w_t\})$, where $r \leq d$. This upper bound on r holds because $\left\lceil \frac{N}{\lceil N/d \rceil} \right\rceil \leq d$. In the case that $r < d$, we can always add another row while maintaining the separating property, therefore, we may take $r = d$. □

Lemma 2.8. *Suppose* A *is an* $\mathrm{SHF}(N; n, m, \{w_1, w_2, \ldots, w_t\})$*, and let* $w_1' \leq w_1$*. Then* A *is also an* $\mathrm{SHF}(N; n, m, \{w_1', w_2, \ldots, w_t\})$.

Proof. Let X be an input of type $\{w_1, \ldots, w_t\}$ separated by row r of A, and X' be an input of type $\{w_1', \ldots, w_t\}$ such that $w_1' \leq w_1$. Since the output set of size w_1 is pairwise disjoint from the other output sets, so are all of its subsets. □

The next lemma was first noticed by Hollmann et al. [95] and it provides a key idea for many of our proofs. When discussing matrix representations of hash families, the term "isomorphic" should be interpreted as "isomorphic up to permutation of rows and columns". We use "$*$" to denote any symbol.

Lemma 2.9. *[95] Let* A *be the matrix representation of an* $\mathrm{SHF}(3; n, m, \{2, 2\})$*. Then there is no submatrix of* A *isomorphic to the matrix*

a	a	$*$	$*$
$*$	b	b	$*$
$*$	$*$	c	c

.

Proof. The first and third columns cannot be separated from the second and fourth columns, and all pairs of disjoint sets of size two must be separated in an $\mathrm{SHF}(3; n, m, \{2, 2\})$. □

Our proofs will work by generalizing this *forbidden configuration* from type $\{2, 2\}$ to types $\{w, w\}$ and $\{w, w-1\}$ then to type $\{w, d\}$ and finally type $\{w_1, \ldots, w_t\}$.

2.2.1 Bounds For Type $\{3, 2\}$

To illustrate the technique we use, we first prove a result for SHF of type $\{3, 2\}$. We begin by looking at the special case $N = 4$.

Theorem 2.10. *Suppose $m \geq 2$. If an* SHF$(4; n, m, \{3, 2\})$ *exists, then $n \leq 7m - 6$.*

Proof. Suppose A is an SHF$(4; n, m, \{3, 2\})$ with $n = 7m - 5$ (we will obtain a contradiction).

We construct a submatrix A_1 of A where all elements appearing in the first row appear at least four times. To do this, we delete $t_1 + 2t_2 + 3t_3$ columns from A; namely, those in which elements appear exactly once, twice and three times (resp.) in the first row of A. If $t_1 + t_2 + t_3 = m$, then A would have at most $3m$ columns, which is fewer than $7m - 5$ because $m \geq 2$. Thus we delete no more than $3(m - 1)$ columns, leaving A_1 with at least $7m - 5 - 3(m - 1) = 4m - 2$ columns.

Now we create A_2 from A_1 in such a way that elements in the second row of A_2 repeat three or more times. To do this, we delete $u_1 + 2u_2$ columns of A_1; namely, those with an element appearing exactly once or twice (resp.). If $u_1 + u_2 = m$, A_1 would have at most $2m$ columns, which is fewer than $4m - 2$, so we have that $u_1 + 2u_2 \leq 2(m - 1)$. We can then say that A_2 has at least $4m - 2 - 2(m - 1) = 2m$ columns.

Our final submatrix will be A_3, where, in the third row, all elements will appear at least twice. Since A_2 has at least $2m$ columns, the number of elements appearing exactly once must be less than m. Then A_3 will have at least $2m - (m - 1) = m + 1$ columns.

Since A_3 has at least $m + 1$ columns, the fourth row of A_3 contains an element which occurs at least twice. Also, in A_3 every element in the third row occurs at least twice, so A_3 has a submatrix isomorphic to either

case 1:
$*$	$*$
$*$	$*$
c	c
d	d

or case 2:
$*$	$*$	$*$
$*$	$*$	$*$
c	c	$*$
$*$	d	d

.

In case 1, all elements in the second row of A_2 appear at least three times. Furthermore, all elements in A_1 appear at least four times in the first row of A_1. Therefore, A has a submatrix isomorphic to

a	a	$*$	$*$
$*$	b	b	$*$
$*$	$*$	c	c
$*$	$*$	d	d

. (2.1)

In case 2, when we consider the second row (which has elements repeating three times) there are two possibilities. The first is

$*$	$*$	$*$	$*$
$*$	b	b	b
$*$	c	c	$*$
$*$	$*$	d	d

which leads to a submatrix of A having the form

$$\begin{array}{|c|c|c|c|}\hline a & a & * & * \\ \hline * & b & b & b \\ \hline * & c & c & * \\ \hline * & * & d & d \\ \hline \end{array} . \tag{2.2}$$

This is easily seen because the elements in the first row of A_1 occur at least four times.

The second possibility for case 2 is

$$\begin{array}{|c|c|c|c|}\hline * & * & * & * \\ \hline b & b & * & * \\ \hline * & c & c & * \\ \hline * & * & d & d \\ \hline \end{array} .$$

After considering the first row (as a submatrix of A), there are again two possibilities that arise:

$$\begin{array}{|c|c|c|c|}\hline a & a & a & a \\ \hline b & b & * & * \\ \hline * & c & c & * \\ \hline * & * & d & d \\ \hline \end{array} \tag{2.3}$$

and

$$\begin{array}{|c|c|c|c|c|}\hline a & a & * & * & * \\ \hline * & b & b & * & * \\ \hline * & * & c & c & * \\ \hline * & * & * & d & d \\ \hline \end{array} , \tag{2.4}$$

depending on whether an a appears in columns three and four of the previous submatrix.

It is easy to see that all the possible submatrices (2.1), (2.2), (2.3) and (2.4) lead to a contradiction, because the odd columns cannot be separated from the even columns. If (2.4) is a submatrix, then A is not an SHF of type $\{3, 2\}$. If (2.1), (2.2) or (2.3) are submatrices, then A is not an SHF of type $\{2, 2\}$. Applying Lemma 2.8, we see that A is not an SHF of type $\{3, 2\}$ in these cases, as well. In every case, we have shown that A is not an SHF of type $\{3, 2\}$, which completes the proof. □

Corollary 2.11. *In an* SHF$(N; n, m, \{3, 2\})$, $n \leq 7m^{\lceil \frac{N}{4} \rceil} - 6$.

Proof. We proceed by contradiction. Suppose there exists an SHF$(N; n, m, \{3, 2\})$ where $n = 7m^{\lceil \frac{N}{4} \rceil} - 5$. Then, by Lemma 2.7 there exists an SHF$(4; 7m' - 5, m', \{3, 2\})$ with $m' = m^{\lceil \frac{N}{4} \rceil}$, which contradicts the previous theorem. □

2.2.2 Staircases

The following definition and lemmas will be useful tools for generalizing the proof technique used in Theorem 2.10. Staircases are the forbidden configurations we will use to prove necessary conditions for SHF of type $\{w, w\}$ and $\{w, w-1\}$.

Definition 2.12. An (N, t)-staircase is a matrix S with N rows of the form:

x_1	x_1							
	x_2	x_2						
		$\ddots$	$\ddots$					
			x_{t-1}	x_{t-1}				
			x_t	x_t				
				x_{t+1}	x_{t+1}			
					$\ddots$	$\ddots$		
						x_{N-1}	x_{N-1}	
							x_N	x_N

Further,

(i) when $t = 1$, S is a *regular* staircase, having $N + 1$ columns,

(ii) when $1 < t \leq N$, S is a *compressed* staircase having N columns.

Example 2.13. The first of the three matrices below is a $(4, 1)$-staircase, the second is a $(4, 4)$-staircase and the third is a $(4, 3)$-staircase.

a	a	$*$	$*$	$*$
$*$	b	b	$*$	$*$
$*$	$*$	c	c	$*$
$*$	$*$	$*$	d	d

a	a	$*$	$*$
$*$	b	b	$*$
$*$	$*$	c	c
$*$	$*$	d	d

a	a	$*$	$*$
$*$	b	b	$*$
$*$	c	c	$*$
$*$	$*$	d	d

Lemma 2.14. *Let A be a matrix with N rows. If A has submatrices $A_{N-1} \subset \ldots \subset A_1 \subset A$ such that*

- *For $1 \leq i < N-1$, all elements appearing in row i of A_i appear at least $N-i+1$ times in row i of A_i, and*
- *The last row of A_{N-1} contains at least one element that occurs at least twice.*

Then A has a submatrix isomorphic to an (N,t)-staircase, for some t, $1 \leq t \leq n$.

Proof. In this proof, we number the columns in increasing order from right to left. First, permute the columns of A_{N-1} so that there is a repeated element of row N appearing in columns 1 and 2 (call this element x). These two cells form the base of the staircase. We will create the staircase step by step, extending it upwards when moving from a submatrix of A_i to A_{i-1}, for $i = N-1, N-2, \ldots, 1$. In row $N-1$ of A_{N-1}, every element occurs at least twice. Letting y be the element above x in column 2, we have two possibilities (possibly after permuting columns):

case 1:

y	y
x	x

or case 2:

y	y	$*$
$*$	x	x

.

In the first case, all elements in row $N-2$ of the submatrix A_{N-2} occur at least three times. Our staircase is only two columns wide, so we extend it up and to the left by one step. We can continue extending leftward by one column as we consider each successive submatrix. The result is a (compressed) (N,N)-staircase.

In the second case, we consider the element in row $N-2$ and column 3, say z. We know that z occurs at least three times in row $N-2$ of the submatrix A_{N-2}. There are two sub-cases: either z occurs in columns 1, 2 and 3, or z occurs in column 3 and some new column which we can name column 4. These two sub-cases are analogous to sub-cases considered in the proof of Theorem 2.10.

In the second sub-case, we have extended the staircase up and to the left by one step. In the first sub-case, the staircase does not extend leftward at this stage.

This process can be continued until we reach the top of the staircase. If at any stage, we do not extend the staircase leftward, then it must be the case that all further submatrices cause the staircase to extend up and to the left. So we end up with a (N,t)-staircase, for some t, $1 \leq t \leq n$.

□

The usefulness of staircases is established in the following lemma.

Lemma 2.15. *Suppose A is an $(N; n, m)$ matrix having a submatrix isomorphic to an (N, t)-staircase S. Then A is not an SHF of type $\{h, h\}$ if $N = 2h - 1$, and A is not an SHF of type $\{h, h-1\}$ if $N = 2h - 2$.*

Proof. When A has a submatrix isomorphic to an (N, t)-staircase, the odd and even indexed columns of the staircase are inseparable. We consider two cases, depending on the parity of N.

If $N = 2h - 1$, then S has $2h$ columns if S is regular and $2h - 1$ columns if S is compressed. If S is regular, then A is not an SHF of type $\{h, h\}$. If S is compressed, then A is not an SHF of type $\{h, h-1\}$. By Lemma 2.8, A is not an SHF of type $\{h, h\}$.

The case when $N = 2h - 2$ is similar. S has $2h - 1$ columns if S is regular and $2h - 2$ columns if S is compressed. If S is regular, then A is not an SHF of type $\{h, h-1\}$. If S is compressed, then A is not an SHF of type $\{h-1, h-1\}$. By Lemma 2.8, A is not an SHF of type $\{h, h-1\}$. □

2.2.3 Proofs for Types $\{w, w\}$ and $\{w, w-1\}$

In this section, we generalize the technique used for SHF of type $\{3, 2\}$ to types $\{w, w\}$ and $\{w, w-1\}$.

Theorem 2.16. *If A is an* SHF$(2w-1, n, m, \{w, w\})$*, then*

$$n \leq m + (w-1)(2w-1)(m-1).$$

Proof. Let $N = 2w - 1$, and suppose A had one extra column, i.e., suppose A is an SHF$(N, n, m, \{w, w\}))$ with $n = m + 1 + \frac{N(N-1)}{2}(m-1)$ columns. We will derive a contradiction by showing A has a submatrix isomorphic to an (N, t)-staircase.

We will create a series of submatrices $A_{N-1} \subset \ldots \subset A_1 \subset A$. Submatrix A_i has the property that elements in the i-th row repeat $N - i + 1$ or more times (in the ith row). Denote $A_0 = A$. The construction of A_{i+1} from A_i deletes all columns of A_i where elements in the $(i+1)$-st row appear fewer than $N - i - 1$ times. Let $|A_i|$ denote the number of columns in A_i. By repeatedly applying this construction, we claim that

$$|A_i| \geq m + 1 + \frac{1}{2}(N - i - 1)(N - i)(m - 1) \tag{2.5}$$

for $0 \leq i \leq N - 1$. We will prove that (2.5) holds by induction on i.

First note that (2.5) holds for A_0. To construct A_1 from A_0, we must delete the columns containing elements which appear fewer than N times in the first row of A_0. Let t_q be the

number of columns with elements appearing exactly q times in the first row of A_0. We claim that $t_1 + \ldots t_{N-1} \leq m-1$. If $t_1 + \ldots + t_{N-1} = m$, then A_0 has at most $(N-1)m$ columns, fewer than the number given by (2.5). Then we delete at most $(N-1)(m-1)$ columns, so

$$\begin{aligned}
|A_1| &\geq |A_0| - (N-1)(m-1) \\
&\geq m+1+\frac{N(N-1)}{2}(m-1) - (N-1)(m-1) \\
&= m+1+\left(\frac{N(N-1)-2(N-1)}{2}\right)(m-1) \\
&= m+1+\frac{(N-2)(N-1)}{2}(m-1),
\end{aligned}$$

which shows that (2.5) holds for A_1 as well.

Suppose (2.5) holds up to A_i, suppose that $t_1+\ldots+t_{N-i-1} \leq (m-1)$, and suppose that elements in the i-th row of A_i appear at least $N-i$ times. To create A_{i+1} from A_i, we delete columns of A_i where elements in the $(i+1)$-th row appear less than $N-(i+1) = N-i-1$ times. If $t_1 + \ldots + t_{N-i-2} = m$, then A_i would have at most only $m(N-i-1)$ columns, fewer than the number assumed in the inductive hypothesis. Then

$$\begin{aligned}
|A_{i+1}| &\geq |A_i| - (N-(i+1))(m-1) \\
&= m+1+\frac{(N-i-1)(N-i)}{2}(m-1) - (N-i-1)(m-1) \\
&= m+1+\left(\frac{(N-i-1)(N-i)-2(N-i-1)}{2}\right)(m-1) \\
&= m+1+\left(\frac{(N-i-1)(N-i-2)}{2}\right)(m-1) \\
&= m+1+\frac{(N-(i+1)-1)(N-(i+1))}{2}(m-1)
\end{aligned}$$

which proves (2.5) by induction.

The last submatrix is A_{N-1}, with

$$|A_{N-1}| \geq m+1+\frac{(N-(N-1)-1)(N-(N-1))}{2}(m-1) = m+1\ .$$

Since A_{N-1} has at least $m+1$ columns, one of the elements in row N is must repeat. By applying Lemmas 2.14 and 2.15, we see that A is not an SHF, contradicting the assumption that A was an SHF$(2w-1, 4wm-4w, m, \{w, w\})$. This proves the theorem. □

Corollary 2.17. *If an* SHF$(N; n, m, \{w, w\})$ *exists, it holds that*

$$n \leq m^{\lceil \frac{N}{(2w-1)} \rceil} + (w-1)(2w-1)(m^{\lceil \frac{N}{(2w-1)} \rceil} - 1).$$

Proof. We proceed by contradiction (analogous to Corollary 2.11). Suppose there exists an SHF$(N; n, m, \{w, w\})$ where $n = m + 1 + (w-1)(2w-1)(m-1)$. Then by Lemma 2.7, there exists an SHF$(2w-1; m' + (w-1)(2w-1)(m'-1), m', \{w, w\})$ with $m' = m^{\lceil \frac{N}{(2w-1)} \rceil}$, which contradicts the previous theorem. □

Theorem 2.18. *If an* SHF$(2w-2; n, m, \{w, w-1\})$ *exists, then*

$$n \leq m + (w-1)(w - \frac{3}{2})(m-1).$$

Proof. Omitted. Follows the proof of Theorem 2.16 closely, with $N = 2w-2$ instead of $N = 2w-1$. □

The proof of the next corollary is basically the same as the proof of Corollary 2.17.

Corollary 2.19. *If an* SHF$(N; n, m, \{w, w-1\})$ *exists, it holds that*

$$n \leq m^{\lceil \frac{N}{2w-2} \rceil} + (w-1)(w - \frac{3}{2})(m^{\lceil \frac{N}{2w-2} \rceil} - 1).$$

2.3 Necessary Condition for SHF of Type $\{w, d\}$

In this section we give necessary conditions for SHF of type $\{w, d\}$. This is motivated by Section 2.4, where the result for type $\{w, d\}$ leads to a necessary condition for type $\{w_1, \ldots, w_t\}$.

SHF of type $\{w, 1\}$

First we prove a necessary condition for SHF of type $\{w, 1\}$, not because it is new (see [29] or [155]), but to illustrate part of the forbidden configuration that will be used in Section 2.3 for SHF of type $\{w, d\}$.

Theorem 2.20. *In an* SHF$(w; n, m, \{w, 1\})$*, it holds that* $n \leq m + (w-1)(m-1)$.

Proof. Let $N = w$, and suppose A is an SHF$(N; n, m, \{w, 1\})$ with $n = m+1+(N-1)(m-1)$. We will create a series of submatrices $A_{N-1} \subset \ldots \subset A_1 \subset A$. Let $A_0 = A$. Submatrix A_i has the property that all elements in the i-th row repeat at least twice. When we create A_1 from A, we delete no more than $m-1$ columns, since if all m symbols appeared exactly once, A would have m columns, but we assumed it has $m + 1 + (N-1)(m-1)$. Thus

$$|A_1| \geq |A_0| - (m-1) = m + 1 + (N-2)(m-1).$$

Assume that

$$|A_i| \geq |A_{i-1}| - (m-1) = m + 1 + (N-(i+1))(m-1). \tag{2.6}$$

When we create A_{i+1}, we delete at most $m-1$ columns since we assumed A_i has more than m columns. Then

$$\begin{aligned} |A_{i+1}| &\geq |A_i| - (m-1) \\ &= m + 1 + (N-(i+1))(m-1) - (m-1) \\ &= m + 1 + (N-(i+2))(m-1) \end{aligned}$$

and we conclude that (2.6) holds. The last submatrix A_{N-1}, has $m+1$ columns so we can be sure that at least one element repeats twice in row N. We permute columns to place the repeated element in the last two columns, and label elements in the rightmost column.

$*$	a
$\vdots$	$\vdots$
$*$	x
$*$	y
z	z

Now we know that each element in the last column will be repeated in the same row as we move from submatrix A_i to submatrix A_{i-1}. When each repetition occurs in a different column, A has a submatrix isomorphic to

a	$*$	$*$	$*$	$*$	a
	$\ddots$				$\vdots$
$*$	$*$	x	$*$	$*$	x
$*$	$*$	$*$	y	$*$	y
$*$	$*$	$*$	$*$	z	z

,

in which the first w columns cannot be separated from the last. If some repetitions occur in the same column, say all repetitions appear in $t < w$ columns, then A is not an SHF of type $\{t, 1\}$, and can therefore not be an SHF of type $\{w, 1\}$ by the contrapositive of Theorem 2.8. We have shown that A is not an SHF of type $\{w, 1\}$, a contradiction. □

We now use Lemma 2.7 to extend the result to an arbitrary number of rows. We will use this technique repeatedly, and will subsequently omit the proofs since they are analogous to the one below.

Corollary 2.21. *In an* SHF$(N; n, m, \{w, 1\})$, $n \leq wm^{\lceil \frac{N}{w} \rceil} - w + 1$.

Proof. Suppose there exists an SHF$(N; n, m, \{w, 1\})$ where $n = wm^{\lceil \frac{N}{w} \rceil} - w + 2$. Then by Lemma 2.7 there exists an SHF$(w; wm' - w, m', \{w, 1\})$ with $m' = m^{\lceil \frac{N}{w} \rceil}$, which contradicts the previous theorem. □

Note that this bound is weaker (by one) than the bound of Staddon, Stinson and Wei [155]. They show that $n \leq wm^{\lceil \frac{N}{w} \rceil} - w$ holds in an SHF$(N; n, m, \{w, 1\})$.

SHF of type $\{5, d\}$

Now we examine necessary conditions for the existence of SHF of type $\{5, d\}$ for $1 \leq d \leq 5$. Previous results were known for types $\{5, 1\}$ and Section 2.2.3 gives bounds for types $\{5, 4\}$ and $\{5, 5\}$, while the bounds for the case $d = 2, 3$ are new. Proofs of the new bounds will illustrate the forbidden configuration that will be used to obtain an upper bound for type $\{w, d\}$ in Section 2.3. The new forbidden configuration will be a combination of the staircase configuration and the configuration used for type $\{w, 1\}$.

Theorem 2.22. *If an* SHF$(7; n, m, \{5, 3\})$ *exists, then* $n \leq m + 24(m - 1)$.

Proof. Suppose A is an SHF$(7; n, m, \{5, 3\})$ with $n = m+1+24(m-1)$. We will construct submatrices $A_6 \subset A_5 \subset \ldots \subset A_1 \subset A$, by deleting certain columns.

We construct A_1 so that elements appearing in the first row appear at least eight times in the first row. The t_i elements appearing exactly i times, for $1 \leq i \leq 7$, will be deleted. It must be that $t_1 + \ldots + t_7 \leq (m - 1)$ since if $t_1 + \ldots + t_7 = m$ then A would have at most $7m$ columns, fewer than the number we assumed. After deleting columns from A, A_1 has at least $m + 1 + 17(m - 1)$ columns.

We now delete columns of A_1 to form A_2, where all elements in row 2 appear at least seven times in row 2. Since $t_1 + \ldots + t_6 \leq (m-1)$ we delete no more than $6(m-1)$ columns so $|A_2| \geq m + 1 + 11(m - 1)$. Similarly A_3 must have elements repeating six or more times in row 3, and $|A_3| \geq m + 1 + 6(m - 1)$ since we delete at most $5(m - 1)$ columns. In A_4 elements must repeat at least five times, by similar reasoning $|A_4| \geq m + 1 + 2(m - 1)$.

In A_5 and A_6 elements appearing in rows 5 and 6 (resp.) must appear at least twice in these rows. Both times we delete no more than $(m - 1)$ columns. Therefore, $|A_5| \geq m + 1 + (m - 1)$ and $|A_6| \geq m + 1$.

Now we will derive a contradiction, showing that A is not an SHF of type $\{5,3\}$. A_6 has at least $m+1$ columns so we can be sure that there is a repeated element in the seventh row. Call this element a, and permute columns of A_6 so that both a appear in the rightmost columns. Let b be the element in row 6 above the rightmost a. There is a second b in row 6, giving two possibilities

$$\begin{array}{|c|c|}\hline b & b \\ \hline a & a \\ \hline \end{array} \quad \text{or} \quad \begin{array}{|c|c|c|}\hline b & * & b \\ \hline * & a & a \\ \hline \end{array}$$

as a submatrix of A_6. Let c be the element in row 5 in the rightmost column. Since c appears at least twice in row 5, we have the following two possibilities,

$$\begin{array}{|c|c|}\hline c & c \\ \hline b & b \\ \hline a & a \\ \hline \end{array} \quad \text{or} \quad \begin{array}{|c|c|c|c|}\hline \star & \star & \star & c \\ \hline * & b & * & b \\ \hline * & * & a & a \\ \hline \end{array}$$

where one of $\star$ is the other c. In both cases, this submatrix is not an SHF of type $\{3,1\}$ since the rightmost column cannot be separated from the other three.

We suppose the widest case occurs and consider what happens in the next row as a submatrix of A_4. All elements in the fourth row appear at least five times (in the fourth row). Therefore A_4 has a submatrix isomorphic to

$$\begin{array}{|c|c|c|c|c|}\hline d & d & * & * & * \\ \hline * & c & * & * & c \\ \hline * & * & b & * & b \\ \hline * & * & * & a & a \\ \hline \end{array}$$

The elements in the 3rd row of A_3 appear at least six times, giving

$$\begin{array}{|c|c|c|c|c|c|}\hline e & e & * & * & * & * \\ \hline * & d & d & * & * & * \\ \hline * & * & c & * & * & c \\ \hline * & * & * & b & * & b \\ \hline * & * & * & * & a & a \\ \hline \end{array}$$

A similar situation exists for A_2 and A_1, and we finally get

$$\begin{array}{|c|c|c|c|c|c|c|c|}\hline g & g & * & * & * & * & * & * \\ \hline * & f & f & * & * & * & * & * \\ \hline * & * & e & e & * & * & * & * \\ \hline * & * & * & d & d & * & * & * \\ \hline * & * & * & * & c & * & * & c \\ \hline * & * & * & * & * & b & * & b \\ \hline * & * & * & * & * & * & a & a \\ \hline \end{array} \tag{2.7}$$

as a submatrix of A. Number the columns from right to left. The sets of columns $\{2,3,4,6,8\}$ and $\{1,5,7\}$ cannot be separated, therefore A is not an SHF of type $\{5,3\}$.

We assumed the widest case would occur for the bottom three rows in columns 2,3,4. If the repeated elements occurred in fewer than three columns, then A would not even be an SHF of type $\{4,3\}$ or $\{3,3\}$, and the proof still holds. □

Again, from the theorem above and Lemma 2.7 we get a bound on n for SHF of type $\{5,3\}$ with N rows.

Corollary 2.23. *If an* SHF$(N;n,m,\{5,3\})$ *exists,* $n \leq 25m^{\lceil \frac{N}{7} \rceil} - 24$.

Note that Theorem 2.22 started by showing that A could not have a submatrix isomorphic to an SHF of type $\{3,1\}$ (in the bottom three rows). Then a forbidden configuration for SHF of type $\{3,2\}$ was added (in the top three rows). We can prove a similar result for type $\{5,2\}$ by combining a forbidden configuration for type $\{4,1\}$ on the bottom, with one for type $\{2,1\}$ on the top.

Theorem 2.24. *If an* SHF$(6;n,m,\{5,2\})$ *exists, then* $n \leq m + 14(m-1)$.

Extending to N rows gives the following corollary.

Corollary 2.25. *If an* SHF$(N;n,m,\{5,2\})$ *exists, then* $n \leq 15m^{\lceil \frac{N}{6} \rceil} - 14$.

Results for type $\{5,d\}$ $1 \leq d \leq 5$ are gathered up in Table 2.1.

Type	Bound	Source
$\{5,1\}$	$n \leq 5m^{\lceil \frac{N}{5} \rceil} - 5$	[29] or [155]
$\{5,2\}$	$n \leq 15m^{\lceil \frac{N}{6} \rceil} - 14$	Corollary 2.25
$\{5,3\}$	$n \leq 25m^{\lceil \frac{N}{7} \rceil} - 24$	Corollary 2.23
$\{5,4\}$	$n \leq 15m^{\lceil \frac{N}{8} \rceil} - 14$	Corollary 2.19
$\{5,5\}$	$n \leq 37m^{\lceil \frac{N}{9} \rceil} - 36$	Corollary 2.17

Table 2.1: Summary of results for SHF of type $\{5,d\}$ where $1 \leq d \leq 5$.

Necessary condition for type $\{w,d\}$

In this section we generalize the method used in Section 2.3 for SHF of type $\{5,2\}$ and $\{5,3\}$. Our main result will be a necessary condition for the existence of SHF$(N;n,m,\{w,d\})$.

Proving Theorem 2.26 is a matter of adapting the proof of Theorem 2.22 to have a variable number of rows. Recall the forbidden configuration in the proof for type $\{5,3\}$ came from combining forbidden configurations for types $\{3,1\}$ and $\{3,2\}$, with one column overlapping. This generalizes as follows: to show the existence of a forbidden configuration for type $\{w,d\}$ (where $d \le w$), combine one for type $\{d,d-1\}$ with one for type $\{w-d+1,1\}$.

Theorem 2.26. *If an* $\mathrm{SHF}(w+d-1;n,m,\{w,d\})$ *exists, then*

$$n \le m + (2dw - w - 1)(m-1).$$

Proof. Let $N_0 = 2d-2$, $N_1 = w-d+1$ and $N = N_0 + N_1 = w+d-1$. Suppose A is an $\mathrm{SHF}(N;n,m,\{w,d\})$ where $n = m+1+(2dw-w-1)(m-1)$. Then

$$|A| = m + 1 + \left(N_1 - 1 + N_0 N - \frac{N_0(N_0-1)}{2}\right)(m-1)$$

Let $K = N_1 - 1 + N_0 N - N_0(N_0-1)/2$, then $|A| = m + 1 + K(m-1)$.

We will create a series of submatrices of A, each of which satisfy one of two properties, as indicated.

$$\underbrace{A_{N-1} \subset \ldots \subset A_{N_0+1}}_{\text{Property (ii)}} \subset \underbrace{A_{N_0} \subset A_{N_0-1} \subset \ldots \subset A_1}_{\text{Property (i)}} \subset A_0 = A$$

Property (i): elements in row i of A_i, $1 \le i \le N_0$, repeat at least $N-(i-2)$ times, and we claim that

$$|A_i| \ge m + 1 + (K - iN + i(i-1)/2)(m-1)\ . \tag{2.8}$$

Property (ii): elements appearing in row N_0+i of A_{N_0+i}, $1 \le i \le N_1$ appear at least twice, and we claim that

$$|A_{N_0+i}| \ge m + 1 + (N_1 - 1 - i)(m-1)\ . \tag{2.9}$$

Similar to the proof of type $\{5,3\}$ we will show that these submatrices lead to a contradiction. The matrices satisfying Property (i) will contribute the top N_0 rows, while those satisfying Property (ii) will add the bottom N_1 rows. The double line in (2.7) shows the division in the $\{5,3\}$ case.

We now prove the claim of Property (i) by induction on i. When creating A_1 from A_0, we want elements that appear in the first row to appear $N+1$ times (in the first row). Let t_i be the number of elements repeating i times in the first row. We must delete at most

$N(m-1)$ columns from A_0, for if it were the case that $t_1+t_2+\ldots+t_N=m$, A would only have Nm columns. Assume (2.8) holds up to A_i. To ensure the elements appearing in row $i+1$ of A_{i+1} appear at least $N-i+1$ times in said row, we must delete $(N-i)(m-1)$ columns of A_i. If $t_1+\ldots+t_{N-i}=m$, then A_i would have $(N-i)m$ columns, fewer than the number assumed in the inductive hypothesis. Then

$$\begin{aligned}|A_{i+1}| &\geq |A_i| - (N-i)(m-1) \\ &= m+1+(K-iN+i(i-1)/2-N+i)(m-1) \\ &= m+1+(K-(i+1)N+i(i+1)/2)(m-1)\ ,\end{aligned}$$

which proves the claim in Property (i).

Since Property (i) holds and $K=N_1-1+N_0N-N_0(N_0-1)/2$, note that

$$\begin{aligned}|A_{N_0}| &\geq m+1+(K-N_0N-N_0(N_0-1)/2)(m-1) \\ &= m+1+(N_1-1)(m-1)\ .\end{aligned}$$

Now we consider the submatrices of Property (ii). When we create A_{N_0+1} we delete columns from A_{N_0} so that elements in row N_0+1 appear at least twice. The number of deleted columns does not exceed $m-1$, since we know A_{N_0} has more than m columns. Therefore $|A_{N_0+1}|\geq |A_{N_0}|-(m-1)$ as required. Assume Property (ii) holds up to i. We want A_{N_0+i+1} to have elements appearing at least twice in row N_0+i+1. Again, we delete no more than $m-1$ columns, since A_{N_0+i} has more than m columns by the inductive hypothesis. Then

$$|A_{N_0+i+1}| = |A_{N_0+i}| - (m-1) = m+1+(N_1-1-(i+1))(m-1)\ ,$$

proving the claim of Property (ii).

In the very last submatrix,

$$\begin{aligned}|A_{N-1}| = |A_{N_0+N_1-1}| &\geq m+1+(N_1-1-(N_1-1))(m-1) \\ &= m+1\ ,\end{aligned}$$

which means the last row of A_{N-1} has a repeated element. We can permute the columns of A_{N-1} so that the two rightmost columns have the same element in row N. Since row $N-2$ of A_{N-2}, row $N-3$ of A_{N-3}, ..., row N_0+1 of A_{N_0+1} all have elements occurring twice, A_{N_0} has a submatrix isomorphic to

$$\begin{array}{|c|c|c|c|c|c|}\hline a & * & * & * & * & a \\ \hline & \ddots & & & & \vdots \\ \hline * & * & x & * & * & x \\ \hline * & * & * & y & * & y \\ \hline * & * & * & * & z & z \\ \hline\end{array}\ ,$$

in the widest case. This pattern covers N_1 rows, and we proceed assuming the widest case, when the above submatrix is $N_1 + 1$ columns wide.

By Property (i), elements that appear in row $N_0 - 1$ of A_{N_0-1}, appear at least $N - (N_0 - 1 - 2) = N_0 + N_1 - N_0 + 3 = N_1 + 3$ times. Then we have

b	b	$*$	$*$	$*$	$*$	$*$
$*$	a	$*$	$*$	$*$	$*$	a
$*$		$\ddots$				$\vdots$
$*$	$*$	$*$	x	$*$	$*$	x
$*$	$*$	$*$	$*$	y	$*$	y
$*$	$*$	$*$	$*$	$*$	z	z

.

Moving up row by row to A_0, Property (i) ensures that each row has elements repeating one more time than in the previous row. The staircase pattern is incrementally extended leftward by one up to A_0, which has a submatrix isomorphic to (numbered from right to left):

		$\ldots$	N_1	$\ldots$	4	3	2	1
c	c	$*$	$*$	$*$	$*$	$*$	$*$	$*$
$*$	$\ddots$	$\ddots$	$*$	$*$	$*$	$*$	$*$	$*$
$*$	$*$	b	b	$*$	$*$	$*$	$*$	$*$
$*$	$*$	$*$	a	$*$	$*$	$*$	$*$	a
$*$	$*$	$*$		$\ddots$				$\vdots$
$*$	$*$	$*$	$*$	$*$	x	$*$	$*$	x
$*$	$*$	$*$	$*$	$*$	$*$	y	$*$	y
$*$	$*$	$*$	$*$	$*$	$*$	$*$	z	z

.

Consider the following partition of the columns of A_0.

$$\mathcal{S}_d = \{1\} \cup \{N_1 + 2k : 0 \leq k, N_1 + 2k < N + 1\}$$
$$\mathcal{S}_w = \{2, \ldots, N_1\} \cup \{N_1 + 2k + 1 : 1 \leq k, N_1 + 2k + 1 < N + 1\}$$

It is easy to see that the first two subsets $\{1\}$ and $\{2, \ldots, N_1\}$ cannot be separated, this pattern formed the contradiction in the proof of Theorem 2.20. Likewise, the columns in the second subsets of $\mathcal{S}_w$ and $\mathcal{S}_d$ are the odd and even indexed columns of a staircase, which also cannot be separated. Therefore A is not an SHF of type $\{|\mathcal{S}_w|, |\mathcal{S}_d|\}$.

It remains to show that $|\mathcal{S}_w| = w$ and $|\mathcal{S}_d| = d$. First we show the second subsets are both of size $d - 1$. Note that the number of columns greater than or equal to N_1 is odd, since the staircase is a $d, d - 1$ forbidden configuration. The column N_1 overlaps the two patterns, and belongs to the first part of $\mathcal{S}_w$ leaving $d - 1$ for each set. The first subset

of $\mathcal{S}_w$ has size $N_1 - 1 = w - d + 1$, giving $|S_w| = w$. The first subset of $\mathcal{S}_d$ has size 1, so $|S_d| = d$.

Thus A is not an SHF of type $\{w, d\}$, contradicting our original assumption and thereby proving the theorem.

The proof is still valid if the repeated elements in the bottom N_1 rows occur in fewer than N_1 columns. The columns $2, \ldots, N_1$ are in $\mathcal{S}_w$, so if the repeated elements occupy t fewer columns, then A is not an SHF of type $\{w - t, d\}$ and therefore it cannot be an SHF of type $\{w, d\}$. □

Extending to N rows (in the usual way) gives the following corollary.

Corollary 2.27. *If an* SHF$(N; n, m, \{w, d\})$ *exists, then*

$$n \leq (2dw - w)m^{\lceil \frac{N}{w+d-1} \rceil} - 2dw + w + 1.$$

Remark 2.28. The bound above applies to all choices of w, d; however it is the strongest bound known only when $2 \leq d \leq w - 2$. For the case $\{w, 1\}$ see Theorem 2.20, for type $\{w, w\}$ see Corollary 2.17 and for type $\{w, w - 1\}$ see Corollary 2.19. For the PHF case, type $\{1, 1\}$, see [29].

2.4 Necessary Conditions for SHF of Type $\{w_1, \ldots, w_t\}$

This section will discuss three proofs (and present one proof) for the following bound on the size of a separating hash family.

Theorem 2.29. *Suppose an* SHF$(N; n, m, \{w_1, w_2, \ldots, w_t\})$ *exists. Define* $u = \sum_{i=1}^{t} w_i$. *Then*

$$n \leq \gamma m^{\lceil N/(u-1) \rceil},$$

where γ *is a constant which depends only on* $w_1, w_2, \ldots, w_t$.

In Blackburn et al. [28] three proofs of this theorem are provided. The first proof shows that we may take $\gamma = \binom{u}{2}$ in the theorem; this proof uses general results and so the argument is quite short. The second proof (Theorem 2.30) extends the forbidden configuration technique of Stinson *et al.* [164] to reduce the value we may take for γ. It is shown that we may take $\gamma = 2(u - w_1)w_1 - w_1$, where (without loss of generality) w_1 is the smallest of the integers w_i. The final proof obtains a significantly better value for γ: we may take $\gamma = w_1 w_2 + u - w_1 - w_2$, where we assume (without loss of generality) that w_1 and w_2 are the smallest two of the integers w_i.

We claim that the exponent $\lceil N/(u-1) \rceil$ in the bound of Theorem 2.29 is realistic, since the exponent in a bound of this form cannot be improved to a value less than $N/(u-1)$. To prove this, first note that a probabilistic construction due to Blackburn [26] shows that an $(N; n, m, u)$ perfect hash family exists provided that

$$N > \frac{\log 4 \left(\binom{n}{u} - \binom{n-u}{u} \right)}{\log m^u - \log \left(m^u - u! \binom{m}{u} \right)}.$$

In particular, this implies that for any fixed u, and any real number δ such that $\delta < N/(u-1)$, there exists an $(N; \lfloor m^\delta \rfloor, m, u)$ perfect hash family whenever m is sufficiently large. Since an $(N; n, m, u)$ perfect hash family is an SHF$(N; n, m, \{w_1, w_2, \ldots, w_t\})$ for any w_i such that $\sum_{i=1}^{t} w_i = u$, we have established our claim.

Proof Using Forbidden Configurations

The bound for type $\{w, d\}$ from Section 2.3 immediately gives the following bound for type $\{w_1, w_2, \ldots, w_t\}$.

Theorem 2.30. *Suppose A is an* SHF$(N, n, m, \{w_1, \ldots, w_t\})$ *where $w_1 \leq w_2 \leq \ldots \leq w_t$. Let $u = \sum_{i=1}^{t} w_i$. Then*

$$n \leq (2w_1 - 1)(u - w_1)m^{\lceil N/(u-1) \rceil} - w_1(2u - 2w_1 + 1) + 1 \tag{2.10}$$

Proof. By an appeal to Theorem 2.8, A is also an SHF of type $\{w_1, \ldots, w_{t-1} + w_t\}$. Repeating this $t-2$ more times, we have that A is also an SHF of type $\{w_1, w_2 + \ldots + w_t\}$. The upper bound on n follows directly from substituting $w = w_1$ and $d = u - w_1$ into Corollary 2.27. □

A remark similar to Remark 2.28 is appropriate here as better bounds may exist when $t = 2$ and certainly do exist for PHF types. Additionally, a stronger bound is known for type $\{1, 1, 2\}$, see [164].

2.5 Existence Results for SHF of type $\{w_1, w_2\}$

In this section, we consider existence results for SHF obtained using the probabilistic method, and we analyze the asymptotic behaviour of the bounds obtained. We prove that for given values of N, m, w_1 and w_2, if n is less than some bound, an SHF$(N; n, m, \{w_1, w_2\})$ exists.

For future reference, we need to recall some basic facts about chromatic polynomials. Let G be a graph. The number of ways of colouring the vertices of G, using m colours, such that no two adjacent vertices receive the same colour, is a polynomial in m, called the *chromatic polynomial* of G and denoted $\chi(G, m)$. The following fundamental results about chromatic polynomials will be useful.

Theorem 2.31. *[140] Let G be a graph on ν vertices and ϵ edges, and let $\chi(G, m)$ be the chromatic polynomial of G. Then the following hold:*

1. *the degree of $\chi(G, m)$ is ν*
2. *the coefficient of m^ν in $\chi(G, m)$ is 1,*
3. *the coefficient of $m^{\nu-1}$ in $\chi(G, m)$ is $-\epsilon$, and*
4. *the coefficients of $\chi(G, m)$ alternate in sign.*

First, we begin by stating a special case of a general bound proven in [164]. For completeness, we will provide a proof as well. The proof technique is a standard method in probabilistic combinatorics known as the "expurgation method" or the "deletion method" (see, for example, [99, Ch. 21]).

Theorem 2.32. *[164, Theorem 4.1] Let w_1 and w_2 be positive integers. Define*

$$c_T = \begin{cases} \frac{1}{w_1! w_2!} & \text{if } w_1 \neq w_2 \\ \frac{1}{2(w_1!)^2} & \text{if } w_1 = w_2. \end{cases} \tag{2.11}$$

Also, define

$$p_T = 1 - \frac{\chi(K_{w_1,w_2}, m)}{m^{w_1+w_2}}, \tag{2.12}$$

where K_{w_1,w_2} is the complete bipartite graph with w_1 vertices in one part and w_2 vertices in the other part. Then there exists an $\mathrm{SHF}(N; n, m, \{w_1, w_2\})$ provided that $n \leq B(w_1, w_2)$, where

$$B(w_1, w_2) = (1 - c_T) \left(\frac{1}{p_T}\right)^{\frac{N}{w_1+w_2-1}}. \tag{2.13}$$

Proof. Suppose A is an $N \times L$ matrix whose entries are elements chosen from a set X of size m. For two disjoint subsets C_1, C_2 of columns of A, where $|C_i| = w_i$, $i = 1, 2$, define a random variable $R_A(C_1, C_2) = 0$ if there is a row in A which separates the sets C_1, C_2, and define $R_A(C_1, C_2) = 1$ otherwise.

Now suppose A is a random matrix (i.e., the entries of A are chosen independently and uniformly at random from X). If we fix a row r of A, then the probability that row r

separates the sets C_1, C_2 is exactly $\chi(K_{w_1,w_2}, m)/m^w$, where $w = w_1 + w_2$ (this is explained in detail in the proof of Theorem 3.35). Therefore, the probability that no row r of A separates the sets C_1, C_2 is exactly

$$\left(1 - \frac{\chi(K_{w_1,w_2}, m)}{m^w}\right)^N.$$

Hence, the expected value of the random variable $R_A(C_1, C_2)$ is easily computed to be

$$\begin{aligned} & E[R_A(C_1, C_2)] \\ &= 0 \times \Pr[R_A(C_1, C_2) = 1] + 1 \times \Pr[R_A(C_1, C_2) = 1] \\ &= \Pr[R_A(C_1, C_2) = 1] \\ &= \left(1 - \frac{\chi(K_{w_1,w_2}, m)}{m^w}\right)^N. \end{aligned}$$

Next, denote

$$p_T = 1 - \frac{\chi(K_{w_1,w_2}, m)}{m^w}$$

and define the random variable

$$R_A = \sum_{C_1,C_2} R_A(C_1, C_2), \tag{2.14}$$

where C_1, C_2 are disjoint subsets of columns of A and $|C_i| = w_i$, $i = 1, 2$.

The number of terms in the sum (2.14) is equal to $\binom{L}{w_1}\binom{L-w_1}{w_2}$ if $w_1 \neq w_2$. If $w_1 = w_2$, then the number of terms in the sum is $\binom{L}{w_1}\binom{L-w_1}{w_2}/2$. It is clear that

$$\binom{L}{w_1} \leq \frac{L^{w_1}}{w_1!}$$

and

$$\binom{L - w_1}{w_2} \leq \frac{L^{w_2}}{w_2!}.$$

Therefore, it follows that the number of terms in the sum (2.14) is at most $c_T\, L^w$, where c_T is defined in equation (2.11). Hence, we have that

$$E[R_A] \leq c_T\, L^w\, {p_T}^N. \tag{2.15}$$

Now, suppose that the following inequality holds:

$$L \leq \left(\frac{1}{p_T}\right)^{\frac{N}{w-1}}.$$

Then it is easy to see that (2.15) implies that $E[R_A] \leq c_T\, L$. In what follows, we will take

$$L = \left(\frac{1}{p_T}\right)^{\frac{N}{w-1}}. \tag{2.16}$$

Since the expected value of R_A is at most $c_T\, L$, this implies that there exists a particular matrix A such that $R_A \leq c_T\, L$. Recall that R_A is computed as a sum of terms in which each term is equal to 0 or 1. Therefore, for this matrix A, there are at most $c_T\, L$ nonzero terms in the sum (2.14).

Now, for each nonzero term $R_A(C_1, C_2)$ in the sum (2.14), delete one column $y \in C_1$ from A. Call the resulting matrix A'. It is clear that A' has at least $(1 - c_T)L$ columns, since we have deleted at most $c_T L$ columns from A to construct A'.

Informally, we are deleting a certain set of columns from A so that the resulting matrix A' has no pairs C_1, C_2 such that $R_{A'}(C_1, C_2) = 1$. This will imply that $R_{A'} = 0$.

We now formally prove that $R_{A'} = 0$. Let Y denote the set of all the columns of A and let Y' denote the set of all the columns of A'. We have that

$$R_{A'} = \sum_{\{\{C_1,C_2\}:C_1 \cup C_2 \subseteq Y'\}} R_{A'}(C_1, C_2), \tag{2.17}$$

where as usual, we require that $C_1 \cap C_2 = \emptyset$, $|C_1| = w_1$, and $|C_2| = w_2$. Suppose that $R_{A'} > 0$. Then there exists a term $R_{A'}(C_1, C_2)$ in the sum (2.17) such that $R_{A'}(C_1, C_2) = 1$. Since $Y' \subseteq Y$ (i.e., all the columns in A' are also in A), it must be the case that $R_A(C_1, C_2) = 1$. However, in constructing A' from A, we deleted a column $y \in C_1$ because $R_A(C_1, C_2) = 1$. Therefore there is a column $y \in C_1$ such that $y \notin Y'$. Therefore $C_1 \cup C_2 \not\subseteq Y'$, which is a contradiction.

We conclude that $R_{A'} = 0$, and therefore A' is an SHF$(N; L', m, \{w_1, w_2\})$ in which $L' \geq (1 - c_T)L$. Substituting (2.16), we have

$$L' \geq (1 - c_T)\left(\frac{1}{p_T}\right)^{\frac{N}{w_1+w_2-1}},$$

as desired. This completes the proof. □

Next, we determine the asymptotic behaviour of the bound $B(w_1, w_2)$. In order to do this, we need to analyze the quantity $1/p_T$. This can be done by using the properties of chromatic polynomials enumerated in Theorem 2.31.

Corollary 2.33. *For positive integers w_1 and w_2, as $m \to \infty$ it holds that*

$$B(w_1, w_2) = (1 - c_T) \left(\frac{m}{w_1 w_2} (1 + o(1)) \right)^{\frac{N}{w_1 + w_2 - 1}},$$

where c_T is the constant defined in (2.11).

Proof. Applying the results listed in Theorem 2.31, we have that

$$\begin{aligned} p_T &= 1 - \frac{\chi(K_{w_1,w_2}, m)}{m^{w_1+w_2}} \\ &= 1 - \frac{m^{w_1+w_2} - w_1 w_2 m^{w_1+w_2-1} + g(m)}{m^{w_1+w_2}} \\ &= \frac{w_1 w_2}{m} - \frac{g(m)}{m^{w_1+w_2}}, \end{aligned} \tag{2.18}$$

where $g(m)$ is a polynomial of degree not exceeding $w_1 + w_2 - 2$, whose leading coefficient is positive.

Now, if we write $1/p_T$ in the form

$$\frac{1}{p_T} = \frac{m}{w_1 w_2}(1 + \Delta), \tag{2.19}$$

then

$$\Delta = \frac{w_1 w_2}{m\, p_T} - 1. \tag{2.20}$$

Substituting (2.18) into (2.20), we obtain

$$\begin{aligned} \Delta &= \frac{w_1 w_2}{m \left(\frac{w_1 w_2}{m} - \frac{g(m)}{m^{w_1+w_2}} \right)} - 1 \\ &= \frac{1}{1 - \frac{g(m)}{w_1 w_2\, m^{w_1+w_2-1}}} - 1. \end{aligned}$$

Since $g(m)$ has degree at most $w_1 + w_2 - 2$, it follows that

$$\lim_{m \to \infty} \frac{g(m)}{w_1 w_2\, m^{w_1+w_2-1}} = 0,$$

and hence $\lim_{m\to\infty} \Delta = 0$. Therefore, we can rewrite (2.19) in the form

$$\frac{1}{p_T} = \frac{m}{w_1 w_2}(1 + o(1)),$$

and the desired result follows from (2.13). □

The bound $B(w_1, w_2)$, which guarantees existence of the relevant SHF, has the same exponent as the necessary conditions proven in Sections 2.2 and 2.3 (except for the ceiling function). In this sense our bounds can be considered relatively tight. Also, we note that the first term of $B(w_1, w_2)$ quickly approaches one as w_1 and w_2 increase. For instance, when $w_1 = 5$ and $w_2 = 6$, $(1 - c_t) \approx 1 - 2^{-16}$.

Chapter 3

Constructions

In this chapter, we consider ways of constructing SHF of type $T = \{w_1, \ldots, w_t\}$, by generalizing existing methods which apply to special types. Section 3.1 will focus on generalizing constructions from three main sources.

- Stinson et al. [165], which constructs PHF and SHF of type $\{w_1, w_2\}$ using combinatorial methods and codes with large distance.
- Walker and Colbourn [179], which surveys existing constructions of PHF and provides new ones. We also consider constructions from the paper of Martirosyan and van Trung [118], which extends [179].
- Wang and Xing [181] and Liu and Shen [109], which give constructions for PHF, SHF of type $\{w_1, w_2\}$ and SSHF using a construction based on algebraic curves, similar to the construction of algebraic geometry codes (AG codes) [31].

Walker and Colbourn [179], compare 25 constructions for PHF. Essentially, the metric used for comparison is a count of how often, for fixed n, m and strength, a construction gives a PHF with the smallest N (arrays used as input to recursive constructions which produce the PHF with the smallest N are also counted). Unfortunately, in their comparison the AG-based construction was not included.

Another question related to constructions of PHF and SHF is the asymptotic relationship between N and n when m and T are fixed. AG-based constructions of PHF, SHF and SSHF are asymptotically optimal, since they have $N = O(\log n)$. Liu and Shen [109] prove the following.

Theorem 3.1 ([109, Theorem 3.3, 3.7])**.** *For any positive integers m, w_1, w_2, there exists an infinite class of explicitly constructed SSHF of type $\{w_1, w_2\}$ and SHF of type $\{w_1, w_2\}$ for which N is $O(\log n)$.*

Remark 3.2. The constant hidden by the big-O notation in Theorem 3.1 is $w_1 w_2$ for SHF and $\binom{w_1}{2} + w_1 w_2$ for SSHF.

We will extend these results to SHF of the most general type $\{w_1, \ldots, w_t\}$. Since the result is asymptotic and relies on the existence of curves with certain properties, we take a detailed look at the concrete parameters possible with the AG construction and provide an implementation (§3.3). We also define metrics we use to describe and compare the possible variants of the AG construction (obtained by using different curves).

With our implementation it is also possible to use hash families which separate inputs of a specified type with known probability (chosen as needed). The number of rows in the hash family can be significantly reduced, provided the application can tolerate occasional errors (i.e., failing to separate an input). Finally, we analyze the behaviour of random hash families, since they are simple to construct and implement.

3.1 Constructions

In this section we describe constructions of SHF of arbitrary type. We will use T to denote the type of an SHF, which will be $\{w_1, \ldots, w_t\}$ unless specified otherwise.

The next definition is a quantity associated with the strength of an SHF, and will appear in many constructions.

Definition 3.3. For the multiset $T = \{w_1, \ldots, w_t\}$, define

$$P(T) = \sum_{i=1}^{t} \sum_{j=i+1}^{t} w_i w_j$$

(i.e., the sum of the products of all unordered pairs w_i, w_j, $i \neq j$).

It should be noted that for the PHF type, when $T = \{w_1 = 1, \ldots, w_t = 1\}$, $P(T) = \binom{t}{2}$.

3.1.1 Constructions From Codes with Large Distance

One approach to constructing hash families is to show that a code with large distance is also a hash family of the desired type. If the code is constructed explicitly, it immediately yields an explicit construction for the hash family as well. Alon [3] first used this idea to construct PHF, then Sarkar and Stinson [147] gave a condition for SSHF of type $\{w_1, w_2\}$ to be constructed in this way. Shortly after, Cohen et al. [57] gave the distance condition for a code to be an SHF of type $\{w_1, \ldots, w_t\}$.

Theorem 3.4 ([57, Prop. 7])**.** *Let C be an $(N; n, m, D)$-code with minimum distance*

$$D > N\left(1 - \frac{1}{P(T)}\right) . \tag{3.1}$$

Then C is an $\mathrm{SHF}(N; n, m, \{w_1, \ldots, w_t\})$.

Proof. Let $C_1, \ldots, C_t$ be disjoint sets of C, with $|C_i| = w_i$. Let

$$S_{i,j} = \sum_{\substack{x \in C_i, \\ y \in C_j}} \mathrm{dist}(x, y) .$$

Since $\mathrm{dist}(x, y) \geq D$ for any distinct $x, y \in C$, we have $S_{i,j} \geq w_i w_j D$. Further

$$S = \sum_{i<j} S_{i,j} \geq D\left(P(T)\right)$$

and

$$S > N(P(T) - 1) \tag{3.2}$$

by (3.1). Now suppose that C is not an SHF of type $\{w_1, \ldots, w_t\}$. Then the maximum contribution of each row (of the matrix representation of C) to S is $P(T) - 1$. This implies that $S \leq N\left(P(T) - 1\right)$, which contradicts (3.2). Therefore C is an SHF of type $\{w_1, \ldots, w_t\}$. □

Remark 3.5. The above is a sufficient condition, but not a necessary one. We will see an SHF construction where this does not hold (see Remark 3.31).

The next theorem is new, and our first application of the distance condition. This construction is inspired by the construction of Stinson, Wei and Zhu for SHF of type $\{w_1, w_2\}$ [165, Corollary 2.4].

Theorem 3.6. *Let q be a prime power, and ℓ be an integer such that $2 \leq \ell < q$. Then there exists an* $\mathrm{SHF}(q; q^\ell, q, \{w_1, \ldots, w_t\})$*, provided $P(T) < \frac{q}{\ell - 1}$.*

Proof. We can construct a $(q; q^\ell, q, q - (\ell - 1))$ Reed-Solomon code (see [177, §6.8]). In order to apply Theorem 3.4, the distance $q - (\ell - 1)$ must satisfy

$$q - (\ell - 1) > q\left(1 - \frac{1}{P(T)}\right)$$

which can be expressed as a condition on $P(T)$,

$$\frac{\ell-1}{q} < \frac{1}{P(T)}$$
$$P(T) < \frac{q}{\ell-1}\,.$$

Applying Theorem 3.4, the Reed-Solomon code is an SHF$(q; q^\ell, q, \{w_1, \ldots, w_t\})$, as required. □

Note that the case $\ell = 1$ is trivial since $n = m$.

Example 3.7. For type $T = \{2, 3, 4\}$, $P(T) = 26$. By setting $q = 107, \ell = 5$, we get $q/(\ell-1) = 26.75$ and we can therefore construct an SHF$(107; 107^5, 107, \{2, 3, 4\})$ with Theorem 3.6.

The following recursive construction allows us to decrease the alphabet size in an SHF at the expense of an increased number of rows. The next theorem is similar to a theorem of Blackburn for PHF [25], and was first given for the general case by Cohen et al. [57].

Lemma 3.8. *Let $T = \{w_1, \ldots, w_t\}$. Suppose there exists an* SHF$(N_0; n, v, T)$ *and an* SHF$(N_1; v, m, T)$. *Then there exists an* SHF$(N_0N_1; n, m, T)$.

Proof. Denote the SHF$(N_0; n, v, T)$ by A, the SHF$(N_1; v, m, T)$ by B and the SHF$(N_0N_1; n, m, T)$ by AB. We write $\mathcal{F}_A$ and $\mathcal{F}_B$ for the functions in A and B, respectively. Let $C_1, \ldots, C_t \subset X$, $|C_i| = w_i$, be any input sets to AB. Since A is an SHF of type T, there exists some $\phi \in \mathcal{F}_A$ such that $\bigcap_{i=1}^{t} \phi(C_i) = \emptyset$. Each set C_i yields a multiset C_i' of elements from the alphabet of size v, with $|C_i'| = w_i$. We now use the sets C_i' as input to B, where we have $\psi \in \mathcal{F}_B$, which separates C_i', since B is also an SHF of type T.

Thus, $\mathcal{F}_{AB}$ is defined as the N_0N_1 composed functions $\psi \circ \phi$ where $\phi \in \mathcal{F}_A$ and $\psi \in \mathcal{F}_B$. □

We now present a recursive construction for creating SHF of type $\{w_1, \ldots, w_t\}$ with large domain. This construction is a generalization of a similar construction for PHF from Stinson et al. [165, Theorem 3.6].

Theorem 3.9. *Suppose there exists an* SHF$(N_0; q^{l_0}, m, T)$ *where q is a prime power and $q^{l_0} > P(T)$. Then for all $h \geq 1$, there exists an* SHF$(N_0P_h; q^{l_h}, m, T)$, *where $P_0 = 1$ and*

$$P_h = q^{l_{h-1}} P_{h-1}, \quad \textit{and}$$
$$l_h = l_{h-1} \left\lfloor \frac{q^{l_{h-1}}}{P(T)} \right\rfloor\,.$$

Proof. We proceed by induction on h. The case $h = 0$ is true by assumption, and we assume the theorem holds up to $h-1$. To prove the inductive step, we can use Theorem 3.6 with q replaced by $q^{l_{h-1}}$ and $l_h = \left\lfloor q^{l_{h-1}}/P(T) \right\rfloor$ to construct an $\mathrm{SHF}(q^{l_{h-1}}; q^{l_h}, q^{l_{h-1}}, T)$ (call it A). The required conditions $P(T) < q^{l_{h-1}}/(r-1)$ are satisfied since $q^{l_{h-1}} > q^{l_0}$ and $t > 2$. By our inductive hypothesis, an $\mathrm{SHF}(N_0 P_{h-1}; q^{l_{h-1}}, m, T)$ exists (call it B). Applying the product theorem to A and B (Lemma 3.8), an $\mathrm{SHF}(N_0 P_h; q^{l_h}, m, T)$ also exists. □

The next theorem generalizes a theorem of Stinson et al. [165, Theorem 3.7] for PHF, and holds with virtually the same proof (the original proof is for PHF of strength t, and remains valid if we replace $\binom{t}{2}$ with $P(T)$).

Theorem 3.10. *Let* $\log^*(1) = 1$, $\log^*(n) = \log^*(\lceil \log n \rceil) + 1$ *if* $n > 1$. *For any positive integers* m *and* $T = \{w_1, \ldots, w_t\}$ *there exists an infinite class of* $\mathrm{SHF}(N; n, m, \{w_1, \ldots w_t\})$ *such that* $N = O(P(T)^{\log^* n} \log n)$.

Constructing CFF From Codes with Large Distance

This construction is due to Kautz and Singleton [104]. They prove a lemma that gives conditions under which a constant weight binary code is the incidence matrix of a d-CFF $(X, \mathcal{B})$. Here, each codeword will define a column of the incidence matrix, each row corresponds to one point $x \in X$, and each column defines a set $B \in \mathcal{B}$. The codewords of a constant weight code all have a fixed number of nonzero coordinates. Now suppose we have an $(N; n, q, \ell)$ code, i.e., n codewords of length N over q symbols with minimum distance ℓ. We must replace the q-ary alphabet with a binary one in order to construct an incidence matrix. Let $\varphi : \{1, \ldots, q\} \to \{0,1\}^{q_0}$ be a map which encodes $1, \ldots, q$ as binary vectors of length q_0. The codeword $(c_1, \ldots, c_N)$ will be replaced by $(\varphi(c_1), \ldots, \varphi(c_N))$. Different choices of φ are possible, so long as the encoding $\varphi(1), \ldots, \varphi(q)$ forms a d-$\mathrm{CFF}(q_0, q)$. We refer the reader to Kautz and Singleton [104] and Nguyen and Zeisel [128] for the details of such recursive constructions.

A simple choice of φ encodes i as the i-th column of the $q \times q$ identity matrix. The resulting code has constant weight, and gives a d-$\mathrm{CFF}(qN, n)$ if $d < \frac{N-1}{N-\ell}$. This gives the following explicit construction.

Theorem 3.11 ([104])**.** *Let* q *be a prime power,* $2 \le k < q$ *be an integer and let* C *be a* $(q; q^k, n, q-(k-1))$ *Reed-Solomon code (see [177, §6.8]). Then* C *is* d-$\mathrm{CFF}(q^2, n)$ *if*

$$d < \frac{q-1}{k-1}\,.$$

The CFF is uniform if the input code, and the encoding φ are constant weight. Table 3.1 gives sample parameters produced by this construction.

q	m	k	n	d
8	64	3	512	3
		4	4096	2
16	256	3	4096	7
		4	65536	5
		5	1048576	3
27	729	3	19683	13
		4	531441	8
		5	14348907	6

Table 3.1: Sample parameters for d-CFF(m, n) constructed using Theorem 3.11.

3.1.2 Direct and Recursive Constructions

Column Increase

In the work of Walker and Colbourn [179], a construction called *column increase* is presented. This construction does well in the ranking they present; it produces PHF$(N; n, m, w)$ with the lowest known value of N for many n and m. The column increase construction combines two arrays to produce a new array with an additional column and one or more additional row(s), but no additional symbols. Let SHFN(n, m, T) denote the smallest N for which an SHF$(N; n, m, T)$ is known to exist (define PHFN analogously).

Theorem 3.12 ([179, Theorem 4.13])**.** *For $w \geq 3$,*

$$\text{PHFN}(n+1, m, w) \leq \text{PHFN}(n, m, w) + \text{PHFN}(n-1, m-2, w-2) \ .$$

Specifically, Theorem 3.12 takes a PHF$(N_A; n, m, w)$ and a PHF$(N_B; n-1, m-2, w-2)$ as input, and outputs a PHF$(N_A + N_B; n+1, m, w)$.

We give a generalization of the column increase construction to SHF of type $\{w_1, w_2\}$.

Theorem 3.13.

$$\begin{aligned}\text{SHFN}(n+1, m, \{w_1, w_2\}) \leq\ & \text{SHFN}(n, m, \{w_1, w_2\}) + \\ & \text{SHFN}(n-1, m-2, \{w_1 - 1, w_2 - 1\})\end{aligned}$$

Proof. Let A and B be input, and C the output.

- A is an SHF$(N_A; n, m, \{w_1, w_2\})$,

- B is an SHF($N_B; n-1, m-2, \{w_1-1, w_2-1\}$), and
- C is an SHF($N_A + N_B; n+1, m, \{w_1, w_2\}$).

C will have the following form:

	w	w
	x	x
A	$\vdots$	$\vdots$
	y	y
	z	z
	v	v'
	v	v'
B	$\vdots$	$\vdots$
	v	v'
	v	v'

A and B are vertically juxtaposed, the last column of A is repeated and B is extended by two columns consisting of v and v', the two symbols appearing in A which do not appear in B. We now prove that C is an SHF of type $\{w_1, w_2\}$. Consider column sets W_1 and W_2 of C with $|W_1| = w_1$ and $|W_2| = w_2$. We refer to the n-th column of C as ℓ and the $(n+1)$-st as ℓ'. We now consider four cases, depending on which sets the columns ℓ, ℓ' belong to. In the three cases,

- neither of ℓ nor ℓ' is in W_1 or W_2, i.e., $\ell, \ell' \notin (W_1 \cup W_2)$,
- one of ℓ, ℓ' is in a set, i.e., (wlog) $\ell \in W_1$, $\ell' \notin W_2$, and,
- both ℓ and ℓ' are in the same set, i.e., $\{\ell, \ell'\} \in W_1$, or $\{\ell, \ell'\} \in W_2$,

the submatrix A will contain a row separating W_1, W_2.

In the remaining case when (wlog) $\ell \in W_1$ and $\ell' \in W_2$, B will separate $W_1 - \ell$ from $W_2 - \ell'$ since $|W_1 - \ell| = w_1 - 1$ and $|W_2 - \ell'| = w_2 - 1$. Also, by construction ℓ and ℓ' are separate from each other and from all other columns of B. Hence, C is an SHF of type $\{w_1, w_2\}$. □

We can also generalize Theorem 3.12 to SHF of arbitrary type. Unfortunately, in the generalization the second ingredient is not of reduced strength (and hence the number of rows is not reduced).

Theorem 3.14.

$$\text{SHFN}(n+1, m, T) \leq \text{SHFN}(n, m, T) + \text{SHFN}(n-1, m-2, T) \ .$$

Proof. We use the same construction and notation as in Theorem 3.13 except the SHF A, B and C are all of type T, and we consider the same cases for ℓ and ℓ'. If neither belong to $W_1, \ldots, W_t$, one belongs to one of $W_1, \ldots, W_t$, or both are in one of $W_1, \ldots, W_t$, then A will separate $W_1, \ldots, W_t$. In the remaining case when (wlog) $\ell \in W_i$ and $\ell' \in W_j$, B will separate $W_i - \ell$ from $W_j - \ell'$ since B is an SHF of type T and by construction ℓ and ℓ' are separate from each other and the other columns of B. Hence, C is an SHF of type $\{w_1, \ldots, w_t\}$. □

The column increase construction was later generalized to the "columns increase" construction by Martirosyan and van Trung [118].

Theorem 3.15 ([118])**.** *For $w \geq 3$, and for any integer $2 \leq x \leq m - w + 2$,*

$$\mathrm{PHFN}(n + x - 1, m, w) \leq \mathrm{PHFN}(n, m, w) + \mathrm{PHFN}(n - 1, m - x, w - 2) .$$

We now give the corresponding columns increase construction for separating hash families.

Theorem 3.16. *Suppose an $\mathrm{SHF}(N; n, m, T)$ and an $\mathrm{SHF}(N; n, m-x, T)$ exist with $x \geq 2$. Then*

$$\mathrm{SHFN}(n + x, m, T) \leq \mathrm{SHFN}(n, m, T) + \mathrm{SHFN}(n, m - x, T) .$$

Proof. (Sketch). We use the same construction and notation as in Theorems 3.14 and 3.13. After vertically juxtaposing A and B, we duplicate the last column of A x times, and extend B by x columns, each containing one of the symbols of A not used in B. By a similar argument as above, any inputs will be separated by either a row of A or a row of B. □

As was the case for column increase, it should be possible to improve the columns increase construction for certain values of x and T.

A Direct Construction

In this section we generalize another construction for perfect hash families of Walker and Colbourn [179] which they name the "First-N construction". The construction is direct, i.e., it does not require smaller PHF(s) as input.

Theorem 3.17 ([179])**.** *For $a \geq 1$ and $b \geq 2$, there exists a $\mathrm{PHF}(a+1; ba+b, ba+1, 2a+1)$.*

We show that this construction can be used to create SSHF in the following theorem.

Theorem 3.18. *For $a \geq 1$, $b \geq 2$, there exists an* SSHF$(a+1; b(a+1), ba+1, \{w_1, w_2\})$*, provided $w_1 < a+1$ and $w_2 \leq b(a+1) - w_1$.*

Proof. Our array will have $a+1$ rows, and $b(a+1)$ columns, partitioned into $a+1$ groups of size b (pictured below). Fill the j-th partition of the j-th row with the symbol v, where $v = ba+1$. For the remaining positions of row j place the other ba symbols, once each (omitted from the diagram). The j-th partition of the j-th row will be called the *primary partition.*

v	v	$\ldots$	v												
				v	v	$\ldots$	v								
								$\ddots$	$\ddots$	$\ddots$	$\ddots$				
												v	v	$\ldots$	v

We first show that this array is an SHF of type $\{w_1, w_2\}$. Take any disjoint column sets C_i, where $|C_i| = w_i$ for $i = 1, 2$. Since $w_1 < a+1$, and there are $a+1$ partitions, there exists a primary partition which contains no column from C_1, call this partition p. We claim row p separates C_1 and C_2.

First, note that all columns not in the primary partition of row p are distinct (in row p). All the repeated elements in row p appear in the primary partition which does not have columns belonging to C_1. Therefore, repeated elements of row p can only occur in columns of C_2 and we have constructed an SHF of type $\{w_1, w_2\}$. Now since the columns of C_1 are distinct, the above construction is also an SSHF of type $\{w_1, w_2\}$ as required. □

We make a few remarks about Theorem 3.18.

Remark 3.19.

1. Note that this construction also yields an SHF of type $\{w'_1, \ldots, w'_t\}$ if $w_1 = w'_1 + \ldots + w'_{t-1}$ and $w_2 = w'_t$.
2. The condition on w_2 serves only to ensure that there are enough columns; once C_1 is chosen, C_2 can contain all $b(a+1) - w_1$ remaining columns.
3. The output of this construction may serve as a useful input to a recursive construction which increases n but not m, to increase the compression of this SHF since $n - m$ is not very large.

3.1.3 Algebraic Geometry Background

We now give some background on algebraic curves and functions fields required for the construction we will present in Section 3.1.4. Further details are available in [31, 156, 175].

We will specialize our presentation by defining all objects over the finite field $\mathbb{F}_q$ where q is a prime power.

Let $\mathcal{C}/\mathbb{F}_q$ be a plane affine curve over $\mathbb{F}_q$ of genus g. We use C to denote the irreducible polynomial in $\mathbb{F}_q[x, y]$ which defines $\mathcal{C}$. Let $\mathcal{C}(\mathbb{F}_q)$ be the set of points on $\mathcal{C}$ with coordinates in $\mathbb{F}_q$, i.e., $\mathcal{C}(\mathbb{F}_q) = \big\{(x, y) \in \mathbb{F}_q^2 : C(x, y) = 0\big\}$. We will use $\mathcal{C}$ as shorthand for $\mathcal{C}(\mathbb{F}_q)$. The *coordinate ring* of $\mathcal{C}$ is $\mathbb{F}_q[\mathcal{C}] = \mathbb{F}_q[x, y]/\langle C\rangle$, where $\langle C\rangle$ is the ideal generated by C. The coordinate ring equates polynomials in $\mathbb{F}_q[x, y]$ that differ by a multiple of the equation of the curve. The *function field* associated to $\mathcal{C}$ is the set of rational functions on $\mathcal{C}$:

$$\mathbb{F}_q(\mathcal{C}) = \left\{ \frac{a(x, y)}{b(x, y)} : a, b \in \mathbb{F}_q[\mathcal{C}] \right\} .$$

For example, let $\mathcal{C}$ be the elliptic curve defined by $C(x, y) = y^2 - x^3 - ax - b$. The coordinate ring is $\mathbb{F}_q[\mathcal{C}] = \mathbb{F}_q[x, y]/\langle y^2 - x^3 - ax - b\rangle$, and $\mathbb{F}_q(\mathcal{C})$ is the quotient field of $\mathbb{F}_q[\mathcal{C}]$.

For a function $r \in \mathbb{F}_q(\mathcal{C})$ and a point $P \in \mathcal{C}$, we say that P is a *zero* of r if $r(P) = 0$. If all representations of r as a/b for $a, b \in \mathbb{F}_q[\mathcal{C}]$ have $b(P) = 0$ then we say that P is a *pole* of r and write $r(P) = \mathcal{O}$, where $\mathcal{O}$ is an idealized point (called the *point at infinity*).

Now we define the order at a point of a function in $\mathbb{F}_q(\mathcal{C})$. For a point $P \in \mathcal{C}$, a *uniformizing parameter* is a function $z \in \mathbb{F}_q(\mathcal{C})$ such that i) $z(P) = 0$ and ii) for each nonzero polynomial function $f \in \mathbb{F}_q(\mathcal{C})$, there exists an integer d and another function $s \in \mathbb{F}_q(\mathcal{C})$ such that $s(P) \neq 0$ and $f = z^d s$. The *order of f at P* is defined to be $\mathrm{ord}_P(f) = d$. Further, at least one such a function z exists for each point P, and d is independent of the choice of z.

Definition 3.20. A *divisor* of $\mathcal{C}$ is a formal sum

$$D = \sum_{P \in \mathcal{C}} m_P P , \quad m_P \in \mathbb{Z} ,$$

where only finitely many m_P are nonzero. The *degree* of D, denoted $\deg D$ is $\sum_{P \in \mathcal{C}} m_P$. A divisor is called *positive* if all m_P are non-negative. We can compute the sum of two divisors as

$$D_1 + D_2 = \sum_{P \in \mathcal{C}} m_P P + \sum_{P \in \mathcal{C}} n_P P = \sum_{P \in \mathcal{C}} (m_P + n_P) P .$$

Under this operation, the set of all divisors forms a free abelian group, denoted $Div(\mathcal{C})$.

Definition 3.21. Let $D \in Div(\mathcal{C})$ be as defined in Definition 3.20. The *support of a divisor* is the set of points with nonzero m_P, i.e.

$$\mathrm{Supp}(D) = \{P \in \mathcal{C} \mid m_P \neq 0\} .$$

To each function $x \in \mathbb{F}_q(\mathcal{C})$ we can associate three divisors:

1. the *zero divisor*
$$(x)_0 = \sum_{P \in \text{ zeroes}(x)} \text{ord}_P(x)P \ ,$$
2. the *pole divisor*
$$(x)_\infty = \sum_{P \in \text{ poles}(x)} \text{ord}_P(x)P \ ,$$
3. and the *principal divisor* $(x) = (x)_0 - (x)_\infty$.

The object in the next definition will be the basic building block for constructing hash families.

Definition 3.22. Let $\mathcal{C}/\mathbb{F}_q$ be an algebraic curve and G be a divisor of $Div(\mathcal{C})$. The *Riemann-Roch space of* G is defined

$$\mathcal{L}(G) = \{x \in \mathbb{F}_q(\mathcal{C}) \mid (x) + G \text{ is positive}\} \cup \{0\} \ .$$

The Reimann-Roch space $\mathcal{L}(G) \subset \mathbb{F}_q(\mathcal{C})$ of $\mathcal{C}$ is a finite dimensional vector space over $\mathbb{F}_q$. We write $l(G)$ to denote the dimension of $\mathcal{L}(G)$. As a consequence of the Riemann-Roch theorem,

$$l(G) \geq \deg(G) + 1 - g \ ,$$

which holds with equality when $\deg(G) \geq 2g - 1$ [30]. Further, elements of $\mathcal{L}(G)$ can have at most $\deg(G)$ zeroes [109, 181]. The SHF we will construct will hash from the domain $\mathcal{L}(G)$ to the co-domain $\mathbb{F}_q$, therefore, $n = q^{l(G)}$ and $m = q$.

A well studied class of algebraic geometry codes, introduced by Goppa, are the geometric Goppa codes [90]. The SHF constructed by Theorem 3.28 are a special type of Goppa code.

Definition 3.23. A *geometric Goppa code* is defined for two divisors A, G with disjoint supports, and $|\text{Supp}(A)| = N$, as

$$\mathcal{C}_\mathcal{L}(A, G) = \{x(P_1), \ldots, x(P_N) \mid x \in \mathcal{L}(G), P_i \in \text{Supp}(A)\} \ .$$

It is well known that the minimum distance of $\mathcal{C}_\mathcal{L}(A, G)$ satisfies $D \geq N - \deg(G)$ [156, II.2.2]. For further details of AG codes, see [31, 156].

3.1.4 Constructions From Algebraic Geometry

A recent paper of Liu and Shen [109] gives constructions for strong separating hash families. We first give a simple lemma relating SSHF and SHF.

Lemma 3.24. *Let* $v = w_1 + w_2 + \ldots + w_{t-1}$. *Then an* $\mathrm{SSHF}(N; n, m, \{v, w_t\})$ *is an* $\mathrm{SHF}(N; n, m, \{w_1, w_2, \ldots, w_t\})$.

Proof. By definition of an SSHF, an SSHF of type $\{v, w_t\}$ is an SHF of type $\{1^v, w_t\}$. Since the first v inputs are always distinct, they can be arranged into groups of size $w_1, \ldots, w_{t-1}$ provided their sum is v. □

From this lemma, constructions for SSHF immediately yield constructions for more general SHF. It follows that explicit constructions with asymptotically good parameters exist for SHF of type $\{w_1, \ldots, w_t\}$.

Theorem 3.25. *For any positive integers* $m, w_1, \ldots, w_t$, *there exists an infinite class of explicitly constructed* $\mathrm{SHF}(N; n, m, \{w_1, \ldots, w_t\})$ *for which* N *is* $O(\log n)$.

Proof. Use Lemma 3.24 to extend Theorem 3.1 from SSHF to general SHF. □

In Section 3.1.4 we will generalize the construction of of Liu and Shen for SHF of type $\{w_1, w_2\}$ to SHF of type $\{w_1, \ldots, w_t\}$ which will require fewer rows than using their SSHF construction (Theorem 3.1) with Lemma 3.24. This improvement is possible since $P(\{w_1, \ldots, w_t\}) \leq P(\{1^{w_1+w_2+\ldots+w_{t-1}}, w_t\})$.

Construction for SHF of type $\{w_1, \ldots, w_t\}$

Before presenting the construction we need some lemmas. First, it is possible to construct a divisor of arbitrary degree whose support is disjoint from a set of rational points.

Lemma 3.26 ([156])**.** *Let* S *be a subset of* $\mathcal{C}(\mathbb{F}_q) \setminus \{\mathcal{O}\}$. *Then for any* $t \geq 0$, *there exists a divisor* G *such that* $\deg(G) = t$ *and* $S \cap \mathrm{Supp}(G) = \emptyset$.

This can easily be seen by setting $G = t(\mathcal{O})$. Now we define the set of functions $\mathcal{F}$ which will be used in our construction.

Lemma 3.27 ([181])**.** *For every point* P *in* $\mathcal{C}(\mathbb{F}_q)$, *define the map* $h_P : \mathcal{L}(G) \to \mathbb{F}_q$ *by* $h_P(f) = f(P)$. *Let* $S \subset \mathcal{C}(\mathbb{F}_q)$ *and* $\mathcal{F} = \{h_P \mid P \in S\}$. *Then* $|\mathcal{F}| = |S|$.

The function h_P is sometimes called the *residue class map*. The next theorem is the main result of this sub-section. It is a generalization of [109, Theorem 3.1] which was for SHF of type $\{w_1, w_2\}$.

Theorem 3.28. *Let $\mathcal{C}/\mathbb{F}_q$ be an algebraic curve of genus g, and S a set of $\mathbb{F}_q$-rational points of $\mathcal{C}$. Suppose that G is a divisor with $\deg(G) \geq 2g+1$, and $S \cap \mathrm{Supp}(G) = \emptyset$. Then there exists an*

$$\mathrm{SHF}(|S|; q^{\deg(G)-g+1}, q, \{w_1, \ldots, w_t\})$$

if $|S| > \deg(G) \cdot P(T)$ where $P(T) = \sum_{i=1}^{t} \sum_{j=i+1}^{t} w_i w_j$.

Proof. Let $\mathcal{F} = \{h_P \mid P \in S\}$. Each $f \in \mathcal{F}$ will define a row of the matrix representation of the SHF. Recall from Lemma 3.27 that $|\mathcal{F}| = |S|$. The domain of $\mathcal{F}$ will be $\mathcal{L}(G)$ which is a vector space over $\mathbb{F}_q$ of dimension $\deg(G) - g + 1$, since $\deg(G) \geq 2g + 1$. For any disjoint subsets $C_i, \ldots, C_t$ of $\mathcal{L}(G)$ (the columns) with $|C_i| = w_i$, we consider the sets of differences

$$\Phi_{i,j} = \{u - v \mid u \in C_i, v \in C_j\} \ .$$

The set $\Phi_{i,j}$ will have at most $w_i w_j$ elements, each of which can have up to $\deg(G) \leq 2g+1$ zeroes. Therefore, the total number of zeroes for all elements in $\Phi_{i,j}$ is at most $\deg(G) \cdot w_i w_j$. We now take the union over all unordered pairs (i, j),

$$\Phi_{1,\ldots,t} = \bigcup_{i=1}^{t} \bigcup_{j=i+1}^{t} \Phi_{i,j}$$

for which

$$|\Phi_{1,\ldots,t}| \leq \sum_{i=1}^{t} \sum_{j=i+1}^{t} |\Phi_{i,j}| = P(T) \ .$$

The total number of zeroes for the elements of $\Phi_{1,\ldots,t}$ is at most $\deg(G) \cdot P(T)$. Since we have $|S| > \deg(G) \cdot P(T)$, there must exist a point $R \in S$, such that R is not a zero of any function of $\Phi_{1,\ldots,t}$ (multiple such points may exist, see Remark 3.30).

We now show that h_R separates the inputs, by showing that $h_R(C_i) \cap h_R(C_j) = \emptyset$ for all $i \neq j$. Consider $u \in C_i$, $v \in C_j$. R is not a zero of $u - v$, since $u - v \in \Phi_{i,j} \subset \Phi_{1,\ldots,t}$, and R was chosen not to be a zero of any element in $\Phi_{1,\ldots,t}$. That R is not a zero of $u - v$ is equivalent to having $h_R(u) \neq h_R(v)$. Therefore

$$\bigcap_{i=1}^{t} h_R(C_i) = \emptyset$$

and we have an $\mathrm{SHF}(|S|; q^{\deg(G)-g+1}, q, \{w_1, \ldots, w_t\})$. □

We make a few remarks about this theorem.

Remark 3.29. The upper bound on the number of zeroes in $\Phi_{1,\ldots,t}$ is, for most choices of $C_1, \ldots, C_t$, much larger than the actual number. There are therefore many R such that R is not a zero of any element of $\Phi_{1,\ldots,t}$, and therefore many separating rows. This claim will be supported by experimental observations in Section 3.3.2.

Remark 3.30. There are actually $\lambda = |S| - \deg G \cdot P(T)$ functions which separate any input sets. Such SHF are sometimes referred to as λ-SHF [109].

Remark 3.31. Theorem 3.28 is equivalent to the geometric Goppa code $\mathcal{C}_{\mathcal{L}}(A, G)$ with the restrictions that $\deg G \geq 2g + 1$ and $|\mathrm{Supp}(A)| \geq \deg G \cdot P(T)$ (cf. Definition 3.23). Combining the minimum distance of $\mathcal{C}_{\mathcal{L}}(A, G)$, which satisfies $D \geq N - \deg(G)$, and Theorem 3.4, we find that if

$$N - \deg(G) > N\left(1 - \frac{1}{P(T)}\right) ,$$

or equivalently,

$$\deg(G) > \frac{N}{P(T)} , \tag{3.3}$$

then $\mathcal{C}_{\mathcal{L}}(A, G)$ is an $\mathrm{SHF}(N; q^{l(G)}, q, \{w_1, \ldots, w_t\})$.

A natural question is to ask whether Theorem 3.28 improves on the simpler construction via Goppa codes and the distance condition (Theorem 3.4). If we use the smallest values allowed by Theorem 3.28, $\deg(G) = 2g+1$, $N = \deg(G)P(T)$, and ask whether the distance condition (3.3) holds, we find it does not (the LHS and RHS are equal). Therefore, the construction of Theorem 3.28 does better than the straightforward approach of using a Goppa code satisfying (3.1). To construct an SHF using the latter, we will always need an extra row.

Theorem 3.28 is still not completely explicit until a particular choice of curve is made. In Section 3.2 we will give examples, and compare the resulting parameters.

3.1.5 Construction of Compound Types

A separating hash family of *compound type* is one which acts as multiple types simultaneously. For example, consider $A = \mathrm{SHF}(N; n, m, \{w_1, w_2\}+\{w_3, w_4, w_5\})$, where $+$ is used to mean "and". Then A is both an $\mathrm{SHF}(N; n, m, \{w_1, w_2\})$ and an $\mathrm{SHF}(N; n, m, \{w_3, w_4, w_5\})$. Depending on the types, this often occurs trivially, for example SHF of type $\{w_1, w_2\}$ are also SHF of type $\{w_1', w_2\}$ whenever $w_1' \leq w_1$. Similarly, SHF of type $\{w_1, \ldots, w_t\}$ are also SHF of type $\{w_1, w_2 + \ldots + w_t\}$.

A more interesting compound type is the one used to give a necessary condition for w-IPP codes [155]. Staddon and Stinson [155] show that a w-IPP code must be an SHF of type $\{w, w\} + \{1^{w+1}\}$. The converse does not hold in general (but it does hold for $w = 2$ [96]).

Theorem 3.32. *Let A be an* SHF$(N; n, m, \{w, w\})$ *constructed by Theorem 3.28 or 3.4. Then A is also a PHF of strength $w + 1$.*

Proof. From the construction, we have $N = |S|$, where $|S| \geq \deg G \cdot P(T)$. Since $P(\{w, w\}) > P(\{1^{w+1}\})$, the construction for type $\{w, w\}$ is also a construction for type $\{1^{w+1}\}$, which gives A both separating properties. □

Theorem 3.32 generalizes as follows, by noting that a type with a given $P(T)$ value is also a construction for types with smaller $P(T)$.

Theorem 3.33. *Let A be an SHF of type $T = \{w_1, \ldots, w_t\}$ constructed by Theorem 3.28 or 3.4. Then A is simultaneously an SHF of type T', for all types T' for which $P(T') \leq P(T)$.*

Note that this does not hold for SHF constructed arbitrarily as [164] gives (among others) an example of an SHF of type $\{2, 2\}$ which is *not* an SHF of type $\{1, 3\}$ even though $P(\{2, 2\}) = 4 > P(\{1, 3\}) = 3$.

3.1.6 Analysis of Random Hash Families

In this section, instead of explicitly constructing an SHF, we analyze the probability that a matrix chosen at random acts as an SHF of a given type. In other words, if we choose a matrix at random, what is the probability that it separates an input of type T? Such random hash families have the advantage of being simple and efficient to implement. Further, if the entries of the matrix are generated as needed from a pseudorandom function, minimal storage is required. One of the metrics we will present in Section 3.2 compares the behaviour of a particular SHF to the expected behaviour of a family of random functions. Techniques similar to the ones used in this section were used in Section 2.5, to give existence results for SHF of type $\{w_1, w_2\}$ and in [164] for type $\{w_1, \ldots, w_t\}$.

In the following, let $K_{w_1,\ldots,w_t}$ be the complete multipartite graph where each part has w_i nodes and there are $w_1 + \ldots + w_t$ vertices in total. We use $\chi(G, x)$ to denote the chromatic polynomial of the graph G, which counts the number of proper colourings of G using x colours. Recall that a colouring assigns one of the x colours to each vertex, and a colouring is proper if no adjacent vertices have the same colour.

Definition 3.34. An $(N; n, m)$-*random hash family* is an $N \times n$ matrix with entries chosen randomly from an alphabet of size m uniformly and independently.

Theorem 3.35. *Let A be an $(N; n, m)$-random hash family and $C_1, \ldots, C_t$ be disjoint sets of columns, where $|C_i| = w_i$. Then the probability ϵ, that no row of A separates $C_1, \ldots, C_t$, is*

$$\epsilon = \left(1 - \frac{\chi(K_{w_1,\ldots,w_t}, m)}{m^{w_1+\ldots+w_t}}\right)^N .$$

Proof. We first restrict our attention to f, a single row of the hash family. Consider the complete multipartite graph, where set C_i will define the nodes in part i. Then colour the nodes in part $C_i = \{c_1, \ldots, c_{w_i}\}$ with one of m colours, depending on $f(c_j)$. Connect nodes in one part to all other nodes in different parts to form the complete multipartite graph, $K_{w_1,\ldots,w_t}$. The main observation is that an improper colouring corresponds to inseparable inputs. If two nodes (inputs) share an edge, then they belong to different parts (different sets C_i), and if they have the same colour, then they have the same value for f.

There are a total of $m^{w_1+\ldots+w_t}$ colourings, of which $\chi(K_{w_1,\ldots,w_t}, m)$ are proper and since f is a random function, we are choosing a colouring at random. Since the random functions are independent, we can take the product of the relevant probabilities over all N rows. □

By considering the complimentary probability, whether a particular row *will* separate an input, we can give the expected number of rows that will separate an input (which we denote μ).

Corollary 3.36. *In a random $(N; n, m)$-hash family, the expected number of separating rows for inputs $C_1, \ldots, C_t$ is*

$$\mu = N\left(\frac{\chi(K_{w_1,\ldots,w_t}, m)}{m^{w_1+\ldots+w_t}}\right) .$$

Remark 3.37. For many values of $(N; n, m)$ and $\{w_1, \ldots, w_t\}$ there will be multiple separating rows. For example, a random $(64; 2^{32}, 16)$-hash family has $\mu = 21.67$ for inputs of type $\{2, 2, 3\}$.

Remark 3.38. Computing chromatic polynomials of arbitrary graphs is intractable. See [182] for details. However, an explicit and efficiently computable formula specific to complete multipartite graphs is given in Appendix A. Previously, such a formula was known only for complete bipartite graphs [72].

Error Rate of Randomly Behaving Hash Families

As shown in the previous section, a random hash family may have, on average, multiple separating rows for a given input. Suppose a particular construction outputs A, an $(N; n, m)$-hash family, and the largest value of $P(T)$ guaranteed by the construction is

$maxP(T)$. What is the behaviour of A when we hash inputs "stronger" than those guaranteed by the theorem?

For a random hash family, the probability that an input $C_1, \ldots, C_t$ is not separated is

$$\epsilon = \left(1 - \frac{\chi(K_{w_i,\ldots,w_t}, m)}{m^{w_i+\ldots+w_t}}\right)^N \tag{3.4}$$

where N is the number of rows and $\{w_1, \ldots, w_t\}$ is the type.

Therefore, if an explicit construction produces randomly behaving hash families, then (3.4) estimates the probability of failure when

1. hashing inputs of a type T', where $P(T') > maxP(T)$, and
2. using only the first N' rows of A (here we would replace N by N' in (3.4)).

For applications which can tolerate failure at low probabilities, hash families with smaller parameters can be used. Additionally, parameters can be selected so that the probability given by (3.4) is sufficiently low for the application. We will confirm this experimentally in §3.3.4.

The difference between using a random hash family, and an explicitly constructed non-random, but randomly behaving, hash family is the partial guarantee provided by the explicit construction. For example, suppose A is a hash family which is guaranteed to be of type $\{1, 1, 1, 1\}$, and we use A to hash an input of type $\{2, 2, 2, 2\}$, given by the sets $C_1, \ldots, C_4$. If A behaves randomly, then equation (3.4) gives the probability that $C_1, \ldots, C_4$ are separated by some row. In the event that no row separates $C_1, \ldots, C_4$, A does guarantee that for one of the functions in A, all output sets $f(C_i)$ will contain a distinct element. With a random hash family, no such "partial separation" is guaranteed, as it could happen that for all f, there exist some i, j such that $f(C_i) = f(C_j)$.

3.2 A Detailed Look at the AG Construction

In this section we present three metrics that will be used to describe and compare SHF constructed using Theorem 3.28. The parameters of the SHF constructed using Theorem 3.28 depend on the choice of algebraic curve used with the theorem. In this section we will take a detailed look at this construction for five choices of algebraic curves.

3.2.1 Metrics

In this thesis, $\log x$ will always denote the base 2 logarithm of x. Below are the three metrics we will use to compare various constructions of Section 3.1.

Compression: Let $K = \frac{\log n}{\log m}$, which measures the ratio between the size of the domain and co-domain. Larger values of K are generally better.

Cost: Let $C = \frac{N}{\log n}$, which measures how large a domain can be accommodated at a cost of N rows. Lower values of C are better.

Randomness: Let $R = \frac{\mu_{\text{obs}}}{\mu}$, which measures how close the hash family's behaviour, with respect to separation, resembles a set of uniformly random functions with the same size, domain and co-domain. μ is the average number of separating rows in the random hash family (see §3.1.6), and μ_{obs} is the observed average of the hash family in question. A value $R = 1$ is ideal.

From this perspective, a good construction will have high compression, low cost and random behaviour. Note that R must be computed experimentally, while C and K can be calculated directly from the parameters of the construction. Also, an explicitly constructed SHF with $R \approx 1$ is only random in the sense that it separates inputs in the same number of rows, on average, as a random hash family. This last point motivates the randomness metric: knowing that a particular hash family behaves randomly allows us to apply the analysis from Section 3.1.6.

In addition to giving concrete parameters for SHF constructed using the AG construction, we also supplement the details given in Section 3.1.4 with details required for implementation. Our implementation is described in Section 3.3.

3.2.2 Reed-Solomon Codes

In this section we use a $(q; q^\ell, q)$ Reed-Solomon code to construct an SHF$(q; q^\ell, q, T)$ for T having $P(T) < q/(\ell - 1)$. Table 3.2 shows some possible parameters of SHF. We try to construct an SHF with $P(T) = 10$, and choose $q = p^2$ for comparison with the other examples.

We single out and name two Reed-Solomon codes to use for comparison in Section 3.3. Reed Solomon codes are a special case of the AG construction, when the curve used is the projective line $X + Y + Z$. G is the divisor used in the construction of Theorem 3.28.

RS_1 : defined over $\mathbb{F}_{2^4}$, has 16 points, genus 0, $\deg G = 3$, and gives an SHF$(16; 2^{16}, 2^4, maxP(T) = 5)$, with $K = 4$, $C = 1$.

q	ℓ	$\log n$	$maxP(T)$	C	K
4	2	4	3	1	2
16	4	16	5	1	4
64	8	48	9	1.3	8
256	16	128	17	2	16

Table 3.2: The Reed-Solomon code $(q; q^\ell, q)$ gives an SHF$(q; q^\ell, q, T)$ for types T with $P(T) \leq maxP(T)$.

RS_2 : defined over $\mathbb{F}_{2^6}$, has 64 points, genus 0, $\deg G = 12$, and gives an SHF$(64; 2^{78}, 2^6, maxP(T) = 5)$, with $K = 13$, $C = 0.82$.

3.2.3 Elliptic and Hyperelliptic Curves

Here we briefly present a few instantiations of the AG construction with elliptic and hyperelliptic curves. These curves will be used for comparison in Section 3.3. We omit details of the construction and simply present curves and the parameters obtained. Details for the elliptic case (genus 1) are given in [109], while sufficient information about hyperelliptic curves can be found in [59]. In general, explicit formulas are not known for the number of points on these curves, so software must be used to compute the exact number. Again, G refers to the divisor used in the construction to define $\mathcal{L}(G)$.

E_1 : Elliptic curve $Y^2Z + XYZ - X^3 + X^2Z + Z^3$ defined over $\mathbb{F}_{2^4}$, has 15 points, genus 1, $\deg G = 3$, and gives an SHF$(15; 2^{12}, 2^4, maxP(T) = 5)$, with $K = 3$, $C = 1.25$.

E_2 : Elliptic curve E_1, but defined over $\mathbb{F}_{2^6}$, has 55 points, genus 1, $\deg G = 10$ and gives an SHF$(55; 2^{60}, 2^6, maxP(T) = 5)$, with $K = 10$, $C = 0.91$.

G_1 : Hyperelliptic curve $Z^3Y^2 + YZ^4 - X^5 - Z^5$ defined over $\mathbb{F}_{2^4}$, has 33 points, genus 2, $\deg G = 6$ and gives an SHF$(33; 2^{20}, 2^4, maxP(T) = 5)$, with $K = 5$, $C = 1.64$.

G_2 : Hyperelliptic curve $Z^7Y^2 + YZ^8 - X^9 - Z^9$ defined over $\mathbb{F}_{2^6}$, has 128 points, genus 4, $\deg G = 25$ and gives an SHF$(128; 2^{132}, 2^6, maxP(T) = 5)$, with $K = 22$, $C = 0.97$.

3.2.4 Hermitian Curves

A Hermitian curve, denoted H_r, is defined by the affine equation

$$y^r + y = x^{r+1} \ . \tag{3.5}$$

Curve	$\deg G$	$maxP(T)$	$\log n$	N	C	K
E_1	3	5	12	15	1.25	3
E_2	10	5	60	55	0.91	10
G_1	6	5	20	33	1.64	5
G_2	25	5	132	128	0.97	22

Table 3.3: Some possible parameters using the construction of Theorem 3.28 with elliptic and hyperelliptic curves.

Over the finite field $\mathbb{F}_q$, where $q = r^2$ for a prime power r, H_r has r^3 $\mathbb{F}_q$-rational points and 1 point at infinity. The genus is

$$g_r = \frac{r(r-1)}{2} .$$

Let S be the set of points defining the rows of the SHF; we will choose $S \subseteq H_r(\mathbb{F}_q)\backslash\mathcal{O}$. We can get a divisor G of arbitrary degree $\deg(G) = z$, by choosing $G = z(\mathcal{O})$. If $r^2 - r - 2 < \deg G < r^3$, the vector space $\mathcal{L}(G)$ has dimension

$$l(G) = \deg G + 1 - \frac{r^2 - r}{2} \tag{3.6}$$

(see [175, Ex. 4.4.40]).

We can now construct an SHF$(|S|; q^{l(G)}, q, \{w_1, \ldots, w_t\})$, using the construction of Theorem 3.28, provided

(i) $\deg G \geq 2g + 1$, and

(ii) $|S| > P(T) \deg G$.

To satisfy condition (i), we must choose $\deg G \geq r^2 - r + 1$. To satisfy condition (ii), we must have $P(T) < \frac{r^3}{r^2-r+1}$, since $N = |T|$ is at most r^3. For this choice of $\deg G$ the dimension of $\mathcal{L}(G)$ is given by (3.6).

Table 3.4 shows example parameters possible with this construction. Additional discussion of computational aspects of Hermitian function fields are available in [113].

The following two SHF constructed with Hermitian curves will be used for comparison in Section 3.3.

H_4 : $X^5 + Y^5 + Z^5$ defined over $\mathbb{F}_{2^4}$, has 64 points, genus 6, $\deg G = 13$ and gives an SHF$(64; 2^{32}, 2^4, maxP(T) = 4)$, with $K = 8$, $C = 2$.

H_8 : $X^9 + Y^9 + Z^9$ defined over $\mathbb{F}_{2^6}$, has 512 points, genus 28, $\deg G = 57$ and gives an SHF$(512; 2^{180}, 2^6, maxP(T) = 8)$, with $K = 30$, $C = 2.84$.

q	$\deg G$	$maxP(T)$	$l(G)$	N	C	K
4	3	2	3	8	1.33	3
16	13	4	8	64	2.00	8
64	57	8	30	512	2.84	30
256	241	16	122	4096	4.20	122

Table 3.4: Some possible parameters using the construction of Theorem 3.28 with Hermitian curves. Each row represents an $\mathrm{SHF}(N; q^{l(G)}, q, T)$ for any type T with $P(T) < \max P(T)$.

3.2.5 The Garcia-Stichtenoth (GS) Tower

GS towers yield an optimal class of AG codes, a statement we justify after describing one of the GS towers. The languages of algebraic curves and function fields are interchangeable, but the GS towers are typically described as function fields (as we do here). Two similar towers were defined by Garcia and Stichtenoth in [86] and [87]. The following describes the tower from [87].

Again we assume $q = r^2$, and specialize our presentation to the field $\mathbb{F}_q$. The GS towers are sequences of function fields $F_0, F_1, \ldots$.

- $F_0 = \mathbb{F}_q(x_0)$ is the field of rational functions of $\mathbb{F}_q$ in one indeterminate (the function field of the projective line).
- To create F_1, we adjoin an element to F_0, $F_1 = \mathbb{F}_q(x_0, x_1)$, where x_1 satisfies

$$x_1^q + x_1 = \frac{x_0^q}{x_0^{q-1} + 1} .$$

- In general $F_i = \mathbb{F}_q(x_0, x_1, \ldots, x_i)$, where x_i satisfies

$$x_i^q + x_i = \frac{x_{i-1}^q}{x_{i-1}^{q-1} + 1} .$$

We omit description of the other tower, which is given in [86].

Let g be the genus of a curve or a function field, and let $\delta = D/N$ the relative distance of a $(N; n, m, D)$ code. The rate $R = \log(n)/N$ of an AG code is bounded by

$$R \geq 1 - \delta - \frac{g}{N}$$

and the Drinfeld-Vlăduţ bound says that

$$\liminf_{g\to\infty} \frac{g}{N} \geq \frac{1}{\sqrt{r}-1} \ . \tag{3.7}$$

Note that N is the same as the number of rational places in a function field (or rational points on a curve). The tower of GS function fields has $\liminf_{g\to\infty} = 1/(\sqrt{r}-1)$, which is best possible by (3.7). Since the rate is maximized, codes constructed using the GS tower are in this sense the best AG codes possible.

We now apply Theorem 3.28 with the tower of GS function fields above.

Corollary 3.39. *Let $N(F_i)$ be the number of $\mathbb{F}_q$-rational places (where $q = r^2$), and let g_i be the genus of the GS function field F_i defined above. Then there exists an* $\mathrm{SHF}((2g_i + 1)P(T); r^{4g_i+2}, q, T)$ *provided that*

$$P(T) \leq \frac{N(F_i)}{2g_i + 1} \ . \tag{3.8}$$

Proof. Let G_i be a divisor of degree z_i. We will hash from $\mathcal{L}(G_i)$ to $\mathbb{F}_q$. To apply Theorem 3.28 we require that

(i) $z_i \geq 2g_i + 1$, and

(ii) $N \geq z_i P(T)$.

The condition (3.8) comes from (i), (ii) and the fact that $N \leq N(F_i)$. Conditions (i) and (ii) are satisfied if we set $z_i = 2g_i + 1$ and $N = z_i P(T)$. From the Riemann-Roch theorem, $l(G_i) \geq 2g_i + 1$, and $q^{2g_i+1} = r^{4g_i+2}$ which gives the result. □

Before we can see some sample parameters of this construction, we need formulas for g_i and $N(F_i)$. Bounds on these quantities are given in [109], but we will use more precise formulas when possible. For the first two fields we have

$$\begin{aligned} N(F_0) &= r^2 - r \\ N(F_1) &\geq r^3 - r^2 \ . \end{aligned}$$

For $i \geq 2$ the exact number of $\mathbb{F}_{r^2}$ rational places is given by Aleshnikov et. al. [2]:

$$N(F_i) = \begin{cases} r^i(r^2 - r) + 2r, & \text{for odd } r,\ i \geq 2 \\ r^i(r^2 - r) + 2r^2, & \text{for even } r,\ i \geq 2 \ . \end{cases}$$

An exact formula for the genus was given by Garcia and Stichtenoth [87]:

$$g_i = \begin{cases} (r^{(i+1)/2}-1)(r^{(i-1)/2}-1) & \text{for odd } i\ , \\ (r^{i/2}-1)^2 & \text{for even } i\ . \end{cases}$$

Now that we have all the necessary formulas, possible parameters resulting from the construction of Corollary 3.39 are given in Table 3.5 over F_0, Table 3.6 over F_1, and Table 3.7 over F_2. The largest allowable value of $P(T)$ is listed and used to compute C. The construction over F_3 gives codes so large we omit it (for example when $P(T) = 2$, $n \approx 2^{2684}$).

q	$P(T)$	$\deg G$	d	$\log(n)$	N	C	K
4	2	1	2	4	2	0.50	2
16	12	1	2	8	12	1.50	2
64	56	1	2	12	56	4.66	2
256	240	1	2	16	240	15.00	2

Table 3.5: Table showing parameters of SHF$(\deg G \cdot P(T); q^d, q, T)$ constructed over F_0.

q	$P(T)$	$\deg G$	d	$\log(n)$	N	C	K
4	1	3	3	6	3	0.50	3
16	2	19	11	44	38	0.86	11
64	4	99	51	306	396	1.29	51
256	8	451	227	1816	3608	1.98	227

Table 3.6: Table showing parameters of SHF$(\deg G \cdot P(T); q^d, q, T)$ constructed over F_1.

q	$P(T)$	$\deg G$	d	$\log(n)$	N	C	K
4	2	7	5	10	14	1.39	5
16	2	91	47	188	182	0.96	47
64	4	883	443	2658	3532	1.32	443
256	8	7651	3827	30616	61208	1.99	3827

Table 3.7: Table showing parameters of SHF$(\deg G \cdot P(T); q^d, q, T)$ constructed over F_2.

3.3 Experimental Observations

Here we discuss our implementation which we use to compute the randomness metric for some of the examples given in the previous section. We also confirm that the analysis of

random hash families accurately describes, for SHF constructed using Theorem 3.28, the number of separating rows and the failure probabilities discussed in Section 3.1.6.

3.3.1 Implementation of the AG-based Construction

In this section, we describe an implementation of generalized separating hash families using the construction of Theorem 3.28. The main challenge with the implementation is computing a basis for $\mathcal{L}(G)$, where $G = z\mathcal{O}$ (and $\deg G = z$). For an algebraic curve defined by a projective equation, the Brill-Noether algorithm can be used to find a basis [42, 129]. We are aware of two freely available implementations of this algorithm. The first is the `brnoeth` package for the Singular computer algebra system [170], which we passed over in favour of PAFF (Package for Algebraic Function Fields) [93] which runs in the Axiom computer algebra system [169]. This package was created as a supplement to the thesis of Haché [94]. The approach we use to implement hash families from the AG construction is as follows.

Construct the code. Use PAFF to compute a basis for $\mathcal{L}(z\mathcal{O}) = \left\{x_1, \ldots, x_{l(G)}\right\}$, for a choice of z which satisfies Theorem 3.28. Using this basis, compute a generator matrix M for $\mathcal{C}(D, z\mathcal{O})$ where the divisor D is the sum of all rational points $P_1, \ldots, P_N$. The generator matrix is then (see [156, II.2.3])

$$M = \begin{bmatrix} x_1(P_1) & x_1(P_2) & \ldots & x_1(P_N) \\ \vdots & \vdots & \ldots & \vdots \\ x_{l(z\mathcal{O})}(P_1) & x_{l(z\mathcal{O})}(P_2) & \ldots & x_{l(z\mathcal{O})}(P_N) \end{bmatrix}.$$

Store M**.** This will require $O(N \log(n) \log(m))$ bits. Let $d = l(z\mathcal{O})$.

Hash inputs. To hash column sets $C_1, \ldots, C_t$, encode each $c \in C_i$ as a vector in $(\mathbb{F}_q)^d$. Compute vM for each encoded vector v, store all of the columns corresponding to the inputs. Denote this subset of columns by A. For row r we must test whether $A(r, C_i) \cap A(r, C_j) = \emptyset$, for $1 \leq i, j, \leq t$. This can be done by checking whether (as sets)

$$|A(r, C_1)| + \ldots + |A(r, C_t)| = |A(r, C_1) \ \cup \ldots \cup \ A(r, C_t)|.$$

Return the corresponding entries of A when a separating row is found, otherwise fail.

The cost of hashing a set of inputs can be easily calculated. Let $u = \sum_{i=1}^{t} w_i$. Computing the u codewords requires $O(udN) = O(uN \log(n))$ multiplications in $\mathbb{F}_q$, and finding a separating row requires $O(uN)$ $\mathbb{F}_q$-comparisons.

A caveat of the first part of the implementation just described is that it is limited to curves which can be described by a single equation, which excludes the Garcia-Stichtenoth tower discussed in Section 3.2.5. The work of Shum et al. [151] describes algorithms for finding the rational places of the GS field F_i and computing a basis for $\mathcal{L}(z\mathcal{O}) \in F_i$ for any integer z. The result is an explicit construction of AG codes from GS function fields which requires at most $\lceil n \log_r n \rceil^3$ multiplications and divisions in $\mathbb{F}_q$ (where $q = r^2$).

3.3.2 Observed Number of Separating Rows

By randomly sampling the input space (choosing $C_1, \ldots, C_t$ at random), and counting the number of separating rows, we can compute the mean number of separating rows μ_{obs} and compare it to μ, the expected number in the case of a perfectly random hash family. This serves to compare the analysis of §3.1.6 to the behaviour of hash families constructed by Theorem 3.28. We will then compute $R = \frac{\mu_{obs}}{\mu}$ for the example hash families given at the beginning of this section.

Table 3.8 gives this data for hash families over $\mathbb{F}_{16}$ constructed from (hyper)elliptic and Hermitian curves, as well as Reed-Solomon codes, while Table 3.9 gives the data over $\mathbb{F}_{64}$. We use $maxP(T)$ to denote the largest allowable value of $P(T)$ given by Theorem 3.28, and include the standard deviation of μ_{obs}. In both tables, we used 10^4 random input sets to compute the average number of separating rows.

Curve	Type	μ	μ_{obs}	stddev.	ϵ
H_4	$\{1,2,3\}$	30.67	30.72	3.97	7.32×10^{-19}
H_4	$\{2,2,3\}$	21.67	21.67	3.75	3.22×10^{-12}
E_1	$\{1,2,3\}$	7.19	7.22	1.91	5.62×10^{-5}
E_1	$\{2,2,3\}$	5.08	5.12	1.80	0.00202
G_1	$\{1,2,3\}$	15.34	15.38	2.82	4.45×10^{-10}
G_1	$\{2,2,3\}$	10.84	10.85	2.68	1.19×10^{-6}
RS_1	$\{1,2,3\}$	7.68	7.68	2.01	2.92×10^{-5}
RS_1	$\{2,2,3\}$	5.42	5.41	1.89	0.00133

Table 3.8: Comparison of μ and μ_{obs} for SHF with $m = 16$. In all cases R is very close to one. Since $P(\{1,2,3\}) = 11$ and $P(\{2,2,3\}) = 16$ are larger than the strength guaranteed by the construction, the probability of failure, ϵ, is included (as estimated by (3.4)).

For hash families constructed by Theorem 3.28, the quantity μ_{obs} is also the number of zeroes of the set

$$\Phi_{1,\ldots,t} = \bigcup_{i=1}^{t} \bigcup_{j=i+1}^{t} \{f - g : f \in C_i, g \in C_j\} \ .$$

Curve	Type	μ	μ_{obs}	stddev.	ϵ
H_8	$\{1,2,3,4,5\}$	126.65	126.35	9.69	6.46×10^{-64}
H_8	$\{5,5,10\}$	67.51	67.530	7.61	3.64×10^{-32}
E_2	$\{1,2,3,4,5\}$	13.60	13.64	3.20	1.63×10^{-7}
E_2	$\{5,5,10\}$	7.25	7.25	2.50	4.19×10^{-4}
G_2	$\{1,2,3,4,5\}$	31.66	31.68	4.92	1.59×10^{-16}
G_2	$\{5,5,10\}$	16.88	16.88	3.90	1.38×10^{-8}
RS_2	$\{1,2,3,4,5\}$	15.83	15.86	3.43	1.26×10^{-8}
RS_2	$\{5,5,10\}$	8.44	8.43	2.70	1.18×10^{-4}

Table 3.9: Comparison of μ and μ_{obs} for SHF with $m = 64$. In all cases R is very close to one. Since $P(\{1,2,3,4,5\}) = 85$ and $P(\{5,5,10\}) = 125$ are both much larger than the strength guaranteed by the construction, the probability of failure, ϵ, is included (as estimated by (3.4)).

Let $Z(\Phi_{1,\ldots,t})$ be the number of distinct rational zeroes of all $P(T)$ functions in $\Phi_{1,\ldots,t}$. Further, we suppose that all N rational points are used by the construction.

From the proof of Theorem 3.28, $Z(\Phi_{1,\ldots,t})$ is an important quantity: each point which is a zero of some function in $\Phi_{1,\ldots,t}$ corresponds to a row which cannot separate the input $C_1, \ldots, C_t$. To see how $Z(\Phi_{1,\ldots,t})$ and the number of separating rows s are related, recall that the construction uses all rational points and that each point defines a function (row) $h_P : \mathcal{L}(A) \to \mathbb{F}_q$ as $h_P(x) = x(P)$. Each point P corresponding to one of the s separating rows is *not* a zero of any function in $\Phi_{1,\ldots,t}$. If it were, then $f(P) = g(P)$ for some $f \in C_i$, $g \in C_j$ $(i \neq j)$, which would contradict the separating property. On the other hand, the remaining $N - s$ points correspond to rows which do not separate $C_1, \ldots, C_t$, must be a zero of some function in $\Phi_{1,\ldots,t}$. If this were not the case for a point P, then h_P would separate the inputs since $(f - g)(P) \neq 0$ is equivalent to $f(P) \neq g(P)$ for all pairs of f, g from different input sets.

Therefore the large number of separating rows observed in Tables 3.8 and 3.9 support Remark 3.29, that the bound used on $Z(\Phi_{1,\ldots,t})$ in the proof of Theorem 3.28 is not tight for most inputs.

3.3.3 Randomness Metric

For the example SHF in Section 3.2 we compute the randomness metric using our implementation. The curves in Table 3.10 are defined in §3.2.

For type $\{5,5,5,5\}$ and $m = 16$, i.e., for H_4, E_1, G_1 and RS_1, the expected value of μ was so small that 10^4 trials were not enough to accurately measure μ_{obs}, as nearly

Curve\Type	$\{2,3\}$	$\{2,7\}$	$\{4,4,5\}$	$\{5,5,5,5\}$
H_4	1.000065	1.000770	1.001967	—
H_8	0.999989	1.000128	0.999180	0.99968
E_1	1.000909	1.006130	0.994248	—
E_2	0.999128	0.999834	0.998874	1.00180
G_1	0.999438	1.003119	1.045361	—
G_2	0.999849	1.000318	1.001051	1.00033
RS_1	1.002123	0.999530	0.992877	—
RS_2	0.999765	0.999972	0.999916	1.00138

Table 3.10: The randomness metric for hash families produced by Theorem 3.28. The curves and associated parameters are given in §3.2.

every input hashed was not separated by any row. In the other cases, R is very close to 1, suggesting that hash families constructed by Theorem 3.28 will behave like random hash families. Further, the choice of curve or the type of inputs does not affect R, confirming that randomness is independent of parameter selection.

3.3.4 Observed Error Rates

In Section 3.1.6, equation (3.4) gives the probability that a random hash family will fail to separate a set of inputs. This probability will hold for hash families constructed by Theorem 3.28 provided they behave randomly. Indeed this is the case, as shown in §3.3.3. Table 3.11 confirms the accuracy of equation (3.4) experimentally by counting the number of failures. For each of the hash families tested, 10^5 trials were run. The types were chosen to be large enough so that the probability of error was observably large.

Since the observed error rates are close to those given by equation (3.4), some applications may benefit from the reduced number of rows. For example, if an error rate of 0.00024 is tolerable, using H_8 with 212 rows gives a reduction of 300 rows, and type $\{5,10,10\}$ is much stronger than guaranteed by Theorem 3.28.

3.4 Conclusion

We have presented a variety of constructions for generalized SHF. Our detailed examination of the algebraic geometry construction finds that the choice of curve is important in determining the compression and cost of the hash family, but that all curves we examined give randomly behaving hash families. This random behaviour allows our implementation to separate inputs with known probability.

Curve	Type	N'	ϵ	ϵ_{obs}
RS_1	$\{2, 2, 3\}$	16	0.00133	0.00113
H_4	$\{2, 4, 4\}$	64	0.00047	0.00039
H_4	$\{2, 4, 4\}$	44	0.00518	0.00445
RS_2	$\{1^4, 10\}$	12	0.00042	0.00024
E_2	$\{5, 5, 10\}$	55	0.00041	0.00049
H_8	$\{5, 10, 10\}$	212	0.00024	0.00020
G_2	$\{5, 5, 10\}$	58	0.000274	0.00026

Table 3.11: Observed error rates of the SHF defined in §3.2 using the first N' rows. The error rate of a perfectly random hash family is ϵ (computed using equation (3.4) and the observed error rate is ϵ_{obs}.

Exactly which curve will be best for a given application depends on the relative importance of the desired values C, K allowable strength $P(T)$, as none of the examples we have seen is best in all respects. For example, using the GS tower SHF constructed over F_0 have large allowable values of $P(T)$, but low compression, while SHF constructed over F_2 have low cost, high compression but much lower allowable values of $P(T)$.

Another finding is that the AG construction is practical and can be implemented, at least for the parameter choices made in this chapter. We feel this is also an important contribution as previous work related to this construction focus on the asymptotic behaviour of the construction, instead of examining the constructed SHF for concrete parameter choices, as we have done.

Chapter 4

Anonymity in Shared Symmetric Key Primitives

In this chapter, we provide a stronger definition of anonymity in the context of shared symmetric key primitives (§4.2), and show that existing schemes do not provide this level of anonymity. A new scheme is presented to share symmetric key operations amongst a set of participants according to a (t, n)-threshold access structure. We quantify the amount of information the output of the shared operation provides about the group of participants which collaborated to produce it. The material in this chapter was published in [185].

4.1 Introduction

In this chapter, we consider the anonymity provided by schemes for sharing symmetric key operations such as block ciphers or message authentication codes (MACs). We will focus on threshold access structures. Let $\mathcal{P}$ be a set of participants. A (t, n)-*threshold access structure* on $\mathcal{P}$ is defined by Γ, which is the set of all authorized sets, namely $\Gamma = \{A \subseteq \mathcal{P} : |A| = t\}$. Put simply, any subset of participants of size at least t is authorized.[1] In this model, the keys are distributed by an entity called the *receiver*, who will later receive a message from some $A \in \Gamma$.

A first approach to construct such a primitive might be to share the symmetric key using a (t, n) secret sharing scheme, i.e., distribute a share of the key to each of the n participants such that t or more shares are required to recover the key. The problem with

[1] For simplicity, we do not include sets with more than t participants in Γ. Since the threshold access structure is monotonic, the participants in excess of t may be ignored.

this approach is that it requires reconstructing the key from the shares, revealing the key to the participants. To avoid this requires an approach which uses the shares directly.

Work on sharing block ciphers was initiated by Brickell et al. [40], using a variant of secret sharing called sequence sharing. Subsequently, improved schemes for distributing block ciphers using cumulative arrays and perfect hash families were studied by Martin et al. in [116]. They introduce generalized cumulative arrays (GCAs) and give some efficient solutions (in the number of keys). A second paper by Martin et al. extends [116] to consider distributing the computation of MACs as well [115].

If a set of participants $A \in \Gamma$ collaborates to encrypt a message, what information does the receiver learn about A? Informally, anonymity is provided if the identity of A is kept secret from the receiver. We will define anonymity for groups of participants as well as individual participants in Section 4.2. In the rest of Section 4.1 we present the background required to describe the GCA-MAC threshold MAC scheme of [116], as well as the scheme itself (which will be used to illustrate the anonymity metrics in §4.2).

In addition to shared block ciphers and MACs, schemes using generalized cumulative arrays have been used in secret sharing [110] and for sharing pseudo random functions [180]. A GCA is defined below.

Definition 4.1. A *generalized cumulative array* (GCA) is a set $Y = \{y_1, \ldots, y_m\}$, a partition of Y, say $\{K_1, \ldots, K_v\}$, a set of subsets of Y, denoted by $\mathcal{B} = \{B_1, \ldots, B_n\}$, and an integer t, such that the following properties are satisfied.

(i) For any t-set $A \subseteq \mathcal{B}$, $K_i \in \left(\bigcup_{B \in A} B\right)$ for **at least one** $i \in [1, \ldots, v]$.

(ii) For any set $A' \subseteq \mathcal{B}$ with $|A'| < t$, $K_j \notin \left(\bigcup_{B \in A'} B\right)$ for **all** $j \in [1, \ldots, v]$.

When $|B_i| = \ell$ for all $B_i \in \mathcal{B}$, the GCA is *ℓ-uniform*. In this work we consider only ℓ-uniform GCAs for (t, n)-threshold access structures, and adopt the notation GCA$(t, n; \ell, v)$. In the schemes we consider, Y is a set of key components, and each participant is given one of the sets in $\mathcal{B}$. Property (i) ensures that all t-sets of participants can recover one of the keys, while property (ii) ensures that fewer than t participants are unable to recover any key. Given that a particular key was used in a threshold operation, anonymity metrics will quantify the uncertainty about which authorized subset of participants performed the threshold operation. For example, if the receiver knows that key K_1 was used, and that there is only one $A \in \Gamma$ which can recover K_1, then the primitive provides no anonymity. Anonymity of the group would be perfect if all $A \in \Gamma$ were equally likely to have used K_1.

Long et al. explore GCAs in the context of secret sharing and give upper and lower bounds for existence of GCAs [110]. Martin and Ng [114] also give constructions of GCAs and metrics to describe the efficiency of GCAs. The best construction for threshold access structures in both papers uses perfect hash families.

We will usually depict a PHF$(\ell; n, m, t)$ as an $\ell \times n$ array populated with m symbols. Each column represents an element $x_j \in X$, and each row is defined by a function. The (i, j)-th entry is defined to be $f_i(x_j)$. This array has the property that within any $\ell \times t$ subarray, there is a row containing t distinct elements.

A PHF$(\ell; n, t, t)$ naturally defines a GCA$(t, n; \ell, t)$ as follows. Let $Y = \{(i, j) : i \leq \ell, j \leq t\}$, and $K_i = \{(i, 1), \ldots, (i, t)\}$. Each $B_i \in \mathcal{B}$ is a column of the matrix representation of the PHF, along with row indices i.e., $B_k = \{(i, f_i(k)) : 1 \leq i \leq \ell\}$.

Property (i) is satisfied since for any $A = \{B_{i_1}, \ldots, B_{i_t}\}$, there will be some f such that $f(i_1) \neq \ldots \neq f(i_t)$ and hence A will have t distinct values in one row, and therefore contain some K_i. However, $A' = \left\{B_{i_1}, \ldots, B_{i_{t-1}}\right\}$ has at most $t - 1$ distinct values at each row and therefore cannot cover any K_i. Hence property (ii) is satisfied as well.

Long et al. [110] prove that the PHF construction gives asymptotically optimal GCAs for threshold access structures, i.e., GCAs with a minimal number of key components for a fixed number of participants and threshold. Since practical and efficient constructions for GCAs are only known for threshold access structures, our anonymity study will focus on threshold GCAs constructed from PHFs as described above. We now give a small example of this construction.

Example 4.2. Let f_1, f_2 be a PHF$(2; 4, 2, 2)$:

1	2	1	2
1	1	2	2

.

The construction described above is used to construct a GCA$(2, 4; 2, 2)$, where $Y = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$, $K_1 = \{(1, 1), (1, 2)\}$, $K_2 = \{(2, 1), (2, 2)\}$.

B_1	B_2	B_3	B_4
$(1, f_1(1))$	$(1, f_1(2))$	$(1, f_1(3))$	$(1, f_1(4))$
$(2, f_2(1))$	$(2, f_2(2))$	$(2, f_2(3))$	$(2, f_2(4))$

Since the schemes we will present assign B_i from the GCA to P_i for all $P_i \in \mathcal{P}$, it makes sense to talk about a "column of keys" given to P_i and a "row that separates $A \in \Gamma$". If row r separates $A \in \Gamma$, then the keys held by the participants of A in row r are distinct, which will allow them to use K_r as the key in a threshold operation.

4.1.1 Sharing Symmetric Operations

There are a few possible approaches to share a symmetric key primitive amongst a group of participants with a (t, n)-threshold access structure. We briefly review XOR-based

approaches, examined in detail for block ciphers in [116] and for MACs in [103, 115]. An alternative is to use sequences or cascade ciphers (as in [40, 78]); for details on this approach, see [116]. Sharing using XOR has the advantage that the steps in the inverse operation need not be performed in the same order as the forward operation.

Our presentation is specialized to the (t, n)-threshold access structure, but these techniques are also applicable to general access structures. The methods are generic, in that they can be used with any secure block cipher or MAC, and the resulting shared primitives are at least as secure as the underlying function.

Let $\mathcal{K}$ be a keyspace, let $\mathcal{M}$ be a message space, let $\mathcal{T}$ be the set of authentication tags and let $F : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ be a secure MAC. Suppose $K_r = (k_1, \ldots, k_t)$ is a set of keys held by some $A \in \Gamma$. The t-fold XOR MAC, $F^t : \mathcal{K}^t \times \mathcal{M} \to \mathcal{T}$ is defined as follows:

$$F^t(k_1, \ldots, k_t, m) = \left(\bigoplus_{i=1}^{t} F(k_i, m), r \right).$$

The index r is also included for use during verification, in schemes with multiple keys K_i. To verify the tag (σ, r) on the message m, the verifier computes $F(K_r, m) = (\sigma', r)$ and accepts the tag if $\sigma = \sigma'$. When referring to a *key of* F^t we mean an element of $\mathcal{K}^t$, and refer to elements of $\mathcal{K}$ as *key components*. The following lemma proves that F^t is at least as secure as F.

Lemma 4.3 (Lemma 1, [115])**.** *If F is a secure MAC, then F^t is a secure MAC as well. Moreover, an adversary can generate a forged MAC for F^t if and only if they know all key components $(k_1, \ldots, k_t)$.*

We say that F provides *strong unforgeability* if an adversary cannot create a different tag for a previously authenticated message, and *weak unforgeability* if the adversary cannot create a valid tag for a new message (but may be able to create a different tag for an old message). Formal definitions of security for MACs appear in An, Dodis and Rabin [8]. Lemma 4.3 says that if F provides strong (or weak) unforgeability, then F^t provides the same level of unforgeability. If an attacker produces a forgery for F in time τ with probability ϵ after observing q (message, tag) pairs, then producing a forgery for F^t with probability ϵ requires at least time τ (with q observations).

A similar construction is possible for block ciphers. Let $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ be a secure encryption function where $\mathcal{C}$ is the set of ciphertexts. A group of participants $A \in \Gamma$ can encrypt a message using $K_r = (k_1, \ldots, k_t)$ with

$$E^t(k_1, \ldots, k_t, m, n_0) = \left(m \bigoplus_{i=1}^{t} E(k_i, n_0), n_0, r \right),$$

where n_0 is a random nonce. Further details of these constructions and security proofs are given in [116, 115]. In this chapter we will present schemes and give examples using a shared MAC; however, our results can also be applied to shared block ciphers.

4.1.2 The GCA-MAC Authentication Scheme

The following scheme is presented in [115]. We specialize our discussion to the case of a GCA$(t, n; \ell, t)$ constructed from a PHF$(\ell; n, t, t)$.

Setup Let $(Y, K_1, \ldots, K_t, \mathcal{B})$ be a GCA$(t, n; \ell, t)$, constructed from a PHF$(\ell; n, t, t)$. Let F^t be the MAC defined in §4.1.1, and let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of participants.

Key Distribution The receiver chooses a set of ℓt key components and labels them with the elements of Y, then distributes the key components corresponding to B_i to P_i. In other words, the receiver gives the key components indexed by column i to P_i.

Tag Creation The participants $P_{i_1}, \ldots, P_{i_t}$ create a tag for a message m as follows. First they determine j such that $K_j \subseteq (B_{i_1} \cup \ldots \cup B_{i_t})$. This is done by finding a row in the $\ell \times t$ subarray of columns $i_1, \ldots, i_t$ which has t distinct entries. If multiple rows have distinct entries, then j is set to the first such row. Then the tag is computed as

$$(\sigma, j) = F^t(K_j, m),$$

and m and (σ, j) are sent to the receiver.

Verification The receiver uses K_j to check if $(\sigma, j) \stackrel{?}{=} F^t(K_j, m)$ and accepts if they are equal.

4.1.3 GCA-MAC Example

We reproduce an example of a $(2, 8)$ GCA-MAC from [115], which will be used while discussing anonymity in Section 4.2. This GCA$(2, 8; 3, 2)$ is based on a PHF$(3; 8, 2, 2)$.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
1	1	1	1	2	2	2	2
1	1	2	2	1	1	2	2
1	2	1	2	1	2	1	2

The three possible keys corresponding to the three rows are K_1, K_2, K_3, respectively, each constructed by participants having both a 1 and 2 in that row. This example is small enough to inspect all $\binom{8}{2} = 28$ possible cases. Note that pairs of participants can reconstruct one key (for example P_1, P_2), two keys (e.g. P_2, P_3) or all three keys (e.g. P_1, P_8). There are $\ell t = 6$ distinct key components.

4.2 Anonymity

Anonymity for shared symmetric key operations was first considered for the shared MAC schemes in [115]. There are two types of anonymity to consider. The first is *group anonymity*, which asks whether the receiver can learn which authorized subset collaborated to create the tag. The second type is *participant anonymity* (which we introduce in this chapter), which asks whether the receiver can learn if a particular participant was involved in creating the tag. In both cases the receiver is given only the message and the tag output by the threshold scheme. In this section, we provide a general description of our new measures of anonymity, independent of any particular schemes, We use the GCA-MAC example given in Section 4.1.3 to illustrate the various concepts we discuss.

4.2.1 Threat Model

The goal is to prevent the adversary from learning which set of participants $A \in \Gamma$ performed a particular operation such as encryption or authentication. The adversary may be the receiver, who distributes key components and receives a ciphertext or authentication tag created by a group of participants. It may also be anyone knowing how keys were assigned to participants, who later observes the output of a shared primitive.

We *assume that the message does not leak the identity of the participants in* A and that communication of the tag to the receiver is done using an anonymous channel. All details of communications between members of $\mathcal{P}$ are assumed to be hidden from the adversary. The knowledge of the adversary is limited to the output of the shared primitive, which includes all information required to perform the associated operation (decryption or verification). We also assume that all authorized sets are equally likely to use the primitive.

The anonymity provided is unconditional. Since multiple sets of participants can reconstruct the key used to perform the operation, they are all equally likely to have performed the operation in question, and no amount of computation on the part of the attacker will give additional information.

4.2.2 Group Anonymity

In this section we consider anonymity of a single group $A \in \Gamma$ and then define anonymity for a whole scheme $\mathcal{S}$.

Counting-Based Metrics

The counting-based metrics in this section extend the approach of Martin et al. [115]. The idea is to count the number of authorized sets which can reconstruct a key, and determine how likely each is to use it. Then, given that a particular key was used, determine which authorized sets were most likely to use it.

Definition 4.4. Let $\mathcal{S}$ be a (t, n)-threshold MAC over n participants $\mathcal{P}$, with ℓ keys. The (t, n) access structure is $\Gamma = \{A : A \subseteq \mathcal{P}, |A| = t\}$ and $|\Gamma| = \binom{n}{t}$. Let $A \in \Gamma$. For any message m and valid tag (σ, r) generated by A note that there may be multiple authorized subsets that could have created (σ, r). We write $\Pr[A|r]$ to denote the conditional probability that A created the given tag (σ, r). The *degree of anonymity* of $A \in \Gamma$ with respect to $\mathcal{S}$ is defined $d(A, r) = 1 - \Pr[A|r]$.

The following definition of group anonymity is from [115], where it was originally called *anonymity*. We rename it *average degree of anonymity*. This metric is only applicable to schemes where each authorized set uses only one key (as is the case with GCA-MAC).

Definition 4.5. In the notation of Definition 4.4, the *average degree of anonymity* for $\mathcal{S}$ is defined by

$$d_{av}(\mathcal{S}) = \frac{\sum \{d(A, r) : A \in \Gamma\}}{|\Gamma|} ,$$

where r is the index of the key used by A.

In the best case, upon seeing m and (σ, r), the adversary will see all sets in Γ as being equally likely creators of σ. Therefore, at best, $d(A, r) = 1 - 1/\binom{n}{t}$. The average degree of anonymity of GCA-MAC was given in [115].

Theorem 4.6. *The GCA-MAC scheme has average degree of anonymity*

$$d_{av} \geq 1 - \frac{\ell}{\binom{n}{t}} .$$

Proof. (Sketch) Define Γ_i, $1 \leq i \leq \ell$, as the set of all sets $A \in \Gamma$ separated by row i but not separated by any row $j < i$ (i.e., i is the first row which separates A). The result follows by noting that $d(A, i) = 1 - 1/|\Gamma_i|$ for all $A \in \Gamma_i$, and that $\sum_{i=1}^{\ell} |\Gamma_i| \geq \binom{n}{t}$, since all $A \in \Gamma$ are separated by at least one row. □

Since a PHF$(\ell; n, t, t)$ can be constructed when $\ell \geq \lceil te^t \log n \rceil$ (see [121]), the following result holds.

Theorem 4.7. *There exists a (t, n)-threshold GCA-MAC scheme implemented with a* $\mathrm{PHF}(\ell; n, t, t)$ *that has average degree of anonymity* $d_{av} \geq 1 - \lceil te^t \log n \rceil / \binom{n}{t}$.

As with security, anonymity should arguably be evaluated in the worst case, not the average case. We now present a new, stronger, counting-based anonymity metric.

Definition 4.8. Let $\mathcal{S}$ and $d(A)$ be as defined in Definition 4.4. The *anonymity* of $\mathcal{S}$ is defined

$$\mu = \min \{d(A, r) : A \in \Gamma, r \in [1, \ldots, \ell]\} \ .$$

Of course, average-case anonymity is still a useful measure. If a scheme does not provide anonymity on average, anonymity in the worst case (and for all participants) will not be possible.

We now use the example of Section 4.1.3 to illustrate the difference between d_{av} and μ. We also show how the GCA-MAC protocol reduces group anonymity by specifying that a t-set of participants A will always choose the key corresponding to the *first* row separating A. Therefore, those sets of participants separated by rows one and two will never use key K_2. In the example above, keys K_1, K_2 and K_3 can each be reconstructed by 16 pairs of participants. However, since the GCA-MAC protocol selects the first separating row, we have the following:

- for $(\sigma, 1)$ there are 16 possible choices for A, hence $d(A, 1) = 1 - 1/16$,
- for $(\sigma, 2)$ there are 8 possible choices for A, hence $d(A, 2) = 1 - 1/8$, and
- for $(\sigma, 3)$ there are 4 possible choices for A, hence $d(A, 3) = 1 - 1/4$.

It can also be seen directly from the PHF that 16 inputs are separated first by row 1, eight are separated first by row 2 and four are separated first by row 3. Those $A \in \Gamma$ which use K_3 have an anonymity set one quarter the size of those which use K_1. While $d_{av} = 1 - \ell/\binom{8}{2} = 1 - 3/28 \approx 0.89$ (by Theorem 4.6), the worst case anonymity is $\mu = 0.75$.

Entropy-Based Metrics

By computing the entropy of the conditional probability distributions of the previous section, we can measure the number of bits of uncertainty the adversary has about the creator of a given tag. This provides a more accurate measure.

Definition 4.9. Let $\mathcal{S}$ be as defined in Definition 4.4. Let $\mathbf{A}$ be a random variable defined on Γ, and let r be the index of a key in $\mathcal{S}$. The *anonymity of key* r is the entropy of the probability distribution $\Pr[\mathbf{A}|r]$:

$$h_r = H(\mathbf{A}|r) = -\sum_{A \in \Gamma} \Pr[A|r] \log_2 \Pr[A|r] \ .$$

If the adversary observes a tag created with key r, then h_r can be interpreted as the number of bits of uncertainty the adversary has about the group which created the tag. It measures the effect on group anonymity caused by using different keys. In our GCA-MAC example, $h_1 = 4$, $h_2 = 3$ and $h_3 = 2$, which is consistent with the adversary's knowledge (e.g. given that K_3 was used, he can be certain that it was created by one of four groups). More generally, $\mu \approx 1 - 2^{-\min\{h_r : r \in \mathcal{S}\}}$ since this represents the worst case for anonymity. Note that an upper bound on h_r is the log of the number of groups separated by row r, in this case four bits. The sets separated by row 1 are provided best possible anonymity under this metric. In general, if T_r groups are separated by row r, then T_r is size of the largest anonymity set possible given that r was used. Therefore, key entropy close to $\log_2(T_r)$ bits is desirable.

Averaging over the possible keys that A may use gives the average anonymity provided to A.

Definition 4.10. Let $\mathcal{S}$ be as defined in Definition 4.4. The *average anonymity provided to* $A \in \Gamma$ is

$$\mu_{av}(A) = \sum_{r=1}^{\ell} \left(\Pr[r|A] \times h_r \right) \ . \tag{4.1}$$

The *average anonymity of the scheme* $\mathcal{S}$ is defined to be the average of $\mu_{av}(A)$ for all groups $A \in \Gamma$,

$$\mu_{av}(\mathcal{S}) = \frac{1}{|\Gamma|} \sum_{A \in \Gamma} \mu_{av}(A) \ .$$

Since in GCA-MAC groups only use one key, the average anonymity of A coincides with the key anonymity for the first row which separates A. The average anonymity of the scheme is $\frac{1}{28}(16 \cdot 4 + 8 \cdot 3 + 4 \cdot 2) = 3.4$ bits. There is also a natural relation, as we saw between h_r and μ, to the average degree of anonymity, namely $d_{av}(\mathcal{S}) \approx 1 - 2^{\mu_{av}(\mathcal{S})}$.

Remark 4.11. If all rows have the same value for h_r, then $\mu_{av}(A) = h_r$, since

$$\sum_{r=1}^{N} \Pr[A|r] = 1 \ .$$

In this case the average anonymity is the same for all $A \in \Gamma$. This is desirable so that no "weak keys" exist with respect to anonymity, i.e. keys with extremely low entropy (such as K_3 in our GCA-MAC example). When h_r is the same for all keys, we have $\mu_{av}(\mathcal{S}) = \mu_{av}(A) = h_r$, for all $A \in \Gamma$. The improved scheme we will present in Section 4.3 has this property.

4.2.3 Participant Anonymity

The issue of participant anonymity has not been considered in previous work. Participant anonymity is more challenging to provide than group anonymity, since there are only n participants, while there are $\binom{n}{t}$ groups.

Let m be a message and (σ, r) be a valid tag created by an unknown group $A \in \Gamma$, observed by the receiver. For $P_i \in \mathcal{P}$, let $\Pr[P_i|r]$ be the probability that $P_i \in A$ (the group that created the tag), given that r was used. Suppose $\mathcal{S}$ has ℓ keys. We define the *participant anonymity of* $P_i \in \mathcal{P}$ to be

$$\rho(P_i) = 1 - \max\{\Pr[P_i|r] : r \in [1, \ldots, \ell]\} \ ,$$

and the *participant anonymity of the scheme* $\mathcal{S}$ to be

$$\rho(\mathcal{S}) = \min\{\rho(P_i) : P_i \in \mathcal{P}\} \ .$$

With respect to ρ, it is desirable for $\mathcal{S}$ to have two properties. First, $\rho(\mathcal{S}) = 1 - t/n$ is best possible, since if all participants are equally likely, and t are required for an operation, then a participant has probability t/n of being involved. Therefore, it is desirable that $\rho(\mathcal{S})$ be close to $1 - t/n$. Also, participant anonymity should be *equitable*, that is $\rho(P_i)$ should not differ significantly from $\rho(P_j)$ (for all $P_i, P_j \in \mathcal{P}$). Unfortunately, in any scheme constructed from a PHF, since $\ell = \Omega(\log n)$, there is a trade-off between efficiency and participant anonymity.

4.2.4 Malicious Setup Attack on Anonymity

In this attack, the GCA-MAC scheme is set up so that anonymity is reduced for certain participants. The attack works by adding a row to the PHF that separates a small number of $A \in \Gamma$. Suppose we use the following PHF$(4; 9, 3, 3)$ to create a GCA$(3, 9; 4, 3)$ (source: PHFtables [98]).

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
1	3	2	2	3	2	3	1	1
1	3	1	3	1	2	2	2	3
1	2	2	1	3	3	1	2	3
3	3	2	1	1	3	2	1	2

(4.2)

Using the GCA-MAC protocol,

- given $(\sigma, 1)$, there are 27 possible choices for A,
- given $(\sigma, 2)$, there are 21 possible choices for A,
- given $(\sigma, 3)$, there are 18 possible choices for A,
- given $(\sigma, 4)$, there are 18 possible choices for A.

If the attacker adds a "dummy" row to the PHF,

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
1	1	1	1	1	1	1	2	3
1	3	2	2	3	2	3	1	1
1	3	1	3	1	2	2	2	3
1	2	2	1	3	3	1	2	3
3	3	2	1	1	3	2	1	2

,

then any message authenticated using K_1 must have $A = \{P_8, P_9, P_i\}$ (where P_i is any of the other participants). This reduces the number of possible groups to $7, 27, 18, 16, 16$ for keys $1, 2, 3, 4, 5$, respectively. Using the measures defined above, $\mu = 1 - 1/18 = 0.94$ before the attack and $\mu = 1 - 1/7 = 0.86$ after the attack. The effect on participant anonymity is that $\rho(P_8) = \rho(P_9) = 0$ since the receiver can say for sure that P_8 and P_9 participated whenever K_1 is used. The key associated with the "dummy row" has much lower key entropy than the others, reducing uncertainty about which groups use this key.

This attack may be especially effective when combined with other information about the participants. If the attacker knows which participants are most likely to initiate a message, they will be frequent senders, and the attack will be more effective when they are targeted during setup.

4.2.5 Verifiable Setup

More generally, the setup should be verified by the participants to ensure consistency of the key components distributed by the dealer. For example, in (4.2), suppose the key component given to P_1, call it $k'_{1,1}$, in the first row differs from $k_{1,1}$ given to P_8 and P_9 (who also have a 1 in row 1). Since P_1 is the only participant who holds $k'_{1,1}$, upon receiving $(\sigma, 1)$ the receiver may check whether $k'_{1,1}$ was used to create $(\sigma, 1)$ and learn whether P_1 belongs to the set which created it.

A simple approach to verifying consistency is to have each pair of participants engage in a key confirmation protocol to ensure they hold the same keys (when they should, based

on the PHF). If a trusted bulletin board is available, the dealer might alternately publish a cryptographic hash of the key component and the associated PHF to allow participants to check the validity of their key components. In this work we simply assume the setup has been verified for consistency, and leave the problem of efficient verification to future work.

4.3 An Improved Scheme: BPHF-MAC

The new scheme given below, BPHF-MAC, makes two changes to GCA-MAC to improve anonymity. First, when multiple rows separate $A \in \Gamma$, i.e. when A can reconstruct multiple keys K_i, a row/key is chosen at random from all of those possible. This recovers the anonymity lost when only the first row which separates A is used.

The second change is to counter the malicious setup attack, and prevent keys which provide weak anonymity. A special type of hash family, introduced by Stinson [159], will be used in BPHF-MAC instead of an arbitrary PHF.

Definition 4.12. A PHF$(\ell; n, m, t)$ is *balanced* if in every row, each symbol occurs exactly n/m times. The notation BPHF$(\ell; n, m, t)$ is used to denote a balanced PHF.

The balance property will maximize the possible key entropy, and simplify our analysis of group and participant anonymity of BPHF-MAC. Fortunately, many good explicit constructions of PHFs are balanced. A PHF is said to be *linear* if the code formed by taking the columns as codewords is linear. Since any linear PHF is balanced, constructions from Reed-Solomon codes [161] or AG-codes [181] give BPHFs. The examples given in Section 4.1.3 and 4.2.4 are balanced.

BPHF-MAC:

Setup Construct a BPHF$(\ell; n, t, t)$, where $f_i : X \to Y$. Let F^t be the MAC defined in §4.1.1, and let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of participants. The authorized sets are $\Gamma = \{A \in \mathcal{P} : |A| = t\}$. The BPHF should be verified by the participants if it is constructed by a party untrusted with respect to anonymity.

Key Distribution The receiver chooses a set of ℓt key components and labels them with the elements of Y, creates $\mathcal{B}$ using the construction of §4.1, and distributes the key components corresponding to B_i to P_i. In other words, the receiver gives key components indexed by column i to P_i.

Tag Creation The participants $P_{i_1}, \ldots, P_{i_t}$ create a tag for a message m as follows. First they determine the set J such that $K_j \subseteq (B_{i_1} \cup \ldots \cup B_{i_t})$ for all $j \in J$. This is done by finding the rows in the $\ell \times t$ subarray of columns $i_1, \ldots, i_t$ which have distinct entries. Since we are using a PHF of strength t we are guaranteed $|J| \geq 1$. An index j is chosen uniformly at random from J, and the tag is computed to be

$$(\sigma, j) = F^t(K_j, m) = F^t(k_1, \ldots, k_t, m) \ ,$$

and (σ, j) is sent to the receiver as the tag for m.

Verification The receiver uses K_j to check if $(\sigma, j) \stackrel{?}{=} F^t(K_j, m)$ and accepts if they are equal.

We remark that the performance of the tag creation step may be improved by repeatedly selecting a row at random and checking whether it separates the participants, until a separating row is found. There is no need for the set J to be computed explicitly.

The security of **BPHF-MAC** follows from the security of F^t (Lemma 4.3) and the properties of a GCA. By Lemma 4.3 all t components of K_j are required to compute F^t, while property (ii) in the definition of a GCA (Def. 4.1), ensures that no set of fewer than t users hold K_j, for any j. Therefore, sets of participants $A' \notin \Gamma$ will not be able to compute F^t.

4.3.1 Anonymity of BPHF-MAC

To evaluate the anonymity of **BPHF-MAC** we first prove a key property of BPHFs.

Lemma 4.13. *In a* BPHF$(\ell; n, m, t)$*, with functions* $f_i : X \to Y$*, the size of the set* $\{A \subset X : |A| = t, |f_i(A)| = t\}$ *is*

$$\binom{m}{t}\left(\frac{n}{m}\right)^t \ ,$$

for all f_i*,* $1 \leq i \leq \ell$*.*

Proof. From the definition of a BPHF, each of the m symbols occurs n/m times in each row. We ask how many t-sets of columns are distinct when restricted to row i. The t symbols can be chosen in $\binom{m}{t}$ ways, and for each of these t symbols we must choose one of the n/m positions. Therefore in a BPHF$(\ell; n, m, t)$ each row separates $\binom{m}{t}(n/m)^t$ sets of t columns. □

The following definition will also be used to analyze the anonymity of BPHF-MAC.

Definition 4.14. Let $\mathcal{C}$ be a code, and A be a t-set of codewords of $\mathcal{C}$. The *t-separating distance*, denoted $s_{A,t}$, is the number of coordinates in which all t codewords in A differ.

The t-separating distance first appears in the work of Bassalygo et al. [16] (where it was originally called the t-th hash distance). The t-separating distance may be seen as a generalization of the classic Hamming distance, which is the same as the 2-separating distance. A PHF$(\ell; n, m, t)$ guarantees $s_{A,t} \geq 1$ while a λ-PHF$(\ell; n, m, t)$ guarantees $s_{A,t} \geq \lambda$ ([109, 163]). In the (t, n) BPHF-MAC scheme, t is the threshold size, so we will simply write s_A in what follows.

To compute the anonymity of the BPHF-MAC scheme we require knowledge of $\Pr[A|r]$, the probability that A created a tag using key r (row r of the PHF). Let us first consider $\Pr[r|A]$. If A is not separated by row r, it cannot use key r, therefore

$$\Pr[r|A] = \begin{cases} 0 & \text{when } r \text{ does not separate } A \\ \frac{1}{s_A} & \text{when } r \text{ does separate } A\,. \end{cases}$$

The probability that row r is used is $1/s_A$ since A will choose r at random from one of the s_A separating rows.

We have assumed that $\Pr[A] = 1/\binom{n}{t}$, i.e., all sets of participants are equally likely to create a tag. The probability $\Pr[r]$, i.e. the probability that row r is used for a tag, is given by

$$\begin{aligned} \Pr[r] &= \sum_{A \in \Gamma} (\Pr[r|A] \times \Pr[A]) \\ &= \frac{1}{\binom{n}{t}} \sum_{\substack{A \in \Gamma \\ r \text{ separates } A}} \frac{1}{s_A} \end{aligned} \tag{4.3}$$

Since s_A is at most ℓ and the sum in (4.3) has n^t/t^t terms by Lemma 4.13,

$$\begin{aligned} \Pr[r] &\geq \frac{1}{\binom{n}{t}} \left(\frac{n^t}{t^t}\right) \frac{1}{\ell} \\ &= \frac{n^t}{\ell t^t \binom{n}{t}}\,. \end{aligned}$$

From Bayes' theorem,

$$\begin{aligned} \Pr[A|r] &= \frac{\Pr[r|A]\Pr[A]}{\Pr[r]} \\ &= \frac{(1/s_A)\left(1/\binom{n}{t}\right)}{\Pr[r]} \\ &= \frac{1}{s_A \binom{n}{t} \Pr[r]\,.} \end{aligned} \tag{4.4}$$

Now recall that $\mu = 1 - \max\{\Pr[A|r] : A \in \Gamma, r = 1, \ldots, \ell\}$. The probability $\Pr[A|r]$ is maximized when the denominator of (4.4) is smallest, i.e. when $s_A = 1$ and $\Pr[r] = \frac{n^t}{\ell t^t \binom{n}{t}}$. Therefore

$$\begin{aligned}\mu &= 1 - \frac{1}{\binom{n}{t}\frac{n^t}{\ell t^t \binom{n}{t}}} \\ &= 1 - \frac{\ell t^t \binom{n}{t}}{\binom{n}{t} n^t} \\ &= 1 - \frac{\ell t^t}{n^t} \ . \end{aligned} \tag{4.5}$$

Recall that for GCA-MAC, $d_{av} = 1 - \ell/\binom{n}{t}$. Since $(\ell t!)/(n^t) \leq \ell/\binom{n}{t} \leq (\ell t^t)/n^t$, the worst-case anonymity of BPHF-MAC is comparable to the average case anonymity of GCA-MAC. If t is fixed, they are asymptotically equal.

Example 4.15. As an example, we compute μ for the $(2, 8)$ BPHF-MAC scheme implemented with the PHF$(3; 8, 2, 2)$ given in Section 4.1.3, which has $\ell = 3$, $n = 2$, $t = 2$:

$$\mu = 1 - \frac{3(2^2)}{8^2} = 1 - \frac{12}{64} = 0.8125 \ .$$

The average anonymity of GCA-MAC, which we computed in Section 4.2.2, was 0.89.

Key Anonymity and Cyclic BPHF

We now consider the key anonymity of BPHF-MAC. In the following, we will require knowledge of s_A values for each A separated by r. This motivates the following definition. Let $\vec{S}_r$ be the length ℓ vector defined as

$$\vec{S}_r(i) = |\{A \in \Gamma : A \text{ is separated by row } r \text{ and } s_A = i\}| \ .$$

$\vec{S}_r$ is the *distribution of separating distances* for t-sets separated by row r. Ignoring the values in $\vec{S}_r$, we prove that a large class of BPHF have $\vec{S}_{r_i} = \vec{S}_{r_j}$ for all rows r_i, r_j. This implies $h_{r_i} = h_{r_j}$, which is a desirable property for anonymity (recall Remark 4.11). The class in question are BPHF constructed from cyclic codes, which we briefly review here.

Definition 4.16. Let $\mathcal{C}$ be a code of length ℓ with symbols from an alphabet Σ, and let $(c_1, c_2, \ldots, c_{\ell-1}, c_\ell) \in \Sigma^\ell$. $\mathcal{C}$ is *cyclic* if

$$c = (c_1, c_2, \ldots, c_{\ell-1}, c_\ell) \in \mathcal{C}$$

implies

$$c' = (c_\ell, c_1, \ldots, c_{\ell-2}, c_{\ell-1}) \in \mathcal{C} \ .$$

The transformation of a codeword from c to c' is called a *cyclic shift*. If a PHF is constructed from a cyclic code, we say it is a *cyclic* PHF.

Important classes of cyclic codes are BCH codes (which include Reed-Solomon codes) and quadratic residue codes. Reed-Solomon codes with large distance are an easily constructed class of cyclic BPHF (see Stinson et al. [161]). The example of Section 4.1.3 is a cyclic BPHF.

Theorem 4.17. *In a cyclic* BPHF$(\ell; n, m, t)$, $\vec{S}_i = \vec{S}_j$ *for all* $i, j \in \{1, \ldots \ell\}$.

Proof. The case $i = j$ is trivial. Without loss of generality, choose a pair (i, j) where $j > i$, $j = i + g$. Define the following two sets of t-sets of columns separated by rows i, and j,

$$X_i = \{A_1, \ldots, A_T\}$$
$$X_j = \{A'_1, \ldots, A'_T\}$$

where $T = (n/t)^t$. The balance property of the PHF (see Lemma 4.13) provides the value of T and proves T is the same for all rows.

Consider $\phi : X_i \to X_j$ and define $\phi(A)$ as the set of columns obtained by cyclically shifting each column in A by g positions. This mapping is well defined, i.e. $\phi(A)$ is a set of columns in the BPHF by the cyclic property, and $\phi(A)$ is separated by row j, since $j = i + g$ and A is separated by row i. We now show that

(i) ϕ preserves s_A values, i.e., $s_A = s_{\phi(A)}$, and

(ii) ϕ is one-to-one.

Property (i) holds since the rows are shifted but not modified, so a row separating (or not separating) A is intact with a different index in $\phi(A)$. Since the separating distance is the same, $s_A = s_{\phi(A)}$. It is clear that $\phi(A_n) \neq \phi(A_m)$ for all $n \neq m$ since $A_n \neq A_m$. Therefore the image of ϕ in X_j has size T, which implies ϕ is one-to-one.

Since the t-sets of columns separated by row i are in one-to-one correspondence with those separated by row j, and they have the same s_A values, $\vec{S}_i = \vec{S}_j$. □

The following theorem is the implication of Theorem 4.17 on anonymity in BPHF-MAC.

Theorem 4.18. *Let* $\mathcal{S}$ *be an instance of the BPHF-MAC scheme constructed with a cyclic BPHF. Then* $\mu_{av}(A_i) = \mu_{av}(A_j) = \mu_{av}(\mathcal{S})$ *for all* $A_i, A_j \in \Gamma$.

Proof. In the case of cyclic codes we can show that all rows are equally likely to be used in a tag. The probability that a given row r is used (recall equation (4.3)) is:

$$\begin{aligned}\Pr[r] &= \frac{1}{\binom{n}{t}} \sum_{\substack{A\in\Gamma \\ r \text{ separates } A}} \frac{1}{s_A} \\ &= \frac{1}{\binom{n}{t}} \left(\frac{\vec{S}_r(1)}{1} + \frac{\vec{S}_r(2)}{2} + \ldots + \frac{\vec{S}_r(\ell)}{\ell} \right) .\end{aligned}$$

This quantity is the same for all rows since $\vec{S}_{r_i} = \vec{S}_{r_j}$ for all r_i, r_j and hence $\Pr[r_i] = \Pr[r_j]$. Substituting $\Pr[r] = 1/\ell$ in (4.4) gives

$$\Pr[A|r] = \frac{\ell}{s_A \binom{n}{t}} ,$$

and h_r can be expressed as follows:

$$h_r = - \sum_{\substack{A\in\Gamma \\ r \text{ separates } A}} \frac{\ell}{s_A \binom{n}{t}} \log_2 \left(\frac{\ell}{s_A \binom{n}{t}} \right) .$$

We will group the terms of this sum by s_A value. There are ℓ possible s_A values, and $\vec{S}_r(i)$ is the number of terms (i.e. sets A) with $s_A = i$. Therefore,

$$h_r = - \sum_{i=1}^{\ell} \vec{S}_r(i) \frac{\ell}{i \binom{n}{t}} \log_2 \left(\frac{\ell}{i \binom{n}{t}} \right) . \tag{4.6}$$

Since the values $\vec{S}_r$ are the same for all rows, h_r is also the same for all rows. This is sufficient, by Remark 4.11, to show that the entropy-based measures of group anonymity are equal. □

While we are not able to compute μ_{av} and h_r for BPHF-MAC when arbitrary BPHF are used, we can assure that, for cyclic BPHF, anonymity will be equitable. Computing $\vec{S}_r$ for large codes/PHF appears to be a difficult problem.

Example 4.19. For the cyclic BPHF$(3; 8, 2, 2)$ given as an example in Section 4.1.3, $\vec{S}_1 = \vec{S}_2 = \vec{S}_3 = (4, 8, 4)$ since there are 4, 8 and 4 sets separated by exactly 1, 2, and 3 rows, respectively. Since this BPHF is cyclic, and $\vec{S}$ is known, we can use (4.6) to compute the

key entropy of any row r:

$$\begin{aligned} h_r &= -\sum_{i=1}^{3} \vec{S}_r(i) \frac{3}{i\binom{8}{2}} \log_2 \left(\frac{3}{i\binom{8}{2}} \right) \\ &= -4\left(\frac{3}{28}\log\frac{3}{28}\right) + 8\left(\frac{3}{56}\log\frac{3}{56}\right) + 4\left(\frac{3}{84}\log\frac{3}{84}\right) \\ &\approx 3.9 \ . \end{aligned}$$

As noted in the discussion following Definition 4.9, four bits is the maximum possible key entropy in this example, therefore BPHF-MAC provides nearly optimal anonymity for all $A \in \Gamma$.

Bounding the Key Entropy

While computing h_r for BPHF-MAC schemes constructed from arbitrary BPHF appears difficult without knowledge of $\vec{S}_r$, we can use the min-entropy of $\Pr[A|r]$ to guarantee a minimum amount of anonymity.

Let X be a random variable defined on a set $\mathcal{X}$ and $\gamma = \max_{x \in \mathcal{X}} \{\Pr[X = x]\}$. The *min-entropy* of X is defined to be

$$H_\infty(X) = \log_2 \left(\frac{1}{\gamma} \right) = -\log_2 \gamma \ .$$

Since $H(X) \geq H_\infty(X)$, the min-entropy gives a lower bound on the Shannon entropy.

With respect to the key anonymity of BPHF-MAC, defined as $H(A|r)$, to compute $H_\infty(A|r)$, we must determine the maximum value of $\Pr[A|r]$ over all authorized sets A and all rows r. Following equation (4.5) in Section 4.3.1, we have

$$\max\{P[A|r] : A \in \Gamma, 1 \leq r \leq \ell\} = \frac{\ell t^t}{n^t},$$

and therefore

$$\begin{aligned} h_r &\geq \log_2 \left(\frac{n^t}{\ell t^t} \right) \\ &= t\log_2 n - \log_2 \ell - t\log_2 t \ . \end{aligned} \tag{4.7}$$

While this lower bound will, in general, be strictly lower than the actual value of h_r, it is useful since it can be easily computed for any instance of BPHF-MAC, without knowledge of $\vec{S}_r$.

Example 4.20. Recall the example PHF$(3; 8, 2, 2)$ given in Section 4.1.3, for which h_r was computed in Example 4.19 as 3.9 bits. Using the bound of (4.7), we can compute

$$\begin{aligned} h_r &\geq 2\log_2 8 - \log_2 3 - 2\log_2 2 \\ &\approx 2.4 \ . \end{aligned}$$

A similar computation gives $h_r \geq 2.75$ for the PHF$(4; 9, 3, 3)$ given in Section 2.4, while $h_r = 4.6$. In Example 4.24, for $C_{2,2}$, $h_r \geq 1$ while $h_r = 1.91$, for $C_{2,6}$, $h_r \geq 7.41$ while $h_r = 9.87$, and for $C_{2,10}$, $h_r \geq 14.68$ while $h_r = 17.92$.

A Code with Known Separating Distance Distribution

Here we describe a simple code for which $\vec{S}_r$ may be computed explicitly. Let $C_{q,\ell}$ be the $(\ell; q^\ell, q)$ *complete code* over q symbols of length ℓ. Note that $C_{q,\ell}$ is cyclic since the cyclic shift of any codeword is another ℓ-tuple of the q symbols, hence it belongs to $C_{q,\ell}$.

Theorem 4.21. *In the code $C_{q,\ell}$ for a coordinate r, the separating distance distribution vector $\vec{S}_r$ defined*

$$\vec{S}_r(i) = |\,\{A : A \subset C_{q,\ell}, |A| = q, A \text{ is separated by } r \text{ and } s_{A,q} = i\}\,|,$$

has the values

$$\vec{S}_r(i) = \binom{\ell-1}{i-1} (q!)^{i-1} \left(q^q - q!\right)^{\ell-i+1} \ ,$$

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

Proof. We wish to count the number of q-sets of codewords separated by coordinate r with separating distance i. The coordinate r is fixed, and contains symbols $1, \ldots, q$. Each q-set will have $i-1$ separating coordinates (excluding r) and $\ell-i+1$ non-separating coordinates. First we must choose which $i-1$ coordinates will be separated, which can be done in $\binom{\ell-1}{i-1}$ distinct ways. Each of the $i-1$ separating coordinates may be chosen in $q!$ ways, since they must be a permutation of $\{1, \ldots, q\}$. The $\ell - i + 1$ non-separating coordinates can be chosen in $q^q - q!$ different ways, since they are all possible assignments minus those which separate the codewords in this coordinate. Taking the product gives the desired result; the number of q-sets of codewords in $C_{q,\ell}$ which are separated in position r and have $i-1$ other separating coordinates. □

The following corollary applies Theorem 4.21 to the case $q = 2$.

Corollary 4.22. *$C_{2,\ell}$ is a cyclic* BPHF$(\ell, 2^\ell, 2, 2)$ *with*

$$\vec{S}_r(i) = \binom{\ell-1}{i-1} 2^{\ell-1}$$

for all rows r and $i = 1, \ldots \ell$.

Proof. Note that $C_{2,\ell}$ is a PHF of strength 2 because all codewords are distinct, and that $C_{q,\ell}$ is cyclic as remarked above. $C_{2,\ell}$ is balanced since it contains all words of length ℓ over the alphabet $\{0, 1\}$. □

The next theorem gives an explicit formula for our entropy-based anonymity measures for the case when $C_{2,\ell}$ is used with the BPHF-MAC scheme, by applying the formulas of Theorem 4.18.

Theorem 4.23. *Let $\mathcal{S}$ be an instance of the $(2, 2^\ell)$ BPHF-MAC scheme implemented with $C_{2,\ell}$. Then*

$$\mu_{av}(A) = \mu_{av}(\mathcal{S}) = h_r = -\sum_{i=1}^{\ell} \binom{\ell-1}{i-1} 2^{\ell-1} \frac{\ell}{i\binom{n}{t}} \log_2 \left(\frac{\ell}{i\binom{n}{t}} \right)$$

for all $A \in \Gamma$ and key indices r.

Proof. Equality of $\mu_{av}(A), \mu_{av}(S)$ and h_r follows from Theorem 4.18. Recall that in the case of cyclic BPHF, following (4.6), we can express h_r as:

$$h_r = -\sum_{i=1}^{\ell} \vec{S}_r(i) \frac{\ell}{i\binom{n}{t}} \log_2 \left(\frac{\ell}{i\binom{n}{t}} \right) . \tag{4.8}$$

By substituting $\vec{S}_r(i)$ with the value given in Corollary 4.22, we arrive at the desired formula. □

Example 4.24. We give a few examples of the group and participant anonymity (μ and ρ) as well as the average anonymity μ_{av}, provided by the $(2, 2^\ell)$ BPHF-MAC scheme implemented with $C_{2,\ell}$. Details of the participant anonymity of BPHF-MAC are given in Section 4.3.2. The column $\mu_{av}(opt)$ gives $\log_2((n/t)^t)$ which is the largest possible value of h_r (see the discussion following Definition 4.9).

ℓ	t	n	μ	$\mu_{av}(\mathcal{S})$	$\mu_{av}(opt)$	ρ
2	2	4	0.5	1.91 bits	2 bits	0.5
6	2	64	0.994	9.87 bits	10 bits	0.968
10	2	1024	0.99996	17.92 bits	18 bits	0.998

4.3.2 Participant anonymity of BPHF-MAC

In this section we determine the anonymity of individual participants. Analysis of the participant anonymity provided by BPHF-MAC is simpler than group anonymity, and relies only on the balance property. BPHF-MACs provide optimal and equitable participant anonymity, as proven in the following theorem.

Theorem 4.25. *The participant anonymity of BPHF-MAC is*

$$\rho(P_j) = 1 - \frac{t}{n}$$

for all $P_j \in \mathcal{P}$.

Proof. First, recall that every row separates $(n/t)^t$ sets of participants (Lemma 4.13). Let P_j be any participant, and r be any row. Given that K_r was used, we evaluate $\Pr[P_j|r]$, the probability that P_j has participated in the creation of a tag using K_r. P_j has some symbol in row r, hence there remain $t-1$ symbols corresponding to participants which can belong to a set including P_j separated by row r. Since our PHF is balanced, each of these symbols occurs (n/t) times in row r. The other $t-1$ symbols/participants can be chosen in $(n/t)^{t-1}$ ways. Therefore,

$$\begin{aligned}
\Pr[P_j|r] &= \frac{|\{A \in \Gamma : P_j \in A,\ \text{row } r \text{ separates } A\}|}{|\{A \in \Gamma : \text{row } r \text{ separates } A\}|} \\
&= \frac{(n/t)^{t-1}}{(n/t)^t} \\
&= \frac{1}{n/t} \\
&= \frac{t}{n}
\end{aligned}$$

Since $\Pr[P_j|r] = t/n$ is the same for all rows and all participants, $\rho(P_j) = 1 - t/n$ for all $P_j \in \mathcal{P}$ as required. □

4.4 GCA Constructions From Arbitrary PHF

When previously discussing constructions of GCAs using PHF, we gave only the connection between $\text{PHF}(\ell; n, t, t)$ and $\text{GCA}(t, n; \ell, \ell)$. This is a restriction, since we require that the number of symbols in the PHF and the strength be equal (both must be t).

It is also possible to construct GCAs from a general perfect hash family, denoted $\text{PHF}(\ell; n, m, t)$, where $m \geq t$. The new structure is only a GCA by some definitions,

due to the following small difference. Some definitions[2] require that $\mathcal{K} = \{K_1, \ldots, K_v\}$ (the set of keys) be a *partition* of $Y = \{k_1, \ldots, k_{\ell m}\}$ (the set of key components). In the construction we are about to describe this is not the case, however all other GCA properties are satisfied. A GCA where the sets of $\mathcal{K}$ are not necessarily disjoint, which we call a *relaxed* GCA, will be shown adequate for the application at hand.

We use the following PHF$(3; 12, 5, 3)$ to illustrate the construction (source: PHFtables [98]).

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
2	4	4	4	0	1	0	2	3	3	3	1
3	1	0	4	0	1	2	0	2	4	3	3
1	4	0	1	3	2	1	2	4	2	3	0

As in the restricted case, each (row, symbol) pair will correspond to a key component. The total number of key components is ℓm in general, and 15 in our example. Instead of having only one key K_r associated to row r, there will be $\binom{m}{t}$ keys; one key for each subset of t symbols in row r. The total number of keys is therefore $\ell\binom{m}{t}$. In the example, each row has a key associated with each of the ten 3-sets of key components:

$$\{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,5\},$$
$$\{1,3,5\}, \{2,3,5\}, \{1,4,5\}, \{2,4,5\}, \{3,4,5\}$$

for a total of $3 \times 10 = 30$ keys. There are n players, and player i is given the key components corresponding to the i-th column, denoted B_i. The notation $K_{r,i}$ refers to the i-th key of row r. The subsets are numbered using m bits in the following canonical way: if symbol j $(1 \leq j \leq m)$ belongs to the subset then bit j is 1, otherwise bit j is zero. For example, $\{1,2,3\}$ is numbered $(00111)_2 = 7$ and $\{2,4,5\} = (11010)_2 = 26$.

Now we verify the GCA properties.

1. For a set $A = \{P_{i_1}, \ldots, P_{i_t}\} \in \Gamma$, columns $i_1, \ldots, i_t$ will be distinct in at least one row, therefore the key components held by A: $\bigcup_{j=1}^{t} B_{i_j}$, will contain at least one $K_{r,i}$. As in the $m = t$ case, A may have one key from multiple rows, but never multiple keys from any row, since in each row they hold only t key components.

2. For a set $A = \{P_{i_1}, \ldots, P_{i_{t-1}}\} \notin \Gamma$, none of the $K_{r,i}$ are held since A holds at most $t-1$ key components from each row, and $|K_{r,i}| = t$ (for all keys).

[2] The definition of GCAs in the literature is inconsistent. In [116] and [110] it is not specified that the K_is be disjoint, but in [114] and [115] the requirement that the K_is be disjoint appears. The fact that the K_is are disjoint is used once in the proof of [115, Theorem 4].

Note that the necessity of the condition that the keys in $\mathcal{K}$ be disjoint depends crucially on how the key components are distributed to participants. The relaxed GCA construction ensures that no participant gets more than one component of any $K_{r,i}$, to preserve the threshold condition.

For an example, take participants P_5, P_7 and P_8. They have key components 0 and 2 in row 1, components 0 and 2 in row 2, and components 1, 2 and 3 in row 3. Therefore, they may use key $K_{3,7}$. Further, this key depends on components held by all three participants and hence this key could not be used by only one or two of the participants.

The only modification required to the BPHF-MAC scheme when using a relaxed GCA is that the index of the key used must be computed differently. The index information in the tag grows from $\log(\ell)$ bits to $\log(\ell) + m$ bits (using the subset numbering scheme given above, but this can be reduced to $t \log(m)$ or fewer bits if desired).

Using a hash family with these parameters for the BPHF-MAC scheme will improve efficiency significantly, as shown in Section 4.4.1, but at the cost of a small reduction in anonymity, discussed in Section 4.4.2.

A note on strong unforgeability. This MAC scheme does not provide *strong unforgeability*, as defined by An et al. [8]. While it is not possible for an adversary to create a valid tag for a new message, it may be possible to create a different tag on a previously authenticated message. Consider a $(2, n)$ scheme that has four key components per row, fix a row, and denote the key components $k_1, \ldots, k_4$. Given $\sigma_1 = F_{k_1}(m) \oplus F_{k_2}(m)$ and $\sigma_2 = F_{k_3}(m) \oplus F_{k_2}(m)$, the tag $\sigma_1 \oplus \sigma_2$ is valid but different for the same message m.

4.4.1 Impact on Efficiency

We make three general remarks with respect to efficiency of BPHF-MAC schemes based on relaxed GCAs.

1. When considering upper bounds for PHF [29], larger n are possible when ℓ and t are fixed, since $n \leq t^{\ell/(t-1)} < m^{\ell/(t-1)}$ when $m > t$.

2. There are a better variety of constructions available, including those from coding theory, that construct cyclic BPHF. For example, the asymptotically optimal explicit constructions of PHF of Wang and Xing [181] do not construct PHF with $m = t$, nor do PHF constructions from Reed-Solomon codes (see Stinson et al. [161]).

3. The potentially large number of keys $\ell\binom{m}{t}$ has no effect on efficiency, since the *number of key components* is only ℓm.

Intuitively, lifting the requirement that all keys K_i be disjoint (as tuples of key components), increases the number keys available for a given number of key components, which reduces the number of such components required.

The comparison relevant for BPHF-MAC is the number of key components for a (t, n) scheme using a PHF$(\ell; n, t, t)$, versus the number of key components when a PHF$(\ell'; n, m, t)$ is used. Asymptotically, the improvement will be at most a constant factor and will depend on a specific construction, since for fixed m and t, $\ell = O(\log n)$.

Example 4.26. We compare the number of key components required for a $(5, 121)$ threshold scheme. Using the PHF construction based on Reed-Solomon codes in [161], we can construct a PHF$(11; 121, 11, 5)$, and BPHF-MAC requires $11 \times 11 = 121$ key components in total, while each participant must store 11 key components. The best construction of a PHF$(\ell, 121, 5, 5)$ on PHFtables [98] has $\ell = 176$. Here, BPHF-MAC requires $176 \times 5 = 880$ key components in total, about 7.3 times more. Each participant must store 176 key components, which is 16 times more than the Reed-Solomon construction.

Example 4.27. We repeat Example 4.26 but using $t = 6$. Using the Reed-Solomon construction we obtain a PHF$(16; 256, 16, 6)$, which is trivially also a PHF$(16; 121, 16, 6)$. The $(6, 121)$ and $(6, 256)$ BPHF-MAC schemes using this PHF require 256 key components in total, while each participant must store 16 key components. The equivalent best known construction of PHF$(\ell; 121, 6, 6)$ from [98] has $\ell = 1160$ for a total of 6960 total key components and a storage requirement of 1160 key components per participant. In the $(6, 256)$ case, the lowest value of ℓ is 1232, giving a total of 7392 key components, and a storage requirement of 1232 key components per participant.

Additional examples with larger t for comparison are difficult to construct due to the lack of direct constructions for PHF$(N; n, t, t)$. If we consider the existence result of Mehlhorn [121], which states that a PHF$(\ell; n, m, t)$ exists when $\ell \geq te^{t^2/m} \log n$, we can make a more general comparison. In the case that $m = \alpha t$, this bound requires $\ell \geq t \sqrt[\alpha]{e^t} \log n$, so the minimum value of ℓ is reduced from $te^t \log n$ to $t \sqrt[\alpha]{e^t} \log n$.

4.4.2 Impact on Anonymity

To determine the worst-case anonymity of BPHF-MAC based on relaxed GCAs (the $m \geq t$ case), we use an approach similar to the one used when $m = t$ in Section 4.3.1. Let r_i denote the i-th key of row r (recall that there are $\binom{m}{t}$ keys per row). By "r_i separates A" we mean that (i) row r separates A, and (ii) A has the i-th t-set of symbols in row r. In other words, A is not only separated by r, but separated by t specific symbols in row r.

Given a group $A \in \Gamma$, A may use s_A keys, and will choose to use one of them with probability $1/s_A$. Therefore,

$$\Pr[r_i|A] = \begin{cases} 0 & \text{when } r_i \text{ does not separate } A \\ \frac{1}{s_A} & \text{when } r_i \text{ separates } A\,. \end{cases}$$

Now we consider $\Pr[r_i]$ the probability that key r_i is used.

$$\begin{aligned} \Pr[r_i] &= \sum_{A \in \Gamma} (\Pr[r_i|A] \times \Pr[A]) \\ &= \frac{1}{\binom{n}{t}} \sum_{\substack{A \in \Gamma \\ r_i \text{ separates } A}} \frac{1}{s_A} \end{aligned} \tag{4.9}$$

The number of $A \in \Gamma$ separated by a given r_i is $(n/m)^t$ in a BPHF, since each of the t symbols in r_i appears n/m times in row r. Since s_A is at most ℓ and the sum in (4.9) has n^t/m^t terms,

$$\begin{aligned} \Pr[r_i] &\geq \frac{1}{\binom{n}{t}} \left(\frac{n^t}{m^t}\right) \frac{1}{\ell} \\ &= \frac{n^t}{\ell m^t \binom{n}{t}}\,. \end{aligned}$$

From Bayes' theorem,

$$\begin{aligned} \Pr[A|r_i] &= \frac{\Pr[r_i|A]\Pr[A]}{\Pr[r_i]} \\ &= \frac{(1/s_A)\left(1/\binom{n}{t}\right)}{\Pr[r_i]} \\ &= \frac{1}{s_A \binom{n}{t} \Pr[r_i]} \end{aligned} \tag{4.10}$$

Now recall that $\mu = 1 - \max\{\Pr[A|r_i] : A \in \Gamma, r_i = 1, \ldots, \ell\binom{m}{t}\}$. The probability $\Pr[A|r_i]$ is maximized when the denominator of (4.10) is smallest, i.e. when $s_A = 1$ and $\Pr[r_i] = \frac{n^t}{\ell m^t \binom{n}{t}}$. Therefore

$$\begin{aligned} \mu &= 1 - \frac{1}{\binom{n}{t} \frac{n^t}{\ell m^t \binom{n}{t}}} \\ &= 1 - \frac{\ell m^t}{n^t}\,, \end{aligned}$$

which corresponds to equation 4.5 in the case $m = t$.

For fixed ℓ, there is clearly a decrease in anonymity. However, as shown in the efficiency discussion, setting $m > t$ reduces ℓ. Therefore, when $m > t$, ℓ decreases while the other term in the numerator increases. We now present two examples, one where anonymity is significantly decreased, and one where it is decreased only slightly.

Example 4.28. Recall the example PHF$(3; 8, 2, 2)$ from Section 4.1.3. Using this PHF, the $(2, 8)$ BPHF-MAC scheme has $\mu = 0.81$. We can replace this PHF with the following PHF$(2; 8, 4, 2)$.

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

This instance of the $(2, 8)$ BPHF-MAC scheme has $\mu = 0.5$.

Example 4.29. Using the same parameters as in Example 4.26, the $m = t$ instance of the $(5, 121)$ BPHF-MAC has $\mu = 0.9999787$ and $h_r \geq 15.53$, while the relaxed instance has $\mu = 0.9999316$ and $h_r \geq 13.84$. (The bounds on h_r are given by the min-entropy of $\Pr[A|r_i]$.)

When we revisit the $(6, 256)$ scheme from 4.26, we find $\mu = 0.99999979$ and $h_r \geq 22.22$ if $m = t$. The $(6, 256)$ scheme based on the relaxed construction has $\mu = 0.99999904$ and $h_r \geq 20$.

The decrease in anonymity can be explained by the relative sizes of the parameters in the examples. In Example 4.28, $|\Gamma|$ is much smaller than in Example 4.29. Also the impact of the change in ℓ in Example 4.29 counterbalances some of the lost anonymity.

Participant Anonymity. An analysis similar to the proof of Theorem 4.25 shows that (t, n) BPHF-MAC schemes constructed with PHF$(N; n, m, t)$ have participant anonymity $\rho(P) = 1 - m/n$ for all participants P. This is a reduction from $1 - t/n$.

4.5 Conclusion

We have strengthened the definition of anonymity in the context of shared symmetric key primitives. Group anonymity is measured in the worst case, and the concept of participant anonymity was introduced. We have presented modified schemes for sharing symmetric key operations with improved group and participant anonymity using balanced perfect hash families. The relaxed GCA construction of Section 4.4 provides a useful trade-off for practical applications, providing large gains in efficiency with only a small decrease in anonymity.

Chapter 5

Group Testing and Batch Verification

We observe that finding invalid signatures in batches of signatures that fail batch verification is an instance of the classical group testing problem. We survey relevant group testing techniques, and present and compare new sequential and parallel algorithms for finding invalid signatures based on group testing algorithms. The parallel algorithms are based on cover-free families. Of the five new algorithms, three show improved performance for many parameter choices, and the performance gains are especially notable when multiple processors are available. The material in this chapter was published in [184].

5.1 Introduction

A *batch verification algorithm* for a digital signature scheme verifies a list of n (message, signature) pairs as a group. It outputs 1 if all n signatures are valid, and it outputs 0 if one or more are invalid. In the most general case, the messages and signers may be different. Batch verification algorithms may provide large gains in efficiency, as verification of the n signatures is significantly faster than n individual verifications. In this chapter, we address the problem of handling batches which fail verification, i.e., finding the invalid signatures which caused the batch to fail.

It has not been previously observed that finding invalid signatures in bad batches is an instance of the *group testing problem*, which in brief, is as follows. Given a set B, of n items, d of which are defective, determine which items are defective by asking queries of the form "Does $B' \subseteq B$ contain a defective item?". Group testing is an old, well-studied problem, for which many algorithms exist. We re-cast some solutions to the group testing problem as solutions to the invalid signature finding problem, which are then compared for efficiency, parallelizability and accuracy. The group testing algorithms are well known, but have not been considered in the context of batch verification by previous work that

has studied methods to find invalid signatures [80, 108, 119, 133, 132]. Performance will be measured by the number of subset tests required to find d invalid signatures.

In total, five new algorithms for finding invalid signatures are presented and included in our comparison. Of these, three give performance improvements. With a single processor, generalized binary splitting [68] gives a modest improvement over the well-known binary splitting algorithm. In the case of two or more processors, large improvements are possible using one of two new group testing-based algorithms: Li's s-stage algorithm [68] and the Karp, Upfal and Wigderson algorithm [101]. The other two algorithms also have interesting properties. The algorithm based on cover-free families is fully parallelizable, and is an improved instance of a known algorithm for batch verification, the id-code algorithm [133] (for some parameter choices). The random matrices algorithm is probabilistic, fully parallelizable and enjoys a simple implementation. Some algorithms require an *a priori* bound on d (this will be addressed in our comparison).

We also give some general results on the limits of group testing that are also interesting in the context of finding invalid signatures in batches, such as the conditions when the naïve testing strategy is optimal.

Contributions and Outline The first contribution of this chapter is describing the link between finding invalid signatures in bad batches and group testing (§5.1.1, 5.1.2), a connection previously overlooked. We then provide a survey of algorithms from the group testing literature, and describe how they correspond to new algorithms for finding invalid signatures (§5.2). These are classified according to the adaptive (i.e., sequential §5.2.2) or nonadaptive (i.e., parallel §5.2.3) nature of the algorithm. We then compare the performance of the new invalid signature finding algorithms (and some previously known algorithms) and determine the best one under various parameter choices (§5.3). For many parameter choices, especially with multiple processors, the new methods outperform previously known methods.

5.1.1 Batch Verification

Let the algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ specify a signature scheme. Gen takes as input a security parameter k, and outputs a signing and verification keypair (sk, pk). $\mathsf{Sign}(sk, m)$ outputs a signature σ on the message m using the secret key sk, and $\mathsf{Verify}(pk, \sigma, m)$ outputs 1 if σ is a valid signature of m under the secret key sk which corresponds to pk, and 0 otherwise.

Here is the most general definition of batch verification.

Definition 5.1 ([46])**.** Let $P_1, \ldots, P_n$ be n signers, with corresponding keypairs $(sk_1, pk_1), \ldots, (sk_n, pk_n)$ output by $\mathsf{Gen}(k)$ for some security parameter k. Let B be a list containing

$pk_1, \ldots, pk_n$, and n tuples of the form (P_{t_i}, σ_i, m_i) called the *batch* (note that the t_i and m_i values may be repeated.) The algorithm $\mathsf{Batch}(B)$ is a *batch verification algorithm* provided $\mathsf{Batch}(B) = 1$ if and only if $\mathsf{Verify}(pk_{t_i}, \sigma, m_i) = 1$ for all i.

A few variations appear in the literature, including the case with a single signer or the case of multiple signers with a single message. We also mention the related concept of *aggregate signatures*. Suppose $\sigma_1, \ldots, \sigma_n$ are signatures on messages $m_1, \ldots, m_n$ with corresponding verification keys $pk_1, \ldots, pk_n$. An aggregation algorithm is a public algorithm, which given the σ_i, m_i and pk_i $(i = 1, \ldots, n)$ outputs a compressed signature σ. An associated verification algorithm verifies if σ is a valid compressed signature, given pk_i and m_i (for $i = 1, \ldots n$).

A number of signature schemes in the literature support batch verification. Batch cryptography was introduced by Fiat [81, 82] to improve efficiency of an RSA-like scheme, where large numbers of operations are performed at a central site. History shows that secure batch verification algorithms are tricky to construct; a number of schemes were presented and subsequently broken or shown to be otherwise flawed. One example is the scheme of Al-Ibrahim et al. [1], which was broken by Stinson in [160]. Camenisch et al. list and reference ten proposed schemes which were later broken [46, §1.2]. Despite this poor track record, a number of signature schemes have batch verification, many of them based on the general techniques described in Bellare et al. [18].

Of the techniques in [18], the *small exponents test* is commonly used to create batch verification algorithms. It works as follows. An instance consists of g and $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where g and the y_i are elements of a group G of order q, and the x_i are elements of $\mathbb{Z}_q$. The algorithm outputs 1 if $y_i = g^{x_i}$ for $i = 1, \ldots, n$. All n inputs are checked in the following comparison:

$$g^{\sum_{i=1}^{n} x_i d_i} \stackrel{?}{=} \prod_{i=1}^{n} y_i^{d_i} \ ,$$

where the d_i values are randomly chosen ℓ-bit integers. Since this is correct with probability $2^{-\ell}$ (proven in [18]), ℓ may be relatively small, and the test is efficient.

We list a few examples of signature schemes with batch verification, but omit details since the techniques in this work will apply to any scheme with batch verification. RSA* is an RSA-variant with batch verification presented by Boyd and Pavlovski [39]. DSA** is a signature scheme based on DSA, given by Naccache et al. [126], which uses the small exponents test from [18]. Camenisch et al. [46] give a variant of the Camenisch-Lysyanskaya signature scheme [47] which supports batch verification, present a batch verifier for the Π-IBS scheme of Chatterjee and Sarkar [52], and discuss batch verification of BLS signatures [35]. Practical considerations and implementation timings of batch verification are given in Ferrara et al. [80].

5.1.2 Finding Invalid Signatures in Bad Batches

Suppose we are given a batch B such that $\mathsf{Batch}(B) = 0$. We know that B contains at least one invalid signature, but what is the best way to determine *which* of the signatures do not verify? Verifying each signature individually is certainly an option, but can Batch be applied to subsets of B to perform less work overall? This problem can be considered as the computational version of the batch verification problem (which is a decision problem). We name it the *invalid signature finding* (ISF) problem. This does not apply to aggregate signatures, where, since the batch is compressed, we do not have enough information to determine which of the original signatures were invalid.

We will treat the algorithm Batch as a generic test for invalid signatures, and present solutions which work for *any signature scheme* equipped with a Batch function as described in Definition 5.1. There are several advantages of generic ISF algorithms.

1. *Applicability.* A generic ISF algorithm may be used with any signature scheme which provides batch verification. This includes future schemes.

2. *Implementation.* A single implementation may be used to locate bad signatures of multiple signature schemes, reducing the need to maintain multiple ISF algorithm implementations. The single generic ISF algorithm may be optimized, verified and otherwise improved since the effort is amortized over a larger number of applications.

3. *Ability to handle variations of the ISF problem.* The group testing literature has considered many variations of the problem, many of which are applicable to variations of the ISF problem. As examples, group testing with competitive algorithms [68, Ch. 4], or when the size of each test group is restricted [70, 141], or with unreliable tests [68, Ch. 5], all correspond to interesting variations of the ISF problem.

The performance of an ISF algorithm will be evaluated based on the number of calls to Batch and the parallel performance of the algorithm (this is discussed further in Sections 5.2 and 5.3).

Related Work

There have been five papers addressing the ISF problem. The first two are by Pastuszak et al. [132, 133]. They consider a generic Batch function for a signature scheme and study the *divide-and-conquer* method of finding bad signatures in [132]. The divide-and-conquer verifier was originally described in [126] under the name *cut and choose*, and is referred to as *binary splitting* in the group testing literature. In brief, a batch B is divided in half, then Batch is recursively called on each sub-batch, until 1 is output (this sub-batch contains only

valid signatures) or until the sub-batch has size one, which identifies the bad signatures. This method was implemented in the work of Ferrara et al. [80], and we discuss their findings in §5.2.2 when we relate the divide-and-conquer verifier to well-known techniques from group testing.

The second paper [133] approaches the problem using identification codes (id-codes), a code which encodes an ISF algorithm, by specifying subsets of B to test with Batch in such a way that all bad signatures may be identified. This approach is an instance of well-known non-adaptive group testing algorithms based on cover-free, separable and disjunct matrices, discussed in §5.2.3. A limitation of [132, 133] is that either the number of bad signatures in a batch, or a bound on the number of bad signatures is required *a priori*. This is common to most group testing algorithms as well.

The work of Law and Matt [108] improves the divide-and-conquer method by considering the details of the signature scheme. The second part of [108] gives an improved invalid signature finder using a special version of Batch. The batch verification and invalid signature finding tasks are combined, to allow information and intermediate computations from the verification step to be used in the ISF step. This trades off general applicability for improved computational efficiency. Along similar lines, Matt improves the performance of these methods when the number of invalid signatures is large [119]. This addresses a limitation of [108]. The improved techniques of [119] are applicable to the Cha-Cheon signature scheme [51] and the pairing-based schemes discussed in Ferrara et al. [80].

5.2 Group Testing-Based ISF Algorithms

We begin with a general description of the group testing problem called the (d, n)-*problem*. Consider a set of n items which contains exactly d defective items, called the *defective set*. Identification of a defective item requires the application of an error-free test, and we may test an arbitrary subset of the items. The test outcome may be *positive* if the subset of items contains at least one defective item, or *negative* if no defective items are present in the subset. An algorithm A which finds all d defective items is a solution to the problem. An algorithm where the tests are applied sequentially, and subsequent tests depend on the results of previous tests is called an *adaptive algorithm*. *Nonadaptive algorithms* require all tests to be specified at the outset; hence they may be executed in parallel.

Group testing has a long history, originating in World War II, motivated by the task of testing blood samples of draftees to detect syphilis [66, 68]. In this application, a single test on a combination of blood samples will return positive if any of the samples would test positive for syphilis. Since there were only a few thousand cases of the disease in millions of draftees, large subsets would come back negative, saving many individual tests. Group testing later found many industrial applications, a line of research initiated by

Sobel and Groll [152]. In the past 50 years or so, a large literature has grown around the problem, and many variants have been considered. The book of Du and Hwang [67, 68] is a comprehensive reference.

It should now be clear that the ISF problem is a group testing problem: the items are signatures, the test applied to subsets is the batch verification algorithm, and the defectives are invalid signatures. This basic model makes the following assumptions:

- The subset tests all have the same cost, regardless of the number of items being tested.
- The number of defectives d, or a bound on d, is known *a priori*.

The first assumption, which is standard in the group testing literature, is a simplifying assumption for the ISF problem, since the cost of $\mathsf{Batch}(B)$ is typically composed of a fixed overhead cost independent of $|B|$, plus a variable cost which grows with $|B|$. The fixed cost is typically high (e.g. an exponentiation) while the variable cost consists of $|B|$ cheaper operations (e.g. multiplications). This assumption does however, allow us to keep our analysis general, and ignore the details of Batch. The second assumption allows some group testing algorithms to be more efficient. We will discuss the importance of the bound on d for each algorithm, and the behaviour of the algorithm when d is initially bounded incorrectly.

Probabilistic group testing (PGT) assumes a probability distribution on the defective set, while combinatorial group testing (CGT) does not. The only information CGT assumes about the defective set is that it is a d-subset of the n items. Some applications of batch verification may benefit from PGT if it is reasonable to make an assumption about the distribution of invalid signatures; however, we do not consider PGT algorithms.

Denote the minimal number of calls to Batch required to find d invalid signatures in a batch of size n by $M(d, n)$. First note that $M(d, n) \leq n - 1$, by verifying $n - 1$ signatures individually and inferring the validity of the last signature from knowledge of d and the other $n - 1$ signatures. The following general lower bound is proven in [68, Corollary 2.1.11].

Theorem 5.2. $M(d, n) \geq \min\left\{n - 1, 2\ell + \left\lceil \log \binom{n-\ell}{d-\ell} \right\rceil\right\}$ *for* $0 < \ell \leq d < n$.

Unless stated otherwise, $\log x$ is the base two logarithm of x, $\ln x$ is the natural logarithm of x, and e is the natural base.

5.2.1 Individual Testing

The simplest way of identifying all invalid signatures in a bad batch is to individually verify each signature. The question is, when is this naïve testing strategy optimal? Recall that

$M(d, n)$ is the smallest possible number of tests for any (d, n) algorithm. Combining [68, Theorem 3.5.1] and [68, Theorem 3.5.3], we have the following result.

Theorem 5.3. *Let d be the number of invalid signatures in a batch of size n, and let $M(d, n)$ be as defined above. Then*

$$M(d, n) < n - 1 \text{ for } n > 3d, \text{ and}$$
$$M(d, n) = n - 1 \text{ for } n \leq 2.625d.$$

Therefore, when the number of bad signatures is at most $n/3$ it is possible to do better than individual testing, and when there are more than $n/2.625$ bad signatures the naïve strategy is optimal. What is best when $n < 3d$ and $n \geq 2.625d$ remains unknown; however, Hu, Hwang and Wang [97] conjectured that individual testing is optimal whenever $n \leq 3d$.

We note that individual testing is trivially parallelizable.

5.2.2 Adaptive ISF Algorithms

In this section we will present some adaptive ISF algorithms, based on group testing algorithms. In adaptive (or sequential) algorithms, the results of each test determines the items to be tested in subsequent tests. We will use the notation (d, n), where d is an upper bound on the number of bad signatures in the batch of size n.

Binary Splitting

An adaptive group testing algorithm is naturally represented as a binary tree. Nodes of the tree contain elements to be tested, starting at the root, which contains all n items. In binary splitting, at each level of the tree, we halve (i.e. divide as evenly as possible) the set of items in the parent node, to create two child nodes. When a test returns negative, this node becomes a leaf, since we know the set of items at this node is valid. Repeating this process recursively, we ultimately end up with nodes containing a single item, thus identifying the invalid items of the batch. By using depth-first search from the root of the tree we may locate an invalid item using at most $\lceil \log(n) \rceil$ tests. We may remove the invalid item, and repeatedly apply the binary splitting algorithm to find d invalid items using at most $d \lceil \log(n) \rceil$ tests.

An implementation of binary splitting for the BLS signature scheme [35] is discussed in the work of Ferrara et al. [80]. They performed experiments with $n = 1024$ and they found binary splitting was faster than individual verification when $d < 0.15n$. In these experiments, a random fraction of the batch was corrupted, however Ferrara et al. note

that in practice if corrupted signatures occur in bursts, the binary splitting algorithm will have better performance. Ordering of the batch may be an important consideration for applications using binary splitting.

A variant of binary splitting is Hwang's *generalized binary splitting*. The intuition of the algorithm is that there is roughly one defective item in every n/d items, and therefore a group smaller than $n/2$ could be tested and a defective found with fewer tests. When $d = 1$ the number of tests required by generalized binary splitting is $\lfloor \log(n) \rfloor + 1$, and when $d \geq 2$, the number of tests is not more than $d - 1 + \left\lceil \log \binom{n}{d} \right\rceil$, which gives a noticeable saving as d gets larger [68, Corollary. 2.2.4].

Karp, Upfal and Wigderson describe an algorithm to identify a single invalid item using p processors in at most $\lceil \log_{p+1} n \rceil$ parallel tests [101]. The algorithm is identical to binary splitting when $p = 1$, since it uses a $(p+1)$-ary tree in the same way that binary splitting does. At each level, p of the child sets are tested in parallel, and (if necessary) the validity of the $(p+1)$-th set is inferred. We may repeatedly apply this algorithm to identify d invalid items in at most $d \lceil \log_{p+1} n \rceil$ parallel tests. We will refer to this algorithm as the *KUW algorithm*.

Li's s-Stage Algorithm

This algorithm has s rounds of testing, identifying good items at each round, until the last round when the algorithm corresponds to individual testing. Li's algorithm begins by grouping the batch into g_1 groups of size k_1 (some groups might have $k_1 - 1$ items). The groups are tested, and items in valid groups are set aside. The i-th stage divides the remaining elements into g_i groups of size k_i, tests them, and then removes items in valid groups. The final stage has $k_s = 1$, and remaining items are identified as valid or invalid.

When optimal choices are made for g_i, k_i and s (see [68, §2.3]), the number of tests is not more than

$$\frac{e}{\log e} d \log\left(\frac{n}{d}\right) .$$

When p processors are available, Li's algorithm may be parallelized (see [67, p. 33]), and the number of parallel tests is not more than

$$\frac{e}{\log e} \frac{d}{p} \log\left(\frac{n}{dp}\right) + \ln\left(\frac{n}{dp}\right) + d .$$

5.2.3 Nonadaptive Algorithms

As we have seen, some adaptive algorithms are somewhat parallelizable. All nonadaptive algorithms are completely parallelizable. Recall that nonadaptive tests may be completely

specified without information from previous tests. This can be especially useful for online batch verification in a system with time constraints where a batch of n signatures arrive every time interval and must be processed before the next batch arrives, with a known number of tests. This might be applicable in the example of public key authentication in vehicular networks (this example is discussed in [46, 80]) or authentication of data reported periodically from sensors (as discussed in [45]). We continue to use the (d, n) notation defined at the beginning of Section 5.2.

Nonadaptive Group Testing with Cover-Free Families

A useful combinatorial structure for designing nonadaptive CGT (NACGT) algorithms is a cover-free family, introduced in Section 1.2.3. Cover-free families are also studied under the terms *disjunct matrices* [68], *binary superimposed codes* [104], and *strongly selective families* [54]. Stinson et al. [161] discusses relations between these structures. We choose the language of cover-free families since they have found multiple applications in cryptography (see [85, 124, 155] for examples). The following definition of a CFF is easily seen to be equivalent to Definition 1.7, however, this formulation is convenient when discussing group testing.

Definition 5.4. A *d-cover-free family* is an $N \times n$ binary matrix, with $n \geq d+1$, such that for any set of columns C and single column c such that $|C| = d$ and $c \notin C$ the following property holds. Let $U(C)$ be the binary OR of the columns in C. The cover-free property ensures that $c \notin U(C)$, that is, c is 1 in at least one position where $U(C)$ is 0. We will use the notation d-CFF(N, n) for cover-free families.

The cover-free property ensures that no d-set of columns "covers" any other column. A *d-separable* matrix satisfies a weaker property, namely, the OR of any two sets of d columns are distinct. While any d-separable matrix yields a NACGT algorithm, it is not efficient [68, Ch. 7]. We now describe how a d-CFF(N, n) defines an efficient (d, n) NACGT algorithm.

Input: Signatures $\sigma_1, \ldots, \sigma_n$, batch verification function Batch.
Output: Up to d invalid signatures.

1. Construct a matrix A which is a d-CFF(N, n).
2. Associate σ_i to column i of A. Each row of A will define a sub-batch to test; if σ_i has a 1 in row j then σ_i is included in sub-batch j.
3. Compute $\mathsf{Batch}(B_1), \ldots, \mathsf{Batch}(B_N)$ where $B_i = \{\sigma_j : A_{i,j} = 1\}$.

4. For each row i such that $\mathsf{Batch}(B_i) = 1$ mark all $\sigma_j \in B_i$ as valid.

5. Output all the remaining signatures as invalid, i.e., signatures which do not belong to a valid batch.

We now explain how the algorithm correctly identifies valid signatures (and thus correctly outputs invalid signatures in step 5). Suppose σ_i is a valid signature. Let C be the set of columns corresponding to the invalid signatures. We are assuming that $|C| \leq d$. Let C' be any set of d columns that contains C as a subset and does not contain i (C' exists because $n \geq d+1$). Since A is the matrix of a d-CFF(N, n), there exists a row j such that $A_{j,i} = 1$ and $A_{j,c} = 0$ for all c in C'. Therefore $\mathsf{Batch}(B_j) = 1$ and σ_i is recognized as a valid signature in step 4 of the algorithm.

Remark 5.5. Shultz makes the following observation for batches containing $d' > d$ invalid signatures [149]. Let B' be the resulting set of signatures after removing all the signatures belonging to valid sub-batches, in step 4. If $|B'| > d$, the number of invalid signatures in the input batch exceeds d. In this case some valid signatures may be covered by $U(D)$, but are not present in a valid test. Thus B' contains all d' invalid signatures, but may contain some valid signatures as well.

The number of rows, N, in the matrix representation of a d-CFF(N, n) gives the number of tests required in the group testing algorithm given. The bounds on N given in Section 1.2.3 indicate how well (at best) we can expect CFF-based nonadaptive group tests to perform. It is immediately clear that the nonadaptive feature comes at a cost, since the number of tests will always be larger than $d\lceil \log(n) \rceil$, the number of tests required by binary splitting (c.f. 5.2.2).

Using the optimal 1-CFF construction given in Section 1.2.3 gives the following nonadaptive ISF.

Theorem 5.6. *In a batch of n signatures, a single bad signature may be identified using N parallel tests, provided*

$$\binom{N}{\lfloor N/2 \rfloor} \geq n \ .$$

Therefore, 15, 20 and 25 tests may identify a single bad signature in a batch of size at most 6435, 184756 and 5200300, respectively.

A recent paper of Porat and Rothschild [137] explicitly constructs (n, d)-*strongly selective families* from error correcting codes. This structure is equivalent to a $(d-1)$-CFF(N, n) (see [54]), and hence it gives a nonadaptive ISF.

Theorem 5.7 ([137, Theorem 1])**.** *It is possible to construct a d-CFF(N, n) with $N = \Theta((d+1)^2 \log n)$ in $\Theta((d+1)n \log n)$ time.*

In light of the bounds on N given in Section 1.2.3, this construction is asymptotically optimal. We choose to ignore the constant hidden by the Θ-notation, as even with this assumption the CFF algorithm is outperformed by other methods.

Nonadaptive Group Testing with id-codes

The definition of identification codes is very general: any binary matrix which specifies a group testing algorithm is an id-code. Thus CFF are id-codes, and the d-separable property defined earlier in Section 5.2.3 is both necessary and sufficient for an id-code. The construction of id-codes put forward in Pastuszak et al. [133] is a cover-free family with some additional constraints on the number of nonzero row and column entries. Using their construction gives the following ISF.

Theorem 5.8 ([133, Corollary 4])**.** *The number N of tests necessary to identify d bad signatures in a batch of size n satisfies $N \leq (d+1)\sqrt{n}$.*

Clearly, as $n \to \infty$ for fixed d, this method will require a much larger number of tests than CFF-based methods, since $\sqrt{n}$ dominates $\log n$. However, the CFF constructions presented have a quadratic dependence on d, while d is linear in Theorem 5.8. Therefore, for fixed n and increasing d, there will be a crossover point after which the id-code ISF outperforms the CFF ISF. Comparing the formulas,

$$\begin{aligned}(d+1)^2 \log(n) &< (d+1)\sqrt{n} \\ d &< \frac{\sqrt{n}}{\log n} - 1 \ .\end{aligned}$$

This gives the value of d in terms of n before which the CFF ISF outperforms the id-code ISF. For example, when $n = 10^3, 10^4, 10^5, 10^6$, d must be greater than $2, 6, 18, 49$ (resp.) for the id-code ISF to be more efficient.

Random Matrices

In this section we describe a probabilistic nonadaptive ISF which is based on a random matrix, and fails with a given probability. Du and Hwang give the probability that a random matrix is a d-CFF.

Theorem 5.9. *Let C be a random $N \times n$ binary matrix where $C_{i,j} = 1$ with probability $q = 1/(d+1)$. Then C is a d-CFF(N, n) with probability at least*

$$(d+1)\binom{n}{d+1}\left[1 - q(1-q)^d\right]^N \ .$$

Proof. Let D be a set of d columns of C, and let c a single column. In a single row, the probability that $c = 1$ and $D = 0, \ldots, 0$ is $q(1-q)^d$. (Note that $q = 1/(d+1)$ maximizes this probability.) The probability that this pattern does not occur in any of the N rows is $\left[1 - q(1-q)^d\right]^N$. Since the $d+1$ columns of D and c may be chosen in $(d+1)\binom{n}{d+1}$ ways, this gives the bound on the probability that C is a CFF stated in the theorem. □

Now we consider constructing an ISF as described at the beginning of Section 5.2.3 using random matrices. Certainly, this approach would succeed with probability at least that given by Theorem 5.9. However, the ISF will have significantly better performance, since the only case that affects our result is when the d columns corresponding to the bad signatures cover another column. If this occurs, then the covered column may be valid, but it will not appear in a valid test. Columns corresponding to valid signatures which cover each other will have no effect on the ISF. Therefore, we need only consider the probability that a *fixed set of d columns covers another column.* Since the d columns corresponding to defectives are fixed with respect to a batch, the remaining column may be chosen in $n - d$ ways, which gives the following result. The same improvement may be used in DNA library screening (see [68, Theorem 9.3.3] and [12]).

Theorem 5.10. *There exists an ISF which identifies d defectives in a batch of size n using N tests with failure probability $P_{d,n} \leq (n-d)\left[1 - q(1-q)^d\right]^N$, where $q = 1/(d+1)$.*

Remark 5.11. The error of this ISF is one-sided. It may output a valid signature as invalid. To detect this, we must individually test the output signatures, to confirm that they are invalid.

5.3 Comparison of Algorithms

In this section we compare the ISF algorithms given in Section 5.2. We compare them based on the number of tests, and their behaviour when d (the number of defectives) is unknown, or estimated incorrectly. Finally we discuss how the ISFs given by Law and Matt [108, 119] for a specific class of signature schemes compare to the generic ISF algorithms given in this thesis.

5.3.1 Number of Tests

First, for each of the ISF algorithms in Section 5.2, we give the bound on the worst case number of calls to Batch (Table 5.1). Table 5.1 gives the bound for the trivial parallelization of (generalized) binary splitting: divide the original batch into p equal-sized sub-batches. The KUW algorithm is a better parallelization of binary splitting. For generalized binary

Method	Sec.	Tests (worst case)	Tests with p processors
Individual Testing	5.2.1	$n-1$	$\lceil n/p \rceil - p$
Binary Splitting (B.S.)	5.2.2	$d\lceil \log n \rceil$	$d\left\lceil \log\left(\frac{n}{p}\right)\right\rceil$
Gen. Bin. Splitting (G.B.S)*	5.2.2	$d-1+\left\lceil \log\binom{n}{d}\right\rceil$	$d-1+\left\lceil \log\binom{n/p}{d}\right\rceil$
Li's s-stage*	5.2.2	$\frac{e}{\log e} d \log \frac{n}{d}$	$\frac{e}{\log e}\frac{d}{p}\log\frac{n}{dp}+\ln\frac{n}{dp}+d$
PR CFF*	5.2.3	$(d+1)^2 \log n$	$((d+1)^2 \log n)/p$
PPS id-codes* [133]	5.2.3	$(d+1)\sqrt{n}$	$((d+1)\sqrt{n})/p$
KUW	5.2.2	$d\lceil \log_2 n \rceil$	$d\lceil \log_{p+1} n \rceil$

Table 5.1: Summary of the number of tests required for the ISF algorithms presented in §5.2. The number of tests required by the random matrices ISF must be computed using Theorem 5.10. "PR CFF" is the ISF based on Theorem 5.7, and "PPS id-codes" is the ISF in Theorem 5.8. The algorithms marked with an asterisk (*) require an *a priori* bound on d.

splitting, the bounds given hold for $d \geq 2$, while for $d = 1$ the number of required tests is $\lfloor \log n \rfloor + 1$.

Next we compare the number of tests required by each method for various choices of n, d, and p (the number of processors available). In Ferrara et al. [80], the choices $n = 1024$, $d = 1, \ldots, 153$ were used when investigating the practical performance of the binary splitting method. In Pastuszak et al. [132], choices of $n \in [16, 1024]$ are used to give the average number of tests for the binary splitting method when $d = 1, \ldots, 16$. In Law and Matt [108], tables are given with $n = 2^4, 2^6, 2^8, 2^{10}, 2^{12}$ and $d = 1, \ldots, 4$. In Matt [119], the parameters chosen for comparison are $n = 2^4, 2^6, 2^8, 2^{10}$ and $d = 1, \ldots, n$ (here the goal was to show better performance with large d). All previous work considered $p = 1$, i.e., a single processor. We will compare the ISF algorithms with $n = 10^3, 10^4, 10^5, 10^6$, $d = 1, 2, 3, 4, 10$ and $p = 2, 4, 8, 16$. When $p = 1$ the algorithm requiring the fewest tests is always generalized binary splitting, and for smaller values of d, binary splitting performs equally well. Table 5.2 lists the algorithm requiring the fewest number of tests when $p \geq 2$ (according to the bounds in Table 5.1). A finer grained comparison is given in Section 5.4, where Tables 5.4, 5.5 and 5.6 give the actual number of tests required under various combinations of parameters.

n	d	Fewest Tests when $p =$			
		2	4	8	16
10^3	4	KUW	LI	LI	LI
10^4	4	KUW	KUW	LI	LI
10^5	4	KUW	KUW	LI	LI
10^6	4	KUW	KUW	KUW	LI
10^3	10	LI	LI	LI	LI
10^4	10	KUW	LI	LI	LI
10^5	10	KUW	LI	LI	LI
10^6	10	KUW	LI	LI	LI

Table 5.2: Algorithm requiring the fewest number of tests with p processors. The number of tests required by all algorithms listed in Table 5.1 is given in Tables 5.5 and 5.6. Here, LI stands for Li's Algorithm (§5.2.2).

Discussion In the case of a single processor (Table 5.4) we find that the adaptive algorithms have the best performance. In particular, generalized binary splitting slightly outperforms binary splitting, especially as d grows. With a single processor the KUW algorithm has the same performance as binary splitting, hence we have omitted it from the table.

When two or more processors are available to the ISF (Tables 5.2, 5.5 and 5.6), Li's s-stage algorithm and the KUW algorithm begin to show the best performance. The performance gap is most pronounced as the number of processors grows for any of the choices of (n, d) presented. In general, the nonadaptive algorithms improve when more processors are available, as they provide a speedup linear in the number of processors. Regarding the nonadaptive algorithms, the PR CFF algorithm (Theorem 5.7) requires fewer tests than the PPS id-code algorithm (Theorem 5.8) when $d < \sqrt{n}/\log n - 1$. If a failure probability of at most $P_{d,n} = 0.001$ is tolerable (see Remark 5.11 and Theorem 5.10), the random matrix ISF (RM ISF) outperforms the CFF and id-codes methods since it requires a weaker property from the matrix, as discussed following Theorem 5.9. The RM ISF with failure probability 0.001 is best overall when $p = 16$, $d = 4$ and $n = 10^4, 10^5, 10^6$ (see Section 5.4). However, determining whether the RM ISF has failed requires each of the output items to be individually verified.

In the detailed tables of Section 5.4, there are many parameter combinations where multiple ISFs require a nearly equal number of tests. In these cases, implementation factors, average case performance, and the size of subset tests may influence the best choice.

Algorithm	When $d' < d$	When $d' > d$
B.S.	Outputs d' invalid signatures in time $M_{\text{B.S.}}(d', n)$.	
G.B.S., Li	Outputs d' invalid signatures but using sub-optimal parameter choices thus requiring extra work.	
KUW	Outputs d' invalid signatures in time $M_{\text{KUW}}(d', n)$.	
CFF, id-codes	returns d' invalid signatures	returns a set of $d \leq \ell \leq n$ potentially invalid signatures
RM	Outputs d' signatures in $M_{\text{RM}}(d, n)$ tests	Outputs d bad signatures with prob. $P_{d,n}$ and d' bad signatures with probability $P_{d',n}$ (see Theorem 5.10)

Table 5.3: Behaviour of ISFs when the true number of invalid signatures d' differs from the estimated number d. Here, $M_A(d, n)$ represents the number of tests required by algorithm A for a batch of size n with d defectives.

5.3.2 Unknown Number of Invalid Signatures

Table 5.3 lists the behaviour of each of the algorithms when the true number of signatures, is d', a value different from our estimate d.

The binary splitting algorithm has a certain grace with respect to handling arbitrary d, in that the algorithm's behaviour is unchanged, and the bound on the number of tests holds as d changes. On the other hand, Li's s-stage algorithm, and generalized binary splitting begin by computing some parameters based on n and d in order to meet the performance bound stated in Table 5.1. If a batch contains $d' \neq d$ invalid signatures these parameters will not be chosen optimally, and it is unclear to what extent this will hurt the performance of the algorithm. It is also unclear whether better performance is obtained by underestimating or overestimating d'. Therefore, if no *a priori* information about d is available, the best choice is binary splitting when $p = 1$, and KUW when $p > 1$.

When a batch contains $d' > d$ invalid signatures, the CFF and id-code algorithms output a set B' of ℓ signatures, where $d < \ell \leq n$. All d' defectives are in B'; however, it may contain valid signatures as well. As d' increases, ℓ will increase as well, and less information is gained. The case $d' > d$ is easily recognized (if $|B'| > d$), and we may restart the ISF with a larger estimate of d.

The random matrix ISF outputs each $d' > d$ with probability $P_{d',n}$, given in Theorem 5.10. For these algorithms we may run N tests to identify some valid signatures, remove them from the batch, re-estimate d, and re-run the ISF.

Another option when d is unknown is to use a *competitive algorithm*, i.e., one which assumes no *a priori* information about d, yet completes in a bounded number of tests (see [68, Ch. 4]). For example, the "jumping algorithm" of Bar-Noy et al. [13], identifies d invalid signatures in at most $1.65d(\log \frac{n}{d} + 1.031) + 6$ tests, for $0 \leq d \leq n$. Note that this flexibility comes at a cost because the performance of a competitive algorithm when d is known to be small is poorer than the other ISFs presented.

5.3.3 Comparison to Non-Generic ISF Algorithms

Recall from Section 5.1.2 that a non-generic ISF is an ISF which is customized to a particular signature scheme, integrated into the Batch algorithm. In the single processor setting, the ISFs requiring the fewest number of tests were binary splitting and generalized binary splitting. Since the non-generic ISF given by Law and Matt [108, 119] outperforms binary splitting, their ISF will outperform the generic ISF algorithms presented here (for the pairing-based signature schemes to which it applies).

The faster choice in the parallel case would depend on how well the specialized ISFs described by Law and Matt parallelize. If their improved version of binary splitting yields an improved version of the KUW test (which seems possible, since KUW and binary splitting are similar), then the parameter combinations where KUW is the best may be improved upon.

A general comparison is beyond the scope of this work since the units are different: number of calls to Batch() (this work) vs. number of multiplications in a finite field (Law and Matt).

5.4 Comparison Details

Table 5.4 gives the number of tests required by each algorithm when $p = 1$, with varying n and d, while Tables 5.5 and 5.6 fix $d = 4$ and $d = 10$ respectively, with varying n and p.

5.5 Conclusion

We have introduced algorithms based on group testing for finding invalid signatures in bad batches. For many parameter choices, and especially with multiple processors, the

Method	$n = 10^3$, $d =$					$n = 10^4$, $d =$				
	1	2	3	4	10	1	2	3	4	10
Binary Splitting	10	20	30	40	100	14	28	42	56	140
Gen. Bin. Splitting	10	20	30	39	87	14	27	40	52	121
Li's s-stage	18	33	47	60	125	25	46	66	85	187
PR CFF	13	89	159	249	1205	16	119	212	332	1607
PPS id-codes	63	94	126	158	347	200	300	400	500	1100
Random Matrices	49	87	124	162	387	57	101	145	189	452
	$n = 10^5$					$n = 10^6$				
Binary Splitting	17	34	51	68	170	20	40	60	80	200
Gen. Bin. Splitting	17	34	50	65	154	20	40	60	79	187
Li's s-stage	31	58	84	110	250	37	71	103	135	312
PR CFF	20	149	265	415	2009	23	179	318	498	2411
PPS id-codes	632	948	1264	1581	3478	2K	3K	4K	5K	11K
Random Matrices	65	115	166	216	517	73	130	186	243	581

Table 5.4: Table showing the number of tests required by each group testing algorithm from Table 5.1 when $n = 10^3, 10^4, 10^5, 10^6$ and $d = 1, 2, 3, 4, 10$. For random matrices a success probability of 99.9% is required.

new methods outperform known methods. Our comparison shows that the best algorithm depends strongly on the choice of parameters, and no single algorithm is best in all cases. One way to more precisely compare these algorithms, while still maintaining some generality, would be to count the number of calls to Batch() and the size the of input to each, then assign values to the fixed and variable cost, depending on the underlying Batch() function, to arrive at a final performance number. Other topics for future work include: i) comparison of implementations to compensate for not considering the sizes of sub-batches, and ii) specializing the given ISFs to specific signature schemes, perhaps by using techniques from Law and Matt's specialized ISFs for pairing-based signature schemes.

Method	$d = 4$							
	$n = 10^3$, $p =$				$n = 10^4$, $p =$			
	2	4	8	16	2	4	8	16
Binary Splitting	36	32	28	24	52	48	44	40
Gen. Bin. Splitting	35	31	27	23	48	44	40	36
KUW	28	20	16	12	36	24	20	16
Li's s-stage	35	19	12	8	49	27	17	12
PR CFF	125	63	32	16	166	83	42	21
PPS id-codes	79	40	20	10	250	125	63	32
Random Matrices	81	41	21	11	95	48	24	12
	$n = 10^5$				$n = 10^6$			
Binary Splitting	64	60	56	52	76	72	68	64
Gen. Bin. Splitting	61	57	53	49	75	71	67	63
KUW	44	32	24	20	52	36	28	20
Li's s-stage	64	36	22	16	79	45	28	20
PR CFF	208	104	52	26	249	125	63	32
PPS id-codes	719	396	198	99	2.5K	1250	625	313
Random Matrices	108	54	27	14	122	61	31	16

Table 5.5: Table showing the number of tests required by each group testing algorithm from Table 5.1 when $n = 10^3, 10^4$, $d = 4$ and the number of processors available is $p = 2, 4, 8, 16$. For random matrices a success probability of 99.9% is required.

Method	$d = 10$							
	$n = 10^3$, $p =$				$n = 10^4$, $p =$			
	2	4	8	16	2	4	8	16
Binary Splitting	90	80	70	60	130	120	110	100
Gen. Bin. Splitting	77	67	57	47	111	101	91	80
KUW	70	50	40	30	90	60	50	40
Li's s-stage	67	35	21	14	100	53	31	21
PR CFF	603	302	151	76	804	402	201	101
PPS id-codes	174	87	44	22	550	275	138	69
Random Matrices	194	97	49	25	226	113	57	29
	$n = 10^5$, $p =$				$n = 10^6$, $p =$			
	2	4	8	16	2	4	8	16
Binary Splitting	160	150	140	130	190	180	170	160
Gen. Bin. Splitting	144	134	124	114	177	167	157	147
KUW	110	80	60	50	130	90	70	50
Li's s-stage	134	70	41	27	167	88	51	33
PR CFF	1005	503	252	126	1206	603	302	151
PPS id-codes	1739	870	435	218	5500	2750	1375	688
Random Matrices	259	130	65	33	291	146	73	37

Table 5.6: Table showing the number of tests required by each group testing algorithm from Table 5.1 when $n = 10^3, 10^4$, $d = 10$ and the number of processors available is $p = 2, 4, 8, 16$. For random matrices a success probability of 99.9% is required.

Chapter 6

Short One-Time Signatures

In this chapter we present a new one-time signature scheme having short signatures. Our new scheme is also the first one-time signature scheme that supports aggregation, batch verification, and which admits efficient proofs of knowledge. It has a fast signing algorithm, requiring only modular additions, and its verification cost is comparable to ECDSA verification. These properties make our scheme suitable for applications on resource-constrained devices such as smart cards and sensor nodes.

6.1 Introduction

A one-time signature (OTS) scheme is a digital signature scheme that can be used to sign one message per key pair. More generally, we consider w-time signatures, which allow w signatures to be signed securely with each key pair (signing more than w messages breaks the security of the scheme). One-time signatures are an old idea: the first digital signature scheme invented was an OTS (Rabin/Lamport [138, 107]). The two main advantages of OTS is that they may be constructed from *any* one-way function, and the signing and verification algorithms are very fast and (when compared to regular public-key signatures). Common drawbacks, aside from the signature limit, are the signature length and the size of the public and private keys.

Despite the limitations of OTS, they have found many applications. On the more practical side, OTS can be used to authenticate messages in sensor networks [61] and to provide source authentication for multicast (also called broadcast) authentication [135]. One-time signatures are also used in the construction of other primitives, such as online/offline signatures [79] and CCA-secure public-key encryption [32].

With respect to signature length, designing conventional signature schemes with short signatures is not a new problem, and is motivated by applications with strong bandwidth

constraints. For example, signatures which are bar-coded for postage stamps, or which must be entered manually by users as a part of a product registration system, must be as short as possible while maintaining security.

There is a significant gap in signature length between regular public-key signatures and OTS. While signatures in conventional schemes can be very short, e.g., as small as 160 bits for 80-bit security (in the BLS scheme [35]), one-time signatures are usually many times longer (for typical examples, see [135, 142], where signature lengths are over one thousand bits). This motivates the following questions: Are short OTS possible? Can we retain the advantages of OTS but reduce the signature size?

We give a positive answer to these questions. In particular, we make the following contributions:

- We give the first one-time signature scheme with short signatures and a tight security reduction based on the difficulty of the discrete logarithm problem. The signature length in our scheme is constant with respect to the size of the message to be signed, and is about 180 bits long for 80-bit security.
- We give a unified description of five previous schemes and improve parameter selection for these schemes.
- Our new scheme supports aggregation and batch verification, admits efficient proofs of knowledge, and is fail-stop. Ours is the first OTS to have the first two of these properties.
- As a corollary, we give a fail-stop signature scheme with the shortest signatures to date.
- Our new OTS retains fast signing, but has slower verification than most OTS. Nonetheless, verification in our new scheme is only about as expensive as verification of an ECDSA signature.

Informal description of our solution. We consider a general class of OTS schemes based on cover-free families, and make the observation that the one-way function in an OTS scheme is being used as a commitment function. The signer creates commitments during key generation that form the public key, and the openings of these commitments make up the signature. By replacing the one-way function with Pedersen commitments (of the form $g^s h^r$) [134], we can use the algebraic properties of this commitment scheme to compress a number of openings into a single opening. We also show that it is sufficient for security to have the value r in the commitment be very small, leading to short signatures. We make further use of the algebraic structure to prove security, and provide additional features: batch verification, aggregation, and proofs of knowledge.

Chapter organization. First we describe a general construction of OTS based on cover-free families (§6.2), then we review relevant related work on one-time signatures and their applications (§6.3). In §6.4 we present our new scheme and discuss parameter selection. In §6.5 we describe additional features of our scheme, and then we conclude with a discussion of its applications (§6.6).

6.2 General Construction of OTS from Cover-Free Families

A number of existing OTS schemes may be described as special cases of a (unified) general construction based on cover-free families. Our new scheme will also be a variant of this general construction. We start with the definition of a cover-free family (which is somewhat specialized for this chapter).

Definition 6.1. A *w-cover-free family* $(X, \mathcal{B})$ is a set X of m elements, and a set $\mathcal{B}$ of 2^n subsets of X called blocks, with the following property. For any w blocks $B_{i_1}, \ldots, B_{i_w} \in \mathcal{B}$, and all other blocks $B \in \mathcal{B}$, it holds that

$$B \not\subseteq \bigcup_{j=1}^{w} B_{i_j} \ .$$

We say that $B_{i_1}, \ldots, B_{i_w}$ does not *cover* any other $B \in \mathcal{B}$. We will use the notation w-CFF$(m, 2^n)$ for cover-free families.

We now describe the general construction of a w-time signature scheme based on a cover-free family. Throughout this chapter, the message space is $\{0,1\}^n$.

Setup(n)**:** Let $(X, \mathcal{B})$ be a w-CFF with $|X| = m$ elements and $|\mathcal{B}| = 2^n$ sets. Let $f : \{0,1\}^\ell \to \{0,1\}^{\ell_0}$ be any one-way function (where ℓ and ℓ_0 are security parameters). Also, to each message $M \in \{0,1\}^n$ we associate a unique $B_M \in \mathcal{B}$ (this correspondence is public).

Key Generation: Choose m random values $s_1, \ldots, s_m \in \{0,1\}^\ell$ and compute $v_i = f(s_i)$ for $i = 1, \ldots, m$. Output the public key $PK := (v_1, \ldots, v_m)$ and the secret key $SK := (s_1, \ldots, s_m)$.

Sign: To sign $M \in \{0,1\}^n$, compute $B_M \in \mathcal{B}$, the subset corresponding to M. Then output the signature $\sigma = \{(s_i, i) : i \in B_M\}$.

Verify: To verify (σ, M) using PK, compute B_M, then check that $f(s_i) = v_i$ for all $i \in B_M$. If so, output 1; otherwise, output 0.

It is easy to see that security is provided by the one-wayness of f and the cover-free property: given w signatures, all other messages require knowledge of at least one s_i value that has not been revealed (a more detailed analysis appears in [136]). The correspondence between M and B_M depends on the CFF; we will describe an efficient algorithm for our scheme in §6.4.2.

6.3 Related work

We will first describe five existing schemes that are special cases of the general CFF scheme described above. Then we discuss other work related to OTS and fail-stop signatures. The OTS literature is vast, and we do not provide a complete survey here. For a more comprehensive survey, see Menezes et al. [122, Ch. 11.6] and Dods et al. [65]. All of the schemes considered in [65, 122] have signatures that are longer than the new scheme we present in §6.4. For details of the signature length in conventional signature schemes, see [35].

6.3.1 Schemes Based on the CFF Model

The descriptions assume we want to sign n-bit messages. We write M_i for the i-th bit of message M.

In the Lamport scheme [107], the public key consists of $v_{i,b} = f(s_{i,b})$ for $i = 1, \ldots, n$, and for $b = 0, 1$. To sign M, reveal $s_{1,M_1}, \ldots, s_{n,M_n}$. The verifier checks that $v_{i,M_i} = f(s_{i,M_i})$, for $i = 1, \ldots, n$. This can be interpreted as a 1-CFF with $2n$ points, where $X = \{(i, b) : i = 1, \ldots, n \text{ and } b = 0, 1\}$ and $B_M = \{(i, b) : M_i = b\}$ for all $M \in \{0, 1\}^n$.

The Bos and Chaum [38] scheme has m secrets and can be used to sign all weight $\lfloor m/2 \rfloor$ binary vectors. Therefore, we require $\binom{m}{\lfloor m/2 \rfloor} > 2^n$ in order to sign n-bit messages. When these vectors are viewed as the incidence matrix of a CFF, this forms an *optimal* 1-CFF with m points (optimality is proven by Sperner [154]).

The Reyzin-Reyzin schemes [142] use random structures instead of explicitly constructed CFFs (under the name "subset-resilient functions"). Their security analysis considers two probabilities. First, the probability that a random matrix is a w-CFF is used in the security analysis for chosen-message attacks. The adversary has the description of the CFF, and finding w blocks that cover another block allows the adversary to sign the covered message after observing w signatures. Second, the probability that a randomly

chosen set of messages covers another message is used to analyze security when the adversary is passive and observes w signatures on random messages. Some example parameters are given in [142]. To sign two messages with 80-bit security and a forgery probability of 2^{-53}, the public key size is ≈ 82 Kb and the signatures are 1600 bits in length.

The scheme of Pieprzyk, Wang and Xing [136] (the PWX scheme) is the first to explicitly use CFFs, and it uses some constructions based on polynomials and error-correcting codes. All of the constructions in [136] are constructions for w-CFF for general w. The special case $w = 1$ is not singled out (having already been considered in [38]).

Katz [102] defines the same scheme as the PWX scheme, but uses a CFF with a stronger property, namely, that the union of w blocks misses λ (or more) points of any other block. The stronger property is required to provide leakage resilience (a property that guarantees security even if a bounded amount of the signer's secret information is leaked). As in [136], going from signing one message to signing multiple messages is done via a w-CFF.

Parameter Selection In Section 6.4.3 we show that better parameters (shorter signatures and smaller keys) are obtained by using w 1-CFFs rather than a single w-CFF. This improves the schemes above that use w-CFF.

6.3.2 Other Work Related to One-Time Signatures

Fail-Stop Signatures. In a fail-stop signature scheme, the signer may efficiently prove that he did not create a given, valid signature (when the signature is a legitimate forgery). The scheme of van Heyst and Pedersen [176] uses Pedersen commitments in the public key to create a fail-stop signature scheme. For this reason, our schemes have some similarities: both schemes can sign w messages, the public key is a list of commitments, and the secret key is their openings. The signature and verification algorithms differ, however: in [176], the messages are used in the signature directly (rather than being encoded with a CFF), verification is slower (at least twice as slow) and signatures are about twice as long.

Other schemes. Goldwasser, Micali and Rivest [89] present a one-time signature scheme based on trapdoor claw-free permutations. The scheme requires n evaluations of the permutation to sign n-bit messages and the signature is the size of the output. For all known trapdoor claw-free permutations, this means that the signature is at least as long as an RSA modulus (i.e., 1024 bits at the 80-bit security level).

Groth [91] gives an alternate construction of an OTS scheme with the same properties as the van Heyst and Pedersen scheme (though it is not fail-stop).

Bellare and Shoup [19] present a general construction of OTS from three-move identification protocols. Specific instantiations of their construction yield an OTS scheme with short signatures (as short as ours) based on the *one more discrete log* problem [131], while another gives a scheme with signatures about twice as large as ours having discrete log security. Both schemes presented in [19] also require a collision-resistant hash function (CRHF), even when signing short messages, i.e., the hash function is part of the signing algorithm, and not just applied to the input message in the usual way.

A popular approach to convert an OTS scheme to a w-time signature scheme is to use a Merkle tree to authenticate w one-time public keys [65, 122, 123]. Since the signature must include a path through the authentication tree, the signature length is typically thousands of bits. Some examples of multiple-time signatures using Merkle trees are given in [22, 43, 127]. To our knowledge, the Merkle-like scheme with the shortest signatures is due to Dahmen and Krauß [61]. At the 80-bit security level, the signatures produced by their scheme are 330 bits long (in general the signature length is about 4κ to achieve κ-bit security).

A recent paper by Mohassel gives a black-box construction of OTS schemes from chameleon hash functions [125]. This result leads to new OTS schemes based on the hardness of factoring, the discrete logarithm problem, and the short integer solution problem on lattices. The instantiation based on the discrete log assumption has the shortest signatures, which are the same length as the van Heyst and Pedersen OTS, about twice as long as in the new scheme we present.

6.3.3 Applications of One-Time Signatures

Here we review a few example applications of one-time signatures. In §6.6 we discuss using our new OTS for these applications.

Smart cards and sensor networks. Resource-constrained devices that are not capable of using public-key cryptography (RSA, for example) often have sufficient resources for symmetric-key operations and w-time signatures. Rohde et al. [144] show that the Merkle signature scheme is practical on smart cards without a cryptographic coprocessor. Nodes in sensor networks have similar limitations, and one-time signatures were applied in this case by Dahmen and Krauß [61]. Their scheme uses the fact that most messages in sensor networks are short (8–24 bits), for example, basic commands or simple measurements (such as temperature).

Bicakci et al. [21] introduce the concept of *one-time sensors*. In this application, nodes in a wireless sensor network are given enough cryptographic material to produce only one (or a few) authentic messages. This is motivated by nodes with a short lifespan, for

instance, due to limited battery life. Also, the nodes might not be strictly disposable, e.g., they must return to the central authority periodically to obtain new cryptographic keys. Signing must be fast on a sensor node, while verification is done at a central repository by a more powerful computer. Using public key signatures instead of a symmetric key message authentication code allows other entities to verify the authenticity of a message.

Broadcast authentication. Using OTS to authenticate a stream of broadcasted data was initiated by Gennaro and Rohatgi [88, 143]. For streams of data that are unknown in advance (e.g., live broadcasts) their solution uses an OTS to authenticate each block of the stream against the public key transmitted in the previous block.

The BiBa OTS scheme was designed by Perrig [135] as the main component of the BiBa broadcast authentication protocol. In broadcast authentication, a sender wishes to send authenticated packets to multiple receivers that may have limited resources. Fast signing and verification are important in this application to allow high-throughput, low-latency communication. Substituting the Reyzin-Reyzin scheme (see §6.3.1) for the BiBa OTS improves the efficiency of the BiBa broadcast authentication protocol (see [142]).

6.4 A New OTS Scheme with Short Signatures

In this section we describe our new scheme and discuss parameter selection. We also give a set of concrete parameters for 80-bit security. The scheme, given in Figure 6.1, signs only one message, since in §6.4.3 we show that w public keys that each sign one message will be smaller than one public key that signs w messages (for CFF-based OTS).

6.4.1 Scheme Description

We first briefly review Pedersen's commitment scheme [134]. Let G be a group of order q, where q is prime. Let $g, h \in G$ be system parameters. To commit to a message $m \in \mathbb{Z}_q^*$, choose $r \in \mathbb{Z}_q$ at random, and output $C = g^m h^r$ as the commitment. To open C, reveal (m, r). Also note that, given two distinct openings to a Pedersen commitment using distinct bases g and h, it is possible to recover $\log_g h$.

The complete scheme is presented in Figure 6.1.

Remark 6.2. For the scheme in Figure 6.1 to be fail-stop, $\log_g h$ must be unknown to the signer. In practice, g and h may be chosen by a trusted authority, or verifiably at random. A forgery will allow the signer to recover $\log_g h$ (with probability at least $1 - 2^{-\ell_r}$), and use it as a proof of forgery (under the assumption that the signer cannot compute $\log_g h$).

Setup(n)**:** Choose a group G of order q, where q is an ℓ_q-bit prime (ℓ_q is a security parameter). Let $(X, \mathcal{B})$ be an optimal 1-CFF$(m, 2^n)$, and write B_M for the block corresponding to the message $M \in \{0,1\}^n$. Let g and h be generators of G (see Remark 6.2 on choosing g and h).

Key Generation: For $i = 1, \ldots, m$, generate random values $s_i \in_R \mathbb{Z}_q$ and $r_i \in_R \{0,1\}^{\ell_r}$, where ℓ_r is a parameter. The secret key is $SK := (s_i, r_i)_{i=1}^m$. For $i = 1, \ldots, m$, compute $v_i := g^{s_i} h^{r_i}$. The public key is $PK := (v_1, \ldots, v_m)$.

Sign: To sign M, compute and output

$$(\sigma, \rho) := \left(\sum_{i \in B_M} s_i \pmod q, \sum_{i \in B_M} r_i \right) \in \mathbb{Z}_q \times \mathbb{Z}.$$

More precisely, ρ is an integer in $[0, m(2^{\ell_r} - 1)/2]$, since $|B_M| = \lfloor m/2 \rfloor$ in the optimal 1-CFF.

Verify: To verify the signature (σ, ρ), check that $0 \le \rho \le m(2^{\ell_r} - 1)/2$ and

$$g^\sigma h^\rho \stackrel{?}{=} \prod_{i \in B_M} v_i.$$

Output 1 if both conditions hold, and output 0 otherwise.

To sign w messages, simply create w public keys as above, and include a counter with each signature to indicate which of the w public keys is being used.

Figure 6.1: Our new one-time signature scheme.

See the proof of Theorem 6.5 for details. If the fail-stop property is not desired, the signer may use any distinct g and h, provided $\log_g h$ is not publicly known.

Our scheme is secure if the discrete log problem is hard in G.

Definition 6.3. Let G be a cyclic group of prime order q, and let g be a generator of G. The *discrete log problem (DLP)* is, when given $g, h \in G$, to compute $x = \log_g h$, i.e., x such that $h = g^x$.

We say that an adversary A (t, ϵ)-solves the DLP in G if after time t, A outputs a correct solution to a DLP instance with probability ϵ.

The definition of security we use is *strong unforgeability under chosen message attacks.*

Strong unforgeability against an adaptive adversary is modelled by the following game between a challenger C and an adversary A.

1. C publishes $Params := \mathsf{Setup}(n)$.
2. C runs the key generation algorithm with $Params$ w times, and publishes $PK_1, \ldots, PK_w$.
3. A adaptively requests up to w signatures, at most one per public key, which C provides. Let Q be the set of (message, signature, public key index) triples queried by A.
4. A outputs (M, σ, i).

We say that A (t, ϵ)-*wins* the game if

$$\Pr[\mathsf{Verify}(PK_i, M, \sigma) = 1 \wedge (M, \sigma, i) \notin Q] = \epsilon \ ,$$

and Step 4 takes time t. Note that A can win by outputting a triple (M, σ, i) where M appears in Q, but with a different σ. This is the *strong* unforgeability property: a new signature on a previously signed message is considered a forgery.

For our proof of security, we will require the following technical lemma. In what follows, for an integer x we write $[x]$ to denote $\{0, \ldots, x-1\}$.

Lemma 6.4. *Let $\mathcal{X}_n$ be the probability distribution on $[n2^\ell]$ defined as $\mathcal{X}_n = X_1 + \ldots + X_n$, where X_i is the uniform distribution on $[2^\ell]$. Then the min-entropy of $\mathcal{X}_n$, $H_\infty(\mathcal{X}_n)$, is at least ℓ bits.*

Proof. Let $P_n(k) = \Pr[\mathcal{X}_n = k]$ for $k \in [n2^\ell]$. Clearly, $P_1(k) = 2^{-\ell}$ for all $k \in [2^\ell]$ and zero otherwise. By applying repeated convolution to P_1,

$$\begin{aligned} P_n(k) &= \sum_{i \in \mathbb{Z}} P_1(i) P_{n-1}(k-i) \\ &= 2^{-\ell}(P_{n-1}(k-1) + \ldots + P_{n-1}(k-2^\ell)) \ . \end{aligned}$$

because $P_1(i) = 0$ for values of $i \notin [2^\ell]$. Since the sum of the terms $P_{n-1}(k-i)$ are at most one (P_{n-1} is a probability distribution), it follows that $P_n(k) \leq 2^{-\ell}$ for all n, k. Now we compute the min-entropy of $\mathcal{X}_n$:

$$H_\infty(\mathcal{X}_n) = -\log\left(\max\left\{\Pr[\mathcal{X}_n = x]\right\}_{x \in [n2^\ell]}\right) \geq -\log 2^{-\ell} = \ell,$$

which proves the lemma. □

We now prove the security of our scheme.

Theorem 6.5. *Let A be an adversary who (t, ϵ)-wins the strong unforgeability security game above for the w-time signature scheme in Figure 6.1. Then A can be used to $(t + c, \epsilon(1 - 2^{-\ell_r}))$-solve the DLP in G, where ℓ_r is a parameter of the signature scheme, and c is a small constant.*

Proof. B will be a new algorithm that uses A to solve an instance of the DLP in G. Let the DLP instance given as input to B be (g, h). B creates $PK_1, \ldots, PK_w$ and $SK_1, \ldots, SK_w$ as above, using g and h, and gives $PK_1, \ldots PK_w$ and the system parameters to A. Since B knows $SK_1, \ldots, SK_i$, the w adaptive queries A makes may all be answered correctly. (Recall that each message is signed with one of the w keys, as the w-time key pair consists of w one-time key pairs.) After seeing signatures on a set of Q messages, A outputs a signature (σ, ρ, i) on a message M, which verifies using PK_i. Let $(\overline{\sigma}, \overline{\rho})$ be the signature on M created by B using SK_i and the signing algorithm above. Now define the Pedersen commitment

$$C := \prod_{j \in B_M} v_j = g^{\overline{\sigma}} h^{\overline{\rho}} = g^{\sigma} h^{\rho} \ .$$

There are two cases to consider: (i) $M \notin Q$ or, (ii) $M \in Q$, but $(\sigma, \rho) \neq (\overline{\sigma}, \overline{\rho})$.

Since the ρ-value of a signature is in $[0, m(2^{\ell_r} - 1)/2]$, there are $m(2^{\ell_r} - 1)/2$ valid openings of C. We must analyze the probability that A outputs the same opening as $(\overline{\sigma}, \overline{\rho})$. A does have some information about $\overline{\rho}$; he knows that $\overline{\rho}$ is the sum of uniformly random values from $\{0, 1\}^{\ell_r}$. By Lemma 6.4, the min-entropy of $\overline{\rho}$ as a random variable defined on $\{0, \ldots, m(2^{\ell_r} - 1)/2\}$ is at least ℓ_r bits. Therefore, in case (i) the forgery equals $(\overline{\sigma}, \overline{\rho})$ with probability not more than $2^{-\ell_r}$, and differs (giving distinct openings of C) with probability at least $1 - 2^{-\ell_r}$. In case (ii), $(\sigma, \rho) \neq (\overline{\sigma}, \overline{\rho})$ by definition, so we always have distinct openings of C.

Thus, with probability at least $\epsilon(1 - 2^{-\ell_r})$ A succeeds *and* B has two distinct openings of C, which allows B to recover $\log_g h$, solving the DLP instance. The time required by B is t (the time required by A) plus the time required to: generate w key pairs, sign w messages, and solve for $\log_g h$, i.e, compute $\log_g h = (\overline{\rho} - \rho)/(\sigma - \overline{\sigma})$. □

Remark 6.6. Note that non-repudiation is provided, despite the signer's ability to choose SK such that two messages have the same signature (the signer can do this by ensuring that the sum of two openings is the same). Suppose Bob has a signature (σ, ρ) from Alice on the message M, and Alice later claims that (σ, ρ) is a signature on M' (and the verification equation holds). If Alice is telling the truth, and she did not give Bob a signature on M, then Bob has produced a forgery on M, which is not possible by Theorem 6.5. Therefore, Alice must have signed M.

A natural question to ask is whether we can change the commitments used in the public key to simple discrete log commitments, i.e., a commitment to m is computed as g^m. This simplifies the scheme and reduces signature size (by 10 bits using our parameters from §6.4.3). We chose the scheme presented because we were not able to find a security proof for the modified scheme with a tight security reduction, and the scheme presented has only a modest increase in signature length.

6.4.2 Encoding a message M as B_M

Encoding messages is considered in detail by Bos and Chaum [38] as well as Reyzin and Reyzin [142]. Pieprzyk et al. [136] discuss coding-theoretic approaches to the problem (encoding using generator matrices). In both [38] and [142], algorithms to encode a message as a weight $\lfloor m/2 \rfloor$ vector are given. Both require a nontrivial amount of computation (at least $m^2 n$ multiplications). Here we mention a simpler approach.

A bijective function $S : \left\{1, \ldots, \binom{a}{b}\right\} \leftrightarrow \{b\text{-subsets of } [a]\}$ is called a *ranking algorithm* from integers to subsets and an *unranking algorithm* from subsets to integers. The well-known ranking algorithm described in [60, 53] requires $\lfloor m/2 \rfloor$ subtractions and $\log d$ comparisons where $d = \binom{n}{\lfloor m/2 \rfloor}$, using precomputed values. The algorithm is as follows:

Input M: an n-bit integer, k: weight of the output
Output y: length d binary vector of weight k

for $i = 1, \ldots, d$

  if $M > \binom{n-i}{k}$
    $y_i = 1$
    $M = M - \binom{n-i}{k}$
    $k = k - 1$
  else $y_i = 0$

return y

This is called the *co-lex ranking* [105]. Bicakci et al. [22] use the *lex ranking* in their implementation of the Bos-Chaum scheme.

6.4.3 Parameter Selection

We first argue that signing w messages using w instances of a 1-CFF will require less storage than using a single w-CFF. This applies to all of the CFF schemes described in Section 6.3.1.

Suppose $w = \ell t$ for integers ℓ and t. In any w-CFF(m, 2^n), we have $m = \Omega(\frac{w^2}{\log w} n)$ (see Stinson et al. [163]). Combining ℓ public keys allowing t signatures each to create a public key permitting $w = \ell t$ signatures therefore has $m = \Omega(\ell \frac{t^2}{\log t} n)$ (where m in this case is the *total* number of secrets needed by the signer). This bound on m is smallest when $\ell = w$ and $t = 1$, i.e., when we use w public keys each allowing one signature (note that the constants hidden by the Ω-notation are the same for all choices of ℓ and t). It is also easy to see that an intermediate choice (e.g., using $\sqrt{w}$ $\sqrt{w}$-CFFs) is not an improvement. Thus, by using w 1-CFF(m, 2^n) instead of one w-CFF(m, 2^n), we can potentially reduce storage by a factor of $w/\log w$. A minor drawback of this approach is that the signature must include a counter, to tell the verifier which part of the public key to use. As we are comparing necessary conditions on m, and in practice one must use an explicit CFF construction, this comparison is not concrete, and may not hold for small parameters. Below we will give an example showing that using multiple 1-CFFs reduces storage space even when $w = 2$.

Constructing an optimal 1-CFF($m, 2^n$) is simple: choose all length m binary vectors with weight $\lfloor m/2 \rfloor$ (there are $\binom{m}{\lfloor m/2 \rfloor}$ such vectors). This is a 1-CFF($m, 2^n$) provided that

$$\binom{m}{\lfloor m/2 \rfloor} > 2^n \; ; \tag{6.1}$$

in fact, it is the same CFF used in the Bos-Chaum scheme [38].[1]

Next we consider ℓ_q. We choose G to be a group of points on an elliptic curve, since its elements have a compact representation. Therefore, ℓ_q must be large enough so that the DLP is hard in G. We would probably use the standard NIST curves [130] for performance reasons.

In most applications, $\ell_r = 10$ will be sufficient. This means that the security reduction succeeds with probability $\epsilon - \epsilon/1024$, or put another way, an instance of the DLP may be solved on average, given a forgery, 1023 out of 1024 times. A probability of proving a forgery of 1023/1024 should also be sufficient for most applications requiring fail-stop signatures.

Parameter sizes for 80-bit security and arbitrary-length messages. We assume that messages will be hashed with a collision-resistant hash function which produces a 160-bit output, and so $n = 160$. We need $\ell_q = 160$ for security, therefore group elements may be represented using 160 bits (recall our choice of G above). In this case the public key size is $160 \cdot 165w$ bits, which is about 26Kb per signature (we take $m \geq 165$ in order to

[1] The 1-CFF with the stronger property required to construct Katz' leakage-resilient scheme also has a simple (but not optimal) construction – a direct generalization of the Sperner construction (see Stinson et al. [163, Lemma 3.2]). The reduction in required storage is even more pronounced here, as m depends more strongly on w.

satisfy (6.1)). When $w = 10$, 100, and 1000, the public key is 264Kb, 2.64Mb and 26.4Mb, respectively. Since $\ell_r = 10$ and $\log_2 \rho \leq \log_2(\lfloor m/2 \rfloor 2^{\ell_r})$ the signature length is not more than $160 + 17 + \log_2 w = 177 + \log_2 w$ bits. (The $\log_2 w$ bits are required for the counter.) Next, the encoding algorithm requires $\log_2 \binom{165}{82} \approx 160$ comparisons, and we must store about 162 160-bit values, requiring about 26Kb space.

Now suppose we had used a w-CFF, instead of w 1-CFFs. For the purpose of comparison, we will generously assume that w-CFF$(m, 2^n)$ may be constructed with $m = \frac{w^2}{\log_2 w} n$. When $w = 2$ and $n = 160$, $m = 640$. The number of key elements is only 330 when using two 1-CFF$(m, 2^n)$ each with $m = 165$. Therefore, even for small w, using w 1-CFFs is preferable to using a single w-CFF.

Computational costs. Recall that $m = 165$ for 80-bit security and G is a prime-order elliptic curve group. Key generation requires m multi-exponentiations in G. Verification requires a single multi-exponentiation (where one exponent is small) and 82 multiplications, which is comparable to ECDSA (which requires one multi-exponentiation with full-length exponents). Signing requires 82 additions mod q. Note that, when computing the multi-exponentiations during key generation and verification, fast exponentiation techniques are applicable [20].

The signing and verification times of our scheme were compared to ECDSA using version 5.6.0 of the Crypto++ library [62] on a Pentium D 3.0GHz processor. The parameter sizes and elliptic curve group implementation used in both schemes was the same. For signing, our OTS is 47.14 times faster than ECDSA, while verification was 1.13 times slower. We note a possible ECDSA tradeoff: if the signer can precompute and (securely) store additional values, the ECDSA signing operation may be sped up significantly.

Parameter sizes for 80-bit security and short messages. We now consider the parameter sizes when our scheme is used to sign 16-bit messages. Recall (§6.3) that short messages are common in sensor networks [61]. To construct the optimal 1-CFF$(m, 2^{16})$ requires $m = 19$. Therefore the public key contains $3040w$ bits to sign w 16-bit messages. While the key sizes are larger than the Dahmen and Krauß scheme, the signature overhead is halved.

Some possible tradeoffs. First, to reduce the signature size even further, we can avoid including the counter at the cost of either (i) increased verification time, by simply omitting it, which may be acceptable when w is small, or (ii) larger public and private parameters, by using a w-CFF. Second, using randomized hashing, we may reduce the public and private storage requirement. The idea is to use a *target collision-resistant hash function*, which is a type of keyed hash function with a short digest (i.e., a κ-bit digest for κ-bit

Scheme	Security	PK size	SK size	Sig. size	Sign	Verify
vHP [176]	DLP	2κ	4κ	2κ	2 mult.	3 exp.
Groth [91]	DLP	3κ	2κ	2κ	3 mult.	3 exp.
BS [19]	DLP + CRHF	κ	2κ	2κ	2 mult.	1 exp.
BS [19]	omDLP + CRHF	κ	3κ	κ	1 mult	1 exp.
Generic CFF [38, 107, 136, 142]	OWF	$O(\gamma n)$	$O(\gamma n)$	$O(\gamma n)$	Encode (§6.4.2)	$O(n)$ OWF computations
This work	DLP	$O(\kappa n)$	$O(\kappa n)$	$\kappa + c$	Encode and $O(n)$ add.	1 exp. + $O(n)$ mult.

Table 6.1: Comparison of various OTS schemes. The DL security parameter is denoted by κ, the one-way function (OWF) security parameter is denoted by γ, n is the number of bits in the message to sign, and c is a small constant ($c = 10$ bits in the examples given above). The "Generic CFF" construction is given in §6.2.

security, instead of a 2κ-bit digest). Since the digest is half the size, the public and private parameters are roughly halved. However, the signer must commit to w hash keys during key generation, and include the hash key with the signature. This idea is discussed in [127] and [143]. Another possible way to reduce signer storage is to store only a short seed and generate SK using a pseudorandom generator (PRG), which increases computation during signing since the PRG must be invoked $\lfloor m/2 \rfloor$ times. Finally, we may reduce computation by precomputing and storing values of h^r for all $r \in \{0,1\}^{\ell_r}$, since ℓ_r is small.

Comparison. In Table 6.1, we compare the size of the keys and signatures, and the computation required to create signatures and verify them, for several OTS. For schemes providing security under the DLP, our scheme has the shortest signatures. This is traded off against the large public key size. The scheme of Bellare and Shoup (BS) has the best all-around performance, but it requires stronger assumptions, namely, the one more DLP (omDLP) and the existence of a collision-resistant hash function (CRHF).

6.5 Additional Features of the OTS Scheme

In this section, we describe four additional features of our OTS scheme. Batch verification and aggregation are techniques for handling multiple signatures, in order to efficiently verify

or compress many signatures. Proving knowledge of a signature and verifiably encrypting a signature are useful properties when using OTS to build more complex cryptographic protocols (e.g., certified email [10]).

6.5.1 Batch Verification

Our new OTS provides batch verification under the most general definition: verification of k signatures on k (possibly different) messages created under k public keys (that have the same generators g and h). Given a batch of signatures, $(\sigma_1, \rho_1, \ldots, \sigma_k, \rho_k, M_1, \ldots, M_k, PK_1, \ldots, PK_k)$, we can efficiently verify them using the *small exponents tests* of Bellare, Garay and Rabin [18], as follows:

$$g^{\sigma_1 d_1 + \ldots + \sigma_k d_k} h^{\rho_1 d_1 + \ldots \rho_k d_k} \stackrel{?}{=} \left(\prod_{i \in B_{M_1}, v_i \in PK_1} v_i \right)^{d_1} \times \cdots \times \left(\prod_{i \in B_{M_k}, v_i \in PK_k} v_i \right)^{d_k}$$

where $d_1, \ldots, d_k$ are randomly chosen small exponents. Using ℓ-bit exponents, there is a $2^{-\ell}$ probability of error during verification. Note that the error is one-sided; the test will never reject a valid batch of signatures. The cost of the test, (assuming m is even) is $km/2$ multiplications plus k small exponentiations plus one full exponentiation. The cost of individually verifying k signatures is $km/2$ multiplications plus k exponentiations.

6.5.2 Aggregation

In this section we prove that k signatures may be securely aggregated by simply computing their sum. We consider the most general definition of aggregation: signatures from k (possibly different) signers on k (possibly different) messages are combined to produce a single signature. Here are the aggregation functions.

$\mathsf{Aggregate}(\sigma_1, \rho_1, \ldots, \sigma_k, \rho_k)$: Output the aggregate signature

$$(\sigma, \rho) = \left(\sum_{i=1}^{k} \sigma_i \pmod q, \sum_{i=1}^{k} \rho_i \pmod q \right) \in \mathbb{Z}_q \times \mathbb{Z}_q \, .$$

Note that, when aggregating a large number of signatures, ρ may be reduced mod q.

$\mathsf{VerAgg}(PK_1, \ldots, PK_k, M_1, \ldots, M_k, \sigma, \rho)$: Output 1 if

$$g^{\sigma} h^{\rho} \stackrel{?}{=} \left(\prod_{i \in B_{M_1}, v_i \in PK_1} v_i \right) \times \cdots \times \left(\prod_{i \in B_{M_k}, v_i \in PK_k} v_i \right) \, ,$$

holds, and output 0 otherwise.

Here we present the security game for aggregate signatures from Boneh et al. [33]. Let C be the challenger and A be the adversary.

1. C runs $\mathsf{Setup}(n)$ to generate $params$, computes $PK_1 = \mathsf{Keygen}(params)$, and gives $params$ and PK_1 to A.
2. A adaptively queries a set of messages Q, and C provides signatures using SK_1.
3. A outputs $PK_2, \ldots, PK_k$, (σ, ρ) and $\mathcal{M} = \{M_1, \ldots, M_k\}$. A's success probability is the probability that $\mathsf{VerAgg}(PK_1, \ldots, PK_k, \mathcal{M}, \sigma, \rho) = 1$ and $M_1 \notin Q$.

In order for $\mathsf{Aggregate}$ to be secure, we must augment the public key PK_i with a zero-knowledge (ZK) proof of knowledge Z_i of the secret key (using standard techniques, e.g., [48, 63]). Alternatively, the signer may prove knowledge of the secret key to the verifier when communicating the public key. This proves that each public key is well formed. We naturally require that A must output valid public keys to win the above game. Since there are multiple ZK proof systems meeting our requirements, and to keep the presentation general, we do not specify that a particular system must be used.

Sketch of the security proof. We show that an adversary who produces a forgery in the above game can be used to solve the DLP in G. The challenger creates PK_1, SK_1, and Z_1 and sends PK_1 and Z_1 to A. A makes adaptive queries, then responds as in Step 3. C then extracts $SK_2, \ldots, SK_k$ from $Z_2, \ldots, Z_k$ (which is possible using the knowledge extractor since the proofs Z_i are valid, see [63]). Given (σ, ρ), and $SK_2, \ldots, SK_k$, the challenger computes $(\sigma', \rho') = \mathsf{Aggregate}(PK_2, \ldots, PK_k, M_2, \ldots, M_k)$. Now, $\sigma'' := \sigma - \sigma'$, $\rho'' := \rho - \rho'$ is a valid signature on M_1 under PK_1 (a forgery). We now proceed as in the proof of Theorem 6.5.

Remark 6.7. No ZK proofs are necessary when all signatures are created by a single signer, or by a group of trusted signers. In this case, since the participants are honest (not under A's control), the above game should be modified to have the challenger choose $PK_2, \ldots, PK_k$, instead of the adversary.

6.5.3 Proving Knowledge of a Signature on the Message M

The following ZK proof of knowledge allows a prover to convince a verifier that he possesses a signature on M under a known public key without revealing the signature.[2]

$$\mathrm{PK}\{(\sigma, \rho) : \prod_{i \in B_M} v_i = g^{\sigma} h^{\rho}\}$$

Note that prover only needs to know g, h, σ, ρ and can remain ignorant of the whole public key. This is easily generalized to proving knowledge of aggregate signatures. Using the non-interactive version of Schnorr's protocol, the computation of the prover is a single multi-exponentiation in both cases. The size of the proof is about $4\ell_q$ bits. (See [48] for more details.)

6.5.4 Verifiably Encrypting a Signature

Since proving knowledge of a signature amounts to proving knowledge of a discrete log, we can verifiably encrypt signatures by combining our OTS with the Camenisch-Shoup verifiable encryption scheme [49]. The ability to verifiably encrypt signatures allows them to be exchanged fairly using an optimistic fair exchange protocol [9]. If we encrypt using an additively homomorphic encryption scheme with efficient proofs of plaintext knowledge (such as the Paillier scheme [139]), encrypted signatures may also be aggregated.

6.6 Impact on Applications

In this section we discuss using our scheme in the applications mentioned in Section 6.3.3. For two of the applications mentioned in the introduction, bar-coded postage and product registration systems, our scheme may be used, but it does not offer any immediate advantages over conventional signatures. So we do not discuss these applications further.

Smart cards and sensor networks. It is important that authentication for smart cards and sensor networks have low overhead, due to the high cost of communication [112]. This is the biggest advantage of using our OTS: the number of signature bits per message bit is smaller than alternative schemes. A specific application where short signatures are especially important is when sensors are used in vehicular networks. Since the vehicles pass

[2]The "PK" (proof of knowledge) notation [48] is a short-hand for the various Schnorr-like proof of knowledge of a discrete logarithm protocols which exist for types of statements such as knowledge of, relations between, and the length of discrete logarithms.

roadside access points at high speed, only a limited amount of data can be transferred [100]. Key management can be handled offline, i.e., when the vehicle is parked.

The computation costs of our scheme favour applications where the card or nodes produce the signatures and verification is performed by a device with more resources. For example, in heterogeneous sensor networks, where signatures are created by low-resource nodes and verified by nodes with greater resources, or in sensor networks where the sensors return authenticated data to a central authority (e.g., weather sensors reporting to a meteorological agency). In the latter example, aggregation can reduce authentication overhead even further, and batch verification can reduce verification time. We stress that that resource-constrained devices are still able to efficiently verify signatures in our scheme. ECDSA has been shown practical for sensor nodes and smart cards [92, 171], and verification in our scheme requires equivalent computational resources.

Broadcast authentication. While our scheme could be used with BiBa, it will not be as efficient as the Reyzin-Reyzin scheme because of the size of the public parameters; we cannot use the hash chain technique to re-use a single public key for many signatures. However, our observation on parameter selection (i.e., to use w 1-CFFs instead of one w-CFF) gives a variant of the Reyzin-Reyzin scheme with smaller public keys and signatures, and thus gives a version of the BiBa protocol with reduced communication and computation, and improved security.

Chapter 7

Future Work

Here we mention some possible topics for related future work. With respect to upper bounds on the size of SHF, the exponent in the bound proven here cannot be improved further, as shown in Section 2.4. Despite this, it may still be possible to improve the constant in Theorem 2.29. It would be interesting to either improve the leading constant, or show that this cannot be done. A second bound of interest would be a lower bound on N for SHF of general type. In Section 2.5, we prove a lower bound on N for type $\{w_1, w_2\}$ but did not generalize this to type $\{w_1, \ldots, w_t\}$.

In Chapter 3 we showed that constructing asymptotically optimal SHF of arbitrary type is possible with the AG construction, and that many constructions for SHF of a specific type may be generalized to construct SHF of arbitrary type. What remains open in this area is the problem of finding constructions that produce good small SHF. This is important since many applications require small SHF, while asymptotically optimal constructions only guarantee good SHF in the limit. We saw an example of this with CFF constructions used in batch verification: the id-codes CFF construction was outperformed asymptotically by the construction of Porat and Rothschild, but gave better results for smaller batch sizes (§5.2.3).

On the subject of anonymity in shared symmetric key primitives, it may be interesting to determine the expected anonymity when the scheme is instantiated with a random BPHF. Are there parameter choices for which trading guaranteed anonymity for anonymity with high probability and reduced key storage are worthwhile?

We suggest two possible follow-up projects related to our batch verification work. First, in our comparison, the function Batch was assumed to take unit time, independent of the input size. A more precise comparison would model Batch with a cost function, which depends on the implementation of batch, and the size of the input. For many of the ISF algorithms, determining the input size for each call to Batch is difficult, therefore, an

empirical comparison would likely be necessary. Second, we considered only generic ISF algorithms, where Batch is treated as a black box. Is it possible to construct improved ISF algorithms for specific signature schemes, using ideas from the group testing literature?

Finally we mention a possible improvement to the short one-time signature scheme presented in Chapter 6. The main drawback of the scheme is the size of the public key, therefore reducing the public key size without increasing the signature size would make a nice contribution. At the moment there is a large gap in public key size between our scheme and the next shortest one-time signatures with discrete log security (the van Heyst and Pedersen scheme). A common technique used in OTS schemes based on one-way functions, recursively committing to a secret (i.e., publishing $f(f(\ldots f(s)\ldots))$) is no longer applicable using Pedersen commitments. This reduces the public key size by allowing multiple signatures to be created from the same public key. Can something similar be done for our short OTS scheme?

Appendix A

Chromatic polynomials of complete multipartite graphs

Let $\chi(G,q)$ be the chromatic polynomial of the graph G, with q colours. Let $\overline{\chi}(G,r)$ count the number of proper colourings of G using exactly r colours, with colour difference. Sometimes $\overline{\chi}(G,r)$ is defined with colour indifference, so we note that the two quantities differ by a factor of $r!$ (see [140, §7]). Finally, we let $S(n,k)$ denote the Stirling numbers of the second kind, which count the number of ways to partition an n-set into k blocks.

We give formulas for computing $K_{w_1,\ldots,w_t}$, the complete multipartite graph with t parts $C_1, C_2, \ldots, C_t$ such that $|C_i| = w_i$. A formula for the special case K_{w_1,w_2} was given by Ehrenborg [72, Prop. 4.4]. The next theorem gives an alternative formula, and gives the main ideas for the more general case which is a straightforward generalization.

Theorem A.1. *Using the notation defined above,*

$$\chi(K_{w_1,w_2},q) = \sum_{r=1}^{q} \binom{q}{r} r! \sum_{i=1}^{r-1} S(w_1,i)S(w_2,r-i) \; .$$

Proof. It is well known that the quantities χ and $\overline{\chi}$ are related [140]

$$\chi(G,q) = \sum_{r=1}^{q} \binom{q}{r} \overline{\chi}(G,r) \; . \tag{A.1}$$

For an arbitrary graph the number of colourings using exactly $r = i+j$ colours with colour indifference is the number of partitions of the vertices with the condition that no edge connects two vertices in the same partition. For a complete bipartite graph, the vertices in C_1 are never connected to each other, but connected to all vertices in C_2 (similarly for C_2).

The proper colourings with i, j colours in C_1, C_2 (resp.) correspond to partitions of the w_1 nodes of C_1 into i blocks and the w_2 nodes of C_2 into j blocks. The Stirling numbers of the second kind $S(n, k)$ count the number of ways to partition an n-set into k blocks. Therefore

$$\overline{\chi}(K_{w_1,w_2}, q) = r! \sum_{i=1}^{r-1} S(w_1, i) S(w_2, r-i) \; . \tag{A.2}$$

The result follows by substituting (A.2) into (A.1). □

This theorem can easily be generalized to multipartite graphs. First we need a generalization of the pairs $(i, r-i)$ used in the previous theorem. Define

$$P_t(n) = \left\{ (x_1, x_2, \ldots, x_t) \in (\mathbb{Z}^+)^t \mid x_1 + x_2 + \ldots + x_t = n \right\} \; ,$$

the set of all additive partitions of the integer n with t terms. We will write $\mathbf{x} = (x_1, \ldots, x_t)$.

Theorem A.2. *Using the notation defined above,*

$$\chi(K_{w_1,\ldots,w_t}, q) = \sum_{r=1}^{q} \binom{q}{r} r! \sum_{\mathbf{x} \in P_t(n)} S(w_1, x_1) S(w_2, x_2) \ldots S(w_t, x_t) \; .$$

Proof. The proof follows the same reasoning as the proof of Theorem A.1. The colours used for vertices of $C_1, \ldots, C_t$ must all be disjoint, since vertices in C_i are connected to all other vertices in the graph, but not those in C_i. We must consider all the ways to divide the r colours amongst the t parts (so that each has at least one colour), this is $P_t(n)$. Then we must consider the possible ways of assigning the x_i colours to the w_i nodes of C_i, which can be done in $S(w_i, x_i)$ ways. □

Implementation

The formula from the previous section was implemented (naïvely) in PARI-GP, to confirm it will be sufficiently fast for our purposes. Sample running times are provided in Table A.1. For large graphs a fast implementation could use the following improvements. 1) Store $S(w_i, x_i)$, as they get reused with each value of r, and 2) store $r!$ then compute $(r+1)!$ as $(r+1)r!$.

$\chi(G, q)$	Time	output size
$\chi(K_{10,20,30,40}, 64)$	6mn, 52s	516 bits
$\chi(K_{10,10,50}, 64)$	23s 423 ms	395 bits
$\chi(K_{10,50}, 64)$	1s, 572 ms	350 bits
$\chi(K_{10,10,10,10}, 64)$	4mn, 54s	225 bits
$\chi(K_{10,50}, 8)$	< 1 ms	145 bits
$\chi(K_{10,10,10,10}, 8)$	8ms	52 bits

Table A.1: Running times of the formula from Theorem A.2.

References

[1] M. Al-Ibrahim, H. Ghodosi, and J. Pieprzyk. Authentication of concast communication. In *Proceedings of INDOCRYPT'02*, volume 2551 of *LNCS*, pages 185–198, 2002. 90

[2] I. Aleshnikov, P. V. Kumar, K. W. Shum, and H. Stichtenoth. On the splitting of places in a tower of function fields meeting the Drinfeld-Vladut bound. *IEEE Transactions on Information Theory*, 47:1613–1619, 2001. 55

[3] N. Alon. Explicit construction of exponential sized families of k-independent sets. *Discrete Mathematics*, 58:191–193, 1986. 35

[4] N. Alon, G. Cohen, M. Krivelevich, and S. Litsyn. Generalized hashing and parent-identifying codes. *J. Combin. Theory Ser. A*, 104:207–215, 2003. 6

[5] N. Alon, E. Fischer, and M. Szegedy. Parent-identifying codes. *J. Combinatorial Theory Series A*, 95:349–359, 2001. 5

[6] N. Alon and M. Naor. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16:434–449, 1996. 4

[7] N. Alon and U. Stav. New bounds on parent-identifying codes: the case of multiple parents. *Combinatorics, Probability and Computing*, 13:795–807, 2004. 5

[8] J.H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Proceedings of EUROCRYPT '02*, volume 2332 of *LNCS*, pages 83–107, 2002. 65, 84

[9] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18:593–610, 2000. Extended abstract published at EUROCRYPT'98. 123

[10] G. Ateniese. Verifiable encryption of digital signatures and applications. *ACM Trans. Inf. Syst. Secur.*, 7:1–20, 2004. 121

[11] M. Atici, S.S. Magliveras, D.R. Stinson, and W.-D. Wei. Some recursive constructions for perfect hash families. *Journal of Combinatorial Designs*, 4:353–363, 1996.

[12] D.J. Balding, W.J. Bruno, E. Knill, and D.C. Torney. A comparative survey of nonadaptive probing designs. In *Genetic Mapping and DNA Sequencing, IMA Vol. in Math. and Its Applications*, pages 133–154. Springer-Verlag, 1996. 99

[13] A. Bar-Noy, F.K. Hwang, I. Kessler, and S. Kutten. Competitive group testing in high speed networks. *Discrete Applied Math.*, 52:29–38, 1994. 103

[14] A. Barg, G. Cohen, S. Encheva, G. Kabatiansky, and G. Zémor. A hypergraph approach to the identifying parent property: the case of multiple parents. *SIAM J. Disc. Math.*, 14:423–431, 2001. 5, 6

[15] A. Barg and G. A. Kabatiansky. A class of I.P.P. codes with efficient identification. *J. Complexity*, 20:137–147, 2004. 5

[16] L.A. Bassalygo, M. Burmester, A. Dyachkov, and G. Kabatianski. Hash codes. In *Proceedings of the 1997 IEEE International Symposium on Information Theory*, page 174, 1997. 75

[17] A. Beimel and Y. Stahl. Robust information-theoretic private information retrieval. *Journal of Cryptology*, 20:295–321, 2007. 4

[18] M. Bellare, J. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Proceedings of EUROCRYPT'98*, volume 1403 of *LNCS*, pages 236–250, 1998. 90, 121

[19] M. Bellare and S. Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In *Public Key Cryptography*, volume 4450 of *LNCS*, pages 201–216, 2007. 112, 120

[20] D. Bernstein. Pippenger's exponentiation algorithm. Manuscript. Available online `http://cr.yp.to/papers.html#pippenger`., 2002. 119

[21] K. Bicakci, C. Gamage, B. Crispo, and A. S. Tanenbaum. One-time sensors: A novel concept to mitigate node-capture attacks. In *Proceedings of Security and Privacy in Ad-hoc and Sensor Networks (ESAS'05)*, volume 3813 of *LNCS*, pages 80–90, 2005. 112

[22] K. Bicakci, G. Tsudik, and B. Tung. How to construct optimal one-time signatures. *Computer Networks*, 43:339–349, 2003. 112, 117

[23] S. R. Blackburn, M. Burmester, Y. Desmedt, and P.R. Wild. Efficient multiplicative sharing schemes. In *EUROCRYPT*, pages 107–118, 1996. 4

[24] Simon R. Blackburn. Frameproof codes. *SIAM J. Discrete Math.*, 16:499–510, 2003. 5, 11

[25] S.R. Blackburn. Combinatorics and threshold cryptography. In *Combinatorial Designs and their Applications*, pages 49–70. Chapman and Hall, 1999. 37

[26] S.R. Blackburn. Perfect hash families: Probabilistic methods and explicit constructions. *Journal of Combinatorial Theory, Series A*, 92:54–60, 2000. 28

[27] S.R. Blackburn. An upper bound on the size of a code with the k-identifiable parent property. *Jounal of Combinatorial Theory, Series A*, 102:179–185, 2003. 5

[28] S.R. Blackburn, T. Etzion, D.R. Stinson, and G.M. Zaverucha. A bound on the size of separating hash families. *Jounal of Combinatorial Theory, Series A*, 115:1246–1256, 2008. 6, 11, 27

[29] S.R. Blackburn and P.R. Wild. Optimal linear perfect hash families. *Journal of Combinatorial Theory, Series A*, 83:233–250, 1998. 10, 19, 23, 27, 84

[30] I.F. Blake. Curves with many points and their applications. In *Proceedings of Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-13)*, volume 1719 of *LNCS*, pages 55–64, 1999. 44

[31] I.F. Blake, C. Heegard, T. Høholdt, and V. Wei. Algebraic-geometry codes. *IEEE Transactions on Information Theory*, 44:2596–2618, 1998. 34, 42, 44

[32] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36:1301–1328, 2007. Extended abstract published at EUROCRYPT'04. 107

[33] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of EUROCRYPT'03*, volume 2656 of *LNCS*, pages 416–432, 2003. 122

[34] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Proceedings of ASIACRYPT'01*, volume 2248 of *LNCS*, pages 514–532, 2001.

[35] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17:297–319, 2004. 90, 94, 108, 110

[36] D. Boneh and J. Shaw. Collusion-free fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44:1897–1905, 1998. 5

[37] A. De Bonis and U. Vaccaro. Constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple access channels. *Theoretical Computer Science*, 306:223–243, 2003. 8

[38] J.N. Bos and D. Chaum. Provably unforgeable signatures. In *Proceedings of CRYPTO'92*, volume 740 of *LNCS*, pages 1–14, 1992. 110, 111, 117, 118, 120

[39] C. Boyd and C. Pavlovski. Attacking and repairing batch verification schemes. In *Proceedings of ASIACRYPT'00*, volume 1976 of *LNCS*, pages 58–71, 2000. 90

[40] E.F. Brickell, G. Di Crescenzo, and Y. Frankel. Sharing block ciphers. In *Information Security and Privacy*, volume 1841 of *LNCS*, pages 457–470, 2000. 63, 65

[41] D. Le Brigand and J.J. Risler. Algorithme de Brill-Noether et codes de Goppa. *Bull. Soc. Math. France*, 116:231–253, 1988.

[42] A. Brill and M. Noether. Über die algebraischen Functionen und ihre Anwendung in der Geometrie. *Math. Ann.*, 7:269–310, 1874. 57

[43] J. Buchmann, E. Dahmen, E. Klintsevich, K. Okeya, and C. Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *Proceedings of ACNS'07*, volume 4521 of *LNCS*, pages 31–45, 2007. 112

[44] K.A. Bush, W.T. Federer, H. Pesotan, and D. Raghavarao. New cobminatorial designs and their application to group testing. *Journal of Statistical Planning and Inference*, 10:335–343, 1984. 6

[45] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 201–210. ACM Press, 2006. 96

[46] J. Camenisch, S. Hohenberger, and M. Østergaard Pedersen. Batch verification of short signatures. In *Proceedings of EUROCRYPT'07*, volume 4515 of *LNCS*, pages 246–263, 2007. 89, 90, 96

[47] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of SCN'02*, volume 2576 of *LNCS*, pages 268–289, 2002. 90

[48] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich, 1997. 122, 123

[49] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proceedings of CRYPTO'03*, volume 2729 of *LNCS*, pages 126–144, 2003. 123

[50] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proceedings of IEEE INFOCOM*, volume 2, pages 708–716, 1999. 6

[51] J. Cha and J. Cheon. An identity-based signature scheme from gap Diffie-Hellman groups. In *Proceedings of PKC'03*, volume 2567 of *LNCS*, pages 18–30, 2003. 92

[52] S. Chatterjee and P. Sarkar. Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. In *Proceedings of the 8th International Conference on Information Security and Cryptology (ICISC)*, volume 3935 of *LNCS*, pages 424–440, 2005. 90

[53] B. Chor and R. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34:901–909, 1988. 117

[54] A.E.F. Clementi, A. Monti, , and R. Silvestri. Distributed broadcast in radio networks of unknown topology. *Theoretical Computer Science*, 302:337–364, 2003. 96, 97

[55] D.M. Cohen, S.R. Dalal, M.L. Fredman, and G.C. Patton. The AETG system: An approach to testing based on combinatiorial design. *IEEE Trans. Software Eng.*, 23:437–444, 1997. 4

[56] G.D. Cohen, S.B. Encheva, S. Litsyn, and H.G. Schaathun. Intersecting codes and separating codes. *Discrete Applied Mathematics*, 128:75–83, 2003. 5

[57] G.D. Cohen, S.B. Encheva, and H.G. Schaathun. On separating codes. Technical Report 2001D003, TELECOM ParisTech, Ecole Nationale Superieure des Telecommunications, 2001. 6, 35, 36, 37

[58] G.D. Cohen and H.G. Schaathun. Upper bounds on separating codes. *IEEE Transactions on Information Theory*, 50:1291–1294, 2004. 5

[59] H. Cohen, G. Frey, and C. Doche (Editors). *Handbook of elliptic and hyperelliptic curve cryptography*. Chapman & Hall/CRC, Boca Raton, 2006. 52

[60] T.M. Cover. Enumerative source coding. *IEEE Transactions on Information Theory*, 19:73–77, 1973. 117

[61] E. Dahmen and C. Krauß. Short hash-based signatures for wireless sensor networks. In *Proceedings of Cryptology and Network Security (CANS'09)*, volume 5888 of *LNCS*, pages 463–476, 2009. 107, 112, 119

[62] W. Dai. Crypto++: A free C++ class library of cryptographic schemes. Accessed January 2010. 119

[63] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *Proeedings of EUROCRYPT'00*, volume 1807 of *LNCS*, pages 418–430, 2000. 122

[64] Y. Desmedt, R. Safavi-Naini, H. Wang, L. Batten, C. Charnes, and J. Pieprzyk. Broadcast anti-jamming systems. *Computer Networks*, 35:223–236, 2001. 6

[65] C. Dods, N.P. Smart, and M. Stam. Hash based digital signature schemes. In *Proceedings of Cryptography and Coding 2005*, volume LNCS of *3796*, pages 96–115, 2005. 110, 112

[66] R. Dorfman. The detection of defective members of large populations. *Ann. Math. Statist.*, 14:436–440, 1943. 92

[67] D. Du and F.K. Hwang. *Combinatorial Group Testing and its Applications*. World Scientific, Singapore, 1993. 93, 95

[68] D. Du and F.K. Hwang. *Combinatorial Group Testing and its Applications (2nd Edition)*. World Scientific, Singapore, 2000. 89, 91, 92, 93, 94, 95, 96, 99, 103

[69] A. G. Dyachkov and V. V. Rykov. Bounds on the length of disjunctive codes. *Problemy Peredachi Informatsii*, 18:7–13, 1982. (Russian). 6

[70] A.G. D'yachkov and V.V. Rykov. Optimal superimposed codes and designs for Renyi's search model. *Journal of Statististical Planning and Inference*, 100:281–302, 2002. 91

[71] M. Dyer, Trevor Fenner, Alan Frieze, and Andrew Thomason. On key storage in secure networks. *Journal of Cryptology*, 8:189–200, 1995. 6

[72] R. Ehrenborg and S. van Willigenburg. Enumerative properties of Ferrers graphs. *Discrete and Computational Geometry*, 32:481–492, 2004. 49, 127

[73] S. Encheva and G. Cohen. Partially identifying codes for copyright protection. In *Proceedings of AAECC-14*, volume 2227 of *LNCS*, pages 260–267, 2001. 5

[74] S. Encheva and G. Cohen. Some new p-ary two-secure frameproof codes. *Applied Mathematics Letters*, 14:177–182, 2001. 5

[75] S. Encheva and G. Cohen. Frameproof codes against limited coalitions of pirates. *Theoretical Computer Science*, 273:295–304, 2002. 5

[76] P. Erdös, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of two others. *Journal of Combinatorial Theory, Series A*, 33:158–166, 1982. 6

[77] 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*, 51:75–89, 1985. 6

[78] S. Even and O. Goldreich. On the power of cascade ciphers. *ACM Transactions on Computer Systems*, 3:108–116, 1985. 65

[79] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9:35–67, 1996. 107

[80] A.L. Ferrara, M. Green, S. Hohenberger, and M. Østergaard Pedersen. Practical short signature batch verification. In *Proceedings of CT-RSA'09*, volume 435 of *LNCS*, pages 309–324, 2009. 89, 90, 92, 94, 96, 100

[81] A. Fiat. Batch RSA. In *Proceedings of CRYPTO'89*, volume 435 of *LNCS*, pages 175–185, 1989. 90

[82] A. Fiat. Batch RSA. *Journal of Cryptology*, 10:75–88, 1997. 90

[83] A. Fiat and M. Naor. Broadcast encryption. In *Proceedings of CRYPTO'93*, volume 773 of *LNCS*, pages 480–491, 1993. 4

[84] M.L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *J. ACM*, 31:538–544, 1984.

[85] J.A. Garay, J.N. Staddon, and A. Wool. Long-lived broadcast encryption. In *Proceedings of CRYPTO'00*, volume 1880 of *LNCS*, pages 333–352, 2000. 6, 96

[86] A. Garcia and H. Stichtenoth. A tower of Artin-Schreier extensions of function fields attaining the Drinfeld-Vladut bound. *Inventiones Mathematicae*, 121:211–222, 1995. 54

[87] A. Garcia and H. Stichtenoth. On the asymptotic behaviour of some towers of function fields over finite fields. *Journal of Number Theory*, 61:248–273, 1996. 54, 56

[88] R. Genarro and P. Rohatgi. How to sign digital streams. In *Proceedings of CRYPTO '97*, volume 1294 of *LNCS*, pages 180–197, 1997. 113

[89] S. Goldwasser, S. Micali, and R.L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17:281–308, 1988. 111

[90] V.D. Goppa. Algebraico-geometric codes. *Math. USSR Izvestiya*, 21:75–91, 1983. 44

[91] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *In proceedings of ASIACRYPT'06*, volume 4284 of *LNCS*, pages 444–459, 2006. 111, 120

[92] N. Gura, A. Patel, A. Wander, H. Eberle, and S.C. Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Proceedings of CHES '04*, volume 3156 of *LNCS*, pages 118–132, 2004. 124

[93] G. Haché. PAFF: Package for algebraic function fields in one variable. 57

[94] G. Haché. *Construction effective des codes géométriques*. PhD thesis, Université Pierre et Marie Curie (Paris 6), 1996. 57

[95] H. Hollmann, J. van Lint, J. Linnartz, and L. Tolhuizen. On codes with the identifiable parent property. *Journal of Combinatorial Theory, Series A*, 82:121–133, 1998. 12

[96] H.D. Hollmann, J.H. van Lint, J-P. Linnartz, and L.M. Tolhuizen. On codes with the identifiable parent property. *Journal of Combinatorial Theory Series A*, 82:121–133, 1998. 5, 48

[97] M.C. Hu, F.K. Hwang, and J.K. Wang. A boundary problem for group testing. *SIAM J. Alg. Disc. Methods*, 2:81–87, 1981. 94

[98] R.A. Walker II. PHFtables.com. Accessed April 2008. 71, 83, 85

[99] S. Jukna. *Extremal Combinatorics with Applications in Computer Science*. Springer, 2001. 29

[100] F. Kargl, P. Papadimitratos, L. Buttyan, M. Müter, E. Schoch, B. Wiedersheim, T.-V. Thong, G. Calandriello, A. Held, A. Kung, and J.-P. Hubaux. Secure vehicular communication systems: Implementation, performance, and research challenges. *IEEE Communications Magazine*, 46:110–118, 2008. 124

[101] R.M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *Journal of Computer and System Sciences*, 36:225–253, 1988. 89, 95

[102] J. Katz. Signature schemes with bounded leakage resilience. Technical Report 2009/220, IACR ePrint Archive, 2009. `http://eprint.iacr.org/2009/220`. 111

[103] J. Katz and A.Y. Lindell. Aggregate message authentication codes. In *Proceedings of CT-RSA '08*, volume 4964 of *LNCS*, pages 155–169, 2008. 65

[104] W.H. Kautz and R.G. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10:363–373, 1964. 6, 38, 96

[105] D.L. Kreher and D.R. Stinson. *Combinatorial Algorithms: Generation, Enumeration and Search*. CRC Press, Boca Raton FL, 1999. 117

[106] R. Kumar, S. Rajagopalan, and A. Sahai. Coding constructions for blacklisting problems without computational assumptions. In *Proceedings of CRYPTO'99*, volume 1666 of *LNCS*, pages 609–623, 1999. 6

[107] L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, 1979. 107, 110, 120

[108] L. Law and B.J. Matt. Finding invalid signatures in pairing-based batches. In *Proceedings of Cryptography and Coding 2007*, volume 4887 of *LNCS*, pages 34–53, 2007. 89, 92, 99, 100, 103

[109] L. Liu and H. Shen. Explicit constructions of separating hash families from algebraic curves over finite fields. *Designs, Codes and Cryptography*, 41:221–233, 2006. 5, 34, 44, 45, 46, 47, 52, 55, 75

[110] S. Long, J. Pieprzyk, H. Wang, and D.S. Wong. Generalised cumulative arrays in secret sharing. *Designs, Codes and Cryptography*, 40:191–209, 2006. 63, 64, 83

[111] B. Lòpez. Codes on Drinfeld modular curves. *Coding Theory, Cryptography and Related Areas*, pages 175–183, 1998.

[112] M. Luk, A. Perrig, and B. Whillock. Seven cardinal properties of sensor network broadcast authentication. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks (SASN '06)*, pages 147–156. ACM Press, 2006. 123

[113] H. Maharaj, G.L. Matthews, and G. Pirsic. Riemann-Roch spaces of the Hermitian function field with applications to algebraic geometry codes and low-discrepancy sequences. *Journal of Pure and Applied Algebra*, 195:261–280, 2005. 53

[114] K.M. Martin and S.-L. Ng. The combinatorics of generalised cumulative arrays. *Journal of Mathematical Cryptology*, 1:13–32, 2007. 63, 83

[115] K.M. Martin, J. Pieprzyk, R. Safavi-Naini, H. Wang, and P.R. Wild. Threshold MACs. In *Proceedings of ICISC 2002*, volume 2587 of *LNCS*, pages 237–252, 2003. 4, 63, 65, 66, 67, 68, 83

[116] K.M. Martin, R. Safavi-Naini, H. Wang, and P.R. Wild. Distributing the encryption and decryption of a block cipher. *Designs, Codes and Cryptography*, 36:263–287, 2005. 4, 63, 65, 66, 83

[117] S. Martirosyan and T. van Trung. On t-covering arrays. *Designs, Codes and Cryptography*, 32:323–339, 2004. 4

[118] S. Martirosyan and T. van Trung. Explicit constructions for perfect hash families. *Designs, Codes and Cryptography*, 41:97–112, 2008. 34, 41

[119] B.J. Matt. Identification of multiple invalid signatures in pairing-based batched signatures. In *Proceedings of PKC 2009*, volume 5443 of *LNCS*, pages 337–356, 2009. 89, 92, 99, 100, 103

[120] K. Mehlhorn. On the program size of perfect and universal hash functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 170–175, Washington, DC, USA, 1982. IEEE Computer Society. 4

[121] K. Mehlhorn. *Data Structures and Algorithms, Vol. 1.* Springer-Verlag, 1984. 4, 68, 85

[122] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press LLC, Boca Raton, FL, 1996. 110, 112

[123] R. Merkle. A certified digital signature. In *Proceedings of CRYPTO '89*, volume 435 of *LNCS*, pages 218–238, 1989. 112

[124] C.J. Mitchell and F.C. Piper. Key storage in secure networks. *Discrete applied mathematics*, 21:215–228, 1988. 6, 96

[125] P. Mohassel. One-time signatures and chameleon hash functions. To appear in the *Proceedings of Selected Areas in Cryptography (SAC)* 2010. 112

[126] D. Naccache, D. M'raihi, S. Vaudenay, and D. Raphaeli. Can DSA be improved? complexity trade-offs with the digital signature standard. In *Proceedings of EUROCRYPT'94*, volume 950 of *LNCS*, pages 77–85, 1994. 90, 91

[127] D. Naor, A. Shenhav, and A. Wool. One-time signatures revisited: Have they become practical? Technical Report 2005/442, IACR ePrint Archive Report, 2005. `http://eprint.iacr.org/2005/442`. 112, 120

[128] Q.A. Nguyen and T. Zeisel. Bounds on constant weight binary superimposed codes. *Problems in Control and Information Theory*, 17:223–230, 1988. 38

[129] M. Noether. Rationale Ausführung der Operationen in der Theorie der algebraischen Functionen. *Math. Ann.*, pages 311–358, 1884. 57

[130] National Institute of Standards and Technology (NIST). Digital signature standard (DSS), 2000. FIPS PUB 186-2. 118

[131] P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *Proceedings of ASIACRYPT'05*, volume 3788 of *LNCS*, pages 1–20, 2005. 112

[132] J. Pastuszak, D. Michalek, J. Pieprzyk, and J. Seberry. Identification of bad signatures in batches. In *Proceedings of PKC'00*, volume 1751 of *LNCS*, pages 28–45, 2000. 89, 91, 92, 100

[133] J. Pastuszak, J. Pieprzyk, and J. Seberry. Codes identifying bad signatures in batches. In *Proceedings of INDOCRYPT'00*, volume 1977 of *LNCS*, pages 143–154, 2000. 89, 91, 92, 98, 100

[134] T.P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of CRYPTO'91*, volume 1440 of *LNCS*, pages 129–140, 1992. 108, 113

[135] A. Perrig. The BiBa one time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS '01)*, pages 28–37, New York, 2001. ACM Press. 107, 108, 113

[136] J. Pieprzyk, H. Wang, and C. Xing. Multiple-time signature schemes against chosen message attacks. In *Proceedings of Selected Areas in Cryptography (SAC '03)*, volume 3006 of *LNCS*, pages 88–100, 2003. 110, 111, 117, 120

[137] E. Porat and A. Rothschild. Explicit non-adaptive combinatorial group testing schemes. In *Proceedings of Automata, Languages and Programming, ICALP'08*, volume 5125 of *LNCS*, pages 748–759, 2008. 97

[138] M.O. Rabin. Digitalized signatures. In *Foundations of Secure Computation*, pages 155–168, New York, 1978. Academic Press. 107

[139] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proceedings of PKC 2001*, volume 1992 of *LNCS*, pages 119–136, 2001. 123

[140] R.C. Read. An introduction to chromatic polynomials. *Journal of Combinatorial Theory*, 4:52–71, 1968. 29, 127

[141] A. Reyni. On the theory of random search. *Bulletin of the AMS*, 71:809–828, 1965. 91

[142] L. Reyzin and N. Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Proceedings of ACISP '02*, volume 2384 of *LNCS*, pages 144–153, 2002. 108, 110, 111, 113, 117, 120

[143] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS '99)*, pages 93–100, New York, 1999. ACM Press. 113, 120

[144] S. Rohde, T. Eisenbarth, E. Dahmen, J. Buchmann, and C. Paar. Fast hash-based signatures on constrained devices. In *Proceedings of CARDIS'08*, volume 5189 of *LNCS*, pages 104–117, 2008. 112

[145] M. Ruszinkó. On the upper bound of the size of the r-cover-free families. *Journal of Combinatorial Theory, Series A*, 66:302–310, 1994. 8

[146] R. Safavi-Naini and H. Wang. Multireceiver authentication codes: Models, bounds, constructions and extensions. *Information and Computation*, 151:148–172, 1999. 6

[147] P. Sarkar and D.R. Stinson. Frameproof and IPP codes. In *Proceedings of INDOCRYPT'01*, volume 2247 of *LNCS*, pages 117–126, 2001. 5, 35

[148] H.G. Schaathun and G.D. Cohen. A trellis-based bound on (2,1)-separating codes. In *Proceedings of Cryptography and Coding 2005*, volume 3796 of *LNCS*, pages 59–67, 2005. 5

[149] D.J. Shultz. *Topics in nonadaptive group testing.* PhD thesis, Temple University, 1992. 97

[150] K. Shum. *A low-complexity construction of algebraic geometric codes better than the Gilbert-Varshamov bound.* PhD thesis, University of South California, Los Angeles, 2001.

[151] K. Shum, I. Aleshnikov, P. V. Kumar, H. Stichtenoth, and V. Deolalikar. A low-complexity algorithm for the construction of algebraic-goemetric codes better than the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 47:2225–2241, 2001. 58

[152] 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. 93

[153] J. Spencer. Minimal completely separating systems. *Journal of Combinatorial Theory*, 8:446–447, 1970. 8

[154] E. Sperner. Ein Satz Uber Untermengen einer endliche Menge. *Math. Zeit.*, 27:544–548, 1928. 8, 110

[155] J.N. Staddon, D.R. Stinson, and R. Wei. Combinatorial properties of frameproof and traceability codes. *IEEE Transactions on Information Theory*, 47:1042–1049, 2001. 4, 5, 6, 10, 11, 19, 21, 23, 48, 96

[156] H. Stichtenoth. *Algebraic Function Fields and Codes*. Springer-Verlag, New York, 1993. 42, 44, 45, 57

[157] D. R. Stinson and T. van Trung. Some new results on key distribution patterns and broadcast encryption. *Designs, Codes and Cryptography*, 14:261–279, 1998. 6

[158] D.R. Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. *Designs, Codes and Cryptography*, 12:215–243, 1997. 4, 6

[159] D.R. Stinson. Some baby-step giant-step algorithms for the low Hamming weight discrete logarithm problem. *Mathematics of Computation*, 71:379–391, 2002. 73

[160] D.R. Stinson. Attack on a concast signature scheme. *Information Processing Letters*, 91:39–41, 2004. 90

[161] 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*, 86:595–617, 2000. 4, 5, 6, 73, 77, 84, 85, 96

[162] D.R. Stinson and R. Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM J. Discrete Math*, 11:41–53, 1998. 6

[163] D.R. Stinson and R. Wei. Generalized cover-free families. *Discrete Mathematics*, 279:463–477, 2004. 75, 118

[164] D.R. Stinson, R. Wei, and K. Chen. On generalized separating hash families. *Journal of Combinatorial Theory, Series A*, 115:105–120, 2008. 6, 10, 11, 27, 28, 29, 48

[165] D.R. Stinson, R. Wei, and L. Zhu. New constructions for perfect hash families and related structures using combinatorial designs and codes. *Journal of Combinatorial Designs*, 8:189–200, 2000. 5, 34, 36, 37, 38

[166] D.R. Stinson, R. Wei, and L. Zhu. Some new bounds for cover-free families. *Journal of Combinatorial Theory, Series A*, 90:224–234, 2000. 8

[167] D.R. Stinson and G.M. Zaverucha. New bounds for generalized separating hash families. Technical Report 2007-21, Center for Applied Cryptographic Research, University of Waterloo, 2007. 11

[168] D.R. Stinson and G.M. Zaverucha. Some improved bounds for secure frameproof codes and related separating hash families. *IEEE Transactions on Information Theory*, 54:2508–2514, 2008. 11

[169] The Axiom Scientific Computation System. `http://www.axiom-developer.org/`. Accessed February 2008. 57

[170] The Singular Computer Algebra System. `http://www.singular.uni-kl.de`. Accessed February 2008. 57

[171] P. Szczechowiak, L.B. Oliveira, M. Scott, M. Collier, and R. Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *Proceedings of EWSN '08*, volume 4913 of *LNCS*, pages 305–320, 2008. 124

[172] V.D. Tô, R. Safavi-Naini, and Y. Wang. A 2-secure code with efficient tracing algorithm. In *Proceedings of INDOCRYPT'02*, volume 2551 of *LNCS*, pages 149–162, 2002. 5

[173] D. Tonien and R. Safavi-Naini. Explicit construction of secure frameproof codes. *International Journal of Pure and Applied Mathematics*, 6:343–360, 2003. 5

[174] D. Tonien and R. Safavi-Naini. Recursive constructions of secure codes and hash families using difference function families. *Journal of Combinatorial Theory, Series A*, 113:664–674, 2006. 5

[175] M. Tsfasman, S. Vlăduţ, and D. Nogin. *Algebraic Geometry Codes: Basic Notions*, volume 139 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2007. 42, 53

[176] E. van Heyst and T.P. Pedersen. How to make efficient fail-stop signatures. In *Proceedings of EUROCRYPT '92*, volume 658 of *LNCS*, pages 366–377, 1993. 111, 120

[177] J.H. van Lint. *Introduction to Coding Theory*. Springer, New York, 1998. 36, 38

[178] T. van Trung and S. Martirosyan. New constructions for ipp codes. *Designs, Codes and Cryptography*, 35:227–239, 2005. 5

[179] R.A. Walker and C.J. Colbourn. Perfect hash families: constructions and existence. *Journal of Mathematical Cryptology*, 1:125–150, 2007. 34, 39, 41

[180] H. Wang and J. Pieprzyk. Shared generation of pseudo-random function with cumulative maps. In *Proceedings of CT-RSA '03*, volume 2612 of *LNCS*, pages 281–294, 2003. 63

[181] H. Wang and C. Xing. Explicit constructions of perfect hash families from algebraic curves over finite fields. *Journal of Combinatorial Theory, Series A*, 93:112–124, 2001. 34, 44, 45, 73, 84

[182] D.J.A. Welsh. *Complexity: Knots, Colourings and Counting*, volume 186 of *LMS Lecture Note Series*. Cambridge University Press, 1993. 49

[183] C. Xing. Asymptotic bounds on frameproof codes. *IEEE Transactions on Information Theory*, 48:2991–2995, 2002. 5

[184] G.M. Zaverucha and D.R. Stinson. Group testing and batch verification. To appear in the *Proceedings of the 4th International Conference on Information Theoretic Security (ICITS'09)*. 88

[185] G.M. Zaverucha and D.R. Stinson. Anonymity in shared symmetric key primitives. *Designs, Codes and Cryptography*, 57:139–160, 2010. 62