Efficient Transparent Redactable Signatures with a Single Signature Invocation[4]

Stuart Haber[1], William Horne[2], and Miaomiao Zhang[3]

[1] Hewlett Packard Enterprise, Princeton, NJ 08550, USA,
stuart.haber@hpe.com
[2] Intertrust Technologies, Sunnyvale, CA 94085, USA
bill.horne@intertrust.com
[3] Stevens Institute of Technology, Hoboken, NJ 07030, USA,
miaomiaozhang@gmail.com
[4] Work performed by all authors while associated with HP Laboratories.

Abstract. A *redactable* signature scheme is one that allows the original signature to be used, usually along with some additional data, to verify certain carefully specified changes to the original document that was signed, namely the removal or *redaction* of subdocuments. For redactable signatures, the term *transparency* has been used to describe a scheme that hides the number and locations of redacted subdocuments. We present here two efficient transparent redactable signature schemes, which are the first such schemes in the literature that are based solely on tools of symmetric cryptography, along with a single application of an ordinary digital signature.

As with several previous schemes for redactable signatures, we sign a sequence of randomized commitments that depend on the contents of the subdocuments of the document to be signed. In order to hide their number and location, we randomize their order, and mix them with a sequence of "dummy nodes" that are indistinguishable from commitment values. Our first scheme uses a data structure of size quadratic in the number of subdocuments, encoding all the precedence relations between pairs of subdocuments. By embedding these precedence relations in a smaller family of graphs, our second scheme is more efficient, with expected cost linear in the number of subdocuments in the document to be signed. We introduce a quantified version of the transparency property, precisely describing the uncertainty about the number of redacted subdocuments that is guaranteed by the two schemes.

We prove that our schemes are secure, i.e. unforgeable, private, and transparent, based on the security of collision-free hash functions, pseudorandom generators, and digital signature schemes. While providing such strong security, our scheme is also efficient, in terms of both computation and communication.

1 Introduction

Motivation Traditional digital signature schemes can be used to check—among other properties—that integrity of data is maintained, i.e. the data has not been modified since it was signed. However, in some scenarios modification of the data is not only allowable, but also desirable.

Several governments in the world have analogues to the US Freedom of Information Act (FOIA). Under this act, formerly confidential documents can be released to the public. Often, before a document is released certain words containing sensitive information, such as names of individuals or national secrets, are *redacted*, i.e. removed or blacked out.

For electronic documents, redaction is problematic when it is also important to protect the authenticity and integrity of the document. Conventional digital signatures and time-stamping schemes can be used to prove the integrity of the original data, and they are meant to identify all changes to the document as invalid. So the problem is to devise a scheme that can be used to attest to the integrity of correctly redacted versions of the original document and no other versions.

As with conventional schemes, it should be computationally infeasible to forge improper signatures or certificates. Specifically, it should be difficult to compute a valid signature or certificate for any document other than one that is a legitimately redacted version of the original. In addition, all information about redacted subdocuments should be hidden.

In certain settings, it is furthermore desirable to hide the length and even the existence of any redacted subdocuments. For example, when a name (e.g. that of a person, a place, or an official agency) is redacted,

revealing the length of the redacted portion may allow a viewer of the redacted version of the document to infer the missing name. A newsworthy demonstration of exactly this vulnerability was given by the authors of [NW04], as widely reported by [Mar04]. Another example is provided by the case of electronic health records in a system with a standard schema for the format of the records, e.g. XML. In this case, the very existence of a redacted sub-record in a particular position of an individual patient's record may attest to the patient having a particular diagnosis or medical condition. Previous authors have considered this property, most often calling it *transparency* [ACMT05, MHI06, BFF+09, CLX09, BBD+10, MPPS14]. Even though "transparent" may not be the best one-word description of the property, we follow previous authors in doing so.

Several prior schemes for transparent redactable signatures make use of aggregate signature schemes, based on techniques from pairing-based cryptography using bilinear maps. Other schemes for certain variations on redactable signature schemes make use of cryptographic accumulators. Still other schemes use many invocations of an ordinary digital signature scheme, with the number of invocations at least linear in the number of subdocuments. Inherent in all of these classes of techniques is the limitation that every subdocument of the original document (or a value that depends on it) is used as input to a relatively expensive number-theoretic or algebraic computation.

In this paper we are interested in designing a redactable signature scheme that hides the existence of redacted subdocuments, using only the tools of symmetric cryptography, along with a single invocation of an ordinary digital signature or time-stamping scheme.

Our security proofs require no further number-theoretic or algebraic assumptions.

Our contribution We present two efficient secure redactable signature schemes that conditionally hide the existence of redacted subdocuments. Along the way, we introduce a new quantified notion that we call $(\alpha(n), \beta(n))$ transparency, capturing the property that the viewer of a possibly redacted document of size n, i.e. containing n subdocuments, may have been redacted from an original document whose length was somewhere in the interval $[\alpha(n), \beta(n)]$.

Our first scheme uses a data structure of size quadratic in n, the number of subdocuments, encoding all the precedence relations between pairs of subdocuments. For our second scheme, we define and design a new graph family, which we call a *redactable precedence graph family*, for each n embedding these precedence relations in a smaller family of graphs; these graphs are of expected size linear in n. This technical tool may be of independent interest.

Ours are the first such schemes that use only basic symmetric cryptographic tools. They are enormously more efficient than previous ones, since they require only a single invocation of a number-theoretic operation, either to sign or to check a signature on a cryptographic hash value. In a practical implementation, all other cryptographic operations are only invocations of a collision-free hash function or of a block cipher.

Without using algebraic techniques, it remains an interesting open problem whether there exists a redactable signature scheme that completely hides the existence of redacted subdocuments, e.g. one that even hides the maximum possible length of the original document.

All of the algorithms presented in this paper can be stated in terms of any proofs of integrity for digital objects that begin by hashing their inputs with a collision-free hash function, including both digital signatures and time-stamping schemes. (Indeed, this is true of many of the redactable-signature schemes in the literature.) Following most previous authors, we limit ourselves here to discussion of digital signatures.

Outline Section 2 gives a brief overview of the related work on redactable signatures, and introduces our model and the cryptographic preliminaries. Section 3 presents our security definitions. Section 4 describes our first transparent redactable signature algorithm, of quadratic cost. Section 5 presents our improved linear-cost algorithm.

2 Background

2.1 Related work

The redactable signature problem was first studied independently by several groups of researchers, under different names: as "content extraction signatures" by Steinfeld et al. [SBZ01], as "redactable signatures" by

Johnson et al. [JMSW02], and as "digitally signed document sanitizing schemes" by Miyazaki et al. [MSI+03, MIM+05]. Hereafter, we will only use the term "redactable".

A related problem, that of "sanitizable" signatures, was introduced by Ateniese et al. [ACMT05]. (These are not to be confused with the "sanitizing" schemes of Miyazaki et al.) Here, instead of allowing redaction as a public operation, in a sanitizable signature scheme a designated party can modify certain parts of the document and sign the modified version. The authors proposed schemes using chameleon hash functions instead of the usual hash functions. Later, Brzuska et al. revisited and formalized the security definitions for sanitizable signatures, and proved the security of a modified scheme that also uses chameleon hash functions [BFF+09].

Researchers have studied a number of other variations on the requirements of what may be called "malleable" signature schemes (to use the term of [MPPS14]), sometimes for redaction signatures, sometimes for sanitizable signatures, and sometimes for both. Such papers include [CKLM13, JMSW02].

The variation that we study here is that of hiding from viewers of a possibly redacted signed document even the existence of redacted portions of the original document [ACMT05, MHI06, BFF+09, CLX09, BBD+10]. Following previous authors, we call this property "transparency", and call a scheme having this property "transparent".

Miyazaki et al. studied cryptographic redaction mechanisms that can prevent as well as allow the redaction of designated parts of a document, calling this feature "disclosure control" [MIM+05]. Haber et al. proposed a more efficient signature scheme supporting redaction, disclosure control, pseudonymization and data deidentification [HHH+08].

Several researchers have extended the study of malleable signatures from documents to trees and graphs, with the motivation of applying these signatures to XML graphs. Kundu and Bertino introduced structural signatures for trees that support public redaction of subtrees by third-party distributors while retaining the integrity of the remaining parts [KB08]. The same authors later proposed two schemes that make use of bilinear maps in order to authenticate directed graphs without leaking information [KB10]. Brzuska et al. formalized a security model for redactable signatures for tree-structured data, and gave a construction that can be proven secure under standard cryptographic assumptions [BBD+10]. In their scheme, the signer computes a new digital signature for every edge of the tree.

Brzuska et al. posed the problem of preventing users who happen to see different redacted versions of the same document from linking them, and they presented a construction to achieve this property of "unlinkability" for redactable signature schemes; their construction makes use of a group signature scheme [BFLS10]. Their construction was improved by Brzuska et al. [BPS14].

Pöhls and Samelin studied the property possessed by certain redactable signature schemes in the literature that allow two legal redactions of a signed document to be merged, and proposed a new scheme based on a trapdoor accumulator function [PS14].

Ahn et al. extended the problem of these and several other kinds of malleable signatures to the general problem of computing on authenticated data [ABC+12].

Chang et al. use a novel randomized variant of a GGM tree (see Section 2.2 below) to supply the randomizing values needed for their transparent redactable signature scheme [CLX09]. This is similar to one part of our construction. Their scheme uses a new hash function based on the strong RSA assumption, whereas we use only a conventional collision-free hash function.

Samelin et al. constructed a redactable signature scheme for a list in $O(n)$ time for computation and storage, assuming the existence of an associative non-abelian hash function [SPB+12].

Ghosh et al. present another $O(n)$ construction of a privacy-preserving authenticated list that supports order queries on its elements, using bilinear maps [GOT14].

We note that each of the latter two papers calculates the complexity of its $O(n)$-time algorithm in terms of number-theoretic or algebraic operations. By contrast, our linear algorithm's complexity is calculated in terms of applications of symmetric-cryptography operations, along with a single digital signature.

2.2 Preliminaries

Model There are three sorts of players in our model: *signers*, *redactors*, and *verifiers*.

The *signer*, having a key pair for a digital signature scheme, prepares and authenticates a document or data set once, producing an ordinary digital signature along with some auxiliary information. We will call

the signature together with the auxiliary information an *extended signature* for the original document. The signature depends on the original data, on a list of operations to be allowed on parts of the data, and on the signer's private signing key.

The document and extended signature may be given to a *redactor*. The redactor may modify the document, according to the signer's list of operations allowed. The redactor makes certain corresponding changes to the auxiliary information, and combines this with the original signature value to form a modified extended signature, which, together with the modified document, may then be "published", i.e. sent to another redactor or sent to a verifier. We emphasize that redactors are able to make these changes without access to the signer's private key.

There may be more than one redaction operation, performed by more than one redactor, where subsequent operations will only be verifiable if they are performed starting from the current modified form of the document and its extended signature (so that, for example, a redacted portion cannot be changed back to the subdocument that it replaced in a previous redaction operation). A user of the system may act as both verifier and redactor.

A *verifier* is able to verify the correctness of the modified document using the (modified) extended signature, capturing the property that the data should only be modified by the redactor according to the specifications of the signer. Unlike the situation with ordinary signature schemes, where any change to the data should cause the signature verification to fail, here we do allow removing certain information from a document, namely the redaction of permitted subdocuments, while disallowing all other changes. We give formal definitions of security for redactable signature schemes in Section 3, and prove that our two algorithms satisfy the definitions in Section 4.4 and Section 5.4.

We describe our cryptographic algorithms in terms of how they apply to single documents, viewed as bitstrings. Let $\mathcal{M}$ denote a document to be signed, segmented into a sequence of subdocuments $m_1, m_2, \ldots, m_n$. In the case of ordinary text documents, these might correspond to words, sentences, or paragraphs, depending on the level of granularity desired. In the case of image or audio or other data files, for example, they might correspond to other convenient divisions of the document; here we ignore these details, and regard the document simply as the sequence of its subdocuments. We measure the size of $\mathcal{M}$ by its number of subdocuments n, sometimes denoted $|\mathcal{M}|$. If $\mathcal{L}$ denotes a subsequence of (the sequence of subdocuments of) $\mathcal{M}$, then we use $\mathcal{M}\backslash\mathcal{L}$ to denote the resulting document after redacting $\mathcal{L}$ from $\mathcal{M}$.

Cryptographic building blocks The security of our algorithms relies on several cryptographic assumptions.

Let H denote a particular choice of *collision-free hash function*, e.g. SHA-256 [NIS12].

Let S be a *digital signature* scheme that is secure against existential forgery attacks by an adaptive chosen-message adversary [GMR88]. (Strictly speaking, we assume that the signing procedure starts by hashing its input. In practice, this is completely without loss of generality.)

Let C be a secure randomized *commitment scheme*, as can be constructed based on the existence of collision-free hash functions [HM96]. The commitment function takes two inputs, a message m and a random string r. Its output $x = \mathsf{C}(m, r)$ is *computationally hiding*, in that it does not leak any information about the particular committed value m, and *computationally binding*, in that it cannot be *opened* by revealing any other string r' so that $x = \mathsf{C}(m', r')$ is also a commitment to another message $m' \neq m$. (In practice—making the stronger assumption that H is a pseudo-random function—one might implement C simply by taking $\mathsf{C}(m, r) = \mathsf{H}(0, m, r)$, with 0 serving as a domain-separation tag indicating input for the commitment scheme.)

Let P be a secure length-doubling *pseudorandom generator*, as used in the GGM construction of pseudorandom functions [GGM86]. Since we use it repeatedly throughout this paper, we sketch here the construction of GGM trees. Beginning with a single random seed s, the construction computes a list of pseudorandom values by building a binary tree from the root to the leaves. Specifically, suppose that s is k bits long. The constructor uses the pseudorandom generator to expand s to a $2k$-bit string, and lets the first and second k bits form, respectively, the left and the right children of s. (In practice—making the same strong assumption on H—this could be implemented by computing $\mathsf{H}(1, s)$ for the left child and $\mathsf{H}(2, s)$ for the right child, with 1 and 2 serving as domain-separation tags.) Continuing in this manner to the necessary depth, we can obtain a sequence of any desired number of leaves.

A *Merkle tree* is a tree such that the values assigned to each internal node is a one-way function of the values assigned to its children [Mer89]. The one-way function is usually instantiated as a standard cryptographic hash function. The root of the tree can be used to authenticate the list of values assigned to its leaves. In our algorithms, we build (possibly unbalanced) binary Merkle trees whose values at the leaves are either "real nodes" that depend on commitments to the contents of the subdocuments, or "dummy nodes", meant to mask the possible existence of subdocuments that may or may not have already been redacted.

2.3 A basic redactable signature scheme

In this section we sketch a particular redactable signature algorithm due to [JMSW02] that does not hide the number or location(s) of redacted subdocuments. This scheme only makes a single signature invocation. It is the basis for designing our new algorithms. In this algorithm, some data are added to form the extended signature for the original document. The size of the additional data grows logarithmically with the number of sequences of consecutively redacted subdocuments. The signature scheme works as follows.

Setup: Given a security parameter, Setup entails the choice of a collision-free hash function H, a secure pseudorandom generator P, a secure commitment scheme C, and a secure signature scheme S. The signer generates a public-private key pair (PK, SK) for S, and adds PK to the list of public parameters $(\mathsf{H}, \mathsf{P}, \mathsf{C}, \mathsf{S})$, while keeping the private key SK secret.

Sign: Given a document $\mathcal{M}$, parsed as the sequence of its subdocuments $(m_1, m_2, \ldots, m_n)$, the signer chooses a random seed s, and computes an n-leaf GGM tree, using P, as explained in §2.2. Let $(r_1, \ldots, r_n)$ denote the list of these leaves. For each subdocument m_i, the signer computes $c_i = C(m_i, r_i)$. Next, the signer builds a Merkle hash tree from the list of leaves $(c_1, \ldots, c_n)$, and signs its root with the private key SK, to obtain a signature σ. The extended signature for m is σ, along with auxiliary information $aux = s$.

Redact: Given the original document $\mathcal{M}$ and its extended signature (σ, aux), along with $\mathcal{L}$, the set of subdocuments in $\mathcal{M}$ to redact, the redactor proceeds as follows. The redactor constructs the GGM tree from the random seed s, and obtains n pseudorandom values $(r_1, \ldots, r_n)$. Let $\mathcal{M}' = (m'_1, \ldots, m'_n)$ denote the resulting redacted document, where $m'_i = m_i$ if $i \notin \mathcal{L}$ and $m'_i = \perp$ if $i \in \mathcal{L}$. (Here, $\perp$ is a special symbol resrved to indicate the presence of redacted subdocuments.) The redactor finds $\mathcal{S}_G$, the minimal set of nodes of the GGM tree that exactly covers $\{r_i | i \notin \mathcal{L}\}$, the set of GGM leaves corresponding to non-redacted subdocuments, as well as $\mathcal{S}_M$, the minimal set of nodes of the Merkle tree that exactly covers $\{c_i | i \in \mathcal{L}\}$, the set of commitment values corresponding to redacted subdocuments. (The choice of these sets $\mathcal{S}_G$ and $\mathcal{S}_M$ for a specific example is illustrated in Figure 3 in Section 4.2 below.) The modified auxiliary information for the redacted document $\mathcal{M}'$ is $aux' = (\mathcal{S}_G, \mathcal{S}_M)$. $\mathcal{M}'$ along with its extended signature (σ, aux') can be passed along for further redaction, which proceeds in similar fashion.

Verify: The verifier is given a possibly redacted document $\mathcal{M}' = (m'_1, \ldots, m'_n)$ and an extended signature of the form $(\sigma, (\mathcal{S}_G, \mathcal{S}_M))$. The verifier uses $\mathcal{S}_G$ to recompute the set of commitments $\{\mathsf{C}(m'_i, r_i) | m'_i \neq \perp\}$ for non-redacted subdocuments. The verifier combines these commitments with the (sub)roots in $\mathcal{S}_M$ to compute the root of the Merkle tree, and verifies the correctness of σ as a signature on that root with respect to the public key PK of the signer.

3 Security definitions

In this section, we formally define the security requirements for a redactable signature scheme. We give formal definitions for three different security requirements: unforgeability, privacy, and transparency.

The principal requirement for any kind of signature scheme is that it should be computationally infeasible to forge illegitimate signatures. In contrast to conventional signature schemes, where no changes to a signed document are permitted, we need a precise characterization of the class of modifications to the original document that we consider to be legitimate. Adapting the definition of [JMSW02] to our scenario, we define a partial order $\preceq$ on documents, as follows.

Definition 1. *Let $\mathcal{M}$ consist of n subdocuments $(m_1, m_2, \ldots, m_n)$, and $\mathcal{M}'$ consist of n' subdocuments $(m'_1, m'_2, \ldots, m'_{n'})$. Then $\mathcal{M}' \preceq \mathcal{M}$ iff $\mathcal{M}'$ is a subsequence of $\mathcal{M}$, considering each document $\mathcal{M}$ or $\mathcal{M}'$ simply as the sequence of its subdocuments.*

In this case, $\mathcal{M}'$ is a possibly redacted version of $\mathcal{M}$, and we may also write $\mathcal{M} \succeq \mathcal{M}'$. If $\mathcal{M}' \preceq \mathcal{M}$ and $\mathcal{M}' \neq \mathcal{M}$, then we write $\mathcal{M}' \prec \mathcal{M}$ (or $\mathcal{M} \succ \mathcal{M}'$). Given a set of documents or messages S, we use $\text{LUB}(S)$ to denote the least upper bound on the elements of S according to the partial order defined by $\preceq$.

Definition 2. *A redactable signature scheme with respect to binary relation $\preceq$ is a quadruple of probabilistic polynomial-time algorithms $\Sigma = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Redact})$, such that:*

- SETUP. *The setup algorithm $\mathsf{Setup}(1^k)$ generates a set of public parameters and a public-private key pair based on the security parameter k.*

$$(params, PK, SK) \leftarrow \mathsf{KeyGen}(1^k)$$

- SIGNING. *The algorithm $\mathsf{Sign}(SK, \mathcal{M})$ takes as input a secret key SK and a document $\mathcal{M}$. It outputs redactable signature $s = (\sigma, aux)$ for $\mathcal{M}$.*

$$(\sigma, aux) \leftarrow \mathsf{Sign}(SK, \mathcal{M})$$

- VERIFICATION. *The verification algorithm $\mathsf{Verify}(PK, \mathcal{M}, s)$ takes the public key PK and signature s as inputs and outputs a bit b reporting whether s is a valid redactable signature on $\mathcal{M}$ with respect to PK.*

$$b \leftarrow \mathsf{Verify}(PK, \mathcal{M}, s)$$

- REDACTION. *On inputs a public key PK, a document $\mathcal{M}$ together with its redactable signature $s = (\sigma, aux)$, as well as a subdocument $\mathcal{M}' \preceq \mathcal{M}$, the redaction algorithm $\mathsf{Redact}(PK, \mathcal{M}, s, \mathcal{M}')$ returns the document $\mathcal{M} \backslash \mathcal{M}'$ and a signature $s' = (\sigma, aux')$.*

$$(\mathcal{M} \backslash \mathcal{M}', \sigma, aux') \leftarrow \mathsf{Redact}(PK, \mathcal{M}, s, \mathcal{M}')$$

We say the redactable signature scheme is correct if it satisfies both of the following conditions:

- SIGNATURE CORRECTNESS. *For any $k \in \mathbb{N}$, key pair $(PK, SK) \leftarrow, \mathsf{KeyGen}(1^k)$, any document $\mathcal{M}$, the signature $s = (\sigma, aux) \leftarrow \mathsf{Sign}(SK, \mathcal{M})$ satisfies $\mathsf{Verify}(PK, s, \mathcal{M}) = 1$.*
- REDACTION CORRECTNESS. *For any $k \in \mathbb{N}$, key pair $(PK, SK) \leftarrow \mathsf{KeyGen}(1^k)$, any document $\mathcal{M}$ and signature $s = (\sigma, aux)$ with $\mathsf{Verify}(PK, s, \mathcal{M}) = 1$, any subdocuments $\mathcal{M}', \mathcal{L}$ satisfying $\mathcal{M}' \preceq \mathcal{M}$, $\mathcal{L} = \mathcal{M} \backslash \mathcal{M}'$, and any $(\mathcal{L}, s') \leftarrow \mathsf{Redact}(PK, \mathcal{M}, s, \mathcal{M}')$ where $s' = (\sigma, aux')$, we require $\mathsf{Verify}(PK, \mathcal{L}, s') = 1$.*

Remark: We denote the (extended) signature $s = (\sigma, aux)$ as having two parts. This is for convenience, since all the algorithms we discuss work this way. After redaction, the new signature becomes $s' = (\sigma, aux')$, where the first component σ is identical to the previous version while the second component is updated. Our notation is no restriction on the form of a redactable signature acheme, since the first component can be empty.

Informally expressed, a transparent redactable signature should satisfy the following properties.

Redactability Given a valid signature tuple $(\mathcal{M}, \sigma, aux)$, another valid signature $(\mathcal{M}', \sigma, aux')$ can be derived for any document $\mathcal{M}' \preceq \mathcal{M}$, without the use of the signer's private key, SK.

Unforgeability Without SK, it should be infeasible to compute a valid extended signature for any document other than one that is a redacted version of a document that *was* signed with SK. Even for an attacker able to request signatures on several different documents $\mathcal{M}_i$, it should remain impossible to forge a signature on any document $\mathcal{M}' \not\preceq \text{LUB}\{\mathcal{M}_i\}$.
However, valid signatures on documents $\mathcal{M}' \preceq \text{LUB}\{\mathcal{M}_i\}$ do not count as successful forgeries. For example, having seen signed versions of AC and AD, both resulting as redactions of the longer signed document $ABCD$, a user might be able to produce a signature for ACD; this would not constitute a forgery for the scheme.

Privacy All information about redacted subdocuments is completely hidden.

Transparency Given a signature tuple $(\mathcal{M}, \sigma, aux)$, it should be infeasible to infer whether $\mathcal{M}$ is a newly signed document or a redacted version of another longer document, nor anything about the locations of previously redacted subdocuments of any longer document.

More formally, we capture these properties in the following definitions, adapted from those given by [JMSW02] and followed by later authors. These security definitions are as strong as those of more recent papers, for example those of [BBD+10, MPPS14].

Definition 3 (Unforgeability). *A redactable signature scheme* $\Sigma = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Redact})$ *is unforgeable under adaptive chosen-message attack if no PPT adversary* $\mathcal{A}$ *can win the following game with non-negligible probability.*
Setup: *The challenger runs the* Setup *algorithm and gives the adversary the public parameters* $(\mathsf{H}, \mathsf{P}, \mathsf{C}, \mathsf{S}, PK)$.
Query: *The adversary adaptively chooses documents* $\mathcal{M}_1, \mathcal{M}_2, \ldots$ *to be signed by an oracle with access to* SK. *In response to the query* $\mathcal{M}_i$, *the oracle returns a freshly signed redactable signature* (σ_i, aux_i) *for* $\mathcal{M}_i$ *with respect to* PK.
Forge: *The adversary* $\mathcal{A}$ *outputs* $(\mathcal{M}', \sigma', aux')$, *winning the game if* $\mathsf{Verify}_{PK}(\mathcal{M}', \sigma', aux') = \mathsf{TRUE}$ *and* $\mathcal{M}' \not\preceq \mathcal{M}_i$ *for every* i *from the Query phase.*

We omit here a formal game-based definition of **privacy**, as in previous papers, because privacy for a redactable signature scheme is implied by the property that we define next. This is our new definition of a quantified version of the stronger subdocument-hiding property of transparency.

Definition 4 (Transparency). *Let* α *and* β *denote functions on positive integers, such that* $\forall n,\ n \leq \alpha(n) < \beta(n)$. *A redactable signature scheme* $\Sigma = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Redact})$ *is* (α, β) *transparent if the following two distributions are computationally indistinguishable, given any document* $\mathcal{M}$ *of size* n *along with documents* $\mathcal{M}_0$ *and* $\mathcal{M}_1$, *both of sizes in the interval* $[\alpha(n), \beta(n)]$ *and such that* $\mathcal{M} \preceq \mathcal{M}_0$ *and* $\mathcal{M} \preceq \mathcal{M}_1$*:*

$$\mathcal{D}_0 = \{(\sigma, aux) : (\sigma, aux) = \mathsf{Redact}_{PK}(\mathcal{M}_0, \mathsf{Sign}_{SK}(\mathcal{M}_0), \mathcal{M}_0 \backslash \mathcal{M}\} \text{ and}$$
$$\mathcal{D}_1 = \{(\sigma, aux) : (\sigma, aux) = \mathsf{Redact}_{PK}(\mathcal{M}_1, \mathsf{Sign}_{SK}(\mathcal{M}_1), \mathcal{M}_1 \backslash \mathcal{M}\},$$

where (PK, SK) *is the public-private key pair generated by an invocation of* Setup.

Here we use the convention that $\mathsf{Redact}_{PK}(\mathcal{M}, \mathsf{Sign}_{SK}(\mathcal{M}), \emptyset) := \mathsf{Sign}_{SK}(\mathcal{M})$.

This definition implies that, given a redactable signature for a document of size n, no adversary can distinguish whether the signature is freshly generated or redacted from a document of size $n' \in [\alpha(n), \beta(n)]$, where the bounds on this uncertainty interval depend on the algorithm used. In practice, leaving a large range of uncertainty will suffice for many applications.

Unconditionally hiding the original size seems to be difficult to achieve using only tools of symmetric cryptography for manipulation of any data associated with the subdocuments. We leave this as an open problem.

We note that Definition 4 is equivalent to the similar game-based definitions of [BBD+10, MPPS14], with the addition of our condition that quantifies the size of possibly redacted subdocuments.

As noted above, (α, β) transparency is a strictly stronger requirement than privacy, as defined in earlier papers with a distinguishability game where the adversary's task is to define which of two original subdocuments (or sequences of subdocuments) has been redacted from the document given to the adversary. An adversary that successfully wins the privacy game against a particular scheme can easily be transformed into an adversary that wins the transparency game, with a similar probability of winning; see e.g. [BBD+10].

4 A quadratic algorithm

As noted above, we base both of our new algorithms on the redactable signature scheme by the authors of [JMSW02], which we described in Section 2.3.

As in their algorithm, we build both a GGM and a Merkle tree. In order to hide what their algorithm does not, we modify the algorithm, incurring some additional cost. For the algorithm we present in this section, the data structures we build will be of size quadratic in the number of subdocuments. For the algorithm we present in Section 5 below, they will be of linear expected size.

As in [JMSW02], we build a Merkle tree, some of whose leaves are computed from the subdocuments. But rather than using a randomized commitment to the contents of each subdocument, for each pair of

subdocuments m_i, m_j with $i < j$ we compute a *precedence node* c_{ij}, which is a randomized commitment value encoding the fact that m_i precedes m_j in the original document (as well as the contents of the two subdocuments). In addition, the signer computes several *dummy nodes*, which are bit-strings of the appropriate length that are indistinguishable from commitment values. Furthermore, rather than using the list of commitment values in their original order as leaves of the Merkle tree as in [JMSW02], intsead the ordered list of the positions of the precedence nodes and dummy nodes as leaves of the Merkle tree is chosen at random.

Again as in [JMSW02], the random bits needed for the computation of randomized commitment values are themselves computed as the leaves of a GGM tree. And also as before, when a subdocument is redacted, the preimages of its associated commitment values are deleted from the signature's associated data, in effect transforming these commitment values—first computed as precedence nodes—into what will appear to be dummy nodes to verifiers of the redacted version.

In other words, we deal with the sequence of subdocuments of $\mathcal{M}$ by considering a graph that models their order, namely the graph with nodes $1, 2, \ldots, n$ having a directed edge (i, j) for every pair of nodes with $i < j$; this is the tournament on $[1 \ldots n]$, with the canonical ordering. (A *tournament* is a directed graph with exactly one edge between every pair of nodes.) For each edge of this graph, the signer computes a corresponding precedence node, and randomly places it among the leaves of a Merkle tree that is built as part of the signing algorithm.

4.1 Details of the quadratic algorithm

The redactable signature scheme can be described in terms of four subalgorithms: $\mathsf{Setup}, \mathsf{Sign}, \mathsf{Redact}$, and Verify. To illustrate the operations of these subalgorithms, we provide a small example in Section 4.2 below.

- $\mathsf{Setup}(1^k)$: Given the security parameter k, Setup entails the choice of a collision-free hash function H, a secure pseudorandom generator P, a secure commitment scheme C, and a secure signature scheme S. The signer generates a public-private key pair (PK, SK) for S, and adds PK to the list of public parameters $(\mathsf{H}, \mathsf{P}, \mathsf{C}, \mathsf{S})$, while keeping the private key SK secret.

- $\mathsf{Sign}(\mathcal{M}, SK)$: Given a document $\mathcal{M}$, parsed as a sequence of n subdocuments $m_1, m_2, \ldots, m_n$, the signer does the following.
  1. For each subdocument, compute $\mathsf{H}(m_i)$ and store it in a lookup table T_2.
  2. Choose a size M for the total number of leaves, $M > \binom{n}{2}$ but $M \in O(n^2)$. (The signer will build an M-leaf Merkle tree, $D = M - \binom{n}{2}$ of whose leaves are dummy nodes.)
  3. Choose a random ordering for these leaves by sampling a vector $\mathbf{x}$ uniformly at random from the set of length M binary vectors of Hamming weight D. We write $\mathbf{x} = x_0 x_1 \ldots x_{M-1}$.
     The signer will construct a sequence of M leaves for the Merkle tree based on $\mathbf{x}$, denoted $a_0, a_1, \ldots, a_{M-1}$, where D positions in the sequence are dummy nodes, indicated by the 1 bits in $\mathbf{x}$, and the remaining $M - D$ leaves are *real nodes*. Define $\phi : [0, \ldots, \binom{n}{2} - 1] \to [0, \ldots, (M-1)]$ by letting $\phi(\ell)$ denote the position of the ℓth real node in $\mathbf{x}$ (i.e. the position of the ℓth 0 in $\mathbf{x}$).
     (a) Write $\{(i, j) | 1 \le i < j \le n\}$ in lexicographic order to form the index column of a new table, T_1.
     (b) Choose a random permutation Π: $\{(i, j)\} \to \left[0, \ldots, \binom{n}{2} - 1\right]$.
     (c) Choose a random seed s and compute an M-leaf GGM tree using the pseudorandom generator P. Let $(r_0, r_1, \ldots, r_{M-1})$ denote the list of these leaves. Compute $\mathcal{S}_G$, the minimum set of nodes of the tree whose descendants at the leaf level is exactly the set of real nodes defined by (the 0 positions in) $\mathbf{x}$.
     (d) For each (i, j) with $1 \le i < j \le n$, insert $\phi(\Pi(i, j))$ into the row indexed by (i, j) in T_1. Then compute the commitment value c_{ij}, the precedence node for the pair of subdocuments m_i and m_j, as follows:
     $$c_{ij} = \mathsf{C}(\ [\phi(\Pi(i, j)), \mathsf{H}(m_i), \mathsf{H}(m_j)],\ r_{\phi(\Pi(i,j))}).$$
     (Note that $\mathsf{H}(m_i), \mathsf{H}(m_j)$ can be retrieved using the the index (i, j) from their precomputed hash values, the ith and jth entries in T_2, and that $r_{\phi(\Pi(i,j))}$ is the leaf in position $\phi(\Pi(i, j))$ in the GGM tree.)

(e) Assign the leaf nodes in the Merkle tree as follows:

$$a_k = \begin{cases} \mathsf{C}(\mathsf{d}, r_k) & \text{if } x_k = 1, \\ c_{ij} & \text{if } x_k = 0, k = \phi(\Pi(i,j)). \end{cases}$$

Each dummy node, indicated by 1 in its position in $\mathbf{x}$, is a randomized commitment to a constant symbol, here denoted d; each real node, indicated by 0 in its position in $\mathbf{x}$, is the appropriate commitment value c_{ij}.

4. Use H to compute a Merkle tree from the list of leaf nodes defined in Step 3e above. Compute $\mathcal{S}_M$, the minimum set of subroots of the Merkle tree that covers the leaves corresponding to the dummy nodes, i.e. those with 1 in their position in $\mathbf{x}$.
5. Sign the root of the Merkle tree, producing the signature σ. The auxiliary information that will accompany σ to form the extended signature is $aux = (\mathsf{T}_1, \mathsf{T}_2, \mathcal{S}_G, \mathcal{S}_M, \mathbf{x})$.

– $\mathsf{Redact}(\mathcal{M}, \sigma, aux, \mathcal{L})$: Given $\mathcal{M}$ and σ, together with the auxiliary data aux and $\mathcal{L}$, the set of subdocuments in $\mathcal{M}$ to redact, the redactor proceeds as follows.

1. Call $\mathsf{Verify}(\mathcal{M}, \sigma, aux)$ to verify the signature; if the signature is invalid, then abort.
2. For each $m_\alpha \in \mathcal{L}$, repeat the following:
   (a) Remove the αth row, which contains $\mathsf{H}(m_\alpha)$, from T_2.
   (b) For each row (i, α) with $i < \alpha$, change the bit in $\mathbf{x}$ at position $\phi(\Pi(i, \alpha))$, given by the mapping value in this row, from 0 to 1, and then remove the row from T_1. This changes the leaf in position $\phi(\Pi(i, \alpha))$ from a real node to an apparent dummy node.
   Make the corresponding changes for each row (α, j) with $\alpha < j$: change the bit in $\mathbf{x}$ at position $\phi(\Pi(\alpha, j))$ from 0 to 1, and then remove the row from T_1.
   (c) Rename $\beta := \beta - 1$ for $\beta = \alpha + 1, \ldots, n$, and relabel the corresponding entries in both T_1 and T_2, obtaining $\mathsf{T}_1{}'$ and $\mathsf{T}_2{}'$, respectively.
3. Let $\mathbf{x}'$ denote the result of all the updates to $\mathbf{x}$.
4. Update $\mathcal{S}_G$ to account for the changes from $\mathbf{x}$ to $\mathbf{x}'$, to obtain $\mathcal{S}'_G$, the minimum set of nodes of the GGM tree whose descendants at the leaf level constitute exactly the set of nodes defined by the 0 positions in $\mathbf{x}'$.
5. Update $\mathcal{S}_M$ to account for the changes from $\mathbf{x}$ to $\mathbf{x}'$, to obtain $\mathcal{S}'_M$, the minimum set of subroots of the Merkle tree that covers exactly the set of leaves with 1 in their positions in $\mathbf{x}'$.

Let $\mathcal{M}'$ denote the redacted document $\mathcal{M} \backslash \mathcal{L}$. The extended signature for $\mathcal{M}'$ is σ with updated auxiliary data $aux' = (\mathsf{T}_1{}', \mathsf{T}_2{}', \mathcal{S}'_G, \mathcal{S}'_M, \mathbf{x}')$. The 1 bits in $\mathbf{x}'$ indicate the positions that were either dummy nodes in $\mathcal{M}$ or nodes that were changed from real to apparent dummy nodes as part of the redaction procedure.

– $\mathsf{Verify}(\mathcal{M}, \sigma, aux)$:
Given the document $\mathcal{M}$ parsed as a sequence of n subdocuments $m_1, m_2, \ldots, m_n$, the signature σ, and the tuple $aux = (\mathsf{T}_1, \mathsf{T}_2, \mathcal{S}_G, \mathcal{S}_M, \mathbf{x})$, the verifier does the following:

1. For each subdocument, compute $\mathsf{H}(m_i)$ and compare it to the value in table T_2. If the hashes do not match, output FALSE.
2. Check that table T_1 correctly encodes the edge set of the tournament on $[1 \ldots n]$. If not, output FALSE.
3. Reconstruct the approriate subgraph of the GGM tree from $\mathcal{S}_G$ and $\mathbf{x}$.
4. For each (i, j), $1 \le i < j \le n$, compute the precedence-proof commitment value for (m_i, m_j), namely

$$c_{ij} = \mathsf{C}(\ [\phi(\Pi(i,j)), \mathsf{H}(m_i), \mathsf{H}(m_j)],\ r_{\phi(\Pi(i,j))}),$$

after accessing $\phi(\Pi(i, j))$ from table T_1 and then $r_{\phi(\Pi(i,j))}$ from the GGM tree.
5. Use H to compute the Merkle tree from the values c_{ij} and $\mathcal{S}_M$, based on $\mathbf{x}$.
6. Check whether σ is a valid signature for the root of the Merkle tree, with respect to the signer's public key. If yes, return TRUE, otherwise return FALSE.

4.2 A small example for the quadratic algorithm

Let $M = 15$, $n = 5$, $D = 15 - \binom{5}{2} = 5$. The vector $\mathbf{x}$ is chosen at random from $\Delta_{15,5}$, say e.g. $\mathbf{x} =$ 0100 1001 1100 000. We choose M not a power of 2 in order to emphasize that the Merkle tree does not have to be balanced.

The signer constructs the table T_1 as shown in Figure 1, and then builds the GGM tree and Merkle tree as shown in Figure 3. The collection of red-circled nodes in the GGM tree form the set $\mathcal{S}_G$, and the blue-circled nodes in the Merkle tree form the set $\mathcal{S}_M$. Note that in Figure 1, the random permutation Π in the second column is only known by the signer, and is never sent as part of the signature. We show this column here in gray just to clarify the mapped indices $\phi(\Pi(i,j))$. We illustrate the redaction of subdocument m_2 from

index (i,j)	$\Pi(i,j)$	$\phi(\Pi(i,j))$
$(1,2)$	4	6
$(1,3)$	1	2
$(1,4)$	9	10
$(1,5)$	5	11
$(2,3)$	0	0
$(2,4)$	7	13
$(2,5)$	8	14
$(3,4)$	3	5
$(3,5)$	2	3
$(4,5)$	6	12

Fig. 1: T_1 for the original document.

(i,j)	$\phi(\Pi(i,j))$
$(1,2)$	6
$(1,3)$	2
$(1,4)$	10
$(1,5)$	11
$(2,3)$	0
$(2,4)$	13
$(2,5)$	14
$(3,4)$	5
$(3,5)$	3
$(4,5)$	12

$\Rightarrow$

(i,j)	$\phi(\Pi(i,j))$
$(1,2)$	2
$(1,3)$	10
$(1,4)$	11
$(2,3)$	5
$(2,4)$	3
$(3,4)$	12

Fig. 2: Updating T_1 by removing rows involving m_2, and relabeling $j := j-1$ for $j = 3,4,5$.

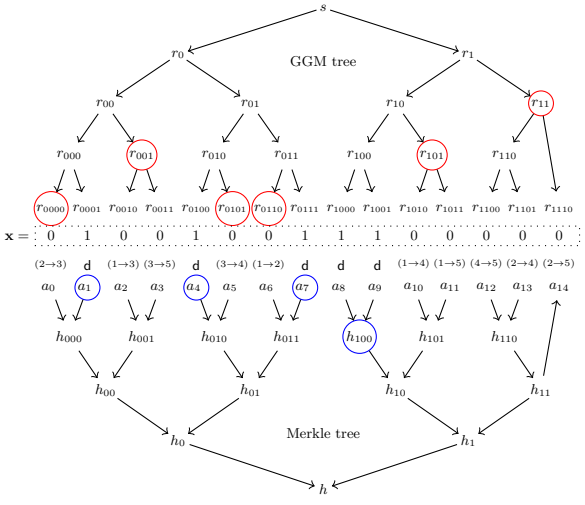

Fig. 3: GGM and Merkle trees for the original document; $\mathcal{S}_G$ nodes are red, and $\mathcal{S}_M$ nodes are blue.

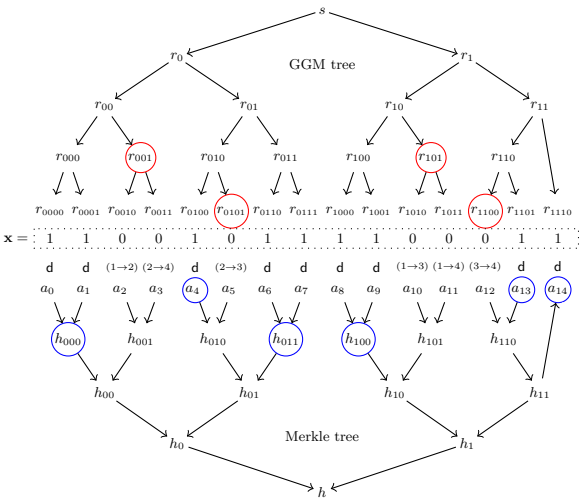

Fig. 4: Updated $\mathcal{S}_G$, $\mathcal{S}_M$ and $\mathbf{x}$ after redaction of m_2.

the orginal document. The result is that T_1 is updated as in Figure 2, where the gray rows in the left-hand table indicate those rows, the ones involving the redacted subdocument m_2, that are to be removed. Then $\mathbf{x}$, $\mathcal{S}_G$, and $\mathcal{S}_M$ are updated as in Figure 4.

4.3 Efficiency

The redactable signature scheme presented above is computationally efficient. It does not require the use of any relatively expensive number-theoretic computations such as modular exponentiation or bilinear pairing. Instead, it can be implemented using only the basic symmetric cryptographic tools of collision-free hash functions and pseudorandom generators, before making a single invocation of an ordinary digital signature scheme to sign the root of a Merkle hash tree.

Let T_L denote the time required by a lightweight operation of symmetric cryptography, and let T_H denote the time required by the heavyweight assymmetric cryptographic operation of signing or verifying an ordinary digital signature. All three of the algorithms Sign, Verify, and Redact run in time $O(n^2) \cdot T_L + O(1) \cdot T_H$,

The communication overhead of the redactable signature is due to σ and $aux = (\mathsf{T_1}, \mathsf{T_2}, \mathcal{S}_G, \mathcal{S}_M, \mathbf{x})$. Given a document of size n, the size of table $\mathsf{T_1}$ is $\binom{n}{2} = O(n^2)$. Because of the random choice of $\mathbf{x}$ to position dummy nodes among real nodes and the random choice of the permutation Π, we expect that for many signed documents, many of the redacted portions as well as many of the dummy nodes will occur as sequences of consecutive leaves in the Merkle tree. If $\mathbf{x}$ contains ℓ such sequences of consecutive 0's (marking real-node positions), then $\mathcal{S}_G$ contains $O(\ell \log \binom{n}{2})$ tree nodes and $\mathcal{S}_M$ contains $O(\ell \log(M - \binom{n}{2}))$ tree nodes. As long as the maximum tree size is $M = O(n^2)$, we incur a total cost of $O(\ell \log n)$ tree nodes in $\mathcal{S}_G$ and $\mathcal{S}_M$. So the communication overhead is dominated by the size of table $\mathsf{T_1}$, which is quadratic in n.

4.4 Security analysis

Here we prove that the algorithm of this section meets the security definitions for a redactable signature scheme defined in Section 3 above, by reducing the existence of an adversary that successfully breaks our scheme to one that breaks one or more of the signature scheme, the pseudorandom generator, the commitment scheme, or the collision-free hash function that we use.

Theorem 1. *Assume that* H *is a collision-free hash function,* S *is a digital signature scheme that is adaptively secure against existential forgery,* C *is a secure commitment scheme, and* P *is a secure pseudorandom generator. Then the algorithm of Section 4.1 is an efficient, secure (unforgeable, transparent, and private) redactable-signature scheme.*

The security assertion of Theorem 1 follows immediately from Theorem 2, Theorem 3, and the fact that (α, β) transparency implies privacy.

Theorem 2. *With the assumptions of Theorem 1, the algorithm of Section 4.1 is unforgeable.*

Proof. Let $\mathcal{A}$ be an adversary having a non-negligible probability of winning the forgery game of Definition 3. We construct below an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to attack one or more of H, C, and S, acting as the challenger for $\mathcal{A}$ and using $\mathcal{A}$'s outputs in turn as her own outputs. $\mathcal{B}$ proceeds as follows.
Setup: $\mathcal{B}$'s challenger chooses hash function H, commitment scheme C, signature scheme S, and pseudorandom generator P, and computes a new public-private key pair (PK, SK) for S. The challenger then gives $\mathcal{B}$ the list of public parameters $(\mathsf{H}, \mathsf{P}, \mathsf{C}, \mathsf{S}, PK)$, and $\mathcal{B}$ passes the list along to $\mathcal{A}$. Note that neither $\mathcal{B}$ nor $\mathcal{A}$ is given the signing key SK.
Query: Adversary $\mathcal{A}$ issues queries to its challenger, $\mathcal{B}$. For each query $\mathcal{M}_i$, $\mathcal{B}$ runs the Sign algorithm of the redactable signature scheme, stopping before its last step, having generated (h_i, aux_i), where h_i is the root of the Merkle tree for $\mathcal{M}_i$. Next, $\mathcal{B}$ submits h_i as a signing query to its own challenger, receiving in return a signature σ_i, and stores return (σ_i, aux_i) as the response to $\mathcal{A}$'s query on $\mathcal{M}_i$.
Forge: Adversary $\mathcal{A}$ outputs a possible forgery, $(\mathcal{M}', \sigma', aux')$. With non-negligible probability, $\mathcal{A}$ wins the game, in which case $\mathcal{M}' \not\preceq \mathcal{M}_i$ for every i, and $\mathsf{Verify}(\mathcal{M}', \sigma', aux') = \mathsf{TRUE}$. $\mathcal{B}$ can use this successful forgery for the overall redaction signature scheme to attack one of the cryptographic primitives it is based on.

Suppose first that σ' is not equal to any of the signatures σ_i that were returned to $\mathcal{A}$ during the **Probe** phase of the game. To convert σ' into a signature forgery for PK, $\mathcal{B}$ uses $(\mathcal{M}', aux')$ to reconstruct the Merkle tree and and obtains its root, h'. Then $\mathcal{B}$ has computed a successful forgery against the signature scheme S, namely (h', σ').

Suppose now that $\mathcal{A}$'s signature σ' *is* equal to one of the responses that were returned to $\mathcal{A}$ during the **Query** phase, namely σ_i, in response to the query $\mathcal{M}_i$, where. Let (aux_i, σ_i) be the extended signature from

this response. Let T' be the Merkle hash tree determined by $(\mathcal{M}', aux')$, and let T_i be the one determined by $(\mathcal{M}_i, aux_i)$. $\mathcal{B}$ compares T' and T_i. If the two trees have different root nodes, then as above $\mathcal{B}$ has computed a successful forgery against the signature scheme S. Otherwise, if the two Merkle trees differ anywhere below the root, then $\mathcal{B}$ can compute a hash collision for H with non-negligible probability.

There remains the case that the two Merkle trees T' and T_i are identical. Let a'_k and $\bar{a}_k$ respectively denote their kth leaf nodes; note that at this point we have $a'_k = \bar{a}_k$, though the two values may have been constructed differently. Next, $\mathcal{B}$ compares the two GGM trees constructed for $\mathcal{M}'$ and for $\mathcal{M}_i$. Because as a successful forgery $\mathcal{M}' \not\preceq \mathcal{M}_i$, along with the fact that by construction every leaf in our Merkle trees is either a precedence node or a dummy node, $\mathcal{B}$ can find at least one of the following two cases among the pairs of corresponding leaf nodes:

- For some index k, a'_k and $\bar{a}_k$ are both precedence nodes: a'_k is a commitment value depending on a pair (m'_i, m'_j) of subdocuments of $\mathcal{M}'$ and similarly $\bar{a}_k$ is a commitment for a pair $(\bar{m}_{\bar{i}}, \bar{m}_{\bar{j}})$ from $\mathcal{M}_i$. We have $(m'_i, m'_j) \neq (\bar{m}_{\bar{i}}, \bar{m}_{\bar{j}})$, while $a'_k = \bar{a}_k$. Thus $\mathcal{B}$ has constructed a pair of inputs for the commitment scheme C that can be used to break its binding property.
- For some index k, one of a'_k and $\bar{a}_k$ is a precedence node, depending as above on a pair of subdocuments of either $\mathcal{M}'$ or $\mathcal{M}_i$, while the other is a dummy node, computed as a commitment to the constant value d. As above, $\mathcal{B}$ has successfully constructed a pair of inputs for the commitment scheme C that can be used to break its binding property. □

Theorem 3. *With the assumptions of Theorem 1, the algorithm of Section 4.1 is $(n, \beta(n))$ transparent, where M is the signer's choice of a maximum size for the GGM tree, and $\beta(n) = \max\{\ell : \binom{\ell}{2} \leq M\}$.*

The proof of Theorem 3 is based on the next lemma and its corollaries. Let $\Delta_{\ell,w}$ be the set of binary vectors of length ℓ and Hamming weight w. For any $k \leq \ell - w$, consider the probabilistic map $flip_k : \Delta_{\ell,w} \to \Delta_{\ell,w+k}$ defined by flipping k randomly chosen 0 bits of its input vector to 1 bits. By straightforward calculation, we have the following lemma.

Lemma 1. *For any $k \leq \ell - w$, the distribution $\{flip_k(\mathbf{x}) : \mathbf{x} \xleftarrow{\$} \Delta_{\ell,w}\}$ is identical to $\{\mathbf{x} : \mathbf{x} \xleftarrow{\$} \Delta_{\ell,w+k}\}$.*

Corollary 1. *Let O_n denote the distribution of the sequence of M leaf nodes in the Merkle tree for a non-redacted document of size n, and $R_{n+1\to n}$ the distribution of the sequence of M leaf nodes in the Merkle tree after redacting any single subdocument in a document of size $n+1$. Then O_n and $R_{n+1\to n}$ are computationally indistinguishable.*

Corollary 2. *Let O_n be as above, and let $R_{n+k\to n}$ be the distribution of the sequence of M leaf nodes in the Merkle tree after redacting any k subdocuments in a document of size $n+k$. Then O_n and $R_{n+k\to n}$ are computationally indistinguishable*

As each leaf node in the Merkle tree is either a commitment to a dummy constant value or a commitment to a precedence proof for two subdocuments, Corollary 1 follows directly from the computationally hiding property of C, and the application of Lemma 1 by setting $\ell = M$, $w = M - \binom{n+1}{2}$ and $k = \binom{n+1}{2} - \binom{n}{2} = n$. Corollary 2 then follows by multiple applications of Corollary 1.

Proof of Theorem 3. We are given a document $\mathcal{M}$ of size n and its signature tuple (σ, aux), computed by the algorithm of Section 4, along with two documents $\mathcal{M}_0$, $\mathcal{M}_1$, both of size $|\mathcal{M}_i| \geq n$ and satisfying $\mathcal{M}_i \succeq \mathcal{M}$. If $\binom{|\mathcal{M}_i|}{2}$ is greater than M, the maximum number of leaves of the Merkle tree, then the verifier knows that $\mathcal{M}$ is not redacted from $\mathcal{M}_i$.

Assume now that for both $i = 0, 1$ we have $|\mathcal{M}_i| \leq \beta(n) = \max\{\ell : \binom{\ell}{2} \leq M\}$. The signature for $\mathcal{M}$ by redacting $\mathcal{M}_i \backslash \mathcal{M}$ is

$$(\sigma^i, aux^i) = \mathsf{Redact}_{PK}(\mathcal{M}_i, \mathsf{Sign}_{SK}(\mathcal{M}_i), \mathcal{M}_i \backslash \mathcal{M}\} = (\sigma^i, \mathsf{T}_1^{\mathsf{i}}, {\mathsf{T}_2}^i, \mathcal{S}_G^i, \mathcal{S}_M^i, \mathbf{x}^i).$$

We would like to prove that the two distributions (σ^0, aux^0) and (σ^1, aux^1) are computationally indistinguishable.

For this purpose we will use the following lemma several times: Given two computationally indistinguishable distributions X and Y, and an efficiently computable (possibly probabilistic) function f, the two joint distributions $(X, f(X))$ and $(Y, f(Y))$ are also computationally indistinguishable.

Since $wt(\mathbf{x}^0) = wt(\mathbf{x}^1) = \binom{n}{2}$, it follows from Lemma 1 that the distributions $\mathbf{x}^0$ and $\mathbf{x}^1$ are identical. Also, the distributions of the two hash tables $\mathsf{T_2}^0$ and $\mathsf{T_2}^1$ are identical. Similarly, in T_1^0 and T_1^1 their respective first columns are identical, and the values in their second columns are based on the random permutation Π and the position information of real nodes in $\mathbf{x}$. Since permutation Π is independent of the generation of $\mathbf{x}$, T_1^0 and T_1^1 are identically distributed.

$\mathcal{S}_G^i$ is a set of pseudorandom numbers generated using P, beginning from a randomly chosen string at the root of the GGM tree. It consists of the minimum list of GGM tree nodes that covers the real nodes indicated by the 0 entries in $\mathbf{x}^i$. Since the output of P is computationally indistinguishable from random, the distributions of $\mathcal{S}_G^0$ and $\mathcal{S}_G^1$ are themselves computationally indistinguishable.

It can be concluded from Corollary 2 above that the respective lists of leaf nodes for the two Merkle trees generated by the computation (or verification) of the two extended signatures are computationally indistinguishable. Because of the way each $\mathcal{S}_M^i$ is computed as the list of internal nodes of the tree that cover the dummy nodes indicated by the 1 entries in $\mathbf{x}^i$, it follows that the distributions $\mathcal{S}_M^0$ and $\mathcal{S}_M^1$ are also computationally indistinguishable. Thus we conclude that the two joint distributions $(\mathsf{T}_1^i, \mathsf{T_2}^i, \mathcal{S}_G^i, \mathcal{S}_M^i, \mathbf{x}^i)$ are computationally indistinguishable.

Finally, the Merkle tree root h^i is a hash value that is calculated by a deterministic computation with input $\mathcal{M}_i \backslash \mathcal{M}$ and $(\mathsf{T}_1^i, \mathsf{T_2}^i, \mathcal{S}_G^i, \mathcal{S}_M^i, \mathbf{x}^i)$, and σ^i is computed with the ordinary signature scheme S with input h^i. Hence σ^0 and σ^1 are computationally indistinguishable, and so are the two distributions (σ^0, aux^0) and (σ^1, aux^1). This concludes the proof of the transparency of our scheme. □

5 A linear algorithm

In order to sign a document containing n subdocuments, the quadratic algorithm of Section 4 above makes crucial use of the canonical tournament on n nodes, and the extended signature is therefore necessarily of size at least $\binom{n}{2}$. To reduce the size of the graphs we need to handle, we can add several nodes while removing many edges; instead of including a single edge in the graph for every pair of subdocuments m_i, m_j with $i < j$, we merely require the existence of a directed path from i to j. For each n, we replace the single graph on n nodes with a distribution on a set of such graphs, $\mathcal{H}_n$. We further define a *redact* operation on these graphs, corresponding to the redaction of one of the n subdocuments. This operation transforms any graph in $\mathcal{H}_n$ into a graph in $\mathcal{H}_{n-1}$, and we require that the result of choosing a graph from the distribution on $\mathcal{H}_n$ and then choosing a random *redact* operation be identical to simply choosing a graph from the distribution on $\mathcal{H}_{n-1}$. We call such an infinite family of distributions on graphs a *redactable precedence graph* family.

We define such a family, $\mathcal{G}_{n,k}$, by adding $k-1$ bridge nodes to $[1 \dots n]$, thereby dividing the n nodes into k groups; constructing tournaments for each of these groups; connecting the bridge nodes in a simple chain; and finally connecting each bridge node to every node in its two neighbouring groups. For properly chosen k, the graphs in $\mathcal{G}_{n,k}$ are of expected size linear in n.

In the rest of this section, we define redactable precedence graph families, describe the construction of the family $\mathcal{G}_{n,k}$ and prove that it possesses the properties we require, and finally show how to use these graphs in a secure redactable signature algorithm.

5.1 Redactable precedence graph families

Definition 5. *Let $V_n = \{1, 2, \dots, n\}$. A redactable precedence graph family is a collection $\mathcal{H} = \{\mathcal{H}_n\}_{n=1,2,\dots}$ of probability distributions on directed acyclic graph families, such that:*

- ***Reachability:*** *for every instance $H = (V', E') \in \mathcal{H}_n$,*
  1. *$V' \supseteq V_n$, and*
  2. *$\forall\ i, j \in V_n$, $i < j$, there is a path in E' from i to j.*
- ***Redactability****: After a choice of $H = (V', E') \xleftarrow{\$} \mathcal{H}_n$ followed by a random choice of $i \in V_n$, the result of removing node i from H along with all its incident arcs is an instance of $\mathcal{H}_{n-1}$, chosen with the correct probability distribution.*

This is a strict generalization of the graphs we used in the quadratic algorithm: If for each n, $\mathcal{H}_n$ consists of a single graph, namely the canonical tournament on n nodes, then $\{\mathcal{H}_n\}$ is a redactable precedence graph family.

In order to define our new redactable precedence graph family, we will need some notation. For nonnegative integers $a_1, \ldots, a_k$ satisfying $n = \sum_{i=1}^{k} a_i$, the *multinomial coefficient* $\binom{n}{a_1, a_2 \ldots, a_k}$ is given by

$$\binom{n}{a_1, a_2 \ldots, a_k} = \frac{n!}{a_1! a_2! \ldots a_k!}.$$

Fact 1. *For each n, the sum of all these multinomial coefficients is*

$$\sum_{a_1+a_2+\ldots+a_k=n} \binom{n}{a_1, a_2 \ldots, a_k} = k^n.$$

Next we describe a new redactable precedence graph family $\mathcal{H}_n = \mathcal{G}_{n,k}$, with k to be chosen as specified below. We define two sets of vertices, $V = \{1, \ldots, n\}$ and $B = \{b_1, \ldots, b_{k-1}\}$. The vertices $i \in V$ correspond to the respective subdocuments m_i of $\mathcal{M}$, and we call vertices $b_i \in B$ *bridge nodes*. An instance from $\mathcal{G}_{n,k}$ is defined by a choice of nonnegative integers $a_1, \ldots, a_k$ satisfying $\sum_{i=1}^{k} a_i = n$. The resulting directed graph $G_{a_1,\ldots,a_k} = (V \cup B, E)$ is defined by choosing its arcs as follows:

1. Use the the $k-1$ bridge nodes to divide the n vertices in V into k contiguous groups $g_1, \ldots, g_k$ of sizes $|g_1| = a_1, \ldots, |g_k| = a_k$, respectively. Then we have $g_1 = \{1, \ldots, a_1\}$, $g_2 = \{a_1 + 1, \ldots, a_1 + a_2\}$, ..., etc.
2. In each group g_i, construct a transitive tournament on the a_i vertices based on their canonical ordering.
3. For every group g_i, $i = 1, \ldots, k$, for every vertex $v \in g_i$, add the arc (b_{i-1}, v).
4. For every group g_i, $i = 1, \ldots, k-1$, for every vertex $v \in g_i$, add the arc (v, b_i).
5. Connect all the bridge nodes $b_i \in B$ sequentially with the canonical ordering.

Each graph $G_{a_1,\ldots,a_k}$ is chosen according to the normalized multinomial coefficient $p_{a_1,\ldots,a_k} = \binom{n}{a_1, a_2 \ldots, a_k} \cdot k^{-n}$.

To illustrate, Figure 5 shows a graph from the family $\mathcal{G}_{12,4}$. After the redaction of a node representing one of the 12 subdocuments, the result is a graph from the family $\mathcal{G}_{11,4}$, as shown in Figure 6.

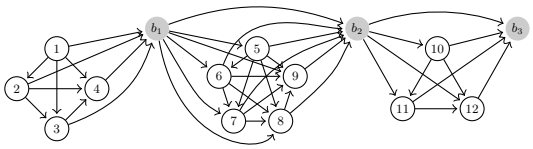

Fig. 5: a graph $G_{4,5,3,0}$ chosen from $\mathcal{G}_{12,4}$

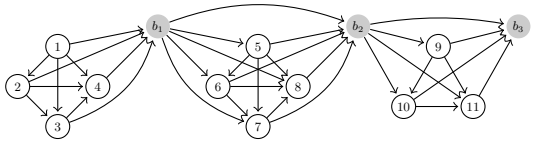

Fig. 6: $G_{4,5,3,0} \xrightarrow[\text{rename } m_{10},m_{11},m_{12}]{\text{redact } m_9} G_{4,4,3,0}$

Now we discuss the the properties of the new graph family $\mathcal{G}_{n,k}$.

Fact 2. *There are $\binom{n+k-1}{k-1}$ graphs in the graph family $\mathcal{G}_{n,k}$.*

Fact 3. *The number of arcs in graph $G_{a_1,\ldots,a_k}$ is*

$$|E_{G_{a_1,\ldots,a_k}}| = \sum_{i=1}^{k} \binom{a_i}{2} + 2\sum_{i=1}^{k} a_i - (a_1 + a_k) + (k-2)$$

Theorem 4. *The average size of a graph sampled from $\mathcal{G}_{n,k}$ according to the multinomial distribution is linear. More precisely, with $k = O(n)$, the number of edges is $\Theta(n)$.*

Theorem 4 is proved by way of a probability calculation whose details are included in the Appendix, Section A.

5.2 Details of the linear algorithm

The quadratic algorithm of Section 4 above can be regarded as operating essentially on the tournament $[1 \dots n]$. The "real nodes" of that algorithm are commitments to the edges of this tournament graph. Every algorithm step that is stated in terms of a pair (i, j) can be restated in terms of an edge of the graph. With this point of view, given a document $\mathcal{M}$ of size n, after the choice of a parameter k as described below and the choice of a graph G according to the distribution $\mathcal{G}_{n,k}$, the previous algorithm can be applied *mutatis mutandis* to the graph G.

As shown in Theorem 4 above, a graph sampled from $\mathcal{G}_{n,k}$ is of expected size $\Theta(n)$ when $k = O(n)$. But if k were picked as an exact function of n, the verifier of a possibly redacted document could infer from k the size of the original document. To avoid this, we have the signer choose k instead from a suitable interval, and quantify the uncertainty provided by this choice.

Specifically, public parameters for an instance of our linear scheme will include bounds L, U for a small interval around 1, e.g. $[L, U] = [1 - \epsilon, 1 + \epsilon]$ with $\epsilon < \frac{1}{2}$. The signer chooses c at random from $[L, U]$, takes $k = \lfloor cn \rfloor$, chooses G according to $\mathcal{G}_{n,k}$, and finally chooses a size M for the number of leaves in the GGM and Merkle trees, $M > |E(G)|$, the size of the edge set of G.

The scheme proceeds now as in our quadratic algorithm, applied to the graph G instead of to the tournament on n nodes, with the following changes in detail:

1. A canonical ordering must be chosen for the edges of any graph in the family $\mathcal{G}_{n,k}$.
2. The table T_2 contains not only hash values $\mathsf{H}(m_i)$ of subdocuments m_i, but also hash values for the individual bridge nodes. For the jth bridge node b_j, this can be computed as $\mathsf{H}(\mathsf{bridge}, j)$.
3. Redacting the ith subdocument entails removing vertex i from G along with *all* its incident edges, including arcs to and from the next and previous bridge nodes, and making the corresponding changes to the data structures in the auxiliary data aux.
4. The graph G is included in aux, updated as needed during calls to Redact.

5.3 Efficiency

As with the analysis of the efficiency of the quadratic algorithm in Section 4.3 above, the communication cost of this algorithm is dominated by the size of the graph G, which has expectation linear in n. The computation cost is dominated by the sizes of the Merkle and GGM trees, which have expectation linear in n. All three of the algorithms Sign, Verify, and Redact run in expected time $O(n) \cdot T_L + O(1) \cdot T_H$ (where, as above, T_L is the running time of a lightweight operation of symmetric cryptography, and T_H is the running time of signing or verifying an ordinary digital signature).

5.4 Security of the linear algorithm

Assuming as before that H is a collision-free hash function, S is a digital signature scheme that is adaptively secure against existential forgery, C is a secure commitment scheme, and P is a secure pseudorandom generator, we prove in this section that the scheme presented immediately above is secure.

Theorem 5. *With the assumptions of Theorem 1, the algorithm of Section 5.2 is unforgeable.*

The argument of Theorem 2 goes through as before, with the point of view that the precedence-proving graph G of the linear algorithm replaces the underlying tournament on n nodes of the quadratic algorithm.

In order to prove transparency, we need to further investigate the distribution of our graph family $\mathcal{G}_{n,k}$. We begin by showing the effect on our choice of graph of redacting a subdocument m_i from the graph chosen to represent a document of length $n + 1$.

Lemma 2. *The probability of obtaining a particular graph $G_{a_1,\dots,a_k}$ with $\sum_{i=1}^{k} a_i = n$ by sampling a graph instance from the distribution $\mathcal{G}_{n+1,k}$ and then redacting one node chosen at random from $[1 \dots n + 1]$ is identical to the probability of sampling $G_{a_1,\dots,a_k}$ from the distribution $\mathcal{G}_{n,k}$.*

The calculaton that proves Lemma 2 is included in the Appendix, Section B.

Theorem 6. *With the assumptions of Theorem 1, the algorithm presented in Section 5.2 is* $(\alpha(n), \beta(n))$ *transparent, where* $\alpha(n) = \max\{n, \frac{k}{U}\}$ *and* $\beta(n) = \min\{e(\ell, k, G), \frac{k}{L}\}$*, where* $e(\ell, k, G)$ *is the quantity defined as follows, depending on the signer's choices of* k*,* M*, and the graph* $G \in \mathcal{G}_{n,k}$*:*

$$e(\ell, k, G) = max\{\ell : \exists G' \in \mathcal{G}_{\ell,k} \text{ satisfying } G' \preceq G, |E_{G'}| \leq M\}.$$

Proof of Theorem 6. The transparency proof for the linear algorithm is similar to the proof of Theorem 3. We are given (σ, aux), a redactable signature for a document $\mathcal{M}$ of size n computed by the linear algorithm, and two documents $\mathcal{M}_0$, $\mathcal{M}_1$ of sizes $\geq n$ and both satisfying $\mathcal{M} \preceq \mathcal{M}_i$.

Let the original document's length be denoted n_{orig}. Then the bounds $\alpha(n) \geq \frac{k}{U}$ and $\beta(n) \leq \frac{k}{L}$ follow directly from the choice of $k = \lfloor cn_{\text{orig}} \rfloor$ with $c \in [L, U]$.

Assume now that $|\mathcal{M}_0|, |\mathcal{M}_1| \in [\alpha(n), \beta(n)]$. We want to show that the distributions of the two extended signatures $(i = 0, 1)$

$$\begin{aligned}(\sigma^i, aux^i) &= \mathsf{Redact}_{PK}(\mathcal{M}_i, \mathsf{Sign}_{SK}(\mathcal{M}_i), \mathcal{M}_i \backslash \mathcal{M}\} \\ &= (\sigma^i, \mathsf{T_1^i}, \mathsf{T_2}^i, \mathcal{S}_G^i, \mathcal{S}_M^i, \mathbf{x}^i, G^i_{a_1^i,\ldots,a_k^i})\end{aligned}$$

are computationally indistinguishable.

The only difference between the extended signatures generated by the linear algorithm and by the quadratic algorithm is that, for the linear algorithm, the auxilary data aux also includes the redactable precedence graph description $G_{a_1,\ldots,a_k}$. Since both (σ^0, aux^0) and (σ^1, aux^1) are valid redactable signatures for $\mathcal{M}$, the parameters of graph $G^i_{a_1^i,\ldots,a_k^i}$ $(i = 0, 1)$ satisfy

$$\sum_{j=1}^{k} a_j^0 = \sum_{j=1}^{k} a_j^1 = n.$$

As a consequence of Lemma 2, the distributions $G^0_{a_1^0,\ldots,a_k^0}$ and $G^1_{a_1^1,\ldots,a_k^1}$ are identical.

Using a similar analysis as in the transparency proof for the quadratic algorithm, we conclude that the distributions aux^0 and aux^1 are computationally indistinguishable, which further implies that the two extended signatures (σ^0, aux^0) and (σ^1, aux^1) are computationally indistinguishable. This concludes the proof. □

If $n \geq \alpha(n) = \frac{k}{U}$, then the uncertainty interval $[\alpha(n), \beta(n)]$ includes n, and thus, as is always the case for the quadratic algorithm, the verifier cannot infer whether the document being checked has been redacted at all.

However, in the case that $n < \alpha(n) = \frac{k}{U}$, the verifier knows that the document in question has been redacted, even though there may remain a long interval of possible lengths for the original document. As $k \geq L \cdot n_{\text{orig}}$, this occurs when

$$n < \frac{L \cdot n_{\text{orig}}}{U} = \frac{L}{U} \cdot n_{\text{orig}},$$

in which case the fraction of the original subdocuments that has been redacted is at least $1 - \frac{L}{U}$.

Acknowledgments

The authors would like to thank Prasad Rao and Antonio Nicolosi for their very helpful discussions during the course of the preparation of this article.

References

[ABC+12] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. In *Proc. of TCC*, number 7194 in LNCS, 2012. `http://eprint.iacr.org/2011/096`.

[ACMT05] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In *ESORICS'05*, pages 159–177, 2005.

[BBD+10] Christina Brzuska, Heike Busch, Oezguer Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable signatures for tree-structured data: definitions and constructions. In *Proceedings of the 8th international conference on Applied cryptography and network security*, ACNS'10, pages 87–104. Springer-Verlag, 2010.

[BFF+09] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of sanitizable signatures revisited. In *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography*, PKC'09, pages 317–336. Springer-Verlag, 2009.

[BFLS10] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In *Public Key Cryptography–PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*. Springer-Verlag, 2010.

[BPS14] Christina Brzuska, Henrich C. Pöhls, and Kai Samelin. Efficient and perfectly unlinkable sanitizable signatures without group signatures. In *Public Key Infrastructures, Services and Applications*, volume 8341 of *Lecture Notes in Computer Science*, pages 12–30. Springer-Verlag, 2014.

[CKLM13] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: Complex unary transformations and delegatable anonymous credentials. Cryptology ePrint Archive, Report 2013/179, 2013.

[CLX09] Ee-Chien Chang, Chee Liang Lim, and Jia Xu. Short redactable signatures using random trees. In *Proceedings of the The Cryptographers' Track at the RSA Conference 2009 on Topics in Cryptology*, CT-RSA '09, pages 133–147. Springer-Verlag, 2009.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988.

[GOT14] Esha Ghosh, Olga Ohrimenko, and Roberto Tamassia. Verifiable member and order queries on a list in zero-knowledge. *CoRR*, abs/1408.3843, 2014.

[HHH+08] 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*, ASIACCS '08, pages 353–362. ACM, 2008.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *in Advances in Cryptology - CRYPTO96, Lecture Notes in Computer Science 1109*, pages 201–215. Springer-Verlag, 1996.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In *Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, CT-RSA '02, pages 244–262. Springer-Verlag, 2002.

[KB08] Ashish Kundu and Elisa Bertino. Structural signatures for tree data structures. *Proc. VLDB Endow.*, 1(1):138–150, August 2008.

[KB10] Ashish Kundu and Elisa Bertino. How to authenticate graphs without leaking. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 609–620. ACM, 2010.

[Mar04] John Markoff. Illuminating blacked-out words. *The New York Times*. Available at `http://www.nytimes.com/2004/05/10/technology/10crypto.html`, 10 May 2004.

[Mer89] Ralph C. Merkle. A certified digital signature. In *Proceedings on Advances in cryptology*, CRYPTO '89, pages 218–238. Springer-Verlag New York, Inc., 1989.

[MHI06] Kunihiko Miyazaki, Goichiro Hanaoka, and Hideki Imai. Digitally signed document sanitizing scheme based on bilinear maps. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, pages 343–354. ACM, 2006.

[MIM+05] 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*, 88-A(1):239–246, 2005.

[MPPS14] Hermann Meer, Henrich C. Pöhls, Joachim Posegga, and Kai Samelin. On the relation between redactable and sanitizable signature schemes. In *Proceedings of the 6th International Symposium on Engineering Secure Software and Systems - Volume 8364*, ESSoS 2014, pages 113–130, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[MSI+03] Kunihiko Miyazaki, Seichi Susaki, Mitsuru Iwamura, Tsutomu Matsumoto, Ryôichi Sasaki, and Hiroshi Yoshiura. Digital Documents Sanitizing Problem. Technical Report ISEC2003-20, IEICE, 2003.

[NIS12] NIST. *FIPS PUB 180-4: Secure Hash Standard (SHS)*. National Institute of Standards and Technology, March 2012.

[NW04] David Naccache and Claire Whelan. Who alerted the CIA? (And other secret secrets), May 2004. Rump session talk at Eurocrypt 2004, Interlaken, Switzerland.

[PS14] Henrich C. Pöhls and Kai Samelin. On updatable redactable signatures. In *ACNS*, pages 457–475. Springer-Verlag, 2014.

[SBZ01] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In *ICISC*, pages 285–304, 2001.

[SPB+12] Kai Samelin, Henrich C. Phls, Arne Bilzhause, Joachim Posegga, and Hermann de Meer. On structural signatures for tree data structures. In *Proceedings of the Applied Cryptography and Network Security 2012*, ACNS 2012, pages 171–187, Berlin, Heidelberg, 2012. Springer-Verlag.

Appendix

A Average graph size

Theorem (Theorem 4). *The average size of a graph sampled from $\mathcal{G}_{n,k}$ according to the multinomial distribution is subquadratic. More precisely, with $k = O(n)$, the number of edges is $\Theta(n)$.*

Proof of Theorem 4. Let $E_{a_1,\ldots,a_k}$ denote the set of arcs for graph $G_{a_1,\ldots,a_k}$. The average number of arcs is

$$|\bar{E}| = \sum_{G_{a_1,\ldots,a_k}} |E_{a_1,\ldots,a_k}| \cdot p_{a_1,\ldots,a_k} \tag{1}$$

$$= \sum_{\substack{a_1+\ldots+a_k=n \\ a_i\geq 0}} \left(\left(\sum_{i=1}^{k} \binom{a_i}{2} + 2\sum_{i=1}^{k} a_i - (a_1+a_k) + (k-2) \right) \cdot \binom{n}{a_1,\ldots,a_k} \cdot k^{-n} \right) \tag{2}$$

$$= k^{-n} \cdot \sum_{\substack{a_1+\ldots+a_k=n \\ a_i\geq 0}} \left(\sum_{i=1}^{k} \binom{a_i}{2} - (a_1+a_k) \right) \cdot \binom{n}{a_1,\ldots,a_k} + k^{-n} \cdot (2n+k-2) \cdot \sum_{\substack{a_1+\ldots+a_k=n \\ a_i\geq 0}} \binom{n}{a_1,\ldots,a_k} \tag{3}$$

$$= \frac{1}{2} k^{-n} \cdot \sum_{\substack{a_1+\ldots+a_k=n \\ a_i\geq 0}} \left(\sum_{i=1}^{k} a_i \cdot (a_i-1) \right) \cdot \binom{n}{a_1,\ldots,a_k} - k^{-n} \cdot \sum_{a_1+\ldots+a_k=n} (a_1+a_k) \cdot \binom{n}{a_1,\ldots,a_k} + 2n + k - 2 \tag{4}$$

Equation 1 follows from summing over all graphs $G_{a_1,\ldots,a_k} \in \mathcal{G}_{n,k}$ the product of the number of arcs in $G_{a_1,\ldots,a_k}$ and the probability of obtaining $G_{a_1,\ldots,a_k}$. Equation 2 follows from Fact 3. Equation 3 is rewritten from Equation 2 by separating two terms and pulling out the constants. Equation 4 follows from Fact 1 and further separation of terms.

To compute $|\bar{E}|$, we first need to separately compute the major components of Equation 4.

$$X_j = \sum_{\substack{a_1+a_2+\ldots+a_k=n \\ \forall a_i\geq 0}} \left(a_j \cdot (a_j-1) \cdot \binom{n}{a_1,a_2,\ldots,a_k} \right)$$

$$= \sum_{\substack{a_1+a_2+\ldots+a_k=n \\ \forall a_i\geq 0, i\neq j;\ a_j=0 \text{ or } 1}} \left(a_j \cdot (a_j-1) \cdot \binom{n}{a_1,a_2,\ldots,a_k} \right) + \sum_{\substack{a_1+\ldots+a_k=n \\ \forall a_i\geq 0, i\neq j;\ a_j\geq 2}} \left(a_j \cdot (a_j-1) \cdot \binom{n}{a_1,\ldots,a_k} \right) \tag{5}$$

$$= 0 + \sum_{\substack{a_1+\ldots+a_k=n \\ \forall a_i\geq 0, i\neq j;\ a_j\geq 2}} \frac{n!}{a_1! \cdot a_2! \ldots a_{j-1}! (a_j-2)! a_{j+1}! \ldots a_k!} \tag{6}$$

$$= \sum_{\substack{a_1+\ldots+a_j+a_k=n-2 \\ \forall a_i\geq 0}} \frac{n \cdot (n-1) \cdot (n-2)!}{a_1! \cdot a_2! \ldots a_{j-1}! a_j! a_{j+1}! \ldots a_k!} \tag{7}$$

$$= n \cdot (n-1) \cdot \sum_{\substack{a_1+\ldots+a_k=n-2 \\ \forall a_i\geq 0}} \binom{n-2}{a_1,\ldots,a_k} \tag{8}$$

$$= n \cdot (n-1) \cdot k^{(n-2)} \tag{9}$$

Equation 6 follows from the fact that a_j or $a_j - 1$ is equal to 0 in the first term, and the fact that $\frac{a_j(a_j-1)}{a_j!} = \frac{1}{(a_j-2)!}$ in the second term. Equation 7 follows by substituting a_j for $a_j - 2$. Equation 8 uses the definition of the multinomial coefficient. Equation 9 follows from Fact 1.

$$\begin{aligned}
Y_1 &= \sum_{\substack{a_1+a_2+\ldots+a_k=n \\ \forall a_i \geq 0}} \left(a_1 \cdot \binom{n}{a_1, a_2, \ldots, a_k} \right) \\
&= \sum_{\substack{a_1+a_2+\ldots+a_k=n \\ \forall a_i \geq 0, i \neq 1;\ a_1 = 0}} \left(a_1 \cdot \binom{n}{a_1, a_2, \ldots, a_k} \right) + \sum_{\substack{a_1+a_2+\ldots+a_k=n \\ \forall a_i \geq 0, i \neq 1;\ a_1 \geq 1}} \left(a_1 \cdot \binom{n}{a_1, a_2, \ldots, a_k} \right) \qquad (10) \\
&= 0 + \sum_{\substack{a_1+a_2+\ldots+a_k=n \\ \forall a_i \geq 0, i \neq 1;\ a_1 \geq 1}} \frac{n!}{(a_1-1)! \cdot a_2! \ldots a_k!} = \sum_{\substack{a_1+\ldots+a_j+\ldots+a_k=n-1 \\ \forall a_i \geq 0}} \frac{n \cdot (n-1)!}{a_1! \cdot a_2! \ldots a_k!} \\
&= n \cdot \sum_{\substack{a_1+a_2+\ldots+a_k=n-1 \\ \forall a_i \geq 0}} \binom{n-1}{a_1, a_2, \ldots, a_k} = n \cdot k^{(n-1)} \qquad (11)
\end{aligned}$$

Equation 10 follows from separating the cases that $a_1 \geq 1$ and $a_1 = 0$. The second part of Equation 11 follows from Fact 1. Similarly,

$$Y_k = \sum_{\substack{a_1+a_2+\ldots+a_k=n \\ \forall a_1 \geq 0}} \left(a_k \cdot \binom{n}{a_1, a_2, \ldots, a_k} \right) = n \cdot k^{(n-1)}$$

Plugging X_j, Y_1 and Y_k into Equation 4, we obtain the expected number of edges:

$$\begin{aligned}
|\bar{E}| &= \frac{1}{2} k^{-n} \cdot \sum_{j=1}^{k} X_j - k^{-n} \cdot (Y_1 + Y_k) + 2n + k - 2 \\
&= \frac{1}{2} k^{-n} \cdot (k \cdot n \cdot (n-1) \cdot k^{(n-2)}) - k^{-n} \cdot (2 \cdot n \cdot k^{(n-1)}) + 2n + k - 2 \\
&= \frac{n \cdot (n-1)}{2k} - \frac{2n}{k} + 2n + k - 2 = \frac{n \cdot (n-5)}{2k} + 2n + k - 2 \qquad (12)
\end{aligned}$$

Differentiating the last part of Equation 12 and setting it equal to 0, we obtain

$$2k^2 = n(n-5).$$

Solving this equation, we find that the optimal number of groups is $k = O(n) \approx 2^{-\frac{1}{2}} n \approx 0.7n$, with resulting expectation $|\bar{E}| = O(n)$. □

B Indistinguishability of graph $G_{a_1,\ldots,a_k}$

Lemma (Lemma 2).

The probability of obtaining a particular graph $G_{a_1,\ldots,a_k}$ with $\sum_{i=1}^{k} a_i = n$ by sampling a graph instance from the distribution $\mathcal{G}_{n+1,k}$ and then redacting one node i chosen at random from $[1 \ldots n+1]$ is identical to the probability of sampling $G_{a_1,\ldots,a_k}$ from the distribution $\mathcal{G}_{n,k}$.

Proof of Lemma 2. According to the graph sampling definition, the probability of choosing $G_{a_1,\ldots,a_k} \in \mathcal{G}_{n,k}$ is $p_{a_1,\ldots,a_k} = \binom{n}{a_1,a_2\ldots,a_k} \cdot k^{-n}$.

If $G_{a_1,\ldots,a_k} \in \mathcal{G}_{n,k}$ is redacted from a graph in $\mathcal{G}_{n-1,k}$ by deleting any one subdocument vertex, there are k possibilities:

Redact one vertex from group g_1 in graph $G_{a_1+1,\ldots,a_k}$
Redact one vertex from group g_2 in graph $G_{a_1,a_2+1,\ldots,a_k}$
...
Redact one vertex from group g_k in graph $G_{a_1,\ldots,a_{k-1},a_k+1}$

Since every subdocument vertex m_i, $i = 1, \ldots, n+1$, may be redacted with equal probability, we find that the probability of obtaining $G_{a_1,\ldots,a_k}$ by redaction is

$$
\begin{aligned}
p'_{a_1,\ldots,a_k} &= \frac{a_1+1}{n+1} \cdot p_{a_1+1,a_2,\ldots,a_k} + \sum_{i=2}^{k-1} \frac{a_i+1}{n+1} \cdot p_{a_1,\ldots,a_i,a_i-1,a_{i+1},\ldots,a_k} + \frac{a_k+1}{n+1} \cdot p_{a_1,\ldots,a_{k-1},a_k+1} \\
&= \frac{a_1+1}{n+1} \cdot \binom{n+1}{a_1+1, a_2, \ldots, a_k} \cdot k^{-(n+1)} + \sum_{i=2}^{k-1} \frac{a_i+1}{n+1} \cdot \binom{n+1}{a_1, \ldots, a_{i-1}, a_i+1, a_{i+1}, \ldots, a_k} \cdot k^{-(n+1)} \\
&\quad + \frac{a_k+1}{n+1} \cdot \binom{n+1}{a_1, \ldots, a_{k-1}, a_k+1} \cdot k^{-(n+1)} \\
&= k^{-(n+1)} \cdot \left(\frac{a_1+1}{n+1} \cdot \frac{(n+1)!}{(a_1+1)!a_2! \ldots a_k!} + \sum_{i=2}^{k-1} \frac{a_i+1}{n+1} \cdot \frac{(n+1)!}{a_1! \ldots a_{i-1}!(a_i+1)!a_{i+1}! \ldots a_k!} \right. \\
&\quad \left. + \frac{a_k+1}{n+1} \cdot \frac{(n+1)!}{a_1! \ldots a_{k-1}!(a_k+1)!} \right) \\
&= k^{-(n+1)} \cdot \sum_{i=1}^{k} \frac{n!}{a_1! \ldots a_{i-1}! a_i! a_{i+1}! \ldots a_k!} = k^{-(n+1)} \cdot \left(k \cdot \frac{n!}{a_1! a_2! \ldots a_k!} \right) \\
&= k^{-n} \cdot \binom{n}{a_1, a_2 \ldots, a_k} \;=\; p_{a_1,\ldots,a_k}
\end{aligned}
$$

□