Homomorphic Signature Schemes

Robert Johnson[1], David Molnar[2], Dawn Song[1*], and David Wagner[1]

[1] University of California at Berkeley
[2] ShieldIP

Abstract. Privacy homomorphisms, encryption schemes that are also homomorphisms relative to some binary operation, have been studied for some time, but one may also consider the analogous problem of homomorphic signature schemes. In this paper we introduce basic definitions of security for homomorphic signature systems, motivate the inquiry with example applications, and describe several schemes that are homomorphic with respect to useful binary operations. In particular, we describe a scheme that allows a signature holder to construct the signature on an arbitrarily redacted submessage of the originally signed message. We present another scheme for signing sets that is homomorphic with respect to both union and taking subsets. Finally, we show that any signature scheme that is homomorphic with respect to integer addition must be insecure.

1 Introduction

A cryptosystem $f : G \to R$ defined on a group $(G, \cdot)$ is said to be homomorphic if f forms a (group) homomorphism. That is, given $f(x)$ and $f(y)$ for some unknown $x, y \in G$, anyone can compute $f(x \cdot y)$ without any need for the private key. Somewhat surprisingly, this property has a wide range of applications, including secure voting protocols [8] and multiparty computation [26].

In a series of talks, Rivest suggested the investigation of homomorphic *signature* schemes. For instance, the RSA signature scheme is a group homomorphism, as $m_1^d \cdot m_2^d = (m_1 \cdot m_2)^d$. This property was previously considered to be undesirable and much energy has been spent on eliminating it [5]. The question is whether this property can be put to positive use instead. More generally, Rivest asked whether homomorphic signature schemes with positive applications can be found.

Our goal is to shed light on this question. In the process, we give a formal definition of what it means to be a secure homomorphic signature scheme (see Section 3). Then, we construct a redactable signature scheme where, given a signature $\mathsf{Sig}(x)$, anyone can compute a signature $\mathsf{Sig}(w)$ on any redaction w of x obtained by rubbing out some positions of x (Section 4). We give proofs of

* This research was supported in part by the Defense Advanced Research Projects Agency under DARPA contract N6601-99-28913 (under supervision of the Space and Naval Warfare Systems Center San Diego) and by the National Science foundation under grants FD99-79852 and CCR-0093337.

security for this scheme (Appendix A). We present a scheme for signing sets that allows anyone to compute the signature on the union of two signed sets, and the signature on any subset of a signed set, and corresponding proofs of security (Section 5).

We also consider additively homomorphic signature schemes $\mathsf{Sig} : \mathbb{Z}/m\mathbb{Z} \to R$. We argue that all such schemes are insecure: they unavoidably possess properties that are likely to render them useless in practice (Section 6). The problematic properties of additive signature schemes are general and completely independent of the way the scheme is implemented. In response, we define and consider *semigroup-homomorphic* signature schemes, which would permit us to avoid these bad properties. We pose as an open problem to find a signature scheme that is semigroup-homomorphic but not group-homomorphic.

2 Related Work

The notion of homomorphic signature schemes was first given by Rivest in a series of talks on "two new signature schemes" [24]. The first of these signature schemes, due to Micali and Rivest, is a "transitive signature scheme" for undirected graphs [19]. In this scheme, given a signature on two graph edges $\mathsf{Sig}((x, y))$, $\mathsf{Sig}((y, z))$, a valid signature $\mathsf{Sig}((x, z))$ on any edge in their transitive closure can be computed without access to the secret key. This scheme works only for undirected graphs; given signatures on the transitive closure edge $\mathsf{Sig}((x, z))$ and the edge $\mathsf{Sig}((x, y))$, a signature on the "intermediate" edge $\mathsf{Sig}((y, z))$ can be computed It was left as an open problem to find a similar scheme for *directed* graphs.

The second signature scheme, due to Rivest, Rabin, and Chari, allows "prefix aggregation" [24]. Given two signatures $\mathsf{Sig}(p||0)$ and $\mathsf{Sig}(p||1)$ on the two messages obtained from p by appending a zero and one bit, their scheme allows computation of a signature $\mathsf{Sig}(p)$ on p without access to the secret key. The scheme as presented has a property similar to the transitive graph signature scheme: the signature $\mathsf{Sig}(p||1)$ can be easily computed. from $\mathsf{Sig}(p)$ and $\mathsf{Sig}(p||0)$. It was left as an open problem to find a similar scheme that does not have this property. Rivest also posed the open problem of finding a "concatenation signature scheme," in which given two signatures $\mathsf{Sig}(x)$ and $\mathsf{Sig}(y)$ a signature $\mathsf{Sig}(x||y)$ on their concatenation can be computed.

Rivest also suggested investigating what other "signature algebras" can be constructed. In Section 4 we give a construction for "redactable signatures." Then in Section 5 we show that RSA accumulators can be used to construct signatures homomorphic with respect to the union and subset operations.

Homomorphic signature schemes are intriguing in part because homomorphic cryptosystems have proved to be so useful. Rivest, Adleman, and Dertouzos noted applications of "privacy homomorphisms" to computing on encrypted data soon after the introduction of RSA [25]. Peralta and Boyar showed that an XOR-homomorphic bit commitment could be exploited to yield more efficient zero-knowledge proofs of circuit satisfiability [23]. Feigenbaum and Merritt noted that a "cryptosystem which is a ring homomorphism on $\mathbb{Z}/2\mathbb{Z}$ could be used to implement completely non-interactive secure circuit evaluation" and called such cryptosystems "algebraic" [14]. Benaloh gave a secure election scheme based on a homomorphic encryption scheme [12]. Cramer and Damgard use homomorphic bit commitments to drastically simplify zero-knowledge proofs [13]. Many other examples exist in which homomorphic properties are used to construct cryptographic protocols.

The initial promise of privacy homomorphisms was tempered by a string of negative results. Ahituv, Lapid, and Neumann showed that any cryptosystem that is XOR-homomorphic on $GF(2^{64})$ is insecure under chosen ciphertext attack [1]. Boneh and Lipton showed that any deterministic cryptosystem that is a field homomorphism must fall victim to a subexponential attack [10]. They further conjectured that any field-homomorphic cryptosystem, which they called "completely malleable," would prove to be insecure. Brickell and Yacobi broke a number of candidate constructions of privacy homomorphisms [11]. These negative results have their analogue in our results of Section 6 showing the triviality of signature schemes that are group homomorphisms on $(\mathbb{Z}, +)$.

Besides RSA, several other homomorphic cryptosystems are currently known. Goldwasser-Micali encryption takes the form of a group homomorphism $\mathbb{Z}/2\mathbb{Z} \to (\mathbb{Z}/n\mathbb{Z})^*$ [17], and others have proposed a number of other public-key encryption schemes that have various useful homomorphic properties [15, 8, 22, 20]. Of particular interest is Sander, Young, and Yung's slick construction of an encryption algorithm that is both AND- and XOR-homomorphic [26]; they note that this is the first cryptosystem homomorphic over a semigroup.

Redactable signature schemes are related in both spirit and construction to the "incremental cryptography" of Goldwasser, Goldreich, and Bellare [3]. Our notion of privacy for redactable signatures has a parallel in their notion of privacy for incremental signatures [4].

3 Definitions

We define the notion of a homomorphic signature scheme as follows. A specification of a signature scheme includes a message space $\mathcal{M}$, a set of private keys $\mathcal{K}$, a set of public keys $\mathcal{K}'$, a (possibly randomized) signature algorithm $\mathsf{Sig} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, and a verification algorithm $\mathsf{Vrfy} : \mathcal{K}' \times \mathcal{M} \times \mathcal{Y} \to \{\mathsf{ok}, \mathsf{bad}\}$ so that $\mathsf{Vrfy}(k', x, \mathsf{Sig}(k, x)) = \mathsf{ok}$ for all $x \in \mathcal{M}$ when (k, k') is a matching private key and public key. As a notational matter, for conciseness we often omit the private and public keys, writing $\mathsf{Sig}(x)$ instead of $\mathsf{Sig}(k, x)$ and $\mathsf{Vrfy}(x, s)$ instead of $\mathsf{Vrfy}(k', x, s)$ when this abbreviation will not cause confusion. Also, for a binary operation $\odot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ and a set $S \subseteq \mathcal{M}$, we let $\mathrm{span}_{\odot}(S)$ denote the least set T with $S \subseteq T$ and $x \odot y \in T$ for all $x, y \in T$.

Definition 1. *Fix a signature scheme* $\mathsf{Sig} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$, $\mathsf{Vrfy} : \mathcal{K}' \times \mathcal{M} \times \mathcal{Y} \to \{\mathsf{bad}, \mathsf{ok}\}$ *and a binary operation* $\odot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$. *We say that* Sig *is homomorphic with respect to* $\odot$ *if it comes with an efficient family of binary operations* $\otimes_{k'} : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$ *so that* $y \otimes_{k'} y' = \mathsf{Sig}(x \odot x')$ *for all* x, x', y, y' *satisfying* $\mathsf{Vrfy}(x, y) = \mathsf{Vrfy}(x', y') = \mathsf{ok}$.

As an example, if $(G, \times_G)$ and $(R, \times_R)$ are two groups and we have a signature scheme $\mathsf{Sig} : \mathcal{K} \times G \to R$ that is also a group homomorphism from G to R for each $k \in \mathcal{K}$, then this will qualify as a signature scheme that is homomorphic with respect to $\times_G$, since we may take $y \otimes_{k'} y' = y \times_R y'$.

This definition requires that signatures derived via $\otimes_{k'}$ be indistinguishable from signatures generated by the private key holder, which we refer to as history-independence. This precludes trivial schemes that, for example, allow the ordered pair (y, y') to serve as a signature on $x \odot x'$. Although the definition above is for deterministic signature schemes, extending it to probabilistic schemes is straightforward; one can simply require that the distribution of $y \otimes_{k'} y'$ be indistinguishable from that of $\mathsf{Sig}(x \odot x')$.

For homomorphic signature schemes, we need a new definition of security. The standard notion of security against existential forgeries is too strong: no homomorphic signature scheme could ever satisfy it, because given two signatures on messages x and x', one can generate a signature on the new message $x'' = x \odot x'$ without asking the signer for a signature on x'' explicitly.

Fortunately, there is a natural extension. Suppose the adversary has obtained signatures on queries $x_1, \ldots, x_q$. Such an adversary can deduce signatures on $x_i \odot x_j$, $(x_i \odot x_j) \odot x_k$, and so on; in other words, no message in $\mathrm{span}_\odot(x_1, \ldots, x_q)$ seems to be safe. Therefore, we will require that no adversary be able to deduce a valid signature on anything outside $\mathrm{span}_\odot(x_1, \ldots, x_q)$.

Definition 2. *We say that a homomorphic signature scheme* Sig *is* (t, q, ϵ)*-secure against existential forgeries with respect to* $\odot$ *if every adversary* A *making at most* q *chosen-message queries and running in time at most* t *has advantage* $Adv\,A \le \epsilon$*. The advantage of an adversary* A *is defined as the probability that, after queries on the messages* $x_1, \ldots, x_q$*,* A *outputs a valid signature* $\langle x', y' \rangle$ *on some message* $x' \notin \mathrm{span}_\odot(x_1, \ldots, x_q)$*. In other words,* $Adv\,A = \Pr[A^{\mathsf{Sig}(k,\cdot)} = \langle x', y' \rangle \wedge \mathsf{Vrfy}(x', y') = \mathsf{ok} \wedge x' \notin \mathrm{span}_\odot(x_1, \ldots, x_q)]$.

In some cases, we might want an additional guarantee that the operation $\odot$ does not allow the adversary to create too many new signatures. Consider the additive signature schemes broken later in Section 6: They are good candidates to satisfy Definition 2, yet knowledge of a single signature could allow an attacker to sign every other possible message. Therefore, we introduce the concept of security against random forgeries:

Definition 3. *The signature scheme* $\mathsf{Sig} : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ *is said to be* (t, q, ϵ)*-secure against random forgeries if every adversary* A *making at most* q *chosen-message queries and running in time at most* t *has advantage* $Adv\,A \le \epsilon$*. The advantage of an adversary is the probability that it can output a valid signature on a new message* x' *chosen by the referee uniformly at random from* $\mathcal{M}$ *and independently of all the adversary's chosen-message queries. In other words,* $Adv\,A = \Pr[\mathsf{Vrfy}(x', A^{\mathsf{Sig}(k,\cdot), c_{x'}}) = \mathsf{ok}]$ *where* $c_{x'}$ *is the constant function that always returns* x' *and where the adversary is not allowed to make any further queries to its signing oracle after looking at the value* x'*.*

For example, one might reasonably conjecture that textbook RSA signatures are secure against random forgeries.

The security of a homomorphic signature scheme seems to depend on two related notions: the size of $\mathrm{span}_{\odot}(x_1, \ldots, x_q)$ for small sets of messages, and the difficulty of the decomposition problem, i.e. given $x \in \mathrm{span}_{\odot}(x_1, \ldots, x_q)$, find an explicit decomposition of x in terms of the x_i's and the $\odot$ operation. If decomposition is hard, then the scheme may be secure even if the spans of small sets of messages are quite large, as is the case with RSA. The scheme may also be secure if the decomposition problem is easy, but most sets of messages only span a small portion of the message space. Redactable signatures are just such a scheme. The trouble occurs when a scheme admits both easy decomposition and large spans. Indeed, this is exactly what renders additive schemes so vulnerable to random forgeries.

One can easily generalize the above notions to operations that take an arbitrary number of operands, and to schemes that are simultaneously homomorphic with respect to more than one operation.

We can also consider signature schemes that are homomorphic with respect to a number of interesting mathematical structures. If $(G, \times_G)$ is a group, then we say that $\mathsf{Sig} : G \rightarrow \mathcal{Y}$ is a group-homomorphic signature scheme if it is homomorphic with respect to the binary operator $\times_G$ as well as the unary inversion operation $x \mapsto x^{-1}$. If $(M, \times_M)$ is a semigroup[1], then we say that $\mathsf{Sig} : M \rightarrow \mathcal{Y}$ is a semigroup-homomorphic signature scheme if it is homomorphic with respect to the binary operator $\times_M$. If $(R, \times_R, +_R)$ is a ring, we say that $\mathsf{Sig} : R \rightarrow \mathcal{Y}$ is a ring-homomorphic signature scheme if it is homomorphic with respect to both $\times_R$ and $+_R$. Boneh et al. have shown that every[2] field-homomorphic signature scheme $\mathsf{Sig} : \mathbb{F} \rightarrow \mathcal{Y}$ can be broken in subexponential time [10].

4 Redactable Signatures

The problem. Redactable signatures are intended to model a situation where a censor can delete certain substrings of a signed document without destroying the ability of the recipient to verify the integrity of the resulting (redacted) document. In particular, we allow the censor to replace arbitrary bit positions in the document with a special symbol $\sharp$ representing the location of the deletions. (Our construction can be readily generalized to any alphabet, so that the signer can limit redactions to whole words, sentences, etc., but for simplicity we describe only the case of bit strings here.)

Redactable signatures might have several applications. They permit deletion of arbitrary substrings of a document, and thus might be useful in proxy cryp-

---

[1] Recall that a *semigroup* is a set M together with an associative binary operator $\times_M$ with the property that M is closed under $\times_M$.

[2] Boneh et al. noted that their attack applies to all *deterministic* field-homomorphic encryption schemes, but not to randomized encryption schemes. We observe that their result also applies to all field-homomorphic signature schemes, whether the signature scheme is deterministic or randomized.

Alice spends 60 hours a week trying to find ways to add value to our bottom line, and never have I known her to shirk her duties. Alice is a true asset to our company, and I cannot think of one person better suited to your requirements.	Alice spends 60 hours a week trying to find ways to shirk her duties, and I can think of one person better suited to your requirements.	Alice spends 60 hours a week trying to find ways to ♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯ ♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯ ♯♯♯ shirk her duties ♯♯♯♯♯♯ ♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯♯ ♯♯, and I can♯♯♯ think of one person better suited to your requirements.

Fig. 1. An illustration of the need to disclose the location of redactions. The left shows a sample document that one might sign with a redactable signature scheme. The middle shows one substring that might be obtained if the location of redactions is not made explicit in the result; note how the meaning of the original document has been violated. The right shows the corresponding redaction obtained if deletions are disclosed explicitly; note how the attempted trickery is revealed by this countermeasure. Because of this potential for this sort of semantic attack when the location of deletions are not made explicit, as illustrated here, all our constructions follow the model shown on the right of making deletions explicit.

tography or incremental cryptography. As their name suggests, they also permit redaction and censorship of signed records with an untrusted censor: the censor cannot forge documents, except those obtained by redaction of a legitimately signed document.

Semantic attacks and our formulation of redaction. Making the problem statement more precise exposes a subtle trap here. A natural first attempt at a definition might require that, given a signature $\mathsf{Sig}(x)$ on x, one can obtain a signature $\mathsf{Sig}(w)$ on any substring w of x that is obtained by deleting some of the symbols in x. However, this formulation of the problem has a serious limitation: such a scheme will conceal the presence of deletions, which introduces the risk of semantic attacks. Without indication of where the redaction has occurred, an attacker might legally be able to truncate the end of one sentence, delete the beginning of the next, and slice them together to get a message the sender would not have authorized. For an example, see Figure 1. To defeat semantic attacks, we instead use a formulation where the presence of redacted segments are made explicit.

We define the concept more formally as follows. Let us take $\Sigma = \{0, 1, \sharp\}$. We define a partial order $\preceq$ on Σ so that $\sharp \preceq 0$, $\sharp \preceq 1$, and $a \preceq a$ for each $a \in \Sigma$ (and no other non-trivial relations hold). This induces a partial order $\preceq$ on Σ^* by pointwise comparison, namely, $w_1 \cdots w_n \preceq x_1 \cdots x_n$ holds if $w_i \preceq x_i$ for each i. We say that the document w is a *redaction* of x if $w \preceq x$, or in other words, if w can be obtained from x by replacing some positions of x with the $\sharp$ symbol. The signature scheme $\mathsf{Sig} : \Sigma^* \to R$ is called *redactable* if we can derive from a signature $\mathsf{Sig}(x)$ on x a signature $\mathsf{Sig}(w)$ on any redaction $w \preceq x$ we like, without the help of the signer.

Note that, in this model, we do not attempt to hide the location of the censored portions of the documents. Thus, a signature $\mathsf{Sig}(w)$ on a redaction w of x may legitimately reveal which portions of x were deleted, and specifically, the presence and location of redacted segments is protected from tampering by the signature. However, it does not reveal the previous contents of the redacted portions of the document. We expect that this privacy property may be important to many applications.

Redactable signatures may be viewed as an instance of a homomorphic signature scheme endowed with operations $D_i : \Sigma^* \to \Sigma^*$ that replace the i-th bit position with a $\sharp$ symbol. The requirement that signed documents be redactable is equivalent to requiring that our signature scheme be homomorphic with respect to these unary operators.

A trivial construction. There is one obvious way to build redactable signatures. We fix a traditional signature scheme Sig_0; no special homomorphic properties are required from Sig_0. We assume for simplicity that our base signature scheme permits message recovery, but this assumption is not essential. In the trivial construction, to sign a message x of length n, we generate a fresh key pair (s, v) for some secure conventional signature scheme, and then the signature on x is

$$\mathsf{Sig}(x) = \langle \mathsf{Sig}_0(n, v), s(1, x_1), \ldots, s(n, x_n) \rangle.$$

Verification is straightforward.

The key pair essentially serves as a document ID. Without it, an attacker could replace the ith component of any signed message with the ith component of any other signed message, which we do not wish to allow.

To redact a signed message, we simply erase the appropriate segments of the message and the corresponding positions in the signature. For instance, redacting the first symbol in $\mathsf{Sig}(x)$ yields a signature $\langle \mathsf{Sig}_0(n, v), s(2, x_2), \ldots, s(n, x_n) \rangle$. This scheme supports a privacy property: redacted signatures do not reveal the redacted parts of the original message. We can see that they reveal the locations that have been redacted, much like typical redaction of paper documents does, but this is all that is leaked.

However, this trivial scheme has a serious limitation: the signature $\mathsf{Sig}(x)$ is very long. If s produces m-bit signatures, then the construction above yields signatures of length $mn + O(1)$, which is likely to be large compared to the length of the message, n. Therefore, the challenge is to find a scheme with much shorter signatures.

Our construction. We describe how to build a secure redactable signature scheme out of any traditional signature scheme. The main idea is to combine Merkle hash trees [18] with the GGM tree construction for increasing the expansion factor of a pseudorandom generator [21]. We first generate a pseudorandom value at each leaf by working down the tree (using GGM), then we compute a hash at the root by working up the tree (using Merkle's tree hashing).

We describe in detail how to sign a message x of length n. First, we arrange the symbols of the message at the leftmost leaves of a binary tree, with each leaf

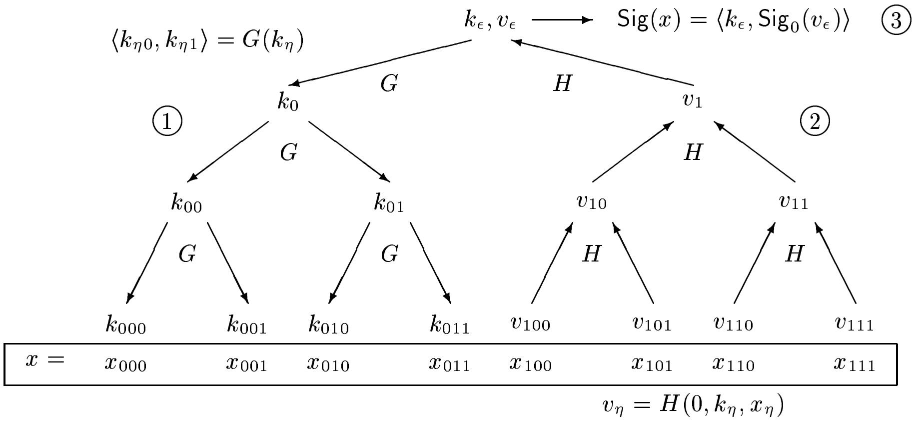

Fig. 2. A redactable signature on the message $x = \langle x_{000}, x_{001}, \ldots, x_{111}\rangle$. The message is signed in three phases: first, we generate k-values by recursing down the tree with the PRG G; then, we generate v-values by recursing up the tree with the hash H; finally, we sign v_ϵ using our conventional signature scheme. To avoid cluttering the diagram, we show the downward phase only on the left branch of the tree, and the upwards phase only on the right branch of the tree, but the full scheme requires we traverse both branches in both directions.

at depth $\lceil \lg n \rceil$. We identify nodes of the tree with elements of $\{0,1\}^*$, so that a node η has children $\eta 0$ and $\eta 1$, and the empty string ϵ denotes the root node. Let $G : \mathcal{K} \to \mathcal{K} \times \mathcal{K}$ be a length-doubling pseudorandom generator, let H be a cryptographic hash function, and pick $k_\epsilon \in \mathcal{K}$ uniformly at random. We use a three-phase algorithm (see Figure 2, which shows phase one on the left and phase two on the right):

Expansion: We use the GGM tree construction to associate a key k_η to each node η. In other words, we define a key for each node of the tree by the recursive relation $\langle k_{\eta 0}, k_{\eta 1}\rangle = G(k_\eta)$.

Hashing: We then compute a hash value v_ℓ for each leaf a as $v_\ell = H(0, k_\ell, x_\ell)$ and apply the Merkle hash construction, i.e., we recursively compute $v_\eta = H(1, v_{\eta 0}, v_{\eta 1})$ (or, if only a left child exists, $H(1, v_{\eta 0})$).

Signing: Finally, we sign the root of the hash tree using our base signature scheme Sig_0, and compute the signature on x as $\mathsf{Sig}(x) = \langle k_\epsilon, \mathsf{Sig}_0(v_\epsilon)\rangle$.

Verification is straightforward, since given k_ϵ and x we can compute v_ϵ and check the signature.

Next, we describe how to redact a signed message. To erase position ℓ from a signed message x, we need to reveal v_ℓ. At first sight, revealing $v_\ell = H(0, k_\ell, x_\ell)$ would appear to be very dangerous, since given k_ϵ we can compute k_ℓ and then iterate over all possible values of x_ℓ to learn the value of the erased symbol. This would violate the secrecy property.

To restore secrecy, we take advantage of the GGM tree. When we erase x_ℓ, we will also erase k_ϵ, and reveal only what is needed to verify the signature without

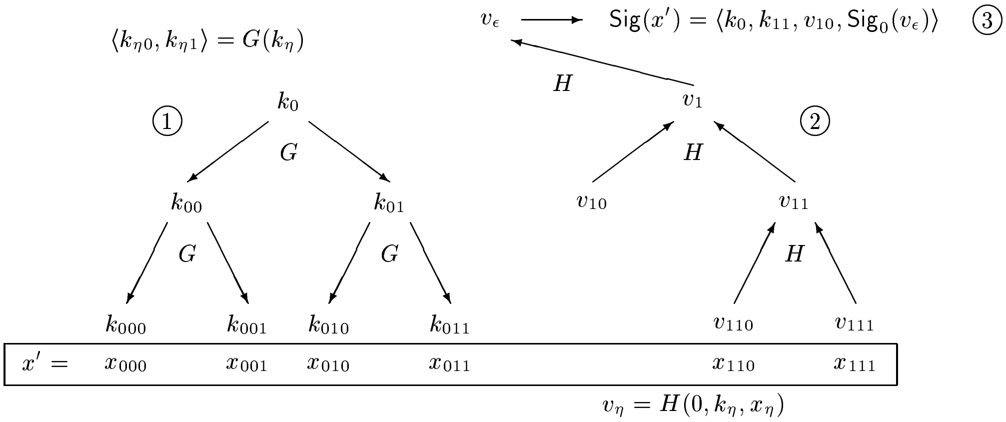

Fig. 3. A redaction of the signature on x, with the two bits x_{100} and x_{101} deleted. Like in Fig. 2, we show only part of the signature process, to avoid cluttering the diagram.

leaking k_ℓ. The notion we need is as follows: define the *co-nodes* associated to a leaf ℓ to be the siblings of the nodes along the path from ℓ to the root. We reveal k_η at each co-node η associated to ℓ, as this is exactly what is needed to check the resulting signature. Thus, redacting x_ℓ from $\mathsf{Sig}(x)$ would yield $\langle v_\ell, \{k_\eta : \eta \text{ a co-node of } \ell\}, \mathsf{Sig}_0(v_\epsilon)\rangle$. Note that there are at most $\lg n$ co-nodes associated to any single leaf.

We can repeat the above procedure to further redact an already-redacted signature, if we like. In the case of redactions in consecutive parts of the document, we compact the signature by using the tree structure: if the signature contains simultaneously $v_{\eta 0}$ and $v_{\eta 1}$ for some η, then we replace these two quantities with the single value v_η, and we similarly replace $k_{\eta 0}, k_{\eta 1}$ with k_η. This both shortens the signature and hides the order in which redactions were performed, ensuring a sort of path-independence property. Thus, though signatures will grow in length, the length may grow only slowly if there is some locality to the sequence of redactions. At worst, each consecutive segment of erasures of length n' from a message of length n adds $O(\lg n')$ hash values and $O(\lg n')$ key values to the signature.

This signature scheme produces signatures that are relatively short. If Sig_0 yields m-bit signatures and if we use m'-bit hash values and keys, then an unredacted signature is only $m + m'$ bits long. This is a constant in n, the length of the message. After erasing s segments, each of length at most n', the signature will be $m + O(sm' \lg(nn'))$ bits long. In general, signatures could be as long as $m + O(nm')$ bits after many redactions, but we can expect that in practice the number of erased segments will typically not be too large and hence signatures should often be quite short. Therefore, this is a considerable improvement over the trivial construction outlined earlier.

This signature scheme also has an additional homomorphic property. For a fixed message x, we can form the lattice of substrings $L_x = \{w \in \Sigma^* : w \preceq x\}$, where the join $v \sqcup v'$ is the unique least $w \in L_x$ satisfying $v \preceq w$ and $v' \preceq w$, and the meet is defined dually. We note that given two redactions $\mathsf{Sig}(v)$ and $\mathsf{Sig}(v')$ of a common signature on a message x, we can compute signatures $\mathsf{Sig}(v \sqcup v')$ and $\mathsf{Sig}(v \sqcap v')$ on their join and meet. In other words, our scheme is also homomorphic with respect to the binary operations $\sqcup$ and $\sqcap$. We imagine that this might be useful in some applications.

Security analysis. We show in Appendix A that this construction is secure against existential forgeries under reasonable cryptographic assumptions.[3] See the appendix for details.

5 Set-Homomorphic Signatures

We now turn to operations on sets and derive a scheme which simultaneously supports the union and subset operations. More precisely, the scheme allows anyone possessing sets U_1, U_2 and $\mathsf{Sig}(U_1), \mathsf{Sig}(U_2)$ to compute $\mathsf{Sig}(U_1 \cup U_2)$ and $\mathsf{Sig}(U)$ for any $U \subseteq U_1$. Note that these two operations, union and subset, can be combined to create many others, such as set difference, symmetric difference, and intersection, so this scheme seems particularly powerful. As an example application, at the end of this section we will construct an alternate redactable signature scheme which has many advantages over the one presented in Section 4.

The construction is based on the accumulator technique [9]. To begin, let h be a public hash function so that, for all x, $h(x)$ is uniformly distributed over the odd primes less than n [16]. If we extend h to sets via $h(U) = \prod_{u \in U} h(u)$, then $h(U_1 \cup U_2) = \mathrm{lcm}(h(U_1), h(U_2))$, assuming there are no h-collisions between elements of U_1 and U_2. Each user creates a rigid RSA modulus n and $v \in (\mathbb{Z}/n\mathbb{Z})^*$ selected uniformly at random. Recall that an RSA modulus $n = pq$ is rigid if $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are prime and $|p| = |q|$ [9]. By choosing n this way, we ensure that $\gcd(h(x), \varphi(n)) = 1$ for almost all x. To sign a set U, one computes $\mathsf{Sig}(U) = v^{\frac{1}{h(U)}} \mod n$. Note that since $h(U)$ is relatively prime to $\varphi(n)$, there is exactly one solution y to the equation $y^{h(U)} = v$. To verify a signature y on a set U, one need only check that $y^{h(U)} = v \mod n$.

Given U_1, U_2 and signatures $v^{\frac{1}{h(U_1)}}, v^{\frac{1}{h(U_2)}}$, computing the signature on $U_1 \cup U_2$ is easy. First use the Euclidean algorithm to find a and b such that $a \cdot h(U_1) + b \cdot h(U_2) = \gcd(h(U_1), h(U_2))$. Then $(v^{\frac{1}{h(U_1)}})^b (v^{\frac{1}{h(U_2)}})^a = v^{\frac{1}{\mathrm{lcm}(h(U_1), h(U_2))}}$, which is the desired signature. If $U \subseteq U_1$ then $\mathsf{Sig}(U_1 \setminus U) = \mathsf{Sig}(U_1)^{h(U)}$, so computing subset signatures is also straightforward.

As with the redactable signature problem, we'd like to show this scheme is resistant to forgeries and that it satisfies the history-independence requirement for homomorphic signature schemes. The latter is clearly satisfied since the signature on a set U' is independent of how we obtain that signature: starting with a signature $\mathsf{Sig}(U_1)$ on U_1 and then removing the elements in U_2 to get

[3] We do not address security against random forgeries, as it is not clear what is the right distribution on the message space.

$\mathsf{Sig}(U_1)^{h(U_2)}$ yields exactly the same result as asking for the signature on $U_1 \setminus U_2$ directly, i.e., $\mathsf{Sig}(U_1)^{h(U_2)} = \mathsf{Sig}(U_1 \setminus U_2)$. A similar property holds for the union operation, and so this scheme is history-independent.

The resistance of this scheme to forgeries requires a more detailed security analysis, but basically it rests on the difficulty of computing kth roots modulo an RSA modulus and on the randomness of the hash function.

Before we give the proof, we should state clearly the parameterization of the difficulty of the RSA problem. We assume that RSA behaves as a good trapdoor permutation, as others have suggested before [6, 7]. This assumption appears to be weaker than the so-called Strong RSA assumption [2].

Definition 4. *Let $H_k = \{pq : p \text{ and } q \text{ are safe primes}, p \neq q, \text{and } |p| = |q| = k\}$. We say RSA is a (t, ϵ_r)-secure trapdoor permutation if for any adversary A with running time less than t, we have $\Pr[A(n, e, x^e) = x] < \epsilon_r$, where the probability is taken over the choice of $n \in H_k$, $e \in (\mathbb{Z}/\varphi(n)\mathbb{Z})^*$, $x \in (\mathbb{Z}/n\mathbb{Z})^*$, and the coin tosses of A.*

The following theorem relates the security of the set-homomorphic signature scheme to the security of RSA, and shows essentially that if the signature scheme is insecure, then an attacker could exploit that weakness to break RSA without too much more work. Note that the theorem guarantees security even when the adversary can make a number of adaptively chosen signature queries, which seems to be an extension of previous results.

Theorem 1. *Let h be a random oracle as above. Assume that RSA is a (t, ϵ_r)-secure one-way function. Then the set-homorphic signature scheme* Sig *defined above is (t', q_h, ϵ)-secure against existential forgery with respect to subset and union operations given that the total number of hash oracle queries performed is less than q_h, where $\epsilon \approx q_h \epsilon_r \log n + q_h^2 \log n / n$ and $t' \approx t$.* [4]

Proof. (sketch) We give a proof by contradiction. Assume the above theorem is false, meaning there exists an adversary A which can (t', ϵ)-break the set-homomorphic signature scheme Sig. Assume after A performs a number of random oracle queries and obtains signatures on sets $x_1, \ldots, x_{q_s}$, A outputs a forgery on x^* where x^* is not in the span of $x_1, \ldots, x_{q_s}$ under subset and union operations. This means there exists an element $y \in x^*$ such that $y \notin x_i$, for all $1 \leq i \leq q_s$, and A knows u such that $u^{h(y)} = v$. There are two cases:

- We have $h(y) \neq h(y')$ for all $y' \in \cup_{1 \leq i \leq q_s} x_i$.
- We have $h(y) = h(y')$ for some $y' \in \cup_{1 \leq i \leq q_s} x_i$. This happens with probability at most $q_h^2 \log n / n$ (by the prime number theorem and the birthday paradox).

If the first case happens with probability higher than $q_h \epsilon_r \log n$, we show as following that we can construct an adversary $B(n, e, z)$ using A which can (t, ϵ_r)-break RSA. In particular, if e is prime, B runs the following game using A. (Otherwise, the simulation fails; this happens with probability at most about $1/\log n$, by the prime number theorem.)

[4] The total number of hash oracle queries include the ones made by A and the ones made by the signing procedure.

- B first selects uniformly randomly q_h primes $p_1, \ldots, p_{q_h}$ not equal to e. Then B sets $f = \prod_{1 \le i \le q_h} p_i$ and $v = z^f \mod n$. Due to the choice of n, we have $\gcd(f, \varphi(n)) = 1$ with overwhelming probability, and thus the map $z \mapsto z^f \mod n$ is bijective on $(\mathbb{Z}/n\mathbb{Z})^*$. This means that v is uniformly distributed on $(\mathbb{Z}/n\mathbb{Z})^*$, since z is, so we can feed v as an input to A and it will have the right distribution. B also selects a random integer $k \in \{1, \ldots, q_h\}$.
- B constructs the hash oracle as follows. Given a hash oracle query on element w, if w has been queried before, then returns the previous corresponding answer. For the j-th unique hash oracle query w_j, if $j \neq k$, then return the prime p_j; otherwise, return e.
- B constructs the signing oracle that A queries as follows. Given a signing query on a set of m elements $U = \{a_1, \ldots, a_m\}$, B queries the hash oracle to obtain $h(a_1), \ldots, h(a_m)$. If for some $1 \le i \le m$, $h(a_i) = e$, then abort; otherwise, return $b = z^{f/(\prod_{1 \le i \le m} h(a_i))}$. It is easy to see that b is a valid signature on the set U.
- If at the end of the game, A outputs an existential forgery on a set of elements that includes an element y such that $h(y) = e$, then B can learn from A's forgery a value u satisfying $u^{h(y)} = v = z^f$. Because e is prime, $\gcd(e, f) = 1$, and we can compute α and β such that $\alpha e + \beta f = 1$ using the Eulidean algorithm. Thus $z = (u^\beta z^\alpha)^e$, and B outputs $u^\beta z^\alpha$ as the e-th root of z. Otherwise, abort the game.

If A outputs an existential forgery and e is prime, the probability that B succeeds is at least $1/q_h$. So if there exists an adversary A that (t', ϵ)-breaks the set-homomorphic signature scheme, we can construct an adversary B that (t, ϵ_r)-breaks RSA.

The set-union scheme suggests another solution to the redactable document signature problem. To sign the document $x = (x_1, \ldots, x_k)$, first select a random unique document identifier, k_x, and then sign the set of triples $D = \{(k_x, i, x_i)\}$, say $y = \mathsf{Sig}(D)$. The complete redactable signature is (k_x, y), and verification is straightforward. To compute the signature on the message with word i redacted, simply compute $y' = \mathsf{Sig}(D \setminus \{(k_x, i, x_i)\}) = y^{h(k_x, i, x_i)}$, and the new signature is (k_x, y'). Note that this scheme also reveals the locations of the redactions, preventing the semantic attacks described in Section 4. Not only is this scheme much simpler to implement and easier to understand than our prior redactable scheme, the signatures produced in this scheme are of constant length.

6 Additive Signature Schemes

We describe next a number of schemes that we have studied in our search for an additive signature scheme. All of them turn out to be insecure, and in interesting ways. More precisely, the schemes we examine all have an undesirable property that is likely to make them useless in practice: given signatures on a small set of known messages, we can forge signatures on all other possible messages. Thus, such schemes are insecure against random forgeries as described in Definition 3.

Constructions built from multiplicative signature schemes. If we have a multiplicative signature scheme $\mathsf{Sig}_\times : G \to R$ as well as a group homomorphism $\varphi : \mathbb{Z}/m\mathbb{Z} \to G$ from the additive group $\mathbb{Z}/m\mathbb{Z}$ to the multiplicative group G, then a natural candidate for an additive signature scheme $\mathsf{Sig}_+ : \mathbb{Z}/m\mathbb{Z} \to R$ is $\mathsf{Sig}_+ = \mathsf{Sig}_\times \circ \varphi$. For instance, we can instantiate $\mathsf{Sig}_\times$ with RSA. In this case, $G = (\mathbb{Z}/n\mathbb{Z})^*$, and every homomorphism from $\mathbb{Z}/m\mathbb{Z}$ to $(\mathbb{Z}/n\mathbb{Z})^*$ takes the form $\varphi(x) = g^x \pmod{n}$ for some $g \in G$, so our construction takes the form $\mathsf{Sig}_+(x) = g^{xd} \pmod{n}$.

To see why any such scheme must be insecure against random forgeries, suppose $\mathsf{Sig}_+ : \mathbb{Z} \to G$ is a group-homomorphic signature scheme. Then Sig_+ is entirely determined by $\mathsf{Sig}_+(1)$. Thus, if we can recover a few messages $m_1, \ldots, m_k$ such that $\gcd(m_1, \ldots, m_k) = 1$, then since there exist $a_1, \ldots, a_k \in \mathbb{Z}$ such that $\sum_i a_i m_i = 1$, we can compute $\mathsf{Sig}_+(1) = \sum_i a_i \mathsf{Sig}_+(m_i)$. Given this information, we can compute $\mathsf{Sig}_+(m) = m\mathsf{Sig}_+(1)$ for any m. This attack easily extends to group homomorphic signatures $\mathsf{Sig}_+ : \mathbb{Z}/n\mathbb{Z} \to G$ by lifting to $\mathbb{Z}$, applying the Euclidean algorithm, and projecting back down to $\mathbb{Z}/n\mathbb{Z}$. Since any small set of messages will likely have gcd 1, this scheme is vulnerable to random forgeries after only a few messages have been signed.

These weaknesses are very general. In particular, they apply to almost any instance of the construction $\mathsf{Sig}_+ = \mathsf{Sig}_\times \circ \varphi$. As another important example, every additive signature scheme that forms a group homomorphism is insecure against random forgeries.

It is perhaps counterintuitive that multiplicative signature schemes are plentiful while fully-additive schemes do not exist. The explanation seems to be the fact that $\mathbb{Z}/m\mathbb{Z}$ has a Euclidean algorithm, but $(\mathbb{Z}/n\mathbb{Z})^*$ does not. In other words, it is not that the span is larger in $\mathbb{Z}/m\mathbb{Z}$, but that the decomposition problem is easy in $\mathbb{Z}/m\mathbb{Z}$ but hard in $(\mathbb{Z}/n\mathbb{Z})^*$.

One might imagine that the problem is the need to be homomorphic with respect to both addition as well as negation, and thus one idea might be to look for a scheme that respects only the addition operator but not the negation. In other words, given $\mathsf{Sig}_+(x), \mathsf{Sig}_+(y)$, we still want to be able to compute $\mathsf{Sig}_+(x + y)$, but it should be hard to compute $\mathsf{Sig}_+(-x)$ from $\mathsf{Sig}_+(x)$. Note that the problems with additive signatures arise because one can find $a, a' \in \mathbb{Z}$ so that $ax + a'x' = 1$ and then compute $\mathsf{Sig}_+(1) = \mathsf{Sig}_+(ax) \times \mathsf{Sig}_+(a'x')$; yet one of a, a' will necessarily be negative. If we can somehow ensure that computing $\mathsf{Sig}_+(-x)$ from $\mathsf{Sig}_+(x)$ is hard, then the Euclidean algorithm will no longer apply, and the above attacks will fail. This suggests that we may want to look for an additive signature scheme without inverses (a semigroup-homomorphism), as such a scheme would resist the attacks described so far.

Further challenges. Yet even an additive signature scheme without inverses still has some properties that might not be expected. In particular, there is the problem that signatures can always be forged on all large enough messages, given signatures on two messages m, m'. This is because the equation

$$am + a'm' = x \qquad (\text{in } \mathbb{Z})$$

typically has solutions with $a, a' \geq 0$ when $x \geq \operatorname{lcm}(m, m')$, and in this case a signature on x can be obtained from signatures on m, m'.

More generally, if we have signatures $\mathsf{Sig}(m_1), \ldots, \mathsf{Sig}(m_k)$, then we can forge a signature $\mathsf{Sig}(x)$ whenever we can write x in the form $x = a_1 m_1 + \cdots + a_k m_k$ for some known $a_1, \ldots, a_k \in \mathbb{Z}_{\geq 0}$. This is a subset sum problem, and the issue is that if the m_i are small enough, the subset sum problem is easy to solve. So our only hope is to choose additive signatures without inverses, only sign messages large enough that subset sum is hard (require $m_i \geq \ell$ for some lower bound ℓ), and refuse to accept unusually large messages (enforce $m_i \ll \ell^2$) in the verification algorithm.

Additive schemes in higher dimensions. More generally, we could look for an additive scheme $\mathsf{Sig}_+ : (\mathbb{Z}/m\mathbb{Z})^d \to R$ in dimension $d > 1$. Unfortunately, this does not seem to offer much opportunity to design a secure signature scheme, either.

Although increasing the dimension does make more work for the attacker, a slight extension of the previous remarks still applies.

Observation 1 *In any additive signature scheme on the lattice $L = (\mathbb{Z}/m\mathbb{Z})^d$, if one can obtain signatures $\mathsf{Sig}(x_1), \ldots, \mathsf{Sig}(x_d)$, where $x_1, \ldots, x_d$ are a basis for L, then one can succeed at any random forgery.*

Knowing the signatures on a basis is useless if computing the representation of a given message in that basis is hard. If we could compute the signatures of the standard basis elements given the signatures on the elements of another basis, then committing random forgeries would be easy. Note that the elements of the standard basis are the shortest vectors in the lattice $\mathbb{Z}^d$, so lattice reduction techniques may be used to discover representations of the standard basis elements in terms of messages with known signatures. The theoretical bounds on the length of the shortest vector returned by LLL are not very tight, but in practice it can find a representation of the standard basis of $(\mathbb{Z}/m\mathbb{Z})^d$ with only $(1+\epsilon)d$ input vectors. Thus, we only need to collect $(1+\epsilon)d$ signed messages before we can commit random forgeries with ease.

Therefore, it seems to be a challenging open problem to find a secure additive signature scheme.

7 Open Problems

Set-homomorphic signatures. We may look for a set-homomorphic signature scheme that is homomorphic with respect to operations other than union and subset. For example, consider the following construction: Let $\mathsf{Sig}_\times : G \to R$ be an arbitrary multiplicative signature scheme on some group G. For a set $U = \{x_1, \ldots, x_k\}$, define the hash function $f_h(U) = h(x_1) \cdots h(x_k)$, where $h : \mathcal{X} \to G$ is a cryptographic hash function on elements in $\mathcal{X}$. Then $\mathsf{Sig} = \mathsf{Sig}_\times \circ f_h$ is a signature scheme with the property that with signatures, $\mathsf{Sig}(U)$ and $\mathsf{Sig}(V)$, on two disjoint sets U and V, one can compute the signature on their union,

$\mathsf{Sig}(U \cup V) = \mathsf{Sig}(U) \times \mathsf{Sig}(V)$. If $U \subseteq V$, one can also compute the signature on their difference, $\mathsf{Sig}(V \setminus U) = \mathsf{Sig}(V) \times \mathsf{Sig}(U)^{-1}$.

We gave another example of a set-homomorphic scheme, based on RSA accumulators, in Section 5. One interesting question is whether we can design a signature scheme that is homomorphic only with respect to the union operation.

Concatenable signatures. Let $\mathsf{Sig} : \{0,1\}^* \to R$ be a signature scheme. We say that it is a concatenable signature scheme if, given $\mathsf{Sig}(x), \mathsf{Sig}(y)$, one can compute (without help from the signer) a signature $\mathsf{Sig}(x||y)$ on the concatenation of x and y. In other words, a concatenable signature scheme should be homomorphic with respect to the semigroup $(\{0,1\}^*, ||)$ of bit-strings with the concatenation operator $||$. Rivest has asked [24]: can we design a concatenable signature scheme?

Semigroup signatures without inverses. Giving an example of a secure semigroup-homomorphic signature scheme seems to be an intriguing open problem that is suggested by this work. We have pointed out instances of this problem several times throughout this paper. Micali and Rivest asked whether there exists a transitive signature scheme on directed graphs [19], and this domain has a semigroup structure. One might seek a redactable signature scheme supporting only redaction but not the join operation. An additive scheme that doesn't respect subtraction might have a chance of being secure. A set-homomorphic scheme that allows only the union operation (but not subsetting) would have a semigroup-homomorphic property, as would a concatenable signature scheme. More generally, we suspect that any scheme that is semigroup-homomorphic but not group-homomorphic might yield insights into these open problems.

8 Conclusions

Homomorphic signature schemes present a promising new direction for research. Since such schemes necessarily do not satisfy traditional definitions of security, we have proposed new definitions of security for these new schemes. We have shown that a variety of homomorphic signature schemes can be designed. We also examined limits on their existence, showing that, for example, no additively group-homomorphic scheme can ever be secure against random forgeries. Perhaps most importantly, we have suggested several open problems that, if solved, might provide useful new schemes supporting a variety of applications.

9 Acknowledgements

We thank Eric Bach and Mika R.S. Kojo for early observations on weaknesses in additive signature schemes. Adrian Perrig originally suggested the idea of using Merkle hash trees to support redaction, which was a key step in the development of the scheme presented in Section 4. We also gratefully acknowledge many helpful comments from Ron Rivest and the anonymous referees.

References

1. Niv Ahituv, Yeheskel Lapid, and Seev Neumann. Processing encrypted data. *Communications of the ACM*, 30(9):777–780, 1987.
2. Niko Baric and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology—EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer-Verlag, 1997.
3. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: the case of hashing and signing. In Yvo Desmedt, editor, *Advances in Cryptology—CRYPTO '94*, pages 216–233, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science Volume 839.
4. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography with application to virus protection. In *FOCS 1995*, Berlin, 1995. Springer-Verlag.
5. M. Bellare and P. Rogaway. The exact security of digital signatures—how to sign with RSA and Rabin. In Ueli Maurer, editor, *Advances in Cryptology—EUROCRYPT '96*, pages 399–416, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science Volume 1070.
6. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993.
7. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures—how to sign with RSA and Rabin. In Ueli Maurer, editor, *Advances in Cryptology—EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.
8. J Benaloh. Dense probabilistic encryption. In *Selected Areas in Cryptography*, 1994.
9. J.C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *EUROCRYPT'93*, 1993.
10. D. Boneh and R. J. Lipton. Algorithms for black-box fields and their application to cryptography. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, pages 283–297, Berlin, 1996. Springer-Verlag. Lecture Notes in Computer Science Volume 1109.
11. E. F. Brickell and Y. Yacobi. On privacy homomorphisms. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology—EUROCRYPT '87*, pages 117–126, Berlin, 1987. Springer-Verlag. Lecture Notes in Computer Science Volume 304.
12. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *26th Symposium on the Foundations of Computer Science*, 1985.
13. Cramer and Damgard. Zero knowledge proofs for finite field arithmetic – or, can zero knowledge be for free? In *Advances in Cryptology—CRYPTO '98*, Berlin, 1998. Springer-Verlag.
14. J. Feigenbaum and Merritt. Open questions, talk abstracts, and summary of discussions. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–45, 1991.
15. E. Fujisaki, T. Okamoto, and Uchiyama. EPOC : Efficient probabilistic encryption. In *Submission to IEEE P1363*, 1998.
16. Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology—EUROCRYPT'99*, pages 123–139. Springer-Verlag, 1999. Lecture Notes in Computer Science Volume 1592.
17. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

18. Ralph Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, April 1980. IEEE Computer Society Press.
19. S. Micali and R. Rivest. Transitive signature schemes. In *RSA Conference 2002*, 2002.
20. D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *5th ACM Symposium on Computer and Communications Security*, 1998.
21. Goldreich Oded, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.
22. P Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *LNCS*, 1999.
23. R Peralta and J. Boyar. Short discreet proofs. In *Journal of Cryptology*, 2000.
24. R. Rivest. Two new signature schemes. Presented at Cambridge seminar; see `http://www.cl.cam.ac.uk/Research/Security/seminars/2000/rivest-tss.pdf`, 2001.
25. R. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–178. Academic Press, 1978.
26. T. Sander, A Young, and M Yung. Non-interactive cryptocomputing in NC^1. In *FOCS '99*, 1999.

A Security analysis for the tree redaction scheme

The combination of redaction and randomization in the tree redaction scheme introduces one tricky aspect into the definition of security. The adversary might choose a message $x \in \{0,1\}^*$, and though the censor might be unwilling to reveal $\mathsf{Sig}(x)$, the adversary might be able to gain access to a redacted signature $\mathsf{Sig}(w)$ on some message $w \preceq x$ of the attacker's choice. In this case our security analysis should ensure that the attacker cannot use this partial knowledge to obtain, e.g., a signature $\mathsf{Sig}(x)$ on the full message x. Therefore, in our model we give the adversary access to two different oracles, denoted R and S. Invoking $R(x)$ denotes registration of a chosen message $x \in \{0,1\}^*$, and the registered messages are stored in a list $x^1, x^2, \ldots, x^q$. Calling $S(i,w)$ returns the signature $\mathsf{Sig}(w)$ on the redacted message $w \in \Sigma^*$, if $w \preceq x^i$; otherwise, if w is not a valid redaction of the message x^i specified in the i-th call to R, the computation aborts. Let W^i denote the join of all values w that appear in some oracle query of the form $S(i,w)$. If an adversary can output a signature on a message that is not a redaction of W^i for some i, we say that the adversary has found an existential forgery.

Theorem 2. *Let G be a (t, ϵ_G)-secure pseudorandom generator, H a (t, p_H)-collision-resistant hash function, and Sig_0 a conventional signature scheme that is (t,q,p_S)-secure against existential forgery. Suppose that $F_k(x) = H(0,k,x)$ is a (t,q,ϵ_H)-secure pseudorandom function. Then the redactable signature scheme Sig defined above for messages of length at most n is (t',q,p')-secure against existential forgery with respect to redaction, where $p' = q\lceil \lg n \rceil \epsilon_G + nq\epsilon_H + p_S + p_H + 2nq/2^{m'}$ and $t' \approx t$.*

Proof. (sketch) Consider any adversary that attempts to break the resulting modified scheme by exhibiting existential forgeries, i.e., by finding a valid signature on a message w^* that is not a redaction of the join of its previous queries. We let x^i denote the i-th message registered with R, k^i_η, v^i_η denote the key and hash value at node η, and we introduce the notation u^i_η for the input to the hash function at η so that $v^i_\eta = H(u^i_\eta)$. For example, k^i_ϵ denotes the key randomly chosen for use with x^i and its redactions, and v^i_ϵ denotes the root of the hash tree on the signature for message x^i.

Suppose the adversary forges a signature $(\{v^*_\ell\}, \{k^*_\eta\}, \mathsf{Sig}\,(v^*_\epsilon))$ on w^*. If $v^*_\epsilon \neq v^i_\epsilon$ for all i, then we have found an existential forgery of Sig_0, which by assumption happens with probability at most p_S. Therefore, we assume that for some i we have $v^*_\epsilon = v^i_\epsilon$. Let T^* denote the tree corresponding to w^*, i.e., the leaf nodes corresponding to unredacted symbols in w^* along with all their ancestors.

Thanks to the properties of Merkle tree hashing, we see that T^* must be a sub-tree of the tree corresponding to x^i. (Otherwise, there is some node ℓ that is a leaf node in the tree for x^i but is an internal node in T^*, but then we have a hash collision $H(u^*_\ell) = H(u^i_\ell)$ since u^*_ℓ starts with a 1 and u^i_ℓ starts with a 0, which contradicts the collision-resistance of H.) Similarly, the leaves of T^* must form leaves in the tree for x^i, and the internal nodes in T^* form internal nodes in the tree for x^i.

Moreover, the hash pre-images u_η must satisfy $u^*_\eta = u^i_\eta$ for each $\eta \in T^*$. This tells us that $v^*_\eta = v^i_\eta$ for all $\eta \in T^*$. It also tells us that $k^*_\ell = k^i_\ell$ and $w^*_\ell = x^i_\ell$ for each leaf node ℓ. This shows that w^* is a redaction of x^i.

The only case left to worry about is that possibly w^* includes some symbol not present in W^i (recall that W^i denotes the join of the redactions of x^i that were queried under oracle S). In this case, there is some leaf node $\ell \in T^*$ so that $w^*_\ell = x^i_\ell$ but ℓ is not found in the tree corresponding to W^i.

In this case, the forged signature must reveal k_η where η is some ancestor of ℓ. But now we can note that k_η was never disclosed by any of the oracle queries (nor was any of the key values at η's ancestors). We argue that this would constitute a break of G.

We can imagine replacing the key k_η and the keys at each of its descendant nodes with truly random values, chosen independently of everything else. If any adversary can recognize this substitution with advantage better than $q\lceil \lg n \rceil \epsilon_G$, then according to the proof of security for the GGM tree construction [21], they can distinguish G from random with advantage better than ϵ_G; thus we can assume that this substitution makes no noticeable difference.

Now the only values related to k_η that are disclosed is $v^i_\ell = H(0, k_\ell, x^i_\ell)$ for leaves ℓ that are descendants of k_η. But, since $H(0, k, \cdot)$ is a good PRF, we can in turn imagine replacing each such v^i_ℓ by truly random values, chosen independently of everything else, and no attacker can recognize this substitution with advantage better than $nq\epsilon_H$ without breaking the PRF assumption.

Since the adversary must present the value k_η in the forged signature, this means that the adversary has guessed the value of k^*_η, even though k^*_η has never been disclosed and was chosen independently of everything else. We can bound the probability that this happens by $1/2^{m'}$ per node, and summing over all the nodes gives at most $2nq/2^{m'}$.

Adding up each of these distinguishing probabilities yields the claimed bound.