Fault-Tolerant Aggregate Signatures

Gunnar Hartung, Björn Kaidel, Alexander Koch, Jessica Koch, and Andy Rupp

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{gunnar.hartung, bjoern.kaidel, alexander.koch, jessica.koch, andy.rupp}@kit.edu

Abstract. Aggregate signature schemes allow for the creation of a short aggregate of multiple signatures. This feature leads to significant reductions of bandwidth and storage space in sensor networks, secure routing protocols, certificate chains, software authentication, and secure logging mechanisms. Unfortunately, in all prior schemes, adding a single *invalid* signature to a valid aggregate renders the whole aggregate invalid. Verifying such an invalid aggregate provides no information on the validity of any individual signature. Hence, adding a single faulty signature destroys the proof of integrity and authenticity for a possibly large amount of data. This is largely impractical in a range of scenarios, e.g. secure logging, where a single tampered log entry would render the aggregate signature of all log entries invalid.

In this paper, we introduce the notion of fault-tolerant aggregate signature schemes. In such a scheme, the verification algorithm is able to determine the subset of all messages belonging to an aggregate that were signed correctly, provided that the number of aggregated faulty signatures does not exceed a certain bound.

We give a generic construction of fault-tolerant aggregate signatures from ordinary aggregate signatures based on cover-free families. A signature in our scheme is a small vector of aggregated signatures of the underlying scheme. Our scheme is bounded, i.e. the number of signatures that can be aggregated into one signature must be fixed in advance. However the length of an aggregate signature is logarithmic in this number. We also present an unbounded construction, where the size of the aggregate signature grows linearly in the number of aggregated messages, but the factor in this linear function can be made arbitrarily small.

The additional information encoded in our signatures can also be used to speed up verification (compared to ordinary aggregate signatures) in cases where one is only interested in verifying the validity of a single message in an aggregate, a feature beyond fault-tolerance that might be of independent interest. For concreteness, we give an instantiation using a suitable cover-free family.

Keywords: Aggregate Signatures · Fault-Tolerance · Cover-free Family

This work was supported by the German Federal Ministry of Education and Research within the framework of the project KASTEL_IoE in the Competence Center for Applied Security Technology (KASTEL).

1 Introduction

Aggregate signature schemes allow anyone to aggregate multiple signatures by different signers into a single combined signature, which is considerably smaller than the size of the individual signatures. This type of digital signature schemes was first proposed and instantiated by Boneh, Gentry, Lynn and Shacham [Bon+03], and has since evolved into a diverse and active research area.

Applications of Aggregate Signatures. The main motivation for aggregate signature schemes is to save bandwidth and storage space. Therefore, their applications are manifold [AGH10].

A well-known field of application are *sensor networks*, which consist of several small sensors that measure an aspect of their physical environment and send their findings to a central base station. Digital signatures ensure the integrity and authenticity of the measurements during transfer from the sensors to the base station. Using a conventional digital signature scheme, the verifying base station would need to receive each signature separately, which is bandwidth-intensive. However, if the signatures were aggregated beforehand using an aggregate signature scheme, the bandwidth consumption on the side of the base station is reduced drastically. Also, verifying an aggregate signature is typically considerably faster than verifying all individual signatures.

Another application is *secure logging*. Log files are used to record events like user actions, system errors, failed log-in attempts as well as general information, and play an important role in computer security by providing, for example, accountability and a basis for intrusion detection. Log files are usually kept for very long periods of time, which means that thousands or even millions of log entries need to be stored. Digital signatures are used to ensure the integrity of the log data. For aggregate signature schemes, it is sufficient to store one single aggregate signature over all log entries, instead of an individual signature per log entry as with a normal digital signature scheme. Whenever a new log entry is added to the log file, one simply calculates a signature for the new entry and aggregates it into the already existing aggregate signature.

Aggregate signatures can also be useful for *authenticating software*. To ensure the validity of software libraries and programs it has become common to sign their code and/or compiled binaries. Mobile operating systems often only allow signed programs to be executed. Again, it is advantageous to use an aggregate signature to save download bandwidth and verification overhead upon execution, e.g. if all programs are verified at boot time. Like in the logging scenario, aggregate signatures allow for installation of new applications without having to store the individual signatures of all installed programs.

Problem Statement. In all known aggregate signature schemes an aggregate signature is invalid (i.e., verification fails) if just one invalid message–signature pair is contained in the aggregate. Note that either this pair was already invalid (i.e., the individual signature was not valid for this particular message) when

the aggregate was created or a "wrong" message is included for verification. In any case, the verification algorithm can give no information about which message–signature pair is the reason for the failure or if other message–signature pairs were valid. This essentially renders the aggregate useless after an invalid signature is added, even though the majority of the messages might have been correctly signed.

For sensor networks, this means that the measurements of all sensors are lost even if only a single sensor sends an invalid signature, for example because of calculation glitches or transmission errors. Usually it is not feasible for computationally weak sensors to ensure the validity of their signature before sending it, since many signature schemes use expensive operations like pairings for verification. An aggregator could ensure the validity of the individual signatures before aggregation, but this would undo one of the advantages of the aggregate signature scheme altogether.

The same problem occurs in the logging scenario: If one log entry is not correctly signed, is tampered with, or is lost (for example through hard disk errors or crashes), the signature for the whole log file becomes invalid. In [MT09] Ma and Tsudik state that this is one of the reasons why they still need to store individual signatures for every log entry, although they use an aggregated signature for the complete log. This is undesirable, since one of the motivations for using the aggregate signature schemes for logging is to save storage space.

This problem also affects the software authentication scenario. If a new program is installed and its signature is invalid or the code of an already installed program gets changed (through hard disk problems etc.), the whole aggregated signature used for authenticating the software becomes invalid. In the worst case this would mean that no program can be executed anymore, because the operating system might block every unauthenticated program.

Contribution. To solve the above mentioned problems, we introduce the concept of fault-tolerant aggregate signature schemes, which are able to tolerate a specific number of invalid (or faulty) signatures while aggregating. In such a scheme, the verification algorithm does not output boolean values like "valid" and "invalid" but instead outputs a list of validly signed messages and will leave out all messages that are invalid.

Note that in contrast to ordinary aggregate signatures, fault-tolerant aggregate signatures cannot offer an aggregate signature size which is independent of the number of individual signatures to be aggregated. In other words, we cannot hope to aggregate an unlimited number of individual signatures using a constant-size aggregate. This easily follows from an information-theoretic argument: Let us assume we fix the size of an aggregate signature to l bits. This l-bit string then needs to be used by the verification algorithm (as the only "source of information") to determine which of its input messages are valid. Hence, based on the l-bit string, the algorithm can distinguish at most 2^l different outputs. However, considering n messages and corresponding individual signatures, d of which are invalid, there are $\binom{n}{n-d}$ possible different subsets (and thus outputs)

which should be distinguishable by the verification algorithm by considering this string. So n is upper bounded by $\binom{n}{n-d} \leq 2^l$. (For a more formal argument using the notation of fault-tolerant aggregate signature introduced later, refer to Appendix A.)

Besides a formal framework for fault-tolerant aggregate signatures, we also present a generic construction which can be used to turn any aggregate signature scheme into a fault-tolerant scheme. This construction makes use of cover-free families [KS64] to provide fault-tolerance and comes with a tight security reduction to the underlying signature scheme. For concreteness, we explicitly describe how to instantiate our scheme with a cover-free family based on polynomials over a finite field [KRS99], which has a compact representation. (We generalize the known family to multivariate polynomials inAppendix B.) This leads to an instantiation featuring short aggregate signatures relative to the number n of individual signatures that are aggregated (provided that the maximal number of faults the scheme should tolerate is relatively small compared to n).

As an additional feature, our construction allows the verification of an individual signature in a fashion that is more efficient (e.g., saving a number of costly pairing operations in the case of pairing-based aggregate signatures) than verifying the complete aggregate. This provides a level of flexibility to the signature scheme as demanded by certain applications such as secure logging [MT09].

As a shortcoming of our scheme, we need to *assume* that aggregates may only contain a previously fixed upper bound d of invalid individual signatures. If for some reason this bound is exceeded, the faulty signatures may affect the verifiability of other messages, as is the case for common aggregate signatures. This is also analogous to error-correction codes (which are related to cover-free families), where only a specific number of errors can be located.

Basic Idea of Our Construction. To get a glimpse of our generic construction of a fault-tolerant aggregate signature scheme, we now informally illustrate the basic idea. Let an ordinary aggregate signature scheme (e.g., BGLS) and n individual signatures $\sigma_1, \ldots, \sigma_n$ generated using this scheme be given. Our goal is to detect $d = 1$ faulty individual signatures. To achieve this, our approach is to choose m subsets $T_1, \ldots, T_m \subset \{\sigma_1, \ldots, \sigma_n\}$ of individual signatures and aggregate the signatures of each subset, thereby yielding aggregate signatures $\tau_1, \ldots, \tau_m$, such that

1. m is (significantly) smaller than n and
2. even if one of the individual signatures is faulty and the corresponding aggregate signatures τ_i will be invalid, all other individual signatures σ_j are aggregated into at least one different, valid signature τ_k.

For example, consider the following binary 4×6 matrix

$$A := (a_{i,j}) := \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

for which $n = 6$ and $m = 4$.

A describes a solution to the above mentioned problem as follows: The 1-entries in column j indicate in which T_i the individual signature σ_j is contained. Consequently, the 1-entries in row i indicate the σ_j contained in T_i. More precisely, $T_i := \{\sigma_j \colon a_{i,j} = 1\}$ and τ_i is the aggregate of all $\sigma_j \in T_i$. Observe that while it is usually unnecessary for the verification to know the order of the claims from the aggregation process, in our fault-tolerant scheme, specifying the correct order is inevitable.

For the matrix above, an aggregate signature $\tau = (\tau_1, \tau_2, \tau_3, \tau_4)$ for individual signatures $\sigma_1, \ldots, \sigma_6$ would be formed in the following manner:

$$\tau = \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{pmatrix} = \begin{pmatrix} \mathsf{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathsf{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathsf{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathsf{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix} \mathrel{\hat{=}} \begin{pmatrix} \sigma_1 & & & \sigma_4 & & \sigma_6 \\ \sigma_1 & \sigma_2 & & & \sigma_5 & \\ & \sigma_2 & \sigma_3 & \sigma_4 & & \\ & & \sigma_3 & & \sigma_5 & \sigma_6 \end{pmatrix},$$

where Agg informally denotes the aggregation function of the underlying aggregate signature scheme.

Let us assume that only one signature σ_j is faulty. Then all τ_i are faulty where $a_{i,j} = 1$. However, because all other σ_k were also aggregated into at least one different τ_i, we can still derive the validity of σ_k.

For a concrete example, suppose σ_1 is faulty. Then τ_1 and τ_2 will be faulty, whereas τ_3 and τ_4 are valid. We see that σ_2, σ_3 and σ_4 occur in τ_3, and σ_5, σ_6 occur in τ_4, and so we may be sure that the corresponding messages were signed.

The matrix A defined above has the property that it can tolerate one faulty signature, i.e., if just one signature is faulty, then all other messages can still be verified. Unfortunately, this is not possible if two or more faulty signatures are aggregated. Lets assume that σ_1 and σ_2 are faulty. In this case, τ_1, τ_2 and τ_3 become invalid and τ_4 is the only valid signature. We could still derive the validity of $\sigma_3, \sigma_5, \sigma_6$, because τ_4 is valid. However, the validity of σ_1, σ_2 and σ_4 can no longer be verified, since they were never aggregated to τ_4.

Note that our scheme does not support fully flexible aggregation: Each column of A can only be used to hold one individual signature, as can be seen in the example above. Two aggregate signatures where the same column is used can not be aggregated further without losing the guarantee of fault-tolerance. However, our scheme still supports a notion of aggregation which is only slightly restricted: Individual signatures can always be aggregated, while aggregate signatures can only be aggregated if no column is used in both. As long as this requirement is met, signatures can be aggregated in any order. This notion is sufficient for many use cases, we discuss this further in Section 3.

The construction of matrices that can tolerate $d > 1$ faulty signatures is more intricate, but incidence matrices belonging to d-cover-free families turned out to imply the desired property. Informally speaking, in such a matrix the "superposition" $\boldsymbol{s}$ of up to d arbitrary column vectors $\boldsymbol{a_{i_1}}, \ldots, \boldsymbol{a_{i_d}}$, i.e., the vector $\boldsymbol{s}$ which has a 1 at position ℓ if at least one of the vectors $\boldsymbol{a_{i_1}}, \ldots, \boldsymbol{a_{i_d}}$ has a 1 at this position, does not "cover" any other distinct column vector $\boldsymbol{a_j}$ ($j \notin \{i_1, \ldots, i_d\}$).

In other words, there is at least one position ℓ such that $\boldsymbol{a_j}$ has a 1 at this position but $\boldsymbol{s}$ shows a 0. This implies that if at most d individual signatures (each belonging to one column) are invalid, then each distinct individual signature is contained in at least one valid aggregate signature, and the corresponding message can therefore be trusted. Hence applying such a matrix, as sketched above, implies that any subset of faulty individual signatures of size up to d will not compromise the trustworthiness of any other message. There are different constructions of d-cover-free families for d and n of unlimited size (where these parameters need to satisfy certain conditions depending on the family) featuring $m \ll n$ for choices of the parameters n, d with $d \ll n$.

Related Work. The first full aggregate scheme was constructed by Boneh et al. [Bon+03] in the random oracle model. Full aggregate schemes allow any user to aggregate signatures of different signers, i.e., aggregation is a public operation. Furthermore it is possible to aggregate individual signatures as well as already aggregated signatures in any order. In [HSW13] Hohenberger, Sahai and Waters give the first construction of such a scheme in the standard model using multilinear maps. Recently, Hohenberger, Koppula, and Waters [HKW15] have constructed a "universal signature aggregator" based on indistinguishability obfuscation. A universal signature aggregator can aggregate signatures from any set of signing algorithms, even if they use different algebraic settings. In [ZS11] Zaverucha and Stinson construct an aggregate one-time-signature.

Since it has proven difficult to construct full aggregate schemes in the standard model, a lot of research was focused on signature schemes with some form of *restricted* aggregation. One major type of restricted aggregation is *sequential* aggregation, as proposed by Lysyanskaya, Micali, Reyzin and Shacham [Lys+04]. In these schemes, the aggregate is sequentially sent from signer to signer and each signer can add new information to the aggregate. Multiple constructions are known, both in the random oracle [Lys+04; Nev08; Bol+07; Ger+12] and the standard model [Lu+06; Sch11; LLY15]. Another type of aggregation is *synchronized* aggregation, as proposed by Gentry and Ramzan [GR06]. Here, a special synchronization information, like the current time period, is used while signing. All signatures sharing the same synchronizing information behave like signatures of a full aggregate scheme, i.e., both individual and aggregated signatures can be aggregated in any order. Again, schemes in the random oracle [AGH10; GR06] and standard model [AGH10] are known. Other authors considered aggregate signature schemes that need interaction between the signers [BN07; BJ10] or can only partially aggregate the signatures [Her06; BGR14].

Our construction is based on cover-free families, which are a combinatorial structure that was first introduced by Kautz and Singleton [KS64] in the language of coding theory. They have several applications in cryptography, for example group testing [STW97], multireceiver authentication codes [SW99], encryption [Cra+07; Dod+02; HK04] and traitor-tracing [TS06]. There are multiple constructions of signature schemes using cover-free families. Hofheinz, Jager, and Kiltz [HJK11] use cover-free families to construct a $(m, 1)$-programmable hash function.

They then use this hash function to construct conventional digital signature schemes from weak assumptions. Zaverucha and Stinson [ZS11] construct an aggregate one-time-signature using cover-free families.

Outline. Section 2 introduces some notations, conventions, and preliminary definitions. Section 3 presents a general definition of fault-tolerant aggregate signature schemes and some properties of such schemes, such as the security definition. Our construction is presented and analyzed in Section 4. Afterwards, we discuss an instantiation of our scheme with a specific class of cover-free families in Section 5.

2 Preliminaries

Let $[n] \coloneqq \{1, \ldots, n\}$. The *multiplicity* of an element m in a multiset M is the number of occurrences of m in M. For two multisets M_1, M_2, the union $M_1 \cup M_2$ is defined as the multiset where the multiplicity of each element is the sum of the multiplicities in M_1, M_2.

If v is a vector or a tuple, $v[i]$ refers to the i-th entry of v. If M is a matrix, $\mathrm{rows}(M)$ and $\mathrm{cols}(M)$ denote the number of rows and columns of M, respectively. For $i \in [\mathrm{rows}(M)], j \in [\mathrm{cols}(M)]$, $M[i, j]$ is the entry in the i-th row and j-th column of M.

Throughout the paper, $\kappa \in \mathbb{N}$ is the security parameter. We say an algorithm A is probabilistic polynomial time (PPT) if the running time of A is polynomial in κ and A is a probabilistic algorithm. All algorithms are implicitly given 1^κ as input, even when not noted explicitly.

In this work, σ usually refers to signatures of standard aggregate signature schemes, whereas τ mostly refers to signatures of a fault-tolerant aggregate signature scheme.

2.1 Aggregate Signatures

Let us quickly review the definition of aggregate signature schemes and the associated security notion, as defined in [Bon+03]. An *aggregate signature scheme* is a tuple of four PPT algorithms:

- $\mathsf{KeyGen}(1^\kappa)$ creates a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$ creates a signature for message m under secret key sk.
- $\mathsf{Agg}(C_1, C_2, \sigma_1, \sigma_2)$ takes as input two multisets of public-key and message pairs C_1 and C_2 and corresponding signatures σ_1 and σ_2 and creates an aggregate signature σ, certifying the validity of the messages in $C_1 \cup C_2$ under the corresponding public keys.
- $\mathsf{Verify}(C, \sigma)$ takes as input a multiset of public-key and message pairs C and an aggregate signature σ for C and outputs 1, if the signature is valid, and 0 otherwise.

For *correctness* we require that any signature that is generated by the signature scheme by applications of Sign and Agg using key pairs of the scheme, is valid, i.e. Verify outputs 1.

Security Notion for Aggregate Signatures. The security experiment for aggregate signatures consists of three phases [Bon+03]:

- *Setup Phase.* The challenger generates a pair of keys $(\mathsf{pk}, \mathsf{sk}) := \mathsf{KeyGen}(1^\kappa)$ and gives the public key pk to the adversary.
- *Query Phase.* The adversary $\mathcal{A}$ may (adaptively) issue signature queries m_i to the challenger, who responds with $\sigma_i := \mathsf{Sign}(\mathsf{sk}, m_i)$.
- *Forgery Phase.* Finally, $\mathcal{A}$ outputs a multiset of public-key and message pairs C^* and a signature σ^*.

The adversary *wins* the experiment iff there is a message m^* such that $c^* = (\mathsf{pk}, m^*)$ is in C^*, $\mathsf{Verify}(C^*, \sigma^*) = 1$, and m^* has never been submitted to the signature oracle.

An aggregate signature scheme Σ is (t, q, ε)-*secure* if there is no adversary $\mathcal{A}$ running in time at most t, making at most q queries to the signature oracle and winning in the above experiment with probability at least ε.

2.2 Cover-free Families

For our construction of a fault-tolerant aggregate signature scheme in Section 3 we need a d-cover-free family, which allows us to detect up to d invalid individual signatures in our aggregate signature.

Definition 1. *A* d-cover-free family $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ *(denoted by* d*-CFF) consists of a set* $\mathcal{S}$ *of* m *elements and a set* $\mathcal{B}$ *of* n *subsets of* $\mathcal{S}$*, where* $d < m < n$*, such that: For any* d *subsets* $B_{i_1}, \ldots, B_{i_d} \in \mathcal{B}$ *and all distinct* $B \in \mathcal{B} \setminus \{B_{i_1}, \ldots, B_{i_d}\}$*, it holds that*

$$|B \setminus \bigcup_{k=1}^{d} B_{i_k}| \geq 1.$$

So, it is not possible to cover a single subset with at most d different subsets. To get a better representation of a d-CFF and to simplify the handling of it, we will use a matrix in the following way:

Definition 2. *For a* d*-CFF* $\mathcal{F} = (\mathcal{S}, \mathcal{B})$*, where the elements of* $\mathcal{S}$ *and* $\mathcal{B}$ *have a well-defined order, such that we can write* $\mathcal{S} = \{s_1, \ldots, s_m\}$*,* $\mathcal{B} = \{B_1, \ldots, B_n\}$*, we define its* incidence matrix $\mathcal{M}$ *as follows:*

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

The i-th row *of* $\mathcal{M}$ *is denoted by* $\mathcal{M}_i \in \{0, 1\}^n$*, for* $i \in [m]$.

So, $s_i \in \mathcal{S}$ corresponds to row i and $B_j \in \mathcal{B}$ corresponds to column j, i.e. $\mathcal{M}$ has m rows and n columns.

3 Fault-Tolerant Aggregate Signatures

Claims and Claim Sequences. As a notational convenience, we introduce the concept of claims. A *claim* c is simply a pair (pk, m) of a public key and a message, conveying the meaning that the owner of pk has authenticated the message m. In this sense, a signature σ for m that is valid under pk is a proof for the claim c. This definition allows for a more compact representation of our algorithms.

The signature scheme we introduce in Section 4 critically requires an order among the claims. While the actual order is arbitrary, it must be maintained by the aggregation and verification algorithms. We therefore define the fault-tolerant signature schemes based on sequences of claims, instead of multisets.

More precisely, when an individual signature τ' for a claim c is first aggregated into an aggregate signature τ, one must assign a unique "position" j to c. If one wishes to verify τ, one must call Verify with a sequence of claims C that has c at its j-th position, i.e. $C[j] = c$. Therefore, two aggregate signatures τ_1, τ_2 for two sequences of claims C_1, C_2 can not be aggregated if $C_1[j] \neq C_2[j]$ for some j.

Thus, our scheme does not support *fully* flexible, arbitrary aggregation. However, if the signers agree in advance on the positions j of their claims, they can aggregate all their signatures into a single combined signature τ. This prerequisite can easily be fulfilled in many applications. In wireless sensor networks for example, one only has to configure each sensor to use a different position j. Moreover, it is always possible to use our scheme as a sequential aggregate signature scheme, since the position j of a claim needs only be determined when it is first aggregated. Our scheme is therefore suitable for all applications where sequential aggregate signatures are sufficient, too, such as secure logging [MT09].

For the general aggregation setting, we will have to deal with "incomplete" claim sequences, i.e. if a claim sequence does not yet contain a claim at position j. We therefore assume the existence of a *claim placeholder* $\perp$ that may be contained in claim sequences. When aggregating the signatures of two such incomplete claim sequences C_1, C_2, the claim sequences will be *merged*, meaning that claim placeholders in C_1 are replaced by actual claims from C_2, for each position j where $C_1[j] = \perp$ and $C_2[j] \neq \perp$, and vice versa. (This merging operation replaces the multiset union used by common aggregate signature schemes.)

For technical reasons, we also require that there is no position where C_1 and C_2 both contain a claim, even if the claims are identical. As a consequence, if a signature τ is aggregated into two different aggregate signatures τ_1, τ_2 using the same position j, τ_1 and τ_2 can not be aggregated later. Note, however, that this does not preclude the possibility to aggregate τ into τ_1 and τ_2 at different positions.

We now move to the formal definition. A *claim sequence* is a tuple of claims and claim placeholders $\perp$. The multiset of elements of a claim sequence C excluding $\perp$ is denoted by $\mathsf{elem}(C)$. Two claim sequences C_1, C_2 are *mergeable* if for all $i \in [\min(|C_1|, |C_2|)]$ it holds that $C_1[i] = \perp$ or $C_2[i] = \perp$ or $C_1[i] = C_2[i]$. C_1, C_2 are called *exclusively mergeable*, if for all such i it holds that $C_1[i] = \perp$ or $C_2[i] = \perp$. (In particular, two exclusively mergeable sequences are mergeable.)

For example, for distinct claims c_1, c_2, c_3, define $C_1 = (\bot, c_2, c_3)$, $C_2 = (c_1, \bot, \bot)$, $C_2' = (c_1, c_2, \bot)$, $C_2'' = (c_1, c_3, c_2)$. Then, C_1, C_2 are exclusively mergeable, C_1, C_2' are mergeable, but not exclusively mergeable, and C_1, C_2'' are not mergeable.

Let C_1 and C_2 be two mergeable claim sequences of length k and l, respectively. Without loss of generality, assume $k \geq l$. Then the *merged claim sequence* $C_1 \sqcup C_2$ is $(c_1, \ldots, c_k)$, where

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

The *empty signature* λ is a signature valid for exactly the claim sequences containing only $\bot$ and the empty claim sequence.

Subsequences. Let $C = (c_1, \ldots, c_n)$ be a tuple and $b \in \{0,1\}^n$ be a bit sequence specifying a selection of indices. Then $C[b]$ is the subsequence of C containing exactly the elements c_j where $b[j] = 1$, replacing all other claims by $\bot$. In particular, if $\mathcal{M}$ is an incidence matrix of a cover-free family, then $C[\mathcal{M}_i]$ is the subsequence containing all c_j where $\mathcal{M}[i,j] = 1$ and $\bot$ at all other positions.

Syntax of Fault-Tolerant Signature Schemes. We are now ready to define fault-tolerant aggregate signature schemes. The intuitive difference of such a scheme to an ordinary aggregate signature scheme is that its verification algorithm does not only output a boolean value 1 or 0 that determines if either all claims are valid or at least one claim is invalid, but it gives (some) information on which claims in C are valid. In particular, it outputs the set of valid claims. If the signature contains more errors than the scheme can cope with, Verify may output just a subset of the valid claims. Other claims may be clearly false or just not certainly true. (The verification algorithm ought to be conservative and reject a claim in case of uncertainty.)

The aggregation algorithm is called with two claim *sequences*, hence, before aggregating, a single claim c must be converted to a claim sequence $C = (\bot, \ldots, \bot, c)$ by assigning a position to c.

Definition 3. *An* aggregate signature scheme with list verification[1] *is a tuple of four PPT algorithms* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Agg}, \mathsf{Verify})$*, where*

- $\mathsf{KeyGen}(1^\kappa)$ *creates a key pair* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$ *creates a signature for message* m *under secret key* sk.
- $\mathsf{Agg}(C_1, C_2, \tau_1, \tau_2)$ *takes as input two exclusively mergeable claim sequences* C_1 *and* C_2 *and corresponding signatures* τ_1 *and* τ_2 *and creates an aggregate signature* τ*, certifying the validity of the claim sequence* $C_1 \sqcup C_2$.

[1] The name "list verification" is chosen to indicate the changes in syntax, in particular that the verification algorithm outputs a multiset (list) instead of just 1 or 0.

– $\mathsf{Verify}(C, \tau)$ *takes as input a claim sequence* C *and an aggregate signature* τ *for* C *and outputs a multiset of claims* $C_{\text{valid}} \subseteq \mathsf{elem}(C)$ *specifying the valid claims in* τ*. Note that this may be a proper subset of* $\mathsf{elem}(C)$*, or even empty, if none of the claims can be derived from* τ *(for certain). Again, here,* C *may contain* $\perp$ *as a claim placeholder.*

Σ *is required to be* correct *as defined in the following paragraphs.*

Regular Signatures. Informally, a signature is regular if it is created by running the algorithms of Σ. More formally, let C be a claim sequence and τ be a signature. We recursively define what it means for τ to be *regular for* C:

– If $(\mathsf{pk}, \mathsf{sk})$ is in the image of $\mathsf{KeyGen}(1^\kappa)$ and $C = ((\mathsf{pk}, m))$ for a message m, and if τ is in the image of $\mathsf{Sign}(\mathsf{sk}, m)$, then τ is said to be regular for C and for any claim sequence obtained by prepending any number of $\perp$ symbols to C.
– If τ_1 is regular for a claim sequence C_1, τ_2 is regular for another claim sequence C_2, and C_1, C_2 are exclusively mergeable, then τ is regular for $C_1 \sqcup C_2$ if τ is in the image of $\mathsf{Agg}(C_1, C_2, \tau_1, \tau_2)$.
– The empty signature λ is regular for the claim sequences containing only $\perp$ and the empty claim sequence $()$.

If a signature τ is not regular for a claim sequence C, it is called *irregular* for C.

Fault Tolerance. Let $M = \{(c_1, \tau_1), \ldots, (c_n, \tau_n)\}$ be a multiset of claim and signature pairs, which is partitioned into two multisets M_{irreg} and M_{reg}, containing the pairs for which τ_i is irregular for $C = (c_i)$ and regular for C, respectively.[2] Then the multiset M *contains* d *errors*, if $|M_{\text{irreg}}|$ is d. An aggregate signature scheme Σ with list verification is *tolerant against* d *errors*, if for any such multiset M containing at most d errors, for any signature τ that was aggregated from the signatures in M (in arbitrary order) and the corresponding claim sequence C, which may additionally contain any number of claim placeholders $\perp$, we have

$$R \subseteq \Sigma.\mathsf{Verify}(C, \tau),$$

where R is the multiset of all the claims (i.e. the first component of the pairs) in M_{reg}. In other words, Verify outputs at least all claims of regular signatures.[3]

A *d-fault-tolerant aggregate signature scheme* is an aggregate signature scheme with list verification that is tolerant against d errors. A *fault-tolerant aggregate signature scheme* is a scheme that is d-fault-tolerant for some $d > 0$.

[2] While there may be schemes with valid signatures which are not regularly generated, like in the usual correctness properties, our guarantees do only concern regular signatures.

[3] Intuitively, one would expect $R = \Sigma.\mathsf{Verify}(C, \tau)$. However, this is not achievable in general, as the aggregation of multiple irregular signatures may contain a new valid claim c_i corresponding to an irregular signature σ_i. This does not contradict security, as crafting such irregular signatures may be hard if one does not know σ_i.

Correctness. Observe that 0-fault-tolerance means that if M contains only regularly created signatures, then Verify must output all claims in M (or C, respectively). This is analogous to the common definition of correctness for aggregate signature schemes. We therefore call an aggregate signature scheme with list verification *correct*, if it is tolerant against 0 errors.

Errors during Aggregation. Our definitions above assume that aggregation is always done correctly. This is a necessary assumption, since it is impossible to give guarantees for arbitrary errors that happen during aggregation. Consider for example a faulty aggregation algorithm that ignores its input and just outputs a random string. It is an interesting open question to find a fault-tolerant signature scheme that can tolerate certain types of aggregation errors, too.

Compression Ratio. Denote by $\mathsf{size}(\sigma)$ the size of a signature σ. Let C be a claim sequence of length n, and σ^* an aggregate signature of maximum size[4] which is regular for C. We say that an aggregate signature scheme has *compression ratio* $\rho(n)$ iff

$$\frac{n}{\mathsf{size}(\sigma^*)} \in \Theta(\rho(n)).$$

Note that if $\mathsf{size}(\sigma^*)$ is upper bounded by a constant, then the compression ratio is $\rho(n) = n$, which is optimal for common aggregate signature schemes. As argued in the introduction this is not possible for fault-tolerant aggregate signatures, cf. Appendix A.

Security Experiment. The security experiment for aggregate signatures with list verification, which is a direct adaption of the standard security experiment of [Bon+03], consists of three phases:

- *Setup Phase.* The challenger generates a pair of keys $(\mathsf{pk}, \mathsf{sk}) := \mathsf{KeyGen}(1^\kappa)$ and gives the public key pk to the adversary.
- *Query Phase.* The adversary $\mathcal{A}$ may (adaptively) issue signature queries m_i to the challenger, who responds with $\tau_i := \mathsf{Sign}(\mathsf{sk}, m_i)$.
- *Forgery Phase.* Finally, $\mathcal{A}$ outputs a claim sequence C^* and a signature τ^*.

The adversary *wins* the experiment iff there is a message m^* such that $c^* = (\mathsf{pk}, m^*) \in \mathsf{Verify}(C^*, \tau^*)$, and m^* has never been submitted to the signature oracle.

Definition 4. *An aggregate signature scheme with list verification is* (t, q, ε)-secure *if there is no adversary* $\mathcal{A}$ *running in time at most* t*, making at most* q *queries to the signature oracle and winning in the above experiment with probability at least* ε*.*

[4] The size of an aggregated signature might depend on the aggregation order.

4 Generic Construction of Fault-Tolerant Aggregate Signatures

In this section, we present our generic construction of fault-tolerant aggregate signature schemes. It is based on an arbitrary aggregate signature scheme Σ, which is used as a black box, and a cover-free family. Our scheme inherits its security from Σ, and can tolerate d faults if it uses a d-cover-free family.

Our Construction. In the following we describe a generic construction of our fault-tolerant aggregate signature scheme. For this let Σ be an ordinary aggregate signature scheme. Moreover, let $\mathcal{M}$ be the incidence matrix of a d-cover-free family $\mathcal{F} = (\mathcal{S}, \mathcal{B})$, as defined in Section 2.2. For the sake of presentation, we first show our bounded construction. In this version of our construction, the maximum number of signatures that can be aggregated is $\mathrm{cols}(\mathcal{M})$. We discuss in Section 4.1 how to remove this restriction.

In our scheme, signatures for just one claim are simply signatures of the underlying scheme Σ, whereas aggregate signatures are short vectors of signatures of Σ. We identify each element of the universe $\mathcal{S}$ with a position in this vector, and each subset $B \in \mathcal{B}$ with an individual signature of the underlying scheme Σ.

Here, we require also that the underlying scheme Σ supports claim sequences and claim placeholders as an input to Agg and Verify, contrary to just multisets, as in the definition of Section 2.1. Moreover, we assume that Σ supports the empty signature λ as an input to Agg and Verify. However, these are not essential restrictions, as for instance any normal aggregate scheme may be easily adapted to a scheme of the modified syntax, by ignoring any order and claim placeholders, i.e. applying $\mathsf{elem}(\cdot)$ on the claim sequences before they are passed to the Agg and Verify algorithm.

- $\mathsf{KeyGen}(1^\kappa)$ creates a key pair $(\mathsf{pk}, \mathsf{sk})$ by using the KeyGen algorithm of Σ.
- $\mathsf{Sign}(\mathsf{sk}, m)$ takes as input a secret key sk and a message m and outputs the signature as given by $\Sigma.\mathsf{Sign}(\mathsf{sk}, m)$.
- $\mathsf{Agg}(C_1, C_2, \tau_1, \tau_2)$ takes as input two exclusively mergeable claim sequences C_1 and C_2 and corresponding signatures τ_1 and τ_2. It proceeds as follows:
  1. If one or both of the claim sequences C_k ($k \in \{1, 2\}$) contains only one (proper) claim c, i.e. τ_k is an individual signature, then σ_k is initialized as τ_k, the corresponding signature given to Agg. Then τ_k is expanded to a vector, by setting

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

     where j is the index of c in the claim sequence.
  2. Then the signatures τ_1, τ_2, which are both vectors now, are aggregated component-wise, i.e.

$$\tau[i] = \Sigma.\mathsf{Agg}(C_1[\mathcal{M}_i], C_2[\mathcal{M}_i], \tau_1[i], \tau_2[i]).$$

Finally, Agg outputs τ.

– $\mathsf{Verify}(C, \tau)$ takes as input a claim sequence C and an aggregate signature τ for C. For each component $\tau[i]$ of τ it computes $b_i \coloneqq \Sigma.\mathsf{Verify}(C[\mathcal{M}_i], \tau[i])$ and outputs the multiset of valid claims

$$C_{\text{valid}} \coloneqq \mathsf{elem}\left(\bigsqcup_{i \in [k], b_i = 1} C[\mathcal{M}_i]\right). \tag{1}$$

We now prove the security of our scheme.

Theorem 1. *If Σ is a (t, q, ε)-secure aggregate signature scheme, then the scheme defined above is a (t', q, ε)-secure aggregate signature scheme with list verification, where t' is approximately the same as t.*

Proof. Let Σ' be the scheme described above. The following argument is rather direct. Assume that $\mathcal{A}$ is an adversary breaking the (t', q, ε)-security of Σ'. We construct an attacker $\mathcal{B}$ breaking the (t, q, ε)-security of Σ.

$\mathcal{B}$ simulates $\mathcal{A}$ as follows. In the setup phase $\mathcal{B}$ starts executing $\mathcal{A}$ and passes its own input pk on to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a signature query for a message m, $\mathcal{B}$ obtains the signature σ by forwarding m to the challenger. $\mathcal{B}$ then passes σ to $\mathcal{A}$ and continues the simulation. When $\mathcal{A}$ outputs a claim sequence C^* and a signature τ^*, $\mathcal{B}$ checks if there is a claim $c^* = (\mathsf{pk}, m^*)$ in C^*, such that m^* was never queried by $\mathcal{A}$.

If this is not the case, then $\mathcal{B}$ outputs $\perp$ and terminates. Otherwise, by definition of $\Sigma'.\mathsf{Verify}$, there must be an index i such that

$$\Sigma.\mathsf{Verify}(C^*[\mathcal{M}_i], \tau^*[i]) = 1 \tag{2}$$

and $m^* \in \mathsf{elem}(C^*[\mathcal{M}_i])$. $\mathcal{B}$ outputs $C^*[\mathcal{M}_i]$ and $\tau^*[i]$. This is a valid signature, because of (2).

Note that $\mathcal{B}$'s queries are exactly the same as $\mathcal{A}$'s. Therefore, if $\mathcal{A}$ did not query m^*, then neither did $\mathcal{B}$. Thus, $\mathcal{B}$ wins exactly iff $\mathcal{A}$ wins, and therefore $\mathcal{B}$ also has success probability ε. We also see that $\mathcal{B}$ makes at most q queries. Finally, it is easy to verify that the running time of $\mathcal{B}$ is approximately the same as the running time of $\mathcal{A}$. □

We now turn to proving the fault-tolerance of our scheme.

Theorem 2. *Let Σ be the aggregate signature scheme with list verification defined above. If Σ is based on a d-CFF, then it is tolerant against d errors, and in particular, it is correct.*

Proof. Let $M = \{(c_1, \tau_1), \ldots, (c_m, \tau_m)\}$ be a multiset of claim and signature pairs, which is partitioned into two multisets M_{irreg} and M_{reg}, containing the pairs for which τ_i is irregular for $C_i = (c_i)$ or regular for C_i, respectively. Let M contain at most d errors, i.e., $|M_{\text{irreg}}| \leq d$. Moreover, let τ be a signature that was aggregated from the signatures in M (in arbitrary order) and C the

corresponding claim sequence. To simplify the proof, we assume without loss of generality that $C = (c_1, \ldots, c_n)$, i.e., the order in the claim sequence is the same as in the indexing of the signatures in M and it does not include any claim placeholders $\perp$. Finally, let $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ be a d-cover-free family used by the scheme above, where $\mathcal{S} = \{s_1, \ldots, s_m\}$ and $\mathcal{B} = \{B_1, \ldots, B_n\}$.

We need to show that $R \subseteq \Sigma.\mathsf{Verify}(C, \tau) =: V$, where R is the multiset of all the claims in M_{reg}. Recall that $\text{rows}(\mathcal{M})$ and $\text{cols}(\mathcal{M})$ denote the number of rows and columns of $\mathcal{M}$, respectively. Let $b_i \coloneqq \Sigma'.\mathsf{Verify}(C[\mathcal{M}_i], \tau[i])$ for all $i \in [\text{rows}(\mathcal{M})]$.

Assume for a contradiction that there is a claim c^* that is contained strictly more often in R than in V. Then there exists an index j^* such that $C[j^*] = c^*$ and $b_i = 0$ for all $i \in [\text{rows}(\mathcal{M})]$ with $\mathcal{M}[i, j^*] = 1$.

In the following, let $I := \{i \in [\text{rows}(\mathcal{M})] \colon \mathcal{M}[i, j^*] = 1\}$ be the set of these indices I, and observe that these are the indices of all rows where the signature for c^* is aggregated into $\tau[i]$.

We now try to obtain a contradiction by showing that the set B_{j^*}, which corresponds to the column j^* of $\mathcal{M}$, is covered by the sets B_k, corresponding to the columns of the claims with irregular signatures.

For each $i \in I$, since $b_i = 0$ and using the correctness of Σ', there must be some $k \in [n]$ such that $(c_k, \sigma_k) \in M_{\text{irreg}}$ and $\mathcal{M}[i, k] = 1$. Since M contains at most d errors, there are at most d such indices k in total. Let K denote the set of these indices. Note that $j^* \notin K$, since $(c^*, \sigma^*) \in M_{\text{reg}}$, according to our assumption.

We now have established that for each $i \in I$, there exists a k with $(c_k, \sigma_k) \in M_{\text{irreg}}$ and $\mathcal{M}[i, k] = 1$, and $|K| \leq d$. Recall that by definition of the incidence matrix $\mathcal{M}$, we have for all $i \in [\text{rows}(\mathcal{M})]$ and $j \in [\text{cols}(\mathcal{M})]$:

$$\mathcal{M}[i, j] = 1 \quad \iff \quad s_i \in B_j.$$

Restating the fact from the above paragraph using this equivalence yields that for all i with $s_i \in B_{j^*}$, there exists a k with $s_i \in B_k$, where there are at most d distinct indices $k \in K$ in total. But this means that $B_{j^*} \subseteq \bigcup_{k \in K} B_k$, where the union is over at most d different subsets B_k of $\mathcal{S}$. This is a direct contradiction to the d-cover-freeness of $\mathcal{F}$, so our assumption must be false, and we must therefore have $R \subseteq V$. □

Compression Ratio. Let C be a claim sequence of length $n \in \mathbb{N}$, and τ be an aggregate signature regular for C. We assume in the following that the length of all signatures of the underlying scheme Σ' is bounded by a constant s and is at least 1. Then the compression ratio of our scheme is $\rho(n) = \frac{n}{\text{rows}(\mathcal{M})}$, since

$$\frac{n}{\mathsf{size}(\tau)} \leq \frac{n}{\text{rows}(\mathcal{M}) \cdot s} \in \mathcal{O}(\rho(n)) \quad \text{and} \quad \frac{n}{\mathsf{size}(\tau)} \geq \frac{n}{\text{rows}(\mathcal{M})} \in \Omega(\rho(n)). \quad (3)$$

Clearly, the compression ratio $\rho(n)$ of our scheme is less than 1 if $n < \text{rows}(\mathcal{M})$, and the resulting aggregate signature is larger than the sum of the individual signature sizes when only few signatures have been aggregated so far. Our scheme

can be easily adapted to fix this behavior, by simply storing all individual signatures instead of immediately aggregating them, until $n = \text{rows}(\mathcal{M})$. When the $n+1$-st signature is added, the individual signatures are aggregated using the aggregation algorithm defined above. When further signatures are added, the size of the aggregate signature remains bounded by $\text{rows}(\mathcal{M}) \cdot s$.

4.1 Achieving Unbounded Aggregation

In order to achieve unbounded aggregation, we do not need just one cover-free family, but a sequence of cover-free families increasing in size, such that we can jump to the next larger one, as soon as we exceed the capacity for the number of aggregatable signatures. This sequence needs to exhibit a monotonicity property, in order to work with our scheme, which we define next.

Definition 5. *We consider a family $(\mathcal{M}^{(l)})_l$ of incidence matrices of corresponding d-cover-free families $(\mathcal{F}_l)_l := (\mathcal{S}_l, \mathcal{B}_l)_l$, where* $\text{rows}(l)$ *denotes the number of rows and* $\text{cols}(l)$ *denotes the number of columns of $\mathcal{M}^{(l)}$. $(\mathcal{M}^{(l)})_l$ is a* monotone family *of incidence matrices of $(\mathcal{F}_l)_l$, if $\mathcal{S}_l \subseteq \mathcal{S}_{l+1}$, $\mathcal{B}_l \subseteq \mathcal{B}_{l+1}$, $l \geq 1$, s.t. $\mathcal{S}_{l+1} = \{s_1, \ldots, s_{\text{rows}(l)}, s_{\text{rows}(l)+1}, \ldots, s_{\text{rows}(l+1)}\}$ and $\mathcal{B}_{l+1} = \{B_1, \ldots, B_{\text{cols}(l)}, B_{cols(l)+1}, \ldots, B_{\text{cols}(l+1)}\}$, where $\mathcal{S}_l = \{s_1, \ldots, s_{\text{rows}(l)}\}$ and $\mathcal{B}_l = \{B_1, \ldots, B_{\text{cols}(l)}\}$.*

Note that Definition 5 implies that

$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & \mathbf{A} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}$$

where for $i = 1, \ldots, \text{rows}(l), j = \text{cols}(l)+1, \ldots, \text{cols}(l+1)$

$$\mathbf{A}[i,j] = \begin{cases} 1, & \text{if } s_i \in B_j, \\ 0, & \text{otherwise} \end{cases}$$

and for $i = \text{rows}(l), \ldots, \text{rows}(l+1), j = \text{cols}(l)+1, \ldots, \text{cols}(l+1)$

$$\mathbf{B}[i,j] = \begin{cases} 1, & \text{if } s_i \in B_j, \\ 0, & \text{otherwise.} \end{cases}$$

So, each $\mathcal{M}^{(l)}$ contains all previous $\mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(l-1)}$.

Now, we are able to achieve unbounded aggregation, i.e. our construction is able to aggregate an arbitrary number of signatures, by replacing the fixed incidence matrix $\mathcal{M}$ of a d-CFF in our construction with a monotone family of incidence matrices $(\mathcal{M}^{(l)})_l$. For this, a run of our aggregation algorithm Agg on inputs C_1, C_2, τ_1, τ_2 first has to determine the smallest l, such that $\text{cols}(l) \geq \max(|C_1|, |C_2|)$ and then proceeds with the corresponding incidence matrix $\mathcal{M}^{(l)}$. Analogously, our verification algorithm Verify on inputs C, τ first determines the smallest l such that $\text{cols}(l) \geq |C|$.

Compression Ratio. The compression ratio of our unbounded scheme is $\rho(n) = n/\text{rows}(l)$, where l is the minimum index such that $\text{cols}(l) \geq n$.

4.2 Additional Features of Our Construction

Selective Verification. Let τ be a regular signature with corresponding claim sequence $C = (c_1, \ldots, c_n)$. Assume we would want to know whether a signature for a specific claim c^* was aggregated into τ, but we want to avoid verifying all the claims in C to save verification time, especially if C is large. It is a unique feature of our fault-tolerant aggregate signature scheme that there is an additional algorithm $\mathsf{SelectiveVerify}(C, \tau, c^*)$ that outputs the number of occurrences of c^* in C that have a valid signature in τ, i.e., the number of occurrences of c^* in $\mathsf{Verify}(C, \tau)$, while being faster than actually calling $\mathsf{Verify}(C, \tau)$.

Let Σ be the aggregate signature scheme with list verification defined above and Σ' be the underlying aggregate signature scheme. Then SelectiveVerify works as follows. First, it determines the set J of indices j where c^* occurs in C, i.e. $c_j = c^*$. Then it determines the set $I := \{i \in \text{rows}(\mathcal{M}) \colon \mathcal{M}[i, j] = 1 \text{ for a } j \in J\}$, i.e. the set of indices of all rows where an individual signature for c^* should have been aggregated. Then, it initializes $M := ()$ and iterates over all $i \in I$, checking if $b_i := \Sigma'.\mathsf{Verify}(C[\mathcal{M}_i], \tau[i]) = 1$. If this is the case for an i, it sets

$$M := M \sqcup C[\mathcal{M}_i].$$

As soon as M contains $|J|$ occurrences of c^*, SelectiveVerify skips all remaining $i \in I$. After the loop is done, SelectiveVerify outputs the number of occurrences of c^* in M.

Since $\Sigma.\mathsf{Verify}$ returns all claims that are contained in a subsequence $C[\mathcal{M}_i]$ with $b_i = 1$, the output of SelectiveVerify is exactly the number of occurrences of c^* in $\Sigma.\mathsf{Verify}$. SelectiveVerify therefore inherits the fault-tolerance and security properties already proven for $\Sigma.\mathsf{Verify}$.

In the best case, SelectiveVerify requires only one call to the underlying verification algorithm $\Sigma'.\mathsf{Verify}$. In the worst case, it still only requires $|I| \leq \sum_{j \in J} |B_j|$ calls to $\Sigma'.\mathsf{Verify}$, where B_j is the set from the cover-free family corresponding to column j.

Going a little further, it is even possible to create a "subsignature" for c^* that allows everyone to check that c^* has a valid signature without requiring the complete claim sequence C and the complete signature τ: It is sufficient to give $C' := \bigsqcup_{i \in I} C[\mathcal{M}_i]$ and the signatures $\tau[i]$ for $i \in I$ to the verifier.

5 A Concrete Instantiation of Our Scheme

In this section, we consider a concrete construction of a d-CFF which can be used to instantiate our generic d-fault-tolerant aggregate signature scheme. There are several d-CFF constructions in the literature, for instance, constructions based on concatenated codes [LVY01; DMR00b; DMR00a], polynomials,

algebraic-geometric Goppa codes as well as randomized constructions [KRS99]. The following theorem gives a lower bound for the number of rows of the incidence matrix in terms of parameter d and the number of columns. Proofs can be found in [DR82; Für96; Rus94].

Theorem 3. *For a d-CFF $\mathcal{F} = (\mathcal{S}, \mathcal{B})$, where $|\mathcal{S}| = m$, $|\mathcal{B}| = n$, it holds*

$$m \geq c \frac{d^2}{\log d} \log n$$

for some constant $c \in (0, 1)$.

In the following construction we use for concreteness only a single incidence matrix, but the next lemma by [LVW06] shows a generic construction to get a monotone family of incidence matrices.

Lemma 1. *If $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ and $\mathcal{F}' = (\mathcal{S}', \mathcal{B}')$ are d-CFFs, then there exist a d-CFF $\mathcal{F}^* = (\mathcal{S}^*, \mathcal{B}^*)$ with $|\mathcal{S}^*| = |\mathcal{S}| + |\mathcal{S}'|$ and $|\mathcal{B}^*| = |\mathcal{B}| + |\mathcal{B}'|$.*

Proof. Suppose $\mathcal{M}$ and $\mathcal{M}'$ are the incidence matrices of d-CFFs $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ and $\mathcal{F}' = (\mathcal{S}', \mathcal{B}')$, respectively. Then

$$\mathcal{M}^* = \begin{pmatrix} \mathcal{M} & \mathbf{0} \\ \mathbf{0} & \mathcal{M}' \end{pmatrix}$$

is an incidence matrix for a d-CFF $\mathcal{F}^* = (\mathcal{S}^*, \mathcal{B}^*)$ with $|\mathcal{S}^*| = |\mathcal{S}| + |\mathcal{S}'|$ and $|\mathcal{B}^*| = |\mathcal{B}| + |\mathcal{B}'|$. □

For our approach we could use a deterministic construction of a d-CFF based on polynomials like [KRS99] did in the following way and for which we propose a generalization to the multivariate case in Appendix B.

For our d-CFF $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ let $\mathbb{F}_q = \{x_1, \ldots, x_q\}$ be a finite field and

$$\mathcal{S} := \mathbb{F}_q^2 = \{(x_i, x_j) \colon i, j = 1, \ldots, q\}, \text{ with } |\mathcal{S}| = q^2.$$

For ease of presentation, we assume that q is a prime (as opposed to a prime power), so we may write $\mathbb{F}_q = \{0, \ldots, q-1\}$. We consider the set of all univariate polynomials $f \in \mathbb{F}_q[X]$ of degree at most k, denoted by $\mathbb{F}_q[X]_{\leq k}$. So,

$$\mathbb{F}_q[X]_{\leq k} := \left\{a_k X^k + \cdots + a_1 X + a_0 \colon a_i \in \mathbb{F}_q,\ i = 0, \ldots, k\right\}.$$

We have $|\mathbb{F}_q[X]_{\leq k}| = q^{k+1}$. Now, for every $f \in \mathbb{F}_q[X]_{\leq k}$, we consider the subsets

$$B_f = \{(x_1, f(x_1)), \ldots, (x_q, f(x_q))\} \subset \mathcal{S} \text{ of size } q,$$

consisting of all tuples $(x, y) \in \mathcal{S}$ which lie on the graph of $f \in \mathbb{F}_q[X]_{\leq k}$, i.e. for which $f(x) = y$. From this we obtain

$$\mathcal{B} := \{B_f \colon f \in \mathbb{F}_q[X]_{\leq k}\}, \text{ which is of size } q^{k+1}.$$

For any distinct $B_f, B_{f_1}, \ldots, B_{f_d} \in \mathcal{B}$ it holds that

$$|B_f \cap B_{f_i}| \leq k,$$

since the degree of each polynomial $g_i := f - f_i$ is at most k and hence they have at most k zeros. Thus, we have

$$\left| B_f \setminus \bigcup_{i=1}^{d} B_{f_i} \right| \geq q - d \cdot k$$

To achieve a d-CFF with this construction, $q \geq d \cdot k + 1$ must be fulfilled.

Now, we consider the incidence matrix $\mathcal{M}$ of a d-CFF, which consists of $|\mathcal{S}|$ rows and $|\mathcal{B}|$ columns. Each row corresponds to an element of $\mathcal{S}$ and each column to an element of $\mathcal{B}$. In the construction above each row corresponds to a tuple $(x, y) \in \mathbb{F}_q^2$, where the order is $(0,0), (0,1), \ldots, (q-1, q-1)$. In the following, let (x_i, y_i) denote the corresponding tuple for row i, $i = 0, \ldots, q^2 - 1$. We start counting from 0 for simplicity, hence,

$$\begin{array}{lcl}
(x_0, y_0) = (0,0), & \ldots, & (x_{q-1}, y_{q-1}) = (0, q-1), \\
(x_q, y_q) = (1,0), & \ldots, & (x_{2q-1}, y_{2q-1}) = (1, q-1), \\
 & \ddots & \\
(x_{q^2-q}, y_{q^2-q}) = (q-1, 0), & \ldots, & (x_{q^2-1}, y_{q^2-1}) = (q-1, q-1).
\end{array}$$

Each column of the incidence matrix $\mathcal{M}$ corresponds to a polynomial of degree at most k, where we decide to start with constant polynomials and end with polynomials of degree k, i.e. $f_0 := 0$, $f_1 := 1$, $f_2 := 2$, …, $f_q := X$, $f_{q+1} := X + 1$, $f_{q+2} := X + 2$, …, $f_{2q} := 2X$, $f_{2q+1} := 2X + 1$, …, $f_{q^{k+1}-1} := (q-1)X^k + (q-1)X^{k-1} + \cdots + (q-1)X + q - 1$.

By f_j we will denote the corresponding polynomial for column j, for $j = 0, \ldots, q^{k+1} - 1$, again starting from 0. Now, the incidence matrix is built as

$$\mathcal{M}[i,j] = \begin{cases} 1, & \text{if } f_j(x_i) = y_i, \\ 0, & \text{otherwise.} \end{cases}$$

Example. For $q = 5$, $d = 2$ and $k = 2$ we have a 2-CFF with

$$\mathcal{S} = \{(0,0), (0,1), \ldots, (4,3), (4,4)\}, \ |\mathcal{S}| = 25.$$

We have

$$\mathcal{B} = \{B_{f_0}, \ldots, B_{f_{124}}\},$$

since $|\mathbb{F}_5[X]_{\leq 2}| = 5^3 = 125$, where

$$B_{f_j} = \{(0, f_j(0)), (1, f_j(1)), \ldots, (4, f_j(4))\}, \quad |B_{f_j}| = 5, \quad j = 0, \ldots, 5^3 - 1$$

and $f_0 := 0$, $f_1 := 1$, …, $f_{124} := 4X^2 + 4X + 4$. Thus, we obtain our incidence matrix $\mathcal{M}$:

$$\begin{array}{c} \\ (0,0) \\ (0,1) \\ \vdots \\ (4,4) \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & \dots & X & \dots & 4X^2+4X+4 \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 1 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & \dots & 1 \end{pmatrix} \end{array} = \mathcal{M}$$

Remark. With this univariate polynomial-based construction of a d-CFF it is very easy to generate our incidence matrix or only some parts of it, which we need for our verification algorithm or if we want to check some information separately.

If, for example, one is interested to verify the validity of only one single claim–signature pair (c_j, σ_j) in an aggregate signature, it is not necessary to generate the whole matrix but only the rows where the related column j has 1-entries. So, you have to know which polynomial corresponds to column j.

For this, we can use the fact, that each positive number $n = 0, \dots, q^{k+1} - 1$ can be written as $a_k \cdot q^k + a_{k-1} \cdot q^{k-1} + \cdots + a_0$, where $a_k, \dots, a_0 \in \{0, \dots, q-1\}$. So, each n corresponds to a $(k+1)$-tuple denoted by $(a_k^{(n)}, \dots, a_0^{(n)})$. For the sake of convenience, we start to count the rows and columns of our matrix by 0, as before. Thus, for column $j = 0, \dots, q^{k+1} - 1$ we assign the polynomial $f_j = a_k^{(j)} X^k + \cdots + a_0^{(j)}$.

Analogously, for each row $i = 0, \dots, q^2 - 1$, we assign the tuple $(b_1^{(i)}, b_0^{(i)}) \in \mathbb{F}_q^2$, where $i = b_1^{(i)} \cdot q + b_0^{(i)}$. Let $I'_j \subset \{0, \dots q^2 - 1\}$ be the subset of all rows i' where $f_j(b_1^{(i')}) = b_0^{(i')}$. So, it suffices to generate only the rows $i' \in I'_j$ to verify the validity of σ_j. To get the 1-entries of these rows, you have to check for each $i' \in I'_j$ which polynomials $f \in \mathbb{F}_q[X]_{\leq k}$ fulfill $f(b_1^{(i')}) = b_0^{(i')}$. For all arbitrary, but fixed values $a_k, \dots a_1 \in \{0, \dots, q-1\}$ compute an appropriate a_0. This results in q^k polynomials, accordingly columns, per row. If the coefficients of the appropriate polynomials are known then we can use them to compute the number of the corresponding columns with 1-entries.

Compression Ratio of Our Bounded Scheme. If our bounded scheme is instantiated with this CFF, and we assume that the length of signatures of the underlying scheme Σ' is bounded by a constant s, then, as shown in (3), the compression ratio is

$$\rho(n) = \frac{n}{\text{rows}(\mathcal{M})} = \frac{n}{|\mathcal{S}|} = \frac{n}{q^2} \ .$$

For $n = |\mathcal{B}|$, we therefore have

$$\rho(n) = \frac{|\mathcal{B}|}{|\mathcal{S}|} = \frac{q^{k+1}}{q^2} \ .$$

Since $q \geq dk + 1$, we have that $|\mathcal{B}|$ grows exponentially in k, whereas $|\mathcal{S}|$ grows only quadratically in k. Hence, $|\mathcal{B}|$ is exponential in $|\mathcal{S}|$, or, stated differently, $|\mathcal{S}|$ is logarithmic in $|\mathcal{B}|$.

Compression Ratio of Our Unbounded Scheme. When our unbounded scheme is instantiated with the monotone family of CFFs obtained by fixing an incidence matrix $\mathcal{M}$ and repeatedly using Lemma 1 on $\mathcal{M}$, then the asymptotic compression ratio is $\rho(n) = 1$, since

$$\frac{n}{\text{rows}(l)} \leq \frac{\text{cols}(l)}{\text{rows}(l)} = \frac{\text{cols}(\mathcal{M})}{\text{rows}(\mathcal{M})} \quad \text{for all } l,$$

which is constant. Therefore, the size of an aggregate signature is linear in the length of the claim sequence.

However, if we assume that all signatures of the underlying scheme Σ' have a size bounded by s, then the concrete size of an aggregate signature is at most

$$\begin{aligned} \text{rows}(l)s &\leq l\,\text{rows}(\mathcal{M})s \\ &\leq (n/\text{cols}(\mathcal{M}) + 1)\,\text{rows}(\mathcal{M})s \\ &= \left(\frac{\text{rows}(\mathcal{M})}{\text{cols}(\mathcal{M})}n + \text{rows}(\mathcal{M})\right)s \ , \end{aligned}$$

since $\text{rows}(l) = l\,\text{rows}(\mathcal{M})$ for the construction of the monotone family of CFFs, and $l = \lceil n/\text{cols}(\mathcal{M})\rceil \leq n/\text{cols}(\mathcal{M}) + 1$.

Therefore we see that the length of the aggregate signature is linear in n, but the factor $\text{rows}(\mathcal{M})/\text{cols}(\mathcal{M})$ can be made arbitrarily small by choosing a proper CFF, such as the one described above.

It is an interesting open problem to construct an unbounded fault-tolerant scheme with better compression ratio, for example by finding a better monotone family of CFFs. A generalization of the above construction to multivariate polynomials, which might be advantageous in some scenarios, is given in Appendix B.

Example Instantiations. Section 5 shows parameters of several cover-free families based on the construction described in this section. For each of the rows given there, there is an instance of our fault-tolerant signature scheme that can compress signatures for up to n claims to a vector of m aggregates, while tolerating up to d errors. (The numbers q and k are needed for the instantiation of the CFF, but do not immediately reflect a property of our fault-tolerant aggregate signature scheme.) Of course, our scheme can be instantiated with different parameters and completely different constructions of CFFs as well.

Acknowledgements. We wish to thank our colleague and friend Julia Hesse for raising the initial research question that led to this work. We would also like to thank the anonymous reviewers for their helpful comments.

References

[AGH10] J. H. Ahn, M. Green, and S. Hohenberger. "Synchronized aggregate signatures: new definitions, constructions and applications". In: *ACM CCS 10*. Ed. by E. Al-Shaer, A. D. Keromytis, and V. Shmatikov. ACM Press, Oct. 2010, pp. 473–484.

Table 1. Example parameters for cover-free families.

q	k	d	$m = \lvert\mathcal{S}\rvert$	$n = \lvert\mathcal{B}\rvert$
5	2	2	25	125
11	2	5	121	1331
17	2	8	289	4913
17	4	4	289	$\approx 1.42 \cdot 10^6$
29	2	14	841	24389
53	2	26	2809	148877
101	2	50	10201	$\approx 1.03 \cdot 10^6$
251	3	83	63001	$\approx 3.97 \cdot 10^9$
1021	2	510	1042441	$\approx 1.06 \cdot 10^9$

[BGR14] K. Brogle, S. Goldberg, and L. Reyzin. "Sequential aggregate signatures with lazy verification from trapdoor permutations". In: *Inf. Comput.* 239 (2014), pp. 356–376. DOI: 10.1016/j.ic.2014.07.001.

[BJ10] A. Bagherzandi and S. Jarecki. "Identity-Based Aggregate and Multi-Signature Schemes Based on RSA". In: *PKC 2010*. Ed. by P. Q. Nguyen and D. Pointcheval. Vol. 6056. LNCS. Springer, May 2010, pp. 480–498.

[BN07] M. Bellare and G. Neven. "Identity-Based Multi-signatures from RSA". In: *CT-RSA 2007*. Ed. by M. Abe. Vol. 4377. LNCS. Springer, Feb. 2007, pp. 145–162.

[Bol+07] A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. "Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing". In: *ACM CCS 07*. Ed. by P. Ning, S. D. C. di Vimercati, and P. F. Syverson. ACM Press, Oct. 2007, pp. 276–285.

[Bon+03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps". In: *EUROCRYPT 2003*. Ed. by E. Biham. Vol. 2656. LNCS. Springer, May 2003, pp. 416–432.

[Cra+07] R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, a. shelat, and V. Vaikuntanathan. "Bounded CCA2-Secure Encryption". In: *ASIACRYPT 2007*. Ed. by K. Kurosawa. Vol. 4833. LNCS. Springer, Dec. 2007, pp. 502–518.

[DMR00a] A. G. Dyachkov, A. J. Macula, and V. V. Rykov. "New applications and results of superimposed code theory arising from the potentialities of molecular biology". In: *Numbers, Information and Complexity*. Springer, 2000, pp. 265–282.

[DMR00b] A. G. Dyachkov, A. J. Macula, and V. V. Rykov. "New constructions of superimposed codes". In: *IEEE Transactions on Information Theory* 46.1 (2000), pp. 284–290. DOI: 10.1109/18.817530.

[Dod+02] Y. Dodis, J. Katz, S. Xu, and M. Yung. "Key-Insulated Public Key Cryptosystems". In: *EUROCRYPT 2002*. Ed. by L. R. Knudsen. Vol. 2332. LNCS. Springer, Apr. 2002, pp. 65–82.

[DR82] A. G. Dyachkov and V. V. Rykov. "Bounds on the length of disjunctive codes". In: *Problemy Peredachi Informatsii* 18.3 (1982), pp. 7–13.

[Für96] Z. Füredi. "On r-Cover-free Families". In: *J. Comb. Theory, Ser. A* 73.1 (1996), pp. 172–173. DOI: 10.1006/jcta.1996.0012.

[Ger+12] M. Gerbush, A. B. Lewko, A. O'Neill, and B. Waters. "Dual Form Signatures: An Approach for Proving Security from Static Assumptions". In: *ASIACRYPT 2012*. Ed. by X. Wang and K. Sako. Vol. 7658. LNCS. Springer, Dec. 2012, pp. 25–42. DOI: 10.1007/978-3-642-34961-4_4.

[GR06] C. Gentry and Z. Ramzan. "Identity-Based Aggregate Signatures". In: *PKC 2006*. Ed. by M. Yung, Y. Dodis, A. Kiayias, and T. Malkin. Vol. 3958. LNCS. Springer, Apr. 2006, pp. 257–273.

[Her06] J. Herranz. "Deterministic Identity-Based Signatures for Partial Aggregation". In: *Comput. J.* 49.3 (2006), pp. 322–330. DOI: 10.1093/comjnl/bxh153.

[HJK11] D. Hofheinz, T. Jager, and E. Kiltz. "Short Signatures from Weaker Assumptions". In: *ASIACRYPT 2011*. Ed. by D. H. Lee and X. Wang. Vol. 7073. LNCS. Springer, Dec. 2011, pp. 647–666.

[HK04] S.-H. Heng and K. Kurosawa. "k-Resilient Identity-Based Encryption in the Standard Model". In: *CT-RSA 2004*. Ed. by T. Okamoto. Vol. 2964. LNCS. Springer, Feb. 2004, pp. 67–80.

[HKW15] S. Hohenberger, V. Koppula, and B. Waters. "Universal Signature Aggregators". In: *EUROCRYPT 2015, Part II*. Ed. by E. Oswald and M. Fischlin. Vol. 9057. LNCS. Springer, 2015, pp. 3–34. DOI: 10.1007/978-3-662-46803-6_1.

[HSW13] S. Hohenberger, A. Sahai, and B. Waters. "Full Domain Hash from (Leveled) Multilinear Maps and Identity-Based Aggregate Signatures". In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Springer, Aug. 2013, pp. 494–512. DOI: 10.1007/978-3-642-40041-4_27.

[KRS99] R. Kumar, S. Rajagopalan, and A. Sahai. "Coding Constructions for Blacklisting Problems without Computational Assumptions". In: *CRYPTO'99*. Ed. by M. J. Wiener. Vol. 1666. LNCS. Springer, Aug. 1999, pp. 609–623.

[KS64] W. H. Kautz and R. C. Singleton. "Nonrandom binary superimposed codes". In: *IEEE Transactions on Information Theory* 10.4 (1964), pp. 363–377. DOI: 10.1109/TIT.1964.1053689.

[LLY15] K. Lee, D. H. Lee, and M. Yung. "Sequential aggregate signatures with short public keys without random oracles". In: *Theor. Comput. Sci.* 579 (2015), pp. 100–125. DOI: 10.1016/j.tcs.2015.02.019.

[Lu+06] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. "Sequential Aggregate Signatures and Multisignatures Without Random Oracles". In: *EUROCRYPT 2006*. Ed. by S. Vaudenay. Vol. 4004. LNCS. Springer, May 2006, pp. 465–485.

[LVW06] P. Li, G. Van Rees, and R. Wei. "Constructions of 2-cover-free families and related separating hash families". In: *Journal of Combinatorial Designs* 14.6 (2006), pp. 423–440.

[LVY01] V. Lebedev, P. Vilenkin, and S. Yekhanin. "Cover-free families and superimposed codes: Constructions, bounds, and applications to cryptography and group testing". In: *In IEEE International Symposium on Information Theory*. 2001.

[Lys+04] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. "Sequential Aggregate Signatures from Trapdoor Permutations". In: *EUROCRYPT 2004*. Ed. by C. Cachin and J. Camenisch. Vol. 3027. LNCS. Springer, May 2004, pp. 74–90.

[MT09] D. Ma and G. Tsudik. "A new approach to secure logging". In: *TOS* 5.1 (2009). DOI: 10.1145/1502777.1502779.

[Nev08] G. Neven. "Efficient Sequential Aggregate Signed Data". In: *EUROCRYPT 2008*. Ed. by N. P. Smart. Vol. 4965. LNCS. Springer, Apr. 2008, pp. 52–69.

[Rus94] M. Ruszinkó. "On the Upper Bound of the Size of the r-Cover-Free Families". In: *J. Comb. Theory, Ser. A* 66.2 (1994), pp. 302–310. DOI: 10.1016/0097-3165(94)90067-1.

[Sch11] D. Schröder. "How to Aggregate the CL Signature Scheme". In: *ESORICS 2011, Proceedings*. Ed. by V. Atluri and C. Díaz. Vol. 6879. LNCS. Springer, 2011, pp. 298–314. DOI: 10.1007/978-3-642-23822-2_17.

[STW97] D. R. Stinson, T. V. Trung, and R. Wei. "Secure Frameproof Codes, Key Distribution Patterns, Group Testing Algorithms and Related Structures". In: *Journal of Statistical Planning and Inference* 86 (1997), pp. 595–617.

[SW99] R. Safavi-Naini and H. Wang. "Multireceiver Authentication Codes: Models, Bounds, Constructions, and Extensions". In: *Information and Computation* 151.1-2 (1999), pp. 148–172. DOI: 10.1006/inco.1998.2769.

[TS06] D. Tonien and R. Safavi-Naini. "An Efficient Single-Key Pirates Tracing Scheme Using Cover-Free Families". In: *ACNS 06*. Ed. by J. Zhou, M. Yung, and F. Bao. Vol. 3989. LNCS. Springer, June 2006, pp. 82–97.

[ZS11] G. M. Zaverucha and D. R. Stinson. "Short one-time signatures". In: *Adv. in Math. of Comm.* 5.3 (2011), pp. 473–488. DOI: 10.3934/amc.2011.5.473.

A Discussion on Signature Size

A typical requirement for aggregate signatures is that the length of an aggregate signature is the same as that of any of the individual signatures [HSW13]. Also, the number of signatures that can be aggregated into a single signature should be unbounded.

We show that these goals are mutually exclusive for an "ideal" fault-tolerant aggregate signature schemes if one wishes to maintain a constant $d \geq 1$.

Proposition 1. *Let $n, d \in \mathbb{N}$, and Σ be a d-fault-tolerant signature scheme. Assume that $\Sigma.\mathsf{Verify}(C, \tau) = R$ for all claim sequences C and corresponding signatures τ constructed from an arbitrary multiset $M = \{(c_1, \tau_1), \ldots, (c_n, \tau_n)\}$ of n claim–signature pairs and containing at most d errors, and where R is the multiset of all claims c_i accompanied by a regular signature τ_i in M. Then we have $|\tau| \geq \Omega(\log_2 n)$ as a function of n, where d is considered constant, and $|\tau|$ is the length of the signature τ in bits.*

Proof. Call an output O of $\Sigma.\mathsf{Verify}$ *in accordance with* C, if O is a sub-multiset of $\mathsf{elem}(C)$ and $|O| \geq |\mathsf{elem}(C)| - d$.

Now, let $n, d, \Sigma, C, \tau, M, R$ be as in the theorem statement. Clearly, since we assumed that Verify always outputs R, $\Sigma.\mathsf{Verify}$'s output must be in accordance with C. For a fixed number of errors $i \in \{0, \ldots, d\}$, there are $\binom{n}{i}$ distinct outputs in accordance with C. Thus, for up to d errors, there are up to

$$s(n) \coloneqq \sum_{i=0}^{d} \binom{n}{i} \geq \binom{n}{d} \geq \frac{(n-d)^d}{d!}$$

distinct outputs in accordance with C.

$\Sigma.\mathsf{Verify}$ must use τ to determine the correct output R among the set of outputs in accordance with C. If the signature size $|\tau|$ is at most $l \in \mathbb{N}$ bits, then $\Sigma.\mathsf{Verify}$ can distinguish at most 2^l cases based on τ. Thus, we must have $2^l \geq s(n)$, or, equivalently,

$$l \geq \log_2 s(n) \geq \log_2 \frac{(n-d)^d}{d!} = d \log_2(n-d) - \log_2(d!) \in \Omega(\log_2 n) \ .$$

This concludes the proof. □

Note that the assumption that $\Sigma.\mathsf{Verify}(C, \tau) = R$ is somewhat artificial: We assume an *ideal* d-fault-tolerant signature scheme, where $\Sigma.\mathsf{Verify}$ always "magically" outputs the correct multiset R, when called with a claim sequence containing n claims.

On the one hand, $\Sigma.\mathsf{Verify}(C, \tau) \supseteq R$ is required by the d-fault-tolerance of Σ. Intuitively, one would expect the other $\Sigma.\mathsf{Verify}(C, \tau) \subseteq R$ direction to follow from the security of Σ. However, this does not appear to follow in general, due to two reasons:

The first reason is that security is only required against adversaries that have running time polynomial in κ, i.e. adversaries that can create at most a polynomial number of claims.

The second reason is that if for two fixed C, τ there is a claim $c = (\mathsf{pk}, m)$ in $\Sigma.\mathsf{Verify}(C, \tau)$ that is not in R, then this does only violate the security definition if the challenge public key randomly drawn by the security experiment happens to be equal to pk by chance.

B Cover-Free Families Using Multivariate Polynomials

For our polynomial based construction, we can also use multivariate polynomials $f \in \mathbb{F}_q[X_1, \ldots, X_t]$, $t \in \mathbb{N}$, of degree at most k. Each multivariate polynomial f with degree $\leq k$ consists of monomials in terms of $a_{i_1,\ldots,i_t} X_1^{i_1} \cdots X_t^{i_t}$, where $a_{i_1,\ldots,i_t} \in \mathbb{F}_q$ and $i_1 + \cdots + i_t \leq k$. We denote by $\mathbb{F}_q[X_1, \ldots, X_t]_{\leq k}$ the set of all multivariate polynomials $f \in \mathbb{F}_q[X_1, \ldots, X_t]$ of degree at most k, i.e.

$$\mathbb{F}_q[X_1, \ldots, X_t]_{\leq k} := \left\{ \sum_{i_1+\cdots+i_n \leq k} a_{i_1,\ldots,i_t} X_1^{i_1} \cdots X_t^{i_t} \colon a_{i_1,\ldots,i_t} \in \mathbb{F}_q \right\}.$$

For the maximal number of monomials of degree exactly k, we obtain $\binom{t+k-1}{k}$. Hence, for degree at most k, we have

$$\sum_{i=0}^{k} \binom{t+i-1}{i} = \binom{t+k}{k}, \text{ and hence, } |\mathbb{F}_q[X_1, \ldots, X_t]_{\leq k}| = q^{\binom{t+k}{k}}.$$

We can now define

$$B_f := \left\{ (\boldsymbol{x}, f(\boldsymbol{x})) \colon \boldsymbol{x} \in \mathbb{F}_q^t \right\}, \text{ with } |B_f| = q^t,$$

and

$$\mathcal{B} := \{ B_f \colon f \in \mathbb{F}_q[X_1, \ldots, X_t]_{\leq k} \}, \text{ with } |\mathcal{B}| = q^{\binom{t+k}{k}}.$$

Now, we set

$$\mathcal{S} := \mathbb{F}_q^{t+1}, \text{ which is of size } q^{t+1}.$$

The number of zeros is at most $k \cdot q^{t-1}$ and thus, for different $B_f, B_{f_1}, \ldots, B_{f_d} \in \mathcal{B}$ it holds

$$\left| B_f \setminus \bigcup_{i=1}^{d} B_{f_i} \right| \geq q^t - d \cdot k \cdot q^{t-1}.$$

To achieve a d-CFF with this construction, $q^t \geq d \cdot k \cdot q^{t-1} + 1$ must be fulfilled.

Compression Ratio of Our Bounded Scheme. If our bounded scheme is instantiated with this multivariate CFF, and we assume for simplicity, that the size of signatures of the underlying scheme Σ' is bounded by a constant, then as shown in (3), the compression ratio is

$$\rho(n) = \frac{n}{\text{rows}(\mathcal{M})} = \frac{n}{|\mathcal{S}|} = \frac{n}{q^{t+1}} \ .$$

For $n = |\mathcal{B}|$, we therefore have

$$\rho(n) = \frac{|\mathcal{B}|}{|\mathcal{S}|} = \frac{q^{\binom{t+k}{k}}}{q^{t+1}} \ .$$

Compression Ratio of Our Unbounded Scheme. By using Lemma 1 on $\mathcal{M}$ we can also obtain a monotone CFF based on multivariate polynomials and use it to instantiate our unbounded scheme. The discussion about the compression ratio of the unbounded scheme in Section 5 also applies to this instantiation.