Policy-Based Redactable Signatures

Zachary A. Kissel[†]

[†]Department of Computer Science, Merrimack College, North Andover, Massachusetts 01845
kisselz@merrimack.edu

Abstract

In this work we make progress towards solving an open problem posed by Bilzhause et. al [5], to give constructions of redactable signature schemes that allow the *signer* to limit the possible redactions performed by a third party. A separate, but related notion, called *controlled disclosure* allows a redactor to limit future redactions. We look at two types of data, sets and linear data (data organized as a sequence). In the case of sets, we limit redactions using a policy modeled by a monotone circuit or any circuit depending on the size of the universe the set is drawn from. In the case of linear data, we give a linear construction from vector commitments that limits redactions using a policy modeled as a monotone circuit. Our constructions have the attractive feature that they are built using only blackbox techniques.

1 Introduction

A redactable signature scheme (RSS) [22, 31] is a signature scheme that allows a signer to sign a message in such a way that a third party may later redact the message and update the signature *without* the signing key. Applications of this primitive include redacting of sensitive of data from documents [22, 31] and anonymous credential systems [29]. One of the problems with redactable signature schemes, as pointed out in [5], is known schemes allow for arbitrary redaction by a third party. In this work we make progress towards solving an open question of Bilzhausse et. al and offer constructions of redactable signature schemes that allow the original signer to limit what can be redacted by a third party. We refer to such signature schemes as *policy-based redactable signatures.* In a policy-based RSS, given only the verification key, the third party may redact message m to m' if and only if m' satisfies the policy associated with the *signature*, specified at *signing time.* Moreover, a third party may further restrict the policy during redaction to further limit future redactions. The last property is not new, the notion of *controlled disclosure* of [26] is already present in some RSS schemes. We emphasize that our policy-based constructions additionally put redaction control in the hands of the signer, unlike controlled disclosure. Readers familiar with P-Homomorphic signatures [1] may see similarities. However, our formulation differs in that our policy is specified at signing time and has a role in redaction and verification.

Contributions Our work has several contributions:

- We formally define *policy-based* redactable signature schemes.
- We offer two small universe set policy-based redactable signature schemes for any policy. One from approximate membership query data structures and one from cryptographic accumulators.
- We give a set policy-based redactable signature scheme for any size universe for policies expressible as monotone circuits from cryptographic accumulators.

- We give a linear document policy-based redactable signature scheme for any size universe for policies expressible as monotone circuits from vector commitments.
- We analyze the performance of our construction for sets with monotone circuit policies in comparison to the classic set construction of Derler et. al [17] and conclude that our construction has minimal slowdown in signing and verification time.
- We demonstrate that work must be undertaken to increase the performance of redactable signature schemes in practice.

1.1 Background

A *redactable signature scheme* (RSS) is a signature scheme where the a signer may sign a message m (resulting in signature σ) in such a way that a third party may redact the original message m to m' updating signature σ to σ' without the signing key. This primitive was defined simultaneously by [22, 31]. There are three security notions that pertain to redactable signatures: a slightly modified form of classic existential unforgeablilty under chosen message attack; a *privacy* notion which guarantees a redacted signature is indistinguishable from the original signature; and *transparency* which says that signing a redacted message is indistinguishable from redacting a signed message.

Early works [22, 31, 17, 12, ...] focused on two types of data: linear data (like text or ordered sets) and data organized in sets. Some initial work has been undertaken on tree structured data [10], A useful survey of the redactable signatures can be found in [5].

A trivial (and inefficient) construction for linear data involves breaking message m into n blocks, $m_1 \parallel m_2 \parallel \cdots \parallel m_n$. For each block i compute $\sigma_i = \mathsf{Sign}_{sk}(r \parallel i \parallel m_i)$, where $r \xleftarrow{\$} \{0,1\}^n$. To redact, message block m_i the redactor deletes m_i and signature σ_i. The trivial construction satisfies unforgeablilty and privacy[1]. By dropping the index of the message block from the signature, one can achieve a trivial construction that supports data organized as sets. This set construction is unforgeable, private, and transparent.

The linear construction of Johnson et. al [22] is based on Merkle trees coupled with the GGM PRF tree construction. Roughly speaking, each of the message blocks m_1 through m_n form the leaves of a GGM tree such that block m_i is paired with $k_i = f_s(i-1)$. Next, the hash of the $k_i \parallel m_i$ is computed for every block i and the results placed at the leaves of a Merkle tree. The signature is $\sigma = (s, v, \mathsf{Sign}_{sk}(v))$ where v is the root of the Merkle tree. The construction only offers unforgeability. Depending on the application this may offer sufficient security. Because of the dependence on Merkle trees, a set construction does not easily follow, unless one wants to restrict the notion to ordered sets. Casting the construction in the language of vector commitments [11], the Merkle tree, a vector commitment, is committing to the $k_i \parallel m_i$ pairs. Therefore, any vector commitment scheme, could be substituted, for this construction.

We can generalize the construction of Johnson et. al by replacing the GGM tree with a constrained pseudorandom function (CPRF)[2] that supports constraining inputs to falling into a range. The Merkle tree portion of the construction remains unchanged. When redacting instead of providing the necessary nodes of the GGM tree (the GGM based CPRF, in actuality) we release the constrained key for the required range. If we use a suitably constructed CPRF, we can achieve predicate privacy and a fixed sized CPRF key (on the order of $\Theta(\lg n)$). In fact, a CPRF for bit-fixing[3] may also serve this purpose.

Derler et. al [17] gave a set construction and a linear construction. The set construction is based on *cryptographic accumulator*s [3]. The idea is to add all elements of set $\mathcal{X}$ to the accumulator, sign the accumulator value with an existentially unforgeable signature scheme, and output a signature that consists of the accumulator value, a witness w_x for all $x \in \mathcal{X}$, and the signed accumulator value. To redact the signature, delete the associated witnesses. The construction is unforgeable, private, and transparent. They arrive at a linear construction by leveraging the basic idea in the set construction together with a tag to

[1][22] offers slightly different trivial construction that only offers existential unforgeability under redaction.

[2]A form of PRF that allows a third party that posses a special constrained key to evaluate the PRF on some subset of the domain.

[3]Some of the bits the of the input are fixed.

aid in preserving relative order of message blocks. In particular to sign the ith block, all tags for the $i-1$ blocks preceding are accumulated into an accumulator, and a signature on the tag, message block, and accumulator value is generated and added along with the ith witness to the output. Redaction consists of deleting accumulators, tags, witnesses, and associated signatures.

More recently, Sanders gave a linear RSS from the Pointcheval-Sanders signature scheme [29]. This scheme requires type-3 bilinear pairings $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$[4]. Of particular interest is this construction allows the key generator to limit which parties can redact messages; as in order to redact, special elements of $\mathbb{G}_1$ must be possessed. However, the size $|\mathcal{S}|$ of the set $\mathcal{S}$ must be known at the time of key generation as the signing and verification keys depend on this size. The size of the signing key is $\Theta(|\mathcal{S}|)$ and the size of the verification key is $\Theta\left(|\mathcal{S}|^2\right)$.

The notion of redactable signature schemes have been cast into a general framework by Ahn et. al through the notion of P-Homomorphic signatures [1] as well as by Derler et. al [17]. While prior works tied to a specific class of data structure (text, sets, etc.) and security notions, Derler et. al focused on generic definitions that unify the space. Ahn et. al refer to their redactable signature schemes as *quoting* schemes, and are merely a type of P-Homomorphic signatures.

Some work has been undertaken related to the notion of redaction control, called *controlled disclosure* [26, 21]. In controlled disclosure the *redactor* is allowed to declare certain regions of a document as unredactable. This declaration is made during initial redaction. The signer has no control over what is and is not marked as redactable. The construction of [26] is based on the content extraction signature schemes of [31]. The work of [21] is based on the linear construction of Johnson et. al, thus transparency is not provided.

2 Preliminaries

In this section we introduce notation and review necessary primitives. The primitives we use include: approximate membership query (AMQ) filters, constrained pseudorandom functions (CPRFs), cryptographic accumulators, vector commitments, and threshold secret sharing. Lastly, we formally introduce redactable signature schemes.

2.1 Notation

We denote by $\mathsf{negl}(\cdot)$ an arbitrary negligible function. We write $x \xleftarrow{\$} A$ to denote that x is sampled from set A uniformly at random. The concatenation of bit-strings m_1 and m_2 is denoted $m_1 \parallel m_2$. We denote by $(\mathrm{DS}.\mathsf{Gen}, \mathrm{DS}.\mathsf{Sign}, \mathrm{DS}.\mathsf{Vrfy})$ an existentially unforgeable under chosen message attack (EUF-CMA) secure digital signature scheme. The signing key will be denoted by sk and the verification key will be denoted vk. We represent a PRF family by $\{f_k : \mathcal{D} \to \mathcal{R} \mid k \in \mathcal{K}\}$ where $\mathcal{K}$ is the key space. Throughout this work, the security parameter is λ. We denote by $m \vdash m'$ that m' can be arrived at by redacting m. We denote by $\mathcal{P}$ a set of policies that a signer can enforce in a redactable signature scheme. For a policy $P \in \mathcal{P}$, we write $m \vdash_P m'$ to denote m can be redacted to m' if and only if $P(m') = 1$. We say that policy $P' \in \mathcal{P}$ is a *subpolicy* of policy $P \in \mathcal{P}$ if for all x where $P'(x) = 1$, $P(x) = 1$. We denote the characteristic bitstring of set A relative to universe $\mathcal{U}$ by $\mathcal{X}_{\mathcal{U}}(A)$.

2.2 Approximate Membership Query (AMQ) Filter

An *Approximate Membership Query* (AMQ) filter compactly stores a set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ such that membership queries will always be correct when a element $x \in \mathcal{X}$ but, may be incorrect when $x \notin \mathcal{X}$ (a false positive). AMQ filters include: Bloom Filters [6], Cuckoo Filters [18], Quotient Filters [13], and XOR Filters [20]. Formally, we denote an AMQ filter as tuple of three algorithms Init, Add, and $\mathsf{Contains}$ where Init may take an expected false positive rate and return an empty filter; Add takes an element x and a filter for set $\mathcal{X}$

[4]There is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

and returns a new filter which represents $\mathcal{X} \cup \{x\}$; and $\mathsf{Contains}$ takes an element x and a filter and returns one if x is probably in the set and zero if x is not in the set.

2.3 Constrained Pseudorandom Functions

A constrained pseudorandom function (CPRF) is a tuple of algorithms
$\Pi = (\mathsf{F.Gen}, \mathsf{F.Constrain}, \mathsf{F.Eval}, \mathsf{F.ConstrainedEval})$ defined over domain $\mathcal{X}$ and range $\mathcal{Y}$ with the following properties:

- $\mathsf{F.Gen}\left(1^\lambda\right)$ takes as input a security parameter, 1^λ and samples a random function from PRF family by sampling a master key msk from the appropriate keyspace.
- $\mathsf{F.Constrain}\left(\mathsf{msk}, \varphi\right)$takes the master key and predicate φ as input and returns a key k_φ that represents the constrained key.
- $\mathsf{F.Eval}\left(\mathsf{msk}, x\right)$ takes the master key and an input $x \in \mathcal{X}$ and returns the appropriate value $y \in \mathcal{Y}$.
- $\mathsf{F.ConstrainedEval}\left(k_\varphi, x\right)$ takes as input a constrained key k_φ and an $x \in \mathcal{X}$. If $\varphi\left(x\right) = 1$ the correct value $y \in \mathcal{Y}$ is returned (i.e. $\mathsf{Eval}\left(k_\varphi, x\right)$); otherwise either a random value in $\mathcal{Y}$ or $\perp$ is returned.

Correctness. We say a CPRF is *correct* if the evaluation of x using the constrained key agrees with the evaluation of x using the master key when the predicate associated with the constrained key is satisfied. More formally, F is *correct* if for all $x \in \mathcal{X}$ where $\varphi\left(x\right) = 1$, master keys msk, and constrained keys k_φ, $\mathsf{F.Eval}(\mathsf{msk}, x) = \mathsf{F.ConstrainedEval}(k_\varphi, x)$.

Security Notion. A constrained pseudorandom function has a nuanced security guarantee. With a traditional PRF, security states that under a polynomial number of adaptive queries a probabilistic-polynomial time (PPT) adversary can not distinguish between the output of a PRF and the output of a randomly selected function of the same domain and range. When it comes to CPRFs security is relative to a predicate $\varphi \in \Phi$ where the adversary will be able to evaluate the PRF at any point $x \in \mathcal{X}$ where $\varphi\left(x\right) = 1$ without the assistance of an evaluation oracle.

Given a CPRF $\mathsf{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, we can cast the security notion as a game $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{CPRF-Secure}}\left(b, \lambda\right)$ where $\mathcal{A}$ is a probabilistic polynomial-time adversary, $b \in \{0, 1\}$, and λ is a security parameter we define the security game $\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Secure}}\left(b, \lambda\right)$ as follows:

$\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Secure}}\left(b, \lambda\right)$:

1. The game runs $\mathsf{F.Gen}\left(1^\lambda\right)$ to obtain master key msk.
2. If $b = 0$, the game samples a random function $R : \mathcal{X} \to \mathcal{Y}$.
3. The adversary is provided access to two oracles:
   (a) A $\mathsf{F.Constrain}$ oracle which allows $\mathcal{A}$ to ask the game for a constrained key k_φ for predicate $\varphi \in \Phi$, adding φ to P, the set of queried predicates.
   (b) An $\mathsf{F.Eval}$ oracle which behaves as follows on query x:
      - If $b = 0$, answer the evaluation queries as follows: if $\varphi\left(x\right) = 1$ for some $\varphi \in P$, respond honestly. If there is no such predicate, output the value of $R\left(x\right)$.
      - If $b = 1$, return the output of $\mathsf{Eval}\left(\mathsf{msk}, x\right)$.
4. Eventually the $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.

We say that a CPRF is secure if for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl such that

$$\left|\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Secure}}(0,\lambda)=1\right]-\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Secure}}(1,\lambda)=1\right]\right|\leq\mathsf{negl}(\lambda).$$

Remark 2.1. If we modify the CPRF-secure game so we force the adversary to commit to some challenge point *before* they get access to the oracle we arrive at the (challenge) selective security game.

Remark 2.2. If in the game the adversary is not given access to a constrain oracle but, rather can ask for exactly one constrained key *before* being given access to the evaluation oracle, we say that the security notion is (constrained-key) selective. This notion of security was (seemingly first) considered by [9].

Predicate Privacy. Roughly speaking, predicate privacy implies that the party that posses the constrained key can not learn the associated predicate. We can formalize this notion as a game $\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Priv}}(\lambda)$ (first defined by [8]) between a challenger and a PPT adversary $\mathcal{A}$ as follows:

$\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Priv}}(\lambda)$:

1. Challenger runs $\mathsf{msk}\leftarrow\mathsf{F.Gen}(1^\lambda)$ and constructs an empty set of queries Q.
2. Adversary $\mathcal{A}$ is given access to an evaluation oracle that on query x returns $y\leftarrow\mathsf{F.Eval}(\mathsf{msk},x)$ to $\mathcal{A}$. Every query x is added to Q.
3. Eventually, $\mathcal{A}$ outputs two predicates φ_0 and φ_1 from Φ such that for all $x\in Q$, $\varphi_0(x)=\varphi_1(x)$. The challenger samples a bit $b\overset{\$}{\leftarrow}\{0,1\}$ and returns $k_{\varphi_b}\leftarrow\mathsf{F.Constrain}(\mathsf{msk},\varphi_b)$ to $\mathcal{A}$.
4. $\mathcal{A}$ is given access to the evaluation oracle but, is not allowed to query on a value of x where $\varphi_0(x)\neq\varphi_1(x)$; otherwise, $\mathcal{A}$ can trivially distinguish between the two keys.
5. Eventually, $\mathcal{A}$ outputs a bit $b'\in\{0,1\}$. The adversary $\mathcal{A}$ wins the game if $b=b'$.

We say that a CPRF F is one-key private if for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathsf{F}}^{\mathrm{CPRF-Priv}}(\lambda)=1\right]\leq\frac{1}{2}+\mathsf{negl}(\lambda).$$

The game, as presented, is known as *one-key privacy* as the adversary is only allowed to make one challenge query. The security notion can be extended to d-key privacy by allowing the adversary to issue d pairs of predicates. In the case of d-key privacy, what is admissible for the adversary must be updated, see [7] for details.

Remark 2.3. We call a CPRF *canonical* if $\mathsf{F.Eval}$ is the same algorithm as $\mathsf{F.ConstrainedEval}$. In other words, a constrained PRF key is indistinguishable from the original PRF master key.

2.4 Cryptographic Accumulator

A cryptographic accumulator [3] is a keyed primitive that allows for the concise representation of a set $\mathcal{X}=\{x_1,x_2,\ldots,x_n\}$. In order to determine if x_i is in $\mathcal{X}$, a witness w_{x_i} is constructed by any party that possess the key. It should, however, be computationally infeasible to construct a witness for any value $y\notin\mathcal{X}$ (a false witness) regardless of whether or not the party possess the key.

More formally a cryptographic accumulator A is a tuple of algorithms $A=(\mathsf{Gen},\mathsf{Eval},\mathsf{Wit},\mathsf{Vrfy})$ where Gen sets up the parameters for the accumulator and returns the accumulator key pair (sk,pk); Eval takes a keypair (sk,pk) and set of elements $\mathcal{X}$ to accumulate and returns an accumulated value acc as well as some potential randomness r; Wit takes the accumulator keypair (sk,pk), the accumulator acc, and an x and returns a witness w_x that proves $x\in\mathcal{X}$ or $\perp$; the Vrfy algorithm takes the accumulator public key pk, a witness w_x, an element x, and the accumulator acc, and returns one if w_x confirms that x is in acc and zero otherwise. In some cases, Eval is allowed to output additional auxiliary information aux and the Wit optionally takes an aux as input.

The correctness notion states that for all honestly generated keys, all honestly computed accumulators and witnesses, the Vrfy algorithm always returns one. The two security notions for cryptographic accumulators are the notion of *collision freeness* and *indistinguishability.* Roughly speaking, an accumulator is *collision free* if it is computationally infeasible to find a witness for a non-accumulated value. Formally, we cast the security as a game, Acc-Coll-Free, where the adversary has access to two oracles: an evaluation oracle $\mathcal{O}_{\mathsf{Eval}}(\cdot,\cdot,\cdot)$ and a witness generation oracle $\mathcal{O}_{\mathsf{Wit}}(\cdot,\cdot,\cdot,\cdot)$.

$\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{Acc-Coll-Free}}(\lambda)$:

1. The game runs $\mathsf{Gen}\left(1^\lambda\right)$ to obtain the keypair (sk, pk).
2. $\mathcal{A}$ is given pk and access to $\mathcal{O}_{\mathsf{Eval}}$ and $\mathcal{O}_{\mathsf{Wit}}$ and eventually outputs $\left(w_{x_i}^*, x_i^*, \mathcal{X}^*, r^*\right)$ where r^* is the randomness used by $\mathcal{A}$ when generating the accumulator.
3. The adversary wins if $\mathsf{Vrfy}_{pk}\left(w_{x_i}^*, x_i^*, acc^*\right) = 1$ and $x_i^* \notin \mathcal{X}^*$, where acc^* was generated using keypair (sk, pk) and randomness r^*.

We say that a cryptographic accumulator Π is collision free if $\Pr\left[\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{Acc-Coll-Free}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$. For accumulator schemes where Eval is deterministic, we elide r^* from the game.

A cryptographic accumulator is said to be *indistinguishable* if it is computationally infeasible to distinguish which of two sets $\mathcal{X}_0$ and $\mathcal{X}_1$ is associated with the accumulator value acc when only witnesses for elements in $x \in \mathcal{X}_0 \cap \mathcal{X}_1$ are available. Formally, we cast the security as a game, Acc-Ind, where the adversary has access to two oracles: an evaluation oracle $\mathcal{O}_{\mathsf{Eval}}(\cdot,\cdot,\cdot)$ and a witness generation oracle $\mathcal{O}_{\mathsf{Wit}}(\cdot,\cdot,\cdot,\cdot)$ that will only produce witnesses for values $x \in \mathcal{X}_0 \cap \mathcal{X}_1$.

$\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{Acc-Ind}}(\lambda)$:

1. The game runs $\mathsf{Gen}\left(1^\lambda\right)$ to obtain the keypair (sk, pk).
2. The game samples bit $b \xleftarrow{\$} \{0,1\}$.
3. $\mathcal{A}$ is run and outputs two sets $\mathcal{X}_0$ and $\mathcal{X}_1$ such that $\mathcal{X}_0 \cap \mathcal{X}_1 \neq \emptyset$.
4. The game computes $acc \leftarrow \mathsf{Eval}(pk, \mathcal{X}_b)$.
5. $\mathcal{A}$ is given acc and access to $\mathcal{O}_{\mathsf{Eval}}$ and $\mathcal{O}_{\mathsf{Wit}}$ as described above and eventually outputs $b^* \in \{0,1\}$.
6. The adversary wins if $b = b^*$.

We say that a cryptographic accumulator Π is indistinguishable if $\Pr\left[\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{Acc-Ind}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

There are several constructions of cryptographic accumulators in the literature, for a survey see [16, 27, 28]. There are three commonly deployed accumulator techniques: Merkle trees, use of the the strong RSA assumption, and bilinear pairings. The drawback of the Merkle tree based accumulator is that the witnesses have size logarithmic in the number of elements in $\mathcal{X}$. The drawback to the strong RSA based accumulator is that it requires the elements of the set to be represented as prime numbers. This of course can be enforced using a random oracle that maps any element in the universe to a prime number, which is computationally expensive. The drawback for most pairing based constructions is key size. One can also construct a cryptographic accumulator from any vector commitment scheme through a generic transformation [11].

Observe that to verify the membership of n elements in an accumulator, requires n witnesses. This linear growth can be problematic. This can be overcome by batching witness creation and verification. In batched accumulator there are two additional algorithms $\mathsf{WitBatch}$ and $\mathsf{VrfyBatch}$. The $\mathsf{WitBatch}$ algorithm takes the accumulator keypair (sk, pk), the accumulator acc, and a subset A of $\mathcal{X}$ and returns a witness w_A for $A \subseteq \mathcal{X}$. The $\mathsf{VrfyBatch}$ algorithm takes the witness w_A, the subset A, and the accumulator acc, one is returned if w_A confirms that $A \subseteq \mathcal{X}$ and zero otherwise. A non-trivial batch accumulator has $\Theta(1)$-sized witnesses.

2.5 Vector Commitments

A vector commitment, proposed by Catalano and Fiore in [11], is a commitment scheme that allows a party to commit to sequence of values (a vector) $m_1, m_2, \ldots, m_q$ such that the committer can prove that m_i is the ith committed message. While Catalano and Fiore defined a vector commitment as a dynamic object, we will restrict ourselves to the static case (i.e., we don't allow updates to the commitment). A (static) vector commitment scheme is a tuple of polynomial type algorithms ($\mathrm{VC}.\mathsf{KeyGen}$, $\mathrm{VC}.\mathsf{Com}$, $\mathrm{VC}.\mathsf{Open}$, $\mathrm{VC}.\mathsf{Ver}$) where,

- $\mathrm{VC}.\mathsf{KeyGen}\left(1^\lambda, q\right)$: Given the security parameter λ and the vector size q, the key generation outputs some public parameters pp.
- $\mathrm{VC}.\mathsf{Com}_{\mathrm{pp}}\left(m_1, m_2, \ldots, m_q\right)$: Output a commitment C and auxiliary information aux.
- $\mathrm{VC}.\mathsf{Open}_{\mathrm{pp}}\left(m, i, \mathrm{aux}\right)$: This algorithm is run by the committer to produce a proof Π_i that proves message m is at position i in the committed vector.
- $\mathrm{VC}.\mathsf{Ver}_{\mathrm{pp}}\left(C, m, i, \Pi_i\right)$: The verification accepts (outputs 1) only if Π_i is a valid proof that C was created for a sequence $m_1, \ldots, m_q$ such that $m = m_i$.

A vector commitment scheme should also be *concise*. By *concise* it is meant that the size of the commitment and the proofs are both independent of q.

Correctness for vector commitments states that for all $\lambda \in \mathbb{N}$, q polynomial in λ, and all honestly generated public parameters pp, if C is a commitment on a vector $(m_1, m_2, \ldots, m_q)$ obtained by running $\mathrm{VC}.\mathsf{Com}$, Π_i is a proof for position i generated by $\mathrm{VC}.\mathsf{Open}$ then, $\mathrm{VC}.\mathsf{Ver}_{\mathrm{pp}}\left(C, m, i, \Pi_i\right) = 1$ with overwhelming probability.

The main security notions for vector commitments is position biding. Position binding informally means that no probabilistic polynomial time adversary can generate a commitment that can be opened to two different values at the same position. Formally, we can cast position binding as a game Pos-Bind defined as follows:

$\mathsf{Exp}^{\mathrm{Pos-Bind}}_{\mathcal{A},\mathrm{VC}}\left(\lambda, q\right)$:

1. The game runs $\mathrm{VC}.\mathsf{KeyGen}\left(1^\lambda, q\right)$ to obtain the public parameters pp.
2. $\mathcal{A}$ is run on pp and outputs a commitment C, two messages m and m', and index i, and two proofs Π_i and Π'_i for message m and m' respectively.
3. The adversary wins (outputs 1) if $\mathrm{VC}.\mathsf{Ver}_{\mathrm{pp}}\left(C, m, i, \Pi_i\right) = 1$ and $\mathrm{VC}.\mathsf{Ver}_{\mathrm{pp}}\left(C, m', i, \Pi'_i\right) = 1$.

We say that a vector commitment scheme VC has position binding if for all probabilistic polynomial time adversaries $\mathcal{A}$ and all honestly generated parameters $\Pr\left[\mathsf{Exp}^{\mathrm{Pos-Bind}}_{\mathcal{A},\mathrm{VC}}\left(\lambda\right) = 1\right] \leq \mathsf{negl}\left(\lambda\right)$.

Readers familiar with traditional commitment schemes might wonder why vector commitment security doesn't discuss hiding, as mentioned in [11], hiding is not a useful property for a majority of use cases for vector commitments. We, however, will require this property in order to obtain a private RSS, so we introduce it here. Informally, a vector commitment scheme is *hiding* if an adversary can not distinguish whether a commitment was constructed from a sequence $m_1, m_2, \ldots, m_q$ or $m'_1, m'_2, \ldots, m'_q$. We formalize our property as a game vc-hiding a follows:

$\mathsf{Exp}^{\mathrm{vc-hiding}}_{\mathcal{A},\mathrm{VC}}(\lambda, q)$:

1. The game runs VC.$\mathsf{KeyGen}\left(1^{\lambda}, q\right)$ to obtain the public parameters pp.
2. $\mathcal{A}$ is run on pp and outputs a pair of message sequences $\mathcal{M}_0$ and $\mathcal{M}_1$.
3. The game samples $b \xleftarrow{\$} \{0,1\}$ and computes $(C_b, aux_b) \leftarrow \mathrm{VC}.\mathsf{Com}_{\mathrm{pp}}(m_{b,1}, m_{b,2}, \ldots, m_{b,q})$
4. $\mathcal{A}$ is given C_b and outputs $b' \in \{0,1\}$.
5. The adversary wins (outputs 1) if $b = b'$.

We say that a vector commitment scheme has the hiding property if for all probabilistic polynomial time adversaries $\mathcal{A}$ and all honestly generated parameters pp,

$$\Pr\left[\mathsf{Exp}^{\mathrm{vc-hiding}}_{\mathcal{A},\mathrm{VC}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Vector commitments have been built from the CDH assumption in bilinear groups and the RSA assumption [11]. One can also think of a Merkle tree as an instantiation of a vector commitment scheme.

2.6 Threshold Secret Sharing

Threshold secret sharing is a primitive developed by Shamir [30] that allows n parties to share a secret in such a way that no party possess the secret. However, $t \leq n$ parties can combine their shares to recover the secret. We will refer to such a scheme as a (t, n)-threshold secret sharing scheme. Formally a secret sharing scheme consists of two algorithms share and $\mathsf{reconstruct}$. The share algorithm takes a message m and outputs a sequence of n shares $s_1, s_2, \ldots, s_n$. The $\mathsf{reconstruct}$ algorithm takes t or more shares and reconstructs the message m.

The security notion for threshold secret sharing is simple: any number of shares $k < t$, will perfectly hide the secret. We formalize our notion as an indistinguishability game.

$\mathsf{Exp}^{\mathrm{share-secure}}_{\mathcal{A},\Pi}(\lambda)$:

1. Select n and $t \leq n$.
2. Sample $b \xleftarrow{\$} \{0,1\}$.
3. Give $\mathcal{A}$, (t, n) and access to a share oracle $\mathcal{O}$ that takes two messages m_0 and m_1, a number of shares $k < t$, and returns k shares of m_b.
4. Eventually $\mathcal{A}$ outputs bit $b' \in \{0,1\}$ we say that $\mathcal{A}$ wins the game if $b = b'$.

We say that a secret sharing scheme Π is secure if for all probabilistic polynomial time adversaries there exists a negligible function such that

$$\Pr\left[\mathsf{Exp}^{\mathrm{share-secure}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

2.7 Redactable Signatures

Formally, a redactable signature is a four tuple $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Redact}, \mathsf{Vrfy})$ of polynomial time algorithms.

- $\mathsf{Gen}\left(1^{\lambda}\right)$: a probabilistic algorithm that takes the security parameter λ as input and returns a signing key sk and a verification key vk.

- $\mathsf{Sign}_{sk}(m)$ a possibly probabilistic algorithm takes a signing key sk and a message m as input and returns a signature σ on message m.

- $\mathsf{Redact}_{vk}(m, \sigma, m')$ a possible probabilistic algorithm that takes the verification key vk, a message m, its signature σ, and and an m' such that $m \vdash m'$ as input. The algorithm returns σ'.
- $\mathsf{Vrfy}_{vk}(m, \sigma)$ a deterministic algorithm that takes the verification key vk, the message m, and purported signature on m, σ and returns 1 if σ is a valid signature for m and 0 otherwise.

There are three main security notions for redactable signatures:

Unforgeability. It should be hard for an adversary to forge a valid signature on a message that is *not* a redaction of a message they have already seen. It should be noted that this notion is based on a small modification to traditional existential unforgeability under a chosen message attack (EUF-CMA). Formally we capture our security notion as a game.

$\mathsf{Exp}^{\mathrm{EUF-CMA-RSS}}_{\mathcal{A},\Pi}(\lambda)$:

1. $\mathsf{Gen}(1^n)$ is run to obtain keys (sk, vk).
2. Adversary $\mathcal{A}$ is given vk and access to an oracle $\mathcal{O}\mathsf{Sign}_{sk}(\cdot)$.
    - Store the queries to the oracle in a set called $\mathcal{Q}$.
3. Eventually the adversary outputs (m^*, σ^*).
4. $\mathcal{A}$ succeeds iff (1) $\mathsf{Vrfy}_{vk}(m^*, \sigma^*) = 1$ and (2) $m^* \notin \{m \vdash m^* \mid m \in \mathcal{Q}\}$.
    - In the case of success, the game outputs 1. Otherwise, the game outputs 0.

We say that an RSS Π is EUF-CMA-RSS secure if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr\left[\mathsf{Exp}^{\mathrm{EUF-CMA-RSS}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.

Privacy. It should be hard for an adversary to determine the original signature from a redacted signature. Formally we capture the notion as a security game.

$\mathsf{Exp}^{\mathrm{RSS-Privacy}}_{\mathcal{A},\Pi}(\lambda)$:

1. $\mathsf{Gen}(1^n)$ is run to obtain keys (sk, vk).
2. $b \xleftarrow{\$} \{0, 1\}$.
3. Adversary $\mathcal{A}$ is given vk and access to an oracle $\mathcal{O}\mathsf{Sign}_{sk}(\cdot)$ and $\mathcal{O}\mathsf{LoRRedact}_{sk,b}(\cdot, \cdot, \cdot, \cdot)$.
4. Eventually the adversary outputs $b' \in \{0, 1\}$.
5. $\mathcal{A}$ succeeds if $b = b'$ and the game outputs 1; otherwise, the game outputs 0.

The oracle $\mathcal{O}\mathsf{LoRRedact}_{sk,b}(\cdot, \cdot, \cdot, \cdot)$ takes as input four messages m_0, m_0', m_1, and m_1' and proceeds as follows:

$\mathcal{O}\mathsf{LoRRedact}_{sk,b}(m_0, m_0', m_1, m_1')$:

1. Compute $\sigma_i = \mathsf{Sign}_{sk}(m_i)$ for $i = 0, 1$.
2. Compute $\sigma_i' = \mathsf{Redact}(m_i, \sigma, m_i')$ for $i = 0, 1$.
3. If $m_0' \neq m_1'$, return $\perp$; else return σ_b'.

We say an RSS Π is RSS-private if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr\left[\mathsf{Exp}^{\mathrm{RSS-Privacy}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.

Transparency. It should be hard for an adversary to distinguish the redacted signature of a message from the signature of a redacted message. Formally we capture the notion as a security game.

$\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{RSS-Transparency}}(\lambda)$:

1. $\mathsf{Gen}(1^n)$ is run to obtain keys (sk, vk).
2. $b \xleftarrow{\$} \{0,1\}$.
3. Adversary $\mathcal{A}$ is given vk and access to an oracle $\mathcal{O}\mathsf{Sign}_{sk}(\cdot)$ and $\mathcal{O}\mathsf{SignRedact}_{sk,b}(\cdot,\cdot)$.
4. Eventually the adversary outputs $b' \in \{0,1\}$.
5. $\mathcal{A}$ succeeds if $b = b'$ and the game outputs 1; otherwise, the game outputs 0.

The oracle $\mathcal{O}\mathsf{SignRedact}_{sk,b}(\cdot,\cdot)$ takes as input two messages m and m' where $m \vdash m'$ and proceeds as follows:

$\mathcal{O}\mathsf{SignRedact}_{sk,b}(m, m')$:

1. Compute $\sigma_0 = \mathsf{Redact}(m, \mathsf{Sign}_{sk}(m), m')$.
2. Compute $\sigma_1 = \mathsf{Sign}_{sk}(m')$
3. Return σ_b

We say an RSS Π is RSS-transparent if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr\left[\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{RSS-Transparency}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.

It is known that transparency implies privacy[17, 10]. One can envision having a scheme that is unforgeable and private (privacy does *not* imply transparency), though to our knowledge, there is no such (efficient) RSS.

3 Policy-Based Redactable Signature Schemes

A policy-based redactable signature scheme is a primitive where a signer can sign a message such that a third party may redact the message (and update the signature) provided the redacted message matches the policy P specified by the signer at the time of signing. We formalize this primitive by adapting the formalism of redactable signatures to include a policy argument to Sign and Redact. Specifically, a *policy-based* redactable signature is a four tuple $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Redact}, \mathsf{Vrfy})$ of polynomial time algorithms.

- $\mathsf{Gen}\left(1^\lambda\right)$: a probabilistic algorithm that takes the security parameter λ as input and returns a signing key sk and a verification key vk.
- $\mathsf{Sign}_{sk}(m, P)$ a possibly probabilistic algorithm takes a signing key sk, a message m, and a policy $p \in \mathcal{P}$ as input and returns a signature σ on message m with redaction policy P.
- $\mathsf{Redact}_{vk}(m, \sigma, m', P')$ a possible probabilistic algorithm that takes the verification key vk, a message m, its signature σ, and a message m' as input. We require that $m \vdash_P m'$ and $P'(m') = 1$, where P is the policy extracted from signature σ. The algorithm returns σ'. Some schemes may require that policy P' be a subpolicy of P.
- $\mathsf{Vrfy}_{vk}(m, \sigma)$ a deterministic algorithm that takes the verification key vk, the message m, and purported signature on m, σ. The algorithm returns 1 if σ is a valid signature for m and $p(m) = 1$, where P is the policy extracted from σ. Otherwise, 0 is returned.

We note that the notion that a user can not redact a message to one not covered by the policy, is captured by the RSS notion of unforgeability with minor changes to the oracle and the $\vdash$ operator being replaced with $\vdash_P$. The Signing oracle will now take a policy $P \in \mathcal{P}$. The changes to the other oracles are summarized in figure 1 Similar to the traditional RSS scheme, policy-based redactable signature schemes also have the

$\mathcal{O}\mathsf{LoRRedact}_{sk,b}\left(m_0, m_0', P_0, P_0', m_1, m_1', P_1, P_1'\right)$:

1. Compute $\sigma_i = \mathsf{Sign}_{sk}\left(m_i, P_i\right)$ for $i = 0, 1$.
2. Compute $\sigma_i' = \mathsf{Redact}\left(m_i, \sigma, m_i', P_i'\right)$ for $i = 0, 1$.
3. If $m_0' \neq m_1'$ or $P_0' \neq P_1'$, return $\perp$; else return σ_b'.

(a) The Left or Right Redaction oracle for policy-based redactable signatures.

$\mathcal{O}\mathsf{SignRedact}_{sk,b}\left(m, m', P, P'\right)$:

1. Compute $\sigma_0 = \mathsf{Redact}\left(m, \mathsf{Sign}_{sk}\left(m, P\right), m', P'\right)$.
2. Compute $\sigma_1 = \mathsf{Sign}_{sk}\left(m', P'\right)$
3. Return σ_b

(b) The Sign or Redact oracle for policy-based redactable signatures.

Figure 1: The changes to the oracles to support policy-based redactable signatures.

property that transparency implies privacy (the proof mirrors the proof of proposition one in [10] with minor changes).

4 Policy-Based Redactable Set Signatures

We begin by looking at signatures for data organized as sets. Specifically, the data elements have no implied order. In what follows, the set universe will be denoted as $\mathcal{U}$. We will call a universe *small* if its size is polylogarithmic in the security parameter.

4.1 Small Universe

Restricting ourselves to a universe $\mathcal{U}$ of size polylogarithmic in the security parameter we can construct a policy-based redactable signature scheme for sets either by utilizing a cryptographic accumulator or using an AMQ Filter. These constructions have the benefit of supporting any policy family $\mathcal{P}$.

Cryptographic Accumulator. We can reuse the basic set construction due to [17] to construct a small universe policy-based set RSS. Assume that there exists an cryptographic accumulator $A = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Wit}, \mathsf{Vrfy})$ To restrict the redactions to some restricted subsets of a set S, we add characteristic strings for the set and all possible subsets (those that could arise from valid redaction) $\mathcal{S}$ to an accumulator acc (i.e., $acc \leftarrow \mathsf{Eval}_{\mathsf{sk}}\left(\left\{\mathcal{X}_{\mathcal{U}}\left(A\right) \mid A \subseteq \mathcal{S} \wedge P\left(A\right) = 1\right\}\right)$). The redactable signature becomes $\Big(acc, \left\{\mathsf{Wit}_{sk}\left(\mathcal{X}_{\mathcal{U}}\left(A\right), acc\right)\right\}_{A \subseteq \mathcal{S} \wedge P(A)=1}, \mathsf{Sign}_{sk}\left(acc\right)\Big)$. The scheme is formalized in figure 2. The scheme is both unforgeable and transparent (implying privacy). Since the construction mirrors the work of [17], the proof is basically the same; we point the interested reader to [17, §C.1] for the proof.

The main drawback of this construction, and the reason to search for better constructions, is the reliance on small universes. Observe that the size of $\mathcal{U}$ must be small as the number of subsets are exponential in the the size of $\mathcal{U}$, causing the number of witnesses to be exponential (in the worst case). The situation would be slightly improved if the associated number of possible redactions was polynomial in the size of $\mathcal{U}$, or the size of $\mathcal{U}$ is polylogarithmic in the security parameter.

$\mathsf{Gen}\left(1^\lambda\right)$: Run DS.$\mathsf{Gen}\left(1^\lambda\right)$ to obtain key pair (s, v) and run the accumulators generator function to obtain (sk_a, pk_a). Output the verification key $vk = (v, pk_a)$ and the signing key $sk = (s, sk_a)$.

$\mathsf{Sign}_{sk}(\mathcal{S}, P)$: Parse the signing key sk as (s, sk_a). If $P \notin \mathcal{P}$, return $\perp$. Otherwise, compute $acc \leftarrow \mathsf{Eval}_{\mathrm{sk}_a}\left(\{\mathcal{X}_{\mathcal{U}}(A) \mid A \subseteq \mathcal{S} \wedge P(A) = 1\}\right)$ and return the signature $\sigma_{\mathcal{S}} = \left(acc, \{\mathsf{Wit}_{sk_a}(\mathcal{X}_{\mathcal{U}}(A), acc)\}_{A \subseteq \mathcal{S} \wedge P(A)=1}, P, \mathsf{Sign}_{sk}(acc)\right)$. We will assume that the witnesses are sorted in lexicographical order by characteristic sequence.

$\mathsf{Redact}_{vk}(\mathcal{S}, \sigma, \mathcal{S}', P')$: Parse σ as $\left(acc, \left\{w_{\mathcal{X}_{\mathcal{U}}(A)}\right\}_{A \subseteq \mathcal{S} \wedge P(A)=1}, P, \sigma^*\right)$. If $\mathcal{S} \vdash_P \mathcal{S}'$ and P' is a subpolicy of P where $P'(\mathcal{S}') = 1$, return
$\sigma_{\mathcal{S}'} = \left(acc, \left\{w_{\mathcal{X}_{\mathcal{U}}(A)}\right\}_{A \subseteq \mathcal{S}' \wedge P(A)=1}, \sigma^*\right)$. Otherwise, return $\perp$.

$\mathsf{Vrfy}_{vk}(\mathcal{S}, \sigma)$: Parse vk as (v, pk_a) and σ as $\left(acc, \left\{w_{\mathcal{X}_{\mathcal{U}}(A)}\right\}_{A \subseteq \mathcal{S} \wedge P(A)=1}, \sigma^*\right)$. Return one if there is a valid witness $w_{\mathcal{X}_{\mathcal{U}}(S)}$, Acc.$\mathsf{Vrfy}_{pk_a}\left(w_{\mathcal{X}_{\mathcal{U}}(S)}, \mathcal{X}_{\mathcal{U}}(S), acc\right) = 1$, and DS.$\mathsf{Vrfy}_v(acc, \sigma^*) = 1$; otherwise, return zero.

Figure 2: The full accumulator-based small universe scheme.

AMQ Filter. Similar to our cryptographic accumulator construction we can utilize a characteristic sequence to limit any possible redactions. In this case, however, we dispense with the need of witnesses present in an accumulator (trading for potential false positives). We store the output of a CPRF F with key k evaluated on the characteristic sequence in an AMQ filter $\mathcal{F}$. The signature on a set S becomes $\sigma_{\mathcal{S}} = (\mathcal{F}, k, \mathsf{Sign}_{sk}(\mathcal{F}))$. To produce a redaction, constrain the key k to $\widetilde{k}$ so that only the necessary sets can have membership checked. The need for constraining keys comes from the privacy requirement, without it, an adversary would be able to distinguish a redacted signed message from a redacted then signed message (violating transparency). This requires the CPRF to hide the policy, be canonical, and have a constant sized key (e.g., [14] or [23]). Unfortunately this construction remains theoretical; there are no existing CPRF constructions that simultaneously offer these properties for arbitrary predicates. The scheme is formalized in figure 3. Our scheme is both unforgeable and transparent. Proofs of our claim are found in appendix A

Unlike the cryptographic accumulator approach the AMQ filter approach does not suffer from exponential sized signatures. This is because the CPRF key is of fixed size and the size of the AMQ filter is governed by the desired false positive rate. We note, the signing time will still be exponential in $\mathcal{U}$ in the worst case, so approaches to limit the size of $\mathcal{U}$ are still required.

Remark 4.1. It is important to note that both of these constructions can be adapted slightly so that the universe can be of arbitrary size by computing the characteristic sequence relative to the original set. In so doing we loose transparency as we will be able to distinguish between redacted and non-redacted messages. This will also increase the size of the signature as we will have to store the index in the characteristic sequence for all elements of the set associated with the signature.

4.2 Arbitrary Universe

To construct a policy-based set RSS for an arbitrary sized universe we could take two main approaches: (1) we could develop a domain extension algorithm for a small universe construction; or (2) design a construction independent of the small universe construction. Neither of our small universe constructions appear amenable to domain extension techniques thus, we offer a new arbitrary universe construction. Unlike our constructions

$\mathsf{Gen}\left(1^{\lambda}\right)$: Run DS.$\mathsf{Gen}\left(1^{\lambda}\right)$ and return the resulting key pair (sk, vk).

$\mathsf{Sign}_{sk}\left(\mathcal{S}, P\right)$: If $P \notin \mathcal{P}$, return $\perp$. Otherwise, run $k \leftarrow$ CPRF.$\mathsf{Gen}\left(1^{\lambda}\right)$; run $\mathcal{F} \leftarrow \mathsf{Init}\left(fpr\right)$ to obtain a new filter with false positive rate fpr. Next, for every $A \subseteq \mathcal{S}$ where $P\left(A\right) = 1$, run $\mathcal{F} \leftarrow \mathsf{Add}\left(\text{CPRF}.\mathsf{Eval}\left(k, \mathcal{X}_{\mathcal{U}}\left(A\right)\right), \mathcal{F}\right)$. Return the signature $\sigma_{\mathcal{S}} = \left(\mathcal{F}, k, P, \mathsf{Sign}_{sk}\left(\mathcal{F}\right)\right)$.

$\mathsf{Redact}_{vk}\left(\mathcal{S}, \sigma, \mathcal{S}', P'\right)$: Parse σ as $\left(\mathcal{F}, k, P, \sigma^{*}\right)$. If $\mathcal{S} \vdash_{P} \mathcal{S}'$ and P' is a subpolicy of P where $P'\left(\mathcal{S}'\right) = 1$, output $\sigma_{\mathcal{S}'} = \left(\mathcal{F}, \widetilde{k}, P', \sigma^{*}\right)$, where $\widetilde{k} \leftarrow$ CPRF.$\mathsf{Constrain}\left(k, P\right)$. Otherwise, return $\perp$.

$\mathsf{Vrfy}_{vk}\left(\mathcal{S}, \sigma\right)$: Parse σ as $\left(\mathcal{F}, k, P, \sigma^{*}\right)$. Return one if $P\left(S\right) = 1$, $\mathsf{Contains}\left(\text{CPRF}.\mathsf{Eval}\left(k, \mathcal{X}_{\mathcal{U}}\left(S\right)\right), \mathcal{F}\right) = 1$, and DS.$\mathsf{Vrfy}_{vk}\left(\mathcal{F}, \sigma^{*}\right) = 1$; otherwise, return zero.

Figure 3: The full AMQ-based small universe scheme.

for a small universe, our arbitrary universe construction restricts the policy family to policies that can be expressed as monotone circuits[5].

Overview. Our starting point is the set construction of [17], which relies on a cryptographic accumulator in proving set membership. We will add a mechanism to enforce a policy cast as a monotone circuit which outputs true if the associated Boolean formula over the elements of the subset evaluate to true. Following the construction in [4], we will encode our circuit using a Shamir threshold secret sharing scheme over each gate of the circuit such that if the circuit outputs true, the shared secret is recovered. We note that our construction exposes the policy to both the redactor and verifier.

Constructing the Circuit. Any (monotone) circuit can be represented as an n-ary tree where the literals appear at the leaves of the tree, all internal nodes (gates) represent either the disjunction or conjunction of their children, and the root node captures the output of the circuit. Let $\mathsf{parent}\left(x\right)$ denote the parent of node x and $\mathsf{index}\left(x\right)$ denote the left-to-right index associated with node x in its level. We label each node x with a value t_x that is one if the node is a disjunction or a leaf; otherwise, the number of children of x.

To enforce the correct evaluation of a circuit we will use Shamir secret sharing in such a way that a secret value $s \in \mathbb{F}_p$ is reconstructed if and only if the set elements present in the signature satisfy the circuit. Our algorithm begins at the root of the tree R associating a $t_R - 1$ degree polynomial $q_R\left(\cdot\right)$ such that $q_R\left(0\right) = s$. Proceeding in a level-order traversal, we associate with internal node x a $t_x - 1$ degree polynomial $q_x\left(\cdot\right)$ such that $q_x\left(0\right) = q_{\mathsf{parent}(x)}\left(\mathsf{index}\left(x\right)\right)$. See figure 4 for an example.

Observe that when we perform the reconstruction algorithm recursively using a bottom up level order traversal of the tree, we arrive at the value s. Moreover, we must only store the values associated with the leaves of the tree. If one wishes to hide the labels on the variables a one-way function may be utilized.

Redactable Signature Scheme. The redactable signature scheme on set $\mathcal{S}$ is described in figure 5. As previously noted we rely on the threshold secret sharing scheme to represent our circuit. The secret serves as the value for true associated with the circuit. We also rely on a cryptographic accumulator for committing to the elements of the set and their shares. We rely on a EUF-CMA secure digital signature scheme to sign the accumulator and secret value associated with the root of the tree that represents the circuit. The scheme as presented is unforgeable and transparent (thus implying privacy). The proof is in appendix B.

[5]A circuit without a logical negation gate.

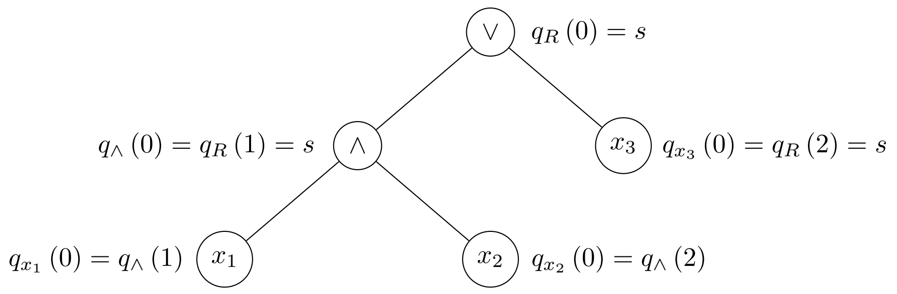

$q_R(0) = s$
$q_\wedge(0) = q_R(1) = s$
$q_{x_3}(0) = q_R(2) = s$
$q_{x_1}(0) = q_\wedge(1)$
$q_{x_2}(0) = q_\wedge(2)$
$\vee$
$\wedge$
x_1
x_2
x_3

Figure 4: The circuit for the expression $(x_1 \wedge x_2) \vee x_3$.

$\mathsf{Gen}\left(1^\lambda\right)$: Run DS.$\mathsf{Gen}\left(1^\lambda\right)$ to obtain key pair (sk_s, vk_s) and run the accumulators generator function to obtain (sk_a, pk_a). Output the verification key $vk = (vk_s, pk_a)$ and the signing key $sk = (sk_s, sk_a)$.

$\mathsf{Sign}_{sk}(\mathcal{S}, P)$: If $P \notin \mathcal{P}$, return $\perp$. Otherwise, parse sk as (sk_s, sk_a); sample an $s \in \mathbb{Z}^*$, construct the tree $\mathcal{T}$ associated with the policy P, compute $acc \leftarrow \mathsf{Eval}_{\mathrm{sk}_a}\left(\{x \parallel \mathsf{share}(q_x(0))\}_{x\in\mathcal{S}}\right)$, and return the signature on $\mathcal{S}$,

$$\begin{aligned}\sigma_{\mathcal{S}} \;\; = \;\; &\big(acc, \mathcal{T}, \{\mathsf{share}(q_x(0))\}_{x\in\mathcal{S}}, \{\mathsf{Wit}_{sk_a}(x \parallel \mathsf{share}(q_x(0)), acc)\}_{x\in\mathcal{S}}, \\ &\mathsf{Sign}_{sk_s}(acc \parallel s)\big).\end{aligned}$$

Notationally, we assume that all elements in $\mathcal{S}$ appear in $\mathcal{T}$ that is, however, not always the case. Elements $x \in \mathcal{S}$ that do not appear in $\mathcal{T}$ should have their share set to a uniformly random sampled element of $\mathbb{Z}^*$.

$\mathsf{Redact}_{vk}(\mathcal{S}, \sigma, \mathcal{S}', P')$: Parse σ as $\left(acc, \mathcal{T}, \{s_x\}_{x\in\mathcal{S}}, \left\{w_{x\parallel s_x}\right\}_{x\in\mathcal{S}}, \sigma^*\right)$ and extract policy P from $\mathcal{T}$. If $\mathcal{S} \vdash_P \mathcal{S}'$ and $P'(\mathcal{S}') = 1$, output the signature $\sigma_{\mathcal{S}'} = \left(acc, \mathcal{T}', \{s_x\}_{x\in\mathcal{S}'}, \left\{w_{x\parallel s_x}\right\}_{x\in\mathcal{S}'}, \sigma^*\right)$, where $\mathcal{T}'$ encodes policy P'. Otherwise, return $\perp$.

$\mathsf{Vrfy}_{vk}(\mathcal{S}, \sigma)$: Parse vk as (vk_s, vk_a) and σ as $\left(acc, \mathcal{T}, \{s_x\}_{x\in\mathcal{S}}, \left\{w_{x\parallel s_x}\right\}_{x\in\mathcal{S}}, \sigma^*\right)$. Return one if for all $x \in \mathcal{S}$, Acc.$\mathsf{Vrfy}_{pk_a}\left(w_{x\parallel s_x}, x \parallel s_x, acc\right) = 1$ and DS.$\mathsf{Vrfy}_{vk_s}(acc \parallel s^*, \sigma^*) = 1$, where s^* results from reconstructing the secret using the shares together with policy tree $\mathcal{T}$; otherwise, return zero.

Figure 5: The monotone circuit RSS scheme.

It should be noted that while we solved the large universe problem, in order for our construction to remain practical $|\mathcal{S}|$ should remain small since our signature has size $\Theta(|\mathcal{S}|)$. This does, however, bring us into line with existing non-policy based constructions which have asymptotically equivalent signature sizes or very large keys.

Remark 4.2. Because of how our policies are enforced, any redactor can impose further limitations on future redactions by modifying the tree representing the policy, thus providing controlled disclosure.

Remark 4.3. If the signer wishes the policy to remain immutable, the policy can be signed. This will, however, affect transparency since the policy may leak the original set.

5 Policy-Based Redactable Signatures for Linear Data

We turn our attention to data that has some implied order and demonstrate a policy-based redactable signature scheme for this data. We offer one construction from vector commitments. This construction mirrors the construction for arbitrary sized universes from cryptographic accumulators. Observe that by substituting a vector commitment scheme for the cryptographic accumulator, we get a construction that enforces position. However, this comes at the cost of transparency and potentially privacy. Our construction from vector commitments only provides unforgeability in all cases, and privacy if the vector commitment scheme has the hiding property (see appendix C). The construction is described in figure 6.

$\mathsf{Gen}\left(1^{\lambda}, q\right)$: Run DS.$\mathsf{Gen}\left(1^{\lambda}\right)$ to obtain key pair (sk_s, vk_s) and run the vector commitment schemes key generator function to obtain pp for a linear document of size q message blocks. Output the verification key $vk = (vk_s, \mathrm{pp})$ and the signing key $sk = (sk_s, \mathrm{pp})$.

$\mathsf{Sign}_{sk}\left(\mathcal{M}, P\right)$: If $P \notin \mathcal{P}$, return $\perp$. Otherwise, parse sk as (sk_s, pp); sample an $s \in \mathbb{Z}^*$, parse $\mathcal{M}$ as $(m_1, m_2, \ldots, m_q)$, construct the tree $\mathcal{T}$ associated with the policy P, compute $(C, aux) \leftarrow \mathrm{VC}.\mathsf{Com}_{\mathrm{pp}}\left(m_1, m_2, \ldots, m_q\right)$, and return the signature,

$$\begin{aligned}\sigma_{\mathcal{S}} \;=\; \Big(&C, \mathcal{T}, \left\{\mathsf{share}\left(q_{m_i}\left(0\right)\right)\right\}_{i\in[q]}, \\ &\left\{\mathrm{VC}.\mathsf{Open}_{pp}\left(m_i \parallel \mathsf{share}\left(q_{m_i}\left(0\right)\right), i, aux\right)\right\}_{i\in[q]}, \\ &\mathsf{Sign}_{sk_s}\left(C \parallel s\right)\Big).\end{aligned}$$

Notationally, we assume that all message blocks in the sequence appear in $\mathcal{T}$ that is, however, not always the case. Message blocks m that do not appear in $\mathcal{T}$ should have their share set to a uniformly random sampled element of $\mathbb{Z}^*$.

$\mathsf{Redact}_{vk}\left(\mathcal{M}, \sigma, \mathcal{M}', P'\right)$: Parse σ as $\left(C, \mathcal{T}, \left\{s_m\right\}_{m\in\mathcal{M}}, \left\{w_{m\parallel s_m}\right\}_{m\in\mathcal{M}}, \sigma^*\right)$. Extract the policy P from $\mathcal{T}$ if $\mathcal{M} \vdash_P \mathcal{M}'$ and P' is a subpolicy of P where $P'\left(\mathcal{M}'\right) = 1$, output the signature $\sigma_{\mathcal{M}'} = \left(C, \mathcal{T}', \left\{s_m\right\}_{m\in\mathcal{M}'}, \left\{\Pi_{m\parallel s_m}\right\}_{m\in\mathcal{M}'}, \sigma^*\right)$, where $\mathcal{T}'$ is associated with P'. Otherwise, output $\perp$.

$\mathsf{Vrfy}_{vk}\left(\mathcal{M}, \sigma\right)$: Parse vk as (vk_s, pp) and σ as $\left(C, \mathcal{T}, \left\{s_m\right\}_{m\in\mathcal{M}}, \left\{w_{m\parallel s_m}\right\}_{m\in\mathcal{M}}, \sigma^*\right)$. Return one if for all $m_i \in \mathcal{M}$, $\mathrm{VC}.\mathsf{Ver}_{\mathrm{pp}}\left(\Pi_{C, m_i\parallel s_{m_i}}, i, m_i \parallel s_{m_i}\right) = 1$ and DS.$\mathsf{Vrfy}_{vk_s}\left(C \parallel s^*, \sigma^*\right) = 1$, where s^* results from reconstructing the secret using the shares together with policy tree $\mathcal{T}$; otherwise, return zero.

Figure 6: The monotone circuit linear RSS scheme.

6 Implementation

We've implemented our arbitrary universe construction as well as our small universe accumulator based construction in Java using a pairing based accumulator scheme from [32] using the Java Pairing Based

Cryptography library [15]. It turns out that standard RSA accumulators like [2] are too inefficient for our purposes (e.g., signing a set of 300 elements under the large construction took close to one minute). This is generally attributed to the cost of hashing to a prime number. The practical performance of various accumulators has been discussed in [24]. We utilized EC-DSA with curve P-256 as our underlying signature scheme. When needed, the threshold secret sharing scheme is Shamir's scheme [30]. Our performance analysis was done using the Apache Netbeans Performance Analyzer. We made our source code available at `https://github.com/kisselz/redactable-sigs`.

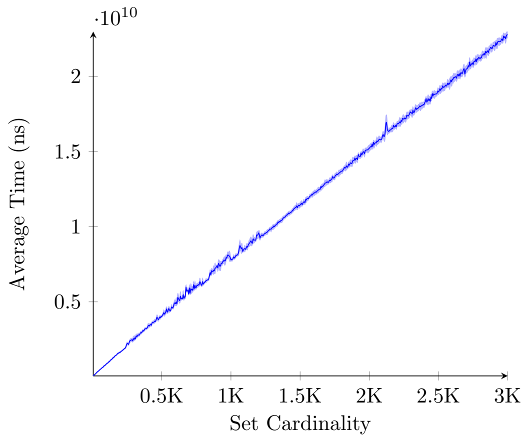
Existing constructions in the literature lack implementation, thus comparison between them is purely theoretical. In practice, our implementation indicates that a majority of the time in our set-based constructions, and other accumulator-based constructions, is spent performing accumulator operations. For all constructions when generating the RSS signing and verification keys a majority of the time was (unsurprisingly) spent in the accumulator code, specifically in loading the group information for our pairing friendly group.

Arbitrary Universe Construction The time in signing is dominated by the generation of accumulator witnesses. This is in turn dominated by the cost of exponentiation in the underlying pairing friendly group. The time for redaction is not dominated by any one factor. Verification is dominated by the verification of accumulator witnesses, which is due to the cost of two pairing operations.

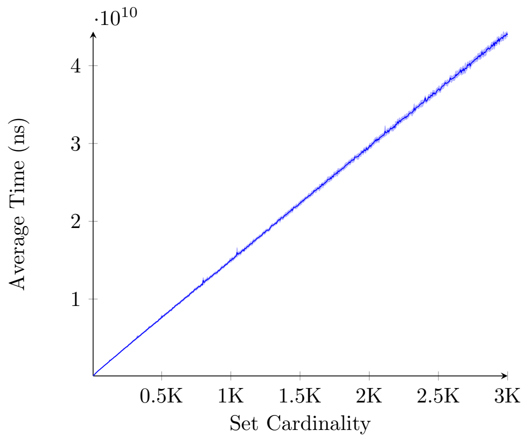
Accumulator-Based Small Universe The time in signing is dominated by the generation of accumulator witnesses. This is in turn dominated by the cost of exponentiating in the underlying pairing friendly group. The time for redaction is not dominated by any one factor. Verification is dominated by the verification of accumulator witnesses, which is due to the cost of the two pairing operations.

Set Construction of Derler et. al We implemented the set based construction of Derler et. al [17], which lacks a policy but, otherwise uses similar cryptographic primitives. This scheme is essentially the accumulator-based small universe construction except we accumulate the elements instead of the characteristic sequences. Since the same underlying instantiation of the primitives are used, the dominating factors in key generation, signing, verification, and redaction mirror the accumulator based construction above.

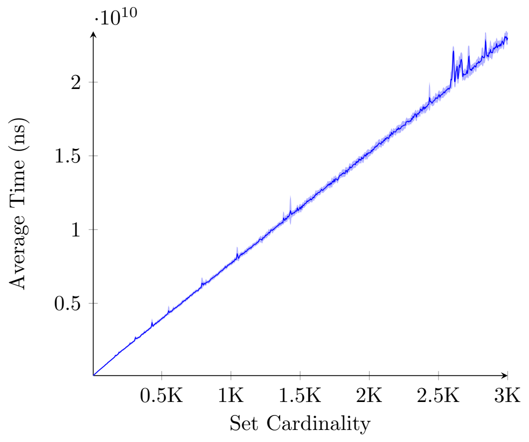
Comparative Analysis We ran our large universe construction and the construction of Derler et. al with various sized sets on a Linux machine with an Intel i7-10750H CPU and 32G of RAM running Fedora 36 (Kernel version 5.19.15-201). Our elements were drawn from the standard Linux word list found on the machine. The sign and verify operations were run 10 times on each set size ($5 - 3000$ elements in 5 element increments). See figure 7 for our performance graphs, the shaded regions indicate one standard deviation above and below the average (solid line). It should be noted that we did not perform an analysis with our small universe construction. This is because the number of accumulator witnesses grows exponentially in the number of elements in the set being signed. Thus, the construction becomes inefficient quickly. From figures 7a and 7c we observe that the addition of the policy only negligibly degrades the performance of the standard redactable set signature scheme of Derler et. al. The sporadic spikes in the graphs can be attributed to system noise over the duration of the test interval. In figures 7b and 7d we observe that the addition of a policy minimally impacts the verification time. The performance of the secret reconstruction in the large construction is aided by the fact that the formulas are encoded in disjunctive normal form, which means the entire tree does not need to be evaluated in order to recover the associated secret. Since any formula not in disjunctive normal form can be rewritten in disjunctive normal form, this does not constrain our analysis. Again, the random spikes in the graphs can be attributed to system noise. We note that in general verification time is approximately twice as long as signing time, this is due to the cost of pairing operations present in the accumulator verification process. We did not collect test data from the redact operations since in both constructions, no cryptographic operations are involved during redaction.



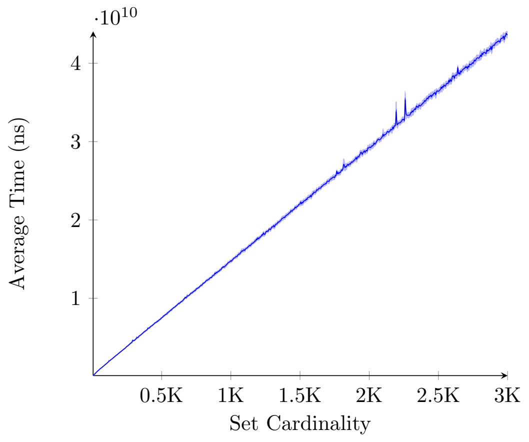
(a) The average time to sign in nanoseconds for the construction of Derler et. al.



(b) The average time to verify in nanoseconds for the construction of Derler et. al.



(c) The average time to sign in nanoseconds for our arbitrary universe construction. The policy size was held constant.



(d) The average time to verify in nanoseconds for our arbitrary universe construction. The policies are in disjunctive normal form.

Figure 7: Performance graphs for sign and verify. The shaded regions indicate the area falling within one standard deviation.

7 Conclusion

In this work we demonstrated constructions of policy-based redactable signature schemes. We further provided implementations to aid in justifying their practicality for smaller set sizes. Since in both our constructions the time grows linear with the accumulator operations, either another technique or faster accumulators must be used. Similarly, our construction for linear documents from vector commitments suffers from the fact that time grows linear with the number of committed values.

Open Questions Our work leaves several open questions around both policy-based redactable set signatures as well as for other data structures like linear and tree based. In terms of set based constructions we would like to support much larger sets, which will require faster accumulators or vastly different techniques. A natural idea is to use batch accumulators; however, it is not clear how to support policy and batching simultaneously other than in a trivial manner (e.g., creating batch witnesses for all allowable redactions).

As for our linear constructions we would similarly need to seek out different techniques from vector commitment or potentially leverage subvector commitments [25], or possibly point proofs [19], similar to the trivial batching of accumulator witnesses for set constructions discussed above. In the case of Pointproofs, the production of an aggregate proof can be performed by a third party thus potentially allowing the third party to shrink signature verification time by first aggregating proofs before verifying.

References

[1] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. *Journal of Cryptology*, 28(2):351–395, 2015.

[2] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *International conference on the theory and applications of cryptographic techniques*, pages 480–494. Springer, 1997.

[3] Josh Benaloh and Michael De Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 274–285. Springer, 1993.

[4] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007.

[5] Arne Bilzhause, Henrich C Pöhls, and Kai Samelin. Position paper: the past, present, and future of sanitizable and redactable signatures. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–9, 2017.

[6] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, jul 1970.

[7] Dan Boneh, Kevin Lewi, and David J Wu. Constraining pseudorandom functions privately. In *IACR International Workshop on Public Key Cryptography*, pages 494–524. Springer, 2017.

[8] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained prfs (and more) from lwe. In *Theory of Cryptography Conference*, pages 264–302. Springer, 2017.

[9] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions. In *Theory of Cryptography Conference*, pages 1–30. Springer, 2015.

[10] Christina Brzuska, Heike Busch, Oezguer Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, et al. Redactable signatures for tree-structured data: Definitions and constructions. In *International Conference on Applied Cryptography and Network Security*, pages 87–104. Springer, 2010.

[11] Dario Catalano and Dario Fiore. Vector commitments and their applications. In *International Workshop on Public Key Cryptography*, pages 55–72. Springer, 2013.

[12] Ee-Chien Chang, Chee Liang Lim, and Jia Xu. Short redactable signatures using random trees. In *Cryptographers' Track at the RSA Conference*, pages 133–147. Springer, 2009.

[13] JG Clerry. Compact hash tables using bidirectional linear probing. *IEEE Transactions on Computers*, 100(9):828–834, 1984.

[14] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, and Shota Yamada. Constrained prfs for bit-fixing from owfs with constant collusion resistance. Technical report, IACR Cryptology ePrint Archive, 2018.

[15] Angelo De Caro and Vincenzo Iovino. jpbc: Java pairing based cryptography. In *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, pages 850–855, Kerkyra, Corfu, Greece, June 28 - July 1, 2011. IEEE.

[16] David Derler, Christian Hanser, and Daniel Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In *Cryptographers' track at the rsa conference*, pages 127–144. Springer, 2015.

[17] David Derler, Henrich C Pöhls, Kai Samelin, and Daniel Slamanig. A general framework for redactable signatures and new constructions. In *ICISC 2015*, pages 3–19. Springer, 2015.

[18] Bin Fan, Dave G Andersen, Michael Kaminsky, and Michael D Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 75–88, 2014.

[19] Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2007–2023, 2020.

[20] Thomas Mueller Graf and Daniel Lemire. Xor filters: Faster and smaller than bloom and cuckoo filters. *Journal of Experimental Algorithmics (JEA)*, 25:1–16, 2020.

[21] Stuart Haber, Yasuo Hatano, Yoshinori Honda, William Horne, Kunihiko Miyazaki, Tomas Sander, Satoru Tezoku, and Danfeng Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 353–362, 2008.

[22] Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *Cryptographers' track at the RSA conference*, pages 244–262. Springer, 2002.

[23] Zachary A Kissel. Constrained pseudorandom functions from pseudorandom synthesizers. Cryptology ePrint Archive, Paper 2022/897, 2022. `https://eprint.iacr.org/2022/897`.

[24] Amrit Kumar, Pascal Lafourcade, and Cédric Lauradoux. Performances of cryptographic accumulators. In *39th Annual IEEE Conference on Local Computer Networks*, pages 366–369. IEEE, 2014.

[25] Russell WF Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In *Annual International Cryptology Conference*, pages 530–560. Springer, 2019.

[26] Kunihiko Miyazaki, Mitsuru Iwamura, Tsutomu Matsumoto, Ryôichi Sasaki, Hiroshi Yoshiura, Satoru Tezuka, and Hideki Imai. Digitally signed document sanitizing scheme with disclosure condition control. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 88(1):239–246, 2005.

[27] Ilker Ozcelik, Sai Medury, Justin Broaddus, and Anthony Skjellum. An overview of cryptographic accumulators. *arXiv preprint arXiv:2103.04330*, 2021.

[28] Yongjun Ren, Xinyu Liu, Qiang Wu, Ling Wang, and Weijian Zhang. Cryptographic accumulator and its application: A survey. *Security and Communication Networks*, 2022, 2022.

[29] Olivier Sanders. Efficient redactable signature and application to anonymous credentials. In *IACR International Conference on Public-Key Cryptography*, pages 628–656. Springer, 2020.

[30] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[31] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In *International Conference on Information Security and Cryptology*, pages 285–304. Springer, 2001.

[32] Giuseppe Vitto and Alex Biryukov. Dynamic universal accumulator with batch update over bilinear groups. In *Cryptographers' Track at the RSA Conference*, pages 395–426. Springer, 2022.

A Security Proofs for the AMQ Small Universe Construction

We begin by showing that the AMQ-based construction is unforgeable. This roughly relies on the underlying unforgeablility of the signature scheme. Our proof is via a standard reduction of an adversary breaking the EUF-CMA security game to an adversary that wins the EUF-CMA-RSS game. In what follows, we assume that Π_{sig} is an EUF-CMA secure signature scheme and Π is our RSS.

Proof. Let $\mathcal{A}_{\text{rss}}$ be an PPT adversary that wins the EUF-CMA-RSS game. We construct an adversary $\mathcal{A}_{\text{sig}}$ that uses $\mathcal{A}_{\text{rss}}$ to win the EUF-CMA game. The code for $\mathcal{A}_{\text{sig}}$ is:

```
$\mathcal{A}_{\text{sig}}^{\mathcal{O}_{\text{sig}}(\cdot)}(vk)$:
  1. Initialize $\mathcal{Q} = \emptyset$.
  2. Run $\mathcal{A}_{\text{rss}}(vk)$ answering signature queries as follows:
     (a) $P \notin \mathcal{P}$ output, $\perp$.
     (b) $\mathcal{F} \leftarrow \mathsf{Init}(fpr)$
     (c) $k \leftarrow \text{CPRF}.\mathsf{Gen}(1^\lambda)$.
     (d) For every $A \subseteq \mathcal{S}$ where $P(A) = 1$, run $F \leftarrow \mathsf{Add}(\text{CPRF}.\mathsf{Eval}(k, \mathcal{X}_{\mathcal{U}}(A)), \mathcal{F})$.
     (e) Return $\sigma_{\mathcal{S}} = (\mathcal{F}, k, \mathcal{O}_{\text{sig}}(\mathcal{F}))$, adding $(P, \mathcal{S}, \sigma_{\mathcal{S}})$ to $\mathcal{Q}$.
  3. Eventually $\mathcal{A}_{\text{rss}}$ outputs $(\mathcal{S}^*, (\mathcal{F}^*, k^*, \sigma^*))$ if there does not exist a $\mathcal{S} \in \mathcal{Q}$ such that $\mathcal{S} \vdash_P \mathcal{S}^*$,
     output $(\mathcal{F}^*, \sigma^*)$ as the forgery.
```

Observe that $\mathcal{A}_{\text{sig}}$ faithfully simulates the view of adversary $\mathcal{A}_{\text{rss}}$ in the EUF-CMA-RSS game. Thus we can conclude that

$$\Pr\left[\mathsf{Exp}_{\mathcal{A}_{\text{rss}},\Pi}^{\text{EUF-CMA-RSS}}(\lambda) = 1\right] = \Pr\left[\mathsf{Exp}_{\mathcal{A}_{\text{sig}},\Pi_{\text{sig}}}^{\text{EUF-CMA}}(\lambda) = 1\right].$$

By assumption we have that $\Pr\left[\mathsf{Exp}_{\mathcal{A}_{\text{sig}},\Pi_{\text{sig}}}^{\text{EUF-CMA}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)$. Thus we conclude that $\Pr\left[\mathsf{Exp}_{\mathcal{A}_{\text{rss}},\Pi'}^{\text{EUF-CMA-RSS}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)$ and our scheme is unforgeable. □

In order to argue privacy we will prove our construction has the transparency property and thus inherit the fact that the construction is private as well. Our proof will proceed via a sequence of games.

Proof. Let W_i denote the event that the adversary wins game i.

Game 0. This is the original game.

Game 1. This is similar to game 0 but, every call to the $\mathsf{SignRedact}$ oracle has b fixed to 0.

We claim that $|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{negl}(\lambda)$. Observe, the only way to distinguish between the two games is if you can distinguish between CPRF keys, which are indistinguishable due to the fact that the CPRF is canonical. Since, there are at most $q(\lambda)$ queries we have that $|\Pr[W_0] - \Pr[W_1]| \leq q(\lambda)\,\mathsf{negl}(\lambda) \leq \mathsf{negl}'(\lambda)$. Since the behavior of the oracle is independent of b in game 1, we have the $\Pr[W_1] = \frac{1}{2}$. Therefore, $|\Pr[W_0] - \Pr[W_1]| \leq \mathsf{negl}(\lambda) \implies \left|\Pr[W_0] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda) \implies \Pr[W_0] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. □

B Security Proofs for the Arbitrary Universe Construction

Our arbitrary universe construction is both unforgeable and transparent. The unforgeability of the scheme, stems from the unforgeability of the underlying signature scheme as well as the collision resistance of the cryptographic accumulator. In what follows we will let Π denote our RSS, Π_{sig} denote our underlying signature scheme, and Π_{acc} our underlying cryptographic accumulator scheme.

Proof. Assume by way of contradiction there exists an adversary $\mathcal{A}_{rss}$ that can win the RSS unforgeability game with greater than negligible probability. Given such an adversary we show how to construct an adversary $\mathcal{A}_{Sig}$ that utilizes $\mathcal{A}_{rss}$ to contradict the EUF-CMA security of the underlying signature scheme and an adversary $\mathcal{A}_{acc}$ that utilizes $\mathcal{A}_{rss}$ to contradict the unforgeablility of the underlying cryptographic accumulator.

We begin by constructing the adversary $\mathcal{A}_{sig}$ as follows:

```
$\mathcal{A}_{\text{sig}}^{\mathcal{O}_{\text{sig}}(\cdot)}(vk)$:
  1. Initialize $\mathcal{Q} = \emptyset$.
  2. Run $(sk_a, pk_a) \leftarrow \text{Acc.}\mathsf{Gen}\left(1^\lambda\right)$.
  3. Run $\mathcal{A}_{\text{rss}}\left((vk, pk_a)\right)$ answering signature queries as follows:
     (a) Operate as Sign except when the signature needs to be computed, query $\mathcal{O}_{\text{sig}}$.
     (b) Compute $acc \leftarrow \mathsf{Eval}\left((sk_a, pk_a), \mathcal{S}\right)$
     (c) Return
           $\sigma_{\mathcal{S}} = \left(acc, \mathcal{T}, \{\mathsf{share}(q_x(0))\}_{x\in\mathcal{S}}, \{\mathsf{Wit}_{sk_a}(x \parallel \mathsf{share}(q_x(0)), acc)\}_{x\in\mathcal{S}}, \mathcal{O}_{\text{Sig}}(acc \parallel s)\right).$
         adding $(P, \mathcal{S}, \sigma_{\mathcal{S}})$ to $\mathcal{Q}$.
  4. Eventually $\mathcal{A}_{\text{rss}}$ outputs $\left(\mathcal{S}^*, \left(acc^*, \mathcal{T}^*, \{s^*_x\}_{x\in\mathcal{S}^*}, \left\{w^*_{x,s_x}\right\}_{x\in\mathcal{S}^*}, \sigma^*\right)\right)$ if there does not exist a
     $\mathcal{S} \in \mathcal{Q}$ such that $\mathcal{S} \vdash_P \mathcal{S}^*$, output $(acc^* \parallel s^*, \sigma^*)$, where $s^*$ was reconstructed according to $\mathcal{T}^*$,
     as the forgery.
```

Observe that $\mathcal{A}_{\text{sig}}$ faithfully simulates the view of adversary $\mathcal{A}_{\text{rss}}$ in the EUF-CMA-RSS game.

We now construct the adversary $\mathcal{A}_{acc}$ as follows:

$\mathcal{A}_{\mathrm{acc}}^{\mathcal{O}_{\mathsf{Wit}}(\cdot),\mathcal{O}_{\mathsf{Eval}}(\cdot)}(pk)$:

1. Initialize $\mathcal{Q} = \emptyset$.
2. Run $(sk, vk) \leftarrow \mathrm{DS}.\mathsf{Gen}\left(1^\lambda\right)$.
3. Run $\mathcal{A}_{\mathrm{rss}}\left((vk, pk)\right)$ answering signature queries as follows:
   (a) Operate as Sign except when the accumulator work needs to be done, hand the computation off to $\mathcal{O}_{\mathsf{Eval}}$ and $\mathcal{O}_{\mathsf{Wit}}$.
   (b) Return

$$\sigma_{\mathcal{S}} = \big(\mathcal{O}_{\mathsf{Eval}}\left(\{x \parallel \mathsf{share}\left(q_x\left(0\right)\right)\}_{x\in\mathcal{S}}\right), \mathcal{T}, \{\mathsf{share}\left(q_x\left(0\right)\right)\}_{x\in\mathcal{S}}, \\ \{\mathcal{O}_{\mathsf{Wit}}\left(x \parallel \mathsf{share}\left(q_x\left(0\right)\right)\right)\}_{x\in\mathcal{S}}, \mathsf{Sign}_{sk}\left(acc \parallel s\right)\big).$$

   adding $(P, \mathcal{S}, \sigma_{\mathcal{S}})$ to $\mathcal{Q}$.
4. Eventually $\mathcal{A}_{\mathrm{rss}}$ outputs $\left(\mathcal{S}^*, \left(acc^*, \mathcal{T}^*, \{s_x^*\}_{x\in\mathcal{S}^*}, \left\{w_{x,s_x}^*\right\}_{x\in\mathcal{S}^*}, \sigma^*\right)\right)$ such that there does not exist a $\mathcal{S} \in \mathcal{Q}$ such that $\mathcal{S} \vdash_P \mathcal{S}^*$. If there does not exist a signature on $acc^* \parallel s^*$, where s^* was reconstructed according to $\mathcal{T}^*$ in $\mathcal{Q}$, abort. Otherwise there is at least one witness w_{x,s_x}^* such that $\mathrm{Acc}.\mathsf{Vrfy}_{pk}\left(acc^*, w_{x,s_x}^*, x\right) = 1$ but $x \notin acc^*$, output the collision $\left(w_{x,s_x}^*, x, acc^*\right)$.

Observe that $\mathcal{A}_{\mathrm{acc}}$ faithfully simulates the view of adversary $\mathcal{A}_{\mathrm{rss}}$ in the EUF-CMA-RSS game if it does not abort. The only way we abort is if the adversary tries to also forge the signature.

Note that there is no way for an adversary to construct new shares for any entries that allow the party to recover s unless the adversary could forge witnesses for the cryptographic accumulator. Moreover, the value of s can't be changed without being able to break the EUF-CMA security of the underlying signature scheme. Since neither adversary outlined above can succeed, we have by the union bound

$$\Pr\left[\mathsf{Exp}_{\mathcal{A}_{\mathrm{rss}},\Pi}^{\mathrm{EUF-CMA-RSS}}(\lambda) = 1\right] \leq \Pr\left[\mathsf{Exp}_{\mathcal{A}_{\mathrm{sig}},\Pi_{\mathrm{sig}}}^{\mathrm{EUF-CMA}}(\lambda) = 1\right] + \Pr\left[\mathsf{Exp}_{\mathcal{A}_{\mathrm{acc}},\Pi_{\mathrm{acc}}}^{\mathrm{Acc-Coll-Free}}(\lambda) = 1\right] \\ \leq \mathsf{negl}(\lambda)$$

thus we conclude that our scheme is unforgeable. □

Our scheme is transparent and thus private. The transparency of our scheme relies on the indistinguishability of the accumulator as well as security of the underlying threshold signature scheme. We proceed to prove our claim through a sequence of games approach.Let W_i denote the event that the adversary wins game i.

Game 0. This is the original game.

Game 1. This is similar to game 0 but, every call to the $\mathsf{SignRedact}$ oracle with b fixed to 0.

Claim B.1. *The adversary will detect the changes from game 0 to game 1 with at most negligible probability.*

Proof. There are two key changes between the games when $b = 1$ in Game 0. The accumulator value has more values accumulated in the accumulator in Game 1 than Game 0. The policies will remain identical and the shares will remain indistinguishable since the secrets are uniformly random. Since the accumulator has the indistinguishability property after k queries there is $k\mathsf{negl}(\lambda)$ probability of distinguishing based on the accumulator. Since, k is polynomial in the security parameter, we have that the probability of distinguishing between the two games is negligible. □

Claim B.2. *The probability the adversary wins Game 0 is at most negligibly more than one half.*

Proof. Observe that in Game 1 the value of the accumulator is independent of the choice of bit b (as the oracle always behaves as if $b = 0$). The remaining components of the signature are identically distributed. This means that $\Pr[W_1] = \frac{1}{2}$ thus,

$$\begin{aligned} &|\Pr[W_0] - \Pr[W_1]| \leq k\mathsf{negl}(\lambda) \\ \implies &\left|\Pr[W_0] - \frac{1}{2}\right| \leq k\mathsf{negl}(\lambda) \\ \implies &\Pr[W_0] \leq \frac{1}{2} + k\mathsf{negl}(\lambda) \end{aligned}$$

□

C Security Proofs for the Linear Construction

Our arbitrary universe linear construction is both unforgeable and private. The unforgeability of the scheme, stems from the unforgeability of the underlying signature scheme as well as the position binding property of the vector commitment scheme. In what follows we will let Π denote our RSS, Π_{sig} denote our underlying signature scheme, and Π_{vc} our underlying vector commitment scheme.

Proof. Assume by way of contradiction there exists an adversary $\mathcal{A}_{rss}$ that can win the RSS unforgeability game with greater than negligible probability. Given such an adversary we show how to construct an adversary $\mathcal{A}_{Sig}$ that utilizes $\mathcal{A}_{rss}$ to contradict the EUF-CMA security of the underlying signature scheme and an adversary $\mathcal{A}_{vc}$ that utilizes $\mathcal{A}_{rss}$ to contradict the position binding property of the underlying vector commitment.

We begin by constructing the adversary $\mathcal{A}_{sig}$ as follows:

$\mathcal{A}_{\text{sig}}^{\mathcal{O}_{\text{sig}}(\cdot)}(vk)$:

1. Initialize $\mathcal{Q} = \emptyset$.
2. Run $\text{pp} \leftarrow \text{VC}.\mathsf{KeyGen}\left(1^\lambda, q\right)$.
3. Run $\mathcal{A}_{\text{rss}}((vk, \text{pp}))$ answering signature queries as follows:
   (a) Operate as Sign except when the signature needs to be computed, query $\mathcal{O}_{\text{sig}}$.
   (b) Compute $(C, aux) \leftarrow \text{VC}.\mathsf{Com}_{pp}(m_1, m_2, \ldots, m_q)$.
   (c) Return

$$\sigma_{\mathcal{M}} = \Big(C, \mathcal{T}, \{\mathsf{share}(q_m(0))\}_{m \in \mathcal{M}}, \left\{\text{VC}.\mathsf{Open}_{pp}(m_i \parallel \mathsf{share}(q_{m_i}(0)), i, aux)\right\}_{i \in [q]}, \mathcal{O}_{\text{Sig}}(C \parallel s)\Big)$$

   adding $(P, \mathcal{M}, \sigma_{\mathcal{M}})$ to $\mathcal{Q}$.
4. Eventually $\mathcal{A}_{\text{rss}}$ outputs $\left(\mathcal{M}^*, \left(C^*, \mathcal{T}^*, \{s_m^*\}_{m \in \mathcal{M}^*}, \left\{\Pi_{m,s_m}^*\right\}_{m \in \mathcal{M}^*}, \sigma^*\right)\right)$ if there does not exist a $\mathcal{S} \in \mathcal{Q}$ such that $\mathcal{S} \vdash_P \mathcal{S}^*$, output $(acc^* \parallel s^*, \sigma^*)$, where s^* was reconstructed according to $\mathcal{T}^*$, as the forgery.

Observe that $\mathcal{A}_{\text{sig}}$ faithfully simulates the view of adversary $\mathcal{A}_{\text{rss}}$ in the EUF-CMA-RSS game.

We now construct the adversary $\mathcal{A}_{\text{vc}}$ as follows:

$\mathcal{A}_{\text{vc}}(pk)$:

```
1. Initialize $\mathcal{Q} = \emptyset$.
2. Run $(sk, vk) \leftarrow \text{DS}.\mathsf{Gen}(1^\lambda)$.
3. Run $\text{pp} \leftarrow \text{VC}.\mathsf{KeyGen}1^\lambda, q$.
4. Run $\mathcal{A}_{\text{rss}}((vk, \text{pp}))$ answering signature queries as follows:
    (a) Operate as $\mathsf{Sign}$ including handling all commitments.
    (b) Compute $(C, aux) \leftarrow \text{VC}.\mathsf{Com}_{pp}(m_1, m_2, \ldots, m_q)$.
    (c) Return
            $\sigma_{\mathcal{M}} = \Big(C, \mathcal{T}, \{\mathsf{share}(q_m(0))\}_{m \in \mathcal{M}}, \big\{\text{VC}.\mathsf{Open}_{pp}(m_i \parallel \mathsf{share}(q_{m_i}(0)), i, aux)\big\}_{i \in [q]},$
                $\mathcal{O}_{\text{Sig}}(C \parallel s))$
        adding $(P, \mathcal{M}, \sigma_{\mathcal{M}})$ to $\mathcal{Q}$.
5. Eventually $\mathcal{A}_{\text{rss}}$ outputs $\Big(\mathcal{M}^*, \Big(C^*, \mathcal{T}^*, \{s^*_m\}_{m \in \mathcal{M}^*}, \big\{\Pi^*_{m,s_m}\big\}_{m \in \mathcal{M}^*}, \sigma^*\Big)\Big)$ such that there does
   not exist a $\mathcal{M} \in \mathcal{Q}$ such that $\mathcal{M} \vdash_P \mathcal{M}^*$. If there does not exist a signature on
   $acc^* \parallel s^*$, where $s^*$ was reconstructed according to $\mathcal{T}^*$ in $\mathcal{Q}$, abort. Let the message
   block sequence associated with the overlapping signature be $\mathcal{M}'$. There is at least one proof
   $\Pi^*_{m_i, s_{m_i}}$ such that $\text{VC}.\mathsf{Ver}_{\text{pp}}\Big(C^*, m_i \parallel s_{m_i}, i, \Pi^*_{m_i, s_{m_i}}\Big) = 1$ but $m_i \notin C^*$, output the violation
   $\Big(C^*, m_i, m'_i, i, \Pi^*_{m_i, s_{m_i}}, \Pi_{m'_i, s_{m'_i}}\Big)$.
```

Observe that $\mathcal{A}_{\text{vc}}$ faithfully simulates the view of adversary $\mathcal{A}_{\text{rss}}$ in the EUF-CMA-RSS game if it does not abort. The only way we abort is if the adversary tries to also forge the signature.

Note that there is no way for an adversary to construct new shares for any entries that allow the party to recover s unless the adversary could forge proofs for the vector commitment. Moreover, the value of s can't be changed without being able to break the EUF-CMA security of the underlying signature scheme. Since neither adversary outlined above can succeed, we have by the union bound

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\text{EUF-CMA-RSS}}_{\mathcal{A}_{\text{rss}},\Pi}(\lambda) = 1\right] &\leq \Pr\left[\mathsf{Exp}^{\text{EUF-CMA}}_{\mathcal{A}_{\text{sig}},\Pi_{\text{sig}}}(\lambda) = 1\right] + \Pr\left[\mathsf{Exp}^{\text{Pos-Bind}}_{\mathcal{A}_{\text{vc}},\Pi_{\text{vc}}}(\lambda) = 1\right] \\ &\leq \mathsf{negl}(\lambda)\end{aligned}$$

thus we conclude that our scheme is unforgeable. □

Our scheme is private provided the underlying vector commitment scheme has the hiding property. The privacy of our scheme relies on the fact that if the messages $\mathcal{M}_0$ and $\mathcal{M}_1$ are position-wise equivalent for non-redacted components, then the two redacted signatures will be indistinguishable. We proceed to prove our claim through a sequence of games approach.Let W_i denote the event that the adversary wins game i.

Game 0. This is the original game.

Game 1. This is similar to game 0 but, every call to the $\mathsf{LoRRedact}$ oracle with b fixed to 0.

Claim C.1. *The adversary will detect the changes from game 0 to game 1 with at most negligible probability.*

Proof. Observe that since the two redacted messages are equivalent and the commitment scheme is hiding, no information is revealed about the message sequence $\mathcal{M}$ that was redacted. Therefore the adversaries power in distinguishing between the two games is negligible. Since we have at most k queries the probability of distinguishing is at most $k\mathsf{negl}(\lambda)$. □

Claim C.2. *The probability the adversary wins Game 0 is at most negligibly more than one half.*

Proof. Observe that in Game 1 the value of the commitment is independent of the choice of bit b (as the oracle always behaves as if $b = 0$). The remaining components of the signature are identically distributed. This means that $\Pr[W_1] = \frac{1}{2}$ thus,

$$\begin{aligned} &\left|\Pr[W_0] - \Pr[W_1]\right| \leq k\mathsf{negl}(\lambda) \\ \implies &\left|\Pr[W_0] - \frac{1}{2}\right| \leq k\mathsf{negl}(\lambda) \\ \implies &\Pr[W_0] \leq \frac{1}{2} + k\mathsf{negl}(\lambda) \end{aligned}$$

□