Transitive Signatures based on Factoring and RSA

Mihir Bellare[1] and Gregory Neven[2]

[1] Dept. of Computer Science & Engineering, University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA.
Email: mihir@cs.ucsd.edu
URL: http://www-cse.ucsd.edu/users/mihir

[2] Dept. of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium.
E-Mail: Gregory.Neven@cs.kuleuven.ac.be
URL: http://www.cs.kuleuven.ac.be/~gregory

Abstract. We present novel realizations of the transitive signature primitive introduced by Micali and Rivest [12], and also provide an answer to an open question they raise regarding the security of their RSA based scheme. Our schemes provide performance improvements over the scheme of [12].

1 Introduction

The concept. The context envisioned by Micali and Rivest [12] is that of dynamically building an authenticated graph, edge by edge. The signer, having secret key *tsk* and public key *tpk*, can at any time pick a pair i, j of nodes and create a signature of $\{i, j\}$, thereby adding edge $\{i, j\}$ to the graph. A composability property is required: given a signature of an edge $\{i, j\}$ and a signature of an edge $\{j, k\}$, anyone in possession of the public key can create a signature of the edge $\{i, k\}$. Security asks that this limited class of forgeries be the only possible ones. (I.e., without *tsk*, it should be hard to create a valid signature of edge $\{i, j\}$ unless i, j are connected by a path whose edges have been explicitly authenticated by the signer.) Thus the authenticated graph at any point is the transitive closure of the graph formed by the edges explicitly authenticated by the signer, whence the name of the concept. We refer the reader to Section 2 for formal definitions and to [12] for motivation and potential applications.

Realizing the concept. A transitive signature scheme can be trivially realized by accepting, as a valid signature of $\{i, j\}$, any chain of signatures that authenticates a sequence of edges forming a path from i to j. Two issues lead [12] to exclude this: the growth in signature size, and the loss of privacy incurred by having signatures carry information about their history. The main result of [12] is a (non-trivial) transitive signature scheme (we call it the MRTS scheme) proven to be (transitively) unforgeable under adaptive chosen-message attack

Scheme	**Signing cost**	**Verification cost**	**Composition cost**	**Signature size**
MRTS	2 stand. sigs 2 exp. in $\mathbb{G}$	2 stand. verifs 1 exp. in $\mathbb{G}$	2 adds in $\mathbb{Z}_q$	2 stand. sigs 2 points in $\mathbb{G}$ 2 points in $\mathbb{Z}_q$
FBTS-1	2 stand. sigs $O(\|N\|^2)$ ops	2 stand. verifs $O(\|N\|^2)$ ops	$O(\|N\|^2)$ ops	2 stand. sigs 3 points in $\mathbb{Z}_N^*$
FBTS-2	4 sq. roots in $\mathbb{Z}_N^*$	$O(\|N\|^2)$ ops	$O(\|N\|^2)$ ops	1 point in $\mathbb{Z}_N^*$
RSATS-1	2 stand. sigs 2 RSA encs	2 stand. verifs 1 RSA enc.	$O(\|N\|^2)$ ops	2 stand. sigs 3 points in $\mathbb{Z}_N^*$
RSATS-2	1 RSA dec.	1 RSA enc.	$O(\|N\|^2)$ ops	1 point in $\mathbb{Z}_N^*$

Fig. 1. Cost comparisons amongst transitive signature schemes. The word "stand." refers to operations of the underlying standard signature scheme, which are eliminated for FBTS-2 and RSATS-2. $\mathbb{G}$ denotes the group of prime order q used in MRTS, and N denotes a modulus product of two primes as used in the other schemes. Abbreviations used are: "exp." for an exponentiation in the group; "RSA enc." for an RSA encryption; "RSA dec." for an RSA decryption performed given the decryption exponent; "sq. root" for a square root modulo N performed using the prime factors of N'; and "ops" for the number of elementary bit operations in big-O notation.

(see Section 2 for formal definitions) assuming that the discrete logarithm problem is hard in an underlying prime-order group and assuming security of an underlying standard signature scheme. They also present a natural RSA based transitive signature scheme but point out that even though it seems secure, and a proof of unforgeability under non-adaptive chosen-message attacks exists, there is no known proof of unforgeability under *adaptive* chosen-message attacks. They thereby highlight the fact that in this domain, adaptive attacks might be harder to provably protect against than non-adaptive ones.

In summary, transitive signatures (unforgeable under adaptive chosen-message attacks) at this point have just a single realization, namely the MRTS scheme. It is standard practice in cryptography to seek new and alternative realizations of primitives of potential interest, both to provide firmer theoretical foundations for the existence of the primitive by basing it on alternative conjectured hard problems and to obtain performance improvements. This paper presents new schemes that accomplish both of these objectives, and also provides an answer to the question about the RSA scheme.

The node certification paradigm. It is worth outlining the node certification based paradigm introduced by the MRTS scheme, which will be our starting point. The signer's keys include those of a standard digital signature scheme, and the public key includes additional items. (In the MRTS scheme, this is a group $\mathbb{G}$ of prime order q and a pair of generators of $\mathbb{G}$.) The signer associates to each node i in the current graph a *node certificate* consisting of a *public label* $L(i)$ and a signature of $(i, L(i))$ under the standard scheme. The signature of an edge contains the certificates of its endpoints plus an *edge label* δ. Verification of the signature of an edge involves relating the edge label to the public labels of its endpoints as provided in the node certificates and verifying the standard signatures in the node certificates. Composition involves algebraic manipulation of edge labels.[3]

The paradigm is useful, but brings an associated cost. Producing a signature for an edge can involve computing two normal signatures. The length of an edge signature, containing two node certificates each including a standard signature, can be large even if the edge labels are small.

1.1 Transitive Signatures based on Factoring

Our first factoring-based transitive signature (FBTS-1) scheme stays within the node certification paradigm but, by implementing label algebra via square-roots modulo a composite, provides security based on factoring while reducing some costs compared to MRTS.

FBTS-1. The signer has keys for a standard signature scheme, and its public key additionally includes a modulus N product of two large primes. The public label of a node i is a quadratic residue $L(i) \in \mathbb{Z}_N^*$, and an edge label of edge $\{i, j\}$ is a square root of $L(i)L(j)^{-1} \bmod N$ assuming $i < j$. Composition involves multiplying edge labels modulo N. We prove that FBTS-1 is unforgeable under adaptive chosen-message attack, assuming the hardness of factoring the underlying modulus, and assuming security of the underlying standard signature scheme. The delicate part of this proof is an information-theoretic lemma showing that, even under an adaptive chosen-message attack, for any $\{i, j\}$ not in the transitive closure of the current graph, an adversary has zero advantage in determining which of the square roots of $L(i)L(j)^{-1}$ is held by the signer.

With regard to costs, we are interested in the computational cost of signing an edge (in the worst case that both endpoints of the edge are not in the current graph); the computational cost of verifying a candidate signature of an edge; the computational cost of composing two edge signatures to obtain another; and the size of a signature. Since FBTS-1 continues to employ the node certification paradigm, it incurs the same costs as MRTS from the use of the standard signature scheme. However, as Figure 1 indicates, it is otherwise cheaper than MRTS for signing and verifying, reducing the extra cost from cubic (exponentiation) to quadratic (a couple of multiplications and an inverse).

FBTS-2: Eliminating node certificates. FBTS-1 is amenable to a modification which eliminates the need for node certificates and thereby removes the standard signature scheme, and all its associated costs, from the picture. The signer's public key is a modulus N product of two primes p, q that make up the signer's secret key. The public label of a node i is not chosen by the signer but

[3] Note that the signer is stateful, and once quantities associated to a node are created, they are stored and re-used for all edges adjacent to this node. This is important for security. See Section 3 for a discussion of how state can be eliminated.

Scheme	Proven to be unforgeable under *adaptive* chosen-message attack assuming	RO Model?
MRTS	Security of standard signature scheme Hardness of discrete logarithm problem in a group of prime order	No
FBTS-1	Security of standard signature scheme Hardness of factoring	No
FBTS-2	Hardness of factoring	Yes
RSATS-1	Security of standard signature scheme RSA is secure against one-more-inversion attack	No
RSATS-2	RSA is secure against one-more-inversion attack	Yes

Fig. 2. Provable security attributes of transitive signature schemes. We indicate the assumptions under which there is a proof of unforgeability under *adaptive* chosen-message attack, and whether or not the random oracle model is used.

rather specified via the output of a public hash function applied to i. (A difficulty, addressed in Section 4, is that the hash output might not be a quadratic residue.) We prove that FBTS-2 is unforgeable under adaptive chosen-message attack, assuming the hardness of factoring the underlying modulus, in a model where the hash function is a random oracle [5].

As Figure 1 indicates, the major cost savings is elimination of all costs associated to the standard scheme. However, signing now requires computation of square roots modulo N by the signer based on the prime factorization of N, which has cost comparable to an exponentiation modulo N. Thus overall the main gain is the reduction in signature size.

This hash based modification is made possible by the fact that squaring modulo a composite is a trapdoor function. The MRTS scheme is not amenable to a similar hash-based modification since the discrete exponentiation function is not trapdoor over the prime order groups used in [12].

1.2 Transitive Signatures based on RSA

RSATS-1. The RSA-based transitive signature scheme noted in [12] (that we call RSATS-1) employs the node certification paradigm. The signer has keys for a standard signature scheme. Its public key additionally includes an RSA modulus N and encryption exponent e, while its secret key includes the corresponding decryption exponent d. The public label of a node i is a point $L(i) \in \mathbb{Z}_N^*$, and the edge label of edge $\{i, j\}$ is $L(i)^d L(j)^{-d} \bmod N$ assuming $i < j$. Composition involves multiplying edge labels modulo N. One can prove that RSATS-1 is unforgeable under *non-adaptive* chosen-message attacks assuming the one-wayness of RSA and the security of the underlying standard signature scheme. No adaptive chosen-message attack that succeeds in forgery has been found, but neither has it been proven that RSATS-1 is unforgeable under adaptive chosen-message attack.

One might wonder why proofs exist for MRTS and FBTS-1 but remain elusive for RSATS-1 in spite of the obvious similarities between these schemes. The proofs for MRTS and FBTS-1 exploit the fact that there are multiple valid edge labels for any given edge in the graph, and that finding two different edge labels implies solving the underlying hard problem. With RSATS-1, the edge label is uniquely determined by the two node certificates, and this paradigm fails.

This situation (namely a scheme that appears to resist both attack and proof) is not uncommon in cryptography, and we suggest that it is a manifestation of the fact that the security of the scheme is relying on properties possessed by RSA but going beyond those captured by the assumption that RSA is one-way. Accordingly we seek an alternative, stronger assumption upon which a proof of security can be based.

We prove that RSATS-1 is unforgeable under adaptive chosen-message attacks under the assumption that RSA is secure under one-more-inversion (and the standard signature scheme is secure). This assumption was introduced by [2], who used it to prove the security of Chaum's blind signature scheme [7]. It was also used in [4] to prove security of the GQ identification scheme [10] against impersonation under active attack.

Security under one more inversion considers an adversary given input an RSA public key N, e, and two oracles. The *challenge* oracle takes no inputs and returns a random target point in $\mathbb{Z}_N^*$, chosen anew each time the oracle is invoked. The *inversion* oracle given $y \in \mathbb{Z}_N^*$ returns $y^d \bmod N$ where d is the decryption exponent corresponding to e. The assumption states that it is computationally infeasible for the adversary to output correct inverses of all the target points if the number of queries it makes to its inversion oracle is strictly less than the number of queries it makes to its challenge oracle. When the adversary makes one challenge query and no inversion queries, this reduces to the standard one-wayness assumption.

RSATS-2. The trapdoorness of the RSA function makes RSATS-1 amenable to the elimination of node certificates via hashing, based on ideas similar to the ones we introduced above. We present RSATS-2, a transitive signature scheme that is unforgeable under adaptive chosen-message attacks in the random oracle model assuming RSA is secure against one-more-inversion. The public label of a node i is not chosen by the signer but rather implicitly specified as the output of a hash function applied to i, and RSA decryption is used to compute edge labels. Finally we note that RSATS-2 is the only one of the schemes discussed here whose signer is naturally stateless.

Figures 1 and 2 summarize, respectively, the costs and provable-security attributes of the various schemes we have introduced, and compare them with the MRTS scheme.

1.3 Definitional Contributions

Regarding the composability property, Micali and Rivest [12, p. 238] (we have modified the notation to be consistent with ours) say: "... if someone sees Alice's signatures on edges $\{i, j\}$ and $\{j, k\}$ then that someone can easily compute a valid signature on edge $\{i, k\}$ that is indistinguishable from a signature on that edge that Alice would have produced herself." This seems to suggest that composition is only required to work when the given signatures were explicitly produced by the signer, but in fact we want composition to work even if the given signatures were themselves obtained via composition. Formulating an appropriate requirement turns out to be more delicate than one might imagine. One could require the simple condition that valid signatures (meaning, ones accepted by the verification algorithm relative to the signer's public key) can be composed to yield valid signatures. (This would follow [11], who require a condition that implies this.) But this requirement is too strong in the current context. Indeed, the MRTS scheme does not meet it, meaning there are valid signatures which, when composed, yield an invalid signature. The same is true for our schemes.

It can be proved that for MRTS and our schemes, finding valid signature inputs that make the composition algorithm return an invalid signature is computationally hard assuming the scheme is secure. But we prefer to not tie correctness of composition to security. Instead, we formulate correctness of composition via a recursive requirement that says that as long as one obtains signatures either directly via the signer or by applying the composition operation to signatures previously legitimately obtained or generated, then the resulting signature is valid. (This would be relatively easy to formulate if the signer was stateless, but needs more care due to the fact that the natural formulation of transitive signature schemes often results in a stateful signer.) As part of the formalization we provide in Definition 1, we follow [11] and require a very strong form of the indistinguishability requirement mentioned in the quote above, namely that the signature output by the composition algorithm is not just indistinguishable from, but identical to, the one the signer would have produced. (As argued in [11], this guarantees privacy.) The MRTS scheme, as well as all our schemes, meet this strong definition.

1.4 Related Work

Transitive signatures are one case of a more general concept promulgated by Rivest [14] in talks, namely that of signature schemes that admit forgery of signatures derived by some specific operation on previous signatures but resist other forgeries. Johnson, Molnar, Song and Wagner [11] formalize a notion of homomorphic signature schemes that captures this. Context Extraction Signatures, as introduced earlier by [15], as well as redactable signatures and set-homomorphic signatures [11], fall in this framework. A signature scheme that is homomorphic with respect to the prefix operation is presented by Chari, Rabin and Rivest [6].

```
$(tpk, tsk) \stackrel{R}{\leftarrow} \mathsf{TKG}(1^k)$
$S \leftarrow \emptyset$ ; $\mathsf{Legit} \leftarrow \mathsf{true}$ ; $\mathsf{NotOK} \leftarrow \mathsf{false}$
Run $A$ with its oracles until it halts, replying to its oracle queries as follows:
    If $A$ makes $\mathsf{TSign}$ query $i, j$ then
        If $i = j$ then $\mathsf{Legit} \leftarrow \mathsf{false}$
        Else
            Let $\sigma$ be the output of the $\mathsf{TSign}$ oracle and let $S \leftarrow S \cup \{(\{i, j\}, \sigma)\}$
            If $\mathsf{TVf}(tpk, i, j, \sigma) = 0$ then $\mathsf{NotOK} \leftarrow \mathsf{true}$
    If $A$ makes $\mathsf{Comp}$ query $i, j, k, \sigma_1, \sigma_2$ then
        If $[(\{i, j\}, \sigma_1) \notin S$ OR $(\{j, k\}, \sigma_2) \notin S$ OR $i, j, k$ are not all distinct] then
            $\mathsf{Legit} \leftarrow \mathsf{false}$
        Else
            Let $\sigma$ be the output of the $\mathsf{Comp}$ oracle and let $S \leftarrow S \cup \{(\{i, k\}, \sigma)\}$
            Let $\tau \leftarrow \mathsf{TSign}(tsk, i, k)$
            If $[(\sigma \neq \tau)$ or $\mathsf{TVf}(tpk, i, k, \sigma) = 0]$ then $\mathsf{NotOK} \leftarrow \mathsf{true}$
When $A$ halts, output $(\mathsf{Legit} \wedge \mathsf{NotOK})$ and halt
```

Fig. 3. Experiment used to define correctness of the transitive signature scheme $\mathsf{TS} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$.

2 Definitions

Notation. We let ε denote the empty string and $\|$ the concatenation operator on strings. We let $\mathbb{N} = \{1, 2, \ldots\}$ be the set of positive integers. The notation $x \stackrel{R}{\leftarrow} S$ denotes that x is selected randomly from set S. If A is a possibly randomized algorithm then the notation $x \stackrel{R}{\leftarrow} A(a_1, a_2, \ldots)$ denotes that x is assigned the outcome of the experiment of running A on inputs $a_1, a_2, \ldots$.

Graphs. All graphs in this paper are undirected. A graph $G^* = (V^*, E^*)$ is said to be *transitively closed* if for all nodes $i, j, k \in V^*$ such that $\{i, j\} \in E^*$ and $\{j, k\} \in E^*$, it also holds that $\{i, k\} \in E^*$; or in other words, edge $\{i, j\} \in E^*$ whenever there is a path from i to j in G^*. If $G = (V, E)$ is a graph, its *transitive closure* is the graph $\widetilde{G} = (V, \widetilde{E})$ where $\{i, j\} \in \widetilde{E}$ iff there is a path from i to j in G. Note that the transitive closure of any graph G is a transitively closed graph. Also note that any transitively closed graph can be partitioned into connected components such that each component is a complete graph.

Transitive signature schemes and their correctness. A *transitive signature scheme* $\mathsf{TS} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ is specified by four polynomial-time algorithms, and the functionality is as follows:

- The randomized *key generation* algorithm TKG takes input 1^k, where $k \in \mathbb{N}$ is the security parameter, and returns a pair (tpk, tsk) consisting of a public key and matching secret key.
- The *signing algorithm* TSign, which could be stateful or randomized (or both), takes input the secret key tsk and *nodes* $i, j \in \mathbb{N}$, and returns a

value called an *original signature* of edge $\{i, j\}$ relative to *tsk*. If stateful, it maintains state which it updates upon each invocation.

- The deterministic *verification* algorithm TVf, given *tpk*, nodes $i, j \in \mathbb{N}$, and a candidate signature σ, returns either 1 or 0. In the former case we say that σ is a *valid* signature of edge $\{i, j\}$ relative to *tpk*.
- The deterministic *composition* algorithm Comp takes *tpk*, nodes $i, j, k \in \mathbb{N}$ and values σ_1, σ_2 to return either a value σ or a symbol $\perp$ to indicate failure.

The above formulation makes the simplifying assumption that the nodes of the graph are positive integers. In practice it is desirable to allow users to name nodes via whatever identifiers they choose, but these names can always be encoded as integers, so we keep the formulation simple.

Naturally, it is required that if σ is an original signature of edge $\{i, j\}$ relative to *tsk* then it is a valid signature of $\{i, j\}$ relative to *tpk*.

As discussed in Section 1.3, formulating a correctness requirement for the composition algorithm is more delicate. Micali and Rivest [12] seem to suggest that composition is only required to work when the given signatures were explicitly produced by the signer, but in fact we want composition to work even if the given signatures were themselves obtained via composition. Furthermore the indistinguishability requirement is not formalized in [12].

Definitions taking these issues into account are however provided in [11]. They ask that whenever the composition algorithm is invoked on valid signatures (valid means accepted by the verification algorithm relative to the signer's public key) it returns the same signature as the signer would produce. This captures indistinguishability in a strong way that guarantees privacy. However, one implication of their definition is that whenever the composition algorithm is invoked on valid signatures, it returns a valid signature, and this last property is not true of known node certification based transitive signature schemes such as $\mathsf{MRTS}, \mathsf{FBTS\text{-}1}$ and $\mathsf{RSATS\text{-}1}$. For these schemes, it is possible to construct examples of valid signature inputs that, when provided to the composition algorithm, result in the latter failing (returning $\perp$ because it cannot compose) or returning an invalid signature. (Roughly, this is because composition of a signature σ_1 of $\{i, j\}$ with a signature σ_2 of $\{j, k\}$ in these schemes requires that the public labels of node j as specified in σ_1 and σ_2 be the same. Validity of the individual signatures cannot guarantee this.)

This is not a weakness in the schemes, because in practice composition is applied not to arbitrary valid signatures but to ones that are legitimate, the latter being recursively defined: a signature is legitimate if it is either obtained directly by the signer, or obtained by applying the composition algorithm to legitimate signatures. What it points to is that we need to formulate a new correctness definition for composition that captures this intuition and results in a notion met by the known transitive signature schemes. Roughly, we would like a formulation that says that if the composition algorithm is invoked on legitimate signatures, then it returns the same signature as the signer would have produced. (Here, we are continuing to follow [11] in capturing indistinguishability by the strong requirement that composed signatures are identical to original ones, but weakening their requirement by asking that this be true not for all valid signature inputs to the composition algorithm, but only for legitimate inputs.)

The formalization would be relatively simple (the informal description above would pretty much be it) if the signing algorithm were stateless, but the natural formulation of numerous transitive signature schemes seems to be in terms of a stateful signing algorithm. In this case, it is not clear what it means that the output of the composition algorithm is the same as that of the signer, since the latter's output depends on its internal state which could be different at different times. To obtain a formal definition of correctness that takes into account the statefulness of the signing algorithm, we proceed as follows. We associate to any algorithm A (deterministic, halting, but not computationally limited) and security parameter $k \in \mathbb{N}$ the experiment of Figure 3, which provides A with oracles

$$\mathsf{TSign}(tsk,\cdot,\cdot) \text{ and } \mathsf{Comp}(tpk,\cdot,\cdot,\cdot,\cdot,\cdot)\ ,$$

where tpk, tsk have been produced by running TKG on input 1^k. In this experiment, the TSign oracle maintains state, and updates this state each time it is invoked. It also tosses coins anew at each invocation if it is randomized.

Definition 1. *We say that the transitive signature scheme* TS *is correct if for every (computationally unbounded) algorithm A and every k, the output of the experiment of Figure 3 is* true *with probability zero.*

The experiment computes a boolean Legit which is set to false if A ever makes an "illegitimate" query. It also computes a boolean NotOK which is set to true if a signature returned by the composition algorithm differs from the original one. To win, A must stay legitimate (meaning $\mathsf{Legit} = \mathsf{true}$) but violate correctness (meaning $\mathsf{NotOK} = \mathsf{true}$). The experiment returns true iff A wins. The definition requires that this happen with probability zero.

Security of transitive signature schemes. We recall the notion of security of [12]. Associated to transitive signature scheme $\mathsf{TS} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$, adversary F and security parameter $k \in \mathbb{N}$ is an experiment, denoted

$$\mathbf{Exp}^{\text{tu-cma}}_{\mathsf{TS},F}(k)\ ,$$

that returns 1 if and only if F is successful in its attack on the scheme. The experiment begins by running TKG on input 1^k to get keys (tpk, tsk). If we are in the random oracle model, it also chooses the appropriate hash functions at random. It then runs F, providing this adversary with input tpk and oracle access to the function $\mathsf{TSign}(tsk,\cdot,\cdot)$. The oracle is assumed to maintain state or toss coins as needed. Eventually, F will output $i', j' \in \mathbb{N}$ and some value σ'. Let E be the set of all edges $\{a, b\}$ such that F made oracle query a, b, and let V be the set of all integers a such that a is adjacent to some edge in E. We say that F *wins* if σ' is a valid signature of $\{i', j'\}$ relative to tpk but edge $\{i', j'\}$ is not in the transitive closure $\widetilde{G}$ of graph $G = (V, E)$. The experiment returns 1 if F wins and 0 otherwise. The *advantage* of F in its attack on TS is the function

$\mathbf{Adv}^{\text{tu-cma}}_{\mathsf{TS},F}(\cdot)$ defined for $k \in \mathbb{N}$ by

$$\mathbf{Adv}^{\text{tu-cma}}_{\mathsf{TS},F}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\text{tu-cma}}_{\mathsf{TS},F}(k) = 1\,\right] ,$$

the probability being over all the random choices made in the experiment. We say that TS is *transitively unforgeable under adaptive chosen-message attack* if the function $\mathbf{Adv}^{\text{tu-cma}}_{\mathsf{TS},F}(\cdot)$ is negligible for any adversary F whose running time is polynomial in the security parameter k.

RO MODEL. Some of our schemes will be defined in the random oracle model [5], which means that the algorithms $\mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp}$ all have oracle access to one or more functions which in the correctness and security experiments are assumed to be drawn at random from appropriate spaces. Formally, both the experiment of Figure 3 and $\mathbf{Exp}^{\text{tu-cma}}_{\mathsf{TS},F}(k)$ are augmented to choose a function mapping $\{0,1\}^*$ to $\{0,1\}^k$ at random, and the adversary, as well as the $\mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp}$ algorithms, then get oracle access to this function. In Definition 1, the probability includes the choice of these functions, and so does the probability in the definition of $\mathbf{Adv}^{\text{tu-cma}}_{\mathsf{TS},F}(k)$. Usually the scheme will need to construct out of the given function a function H with suitable range depending on the public key.

STANDARD SIGNATURE SCHEMES. Some of our schemes use an underlying standard digital signature scheme $\mathsf{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$, described as usual via its polynomial-time key generation, signing and verification algorithms. We use the security definition of [9], where the forger B is given adaptive oracle access to the signing algorithm, and its advantage $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{SDS},B}(k)$ in breaking SDS is defined as the probability that it outputs a valid signature for a message that was not one of its previous oracle queries. The scheme SDS is said to be secure against forgery under adaptive chosen-message attack if $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{SDS},B}(\cdot)$ is negligible for every forger B with running time polynomial in the security parameter k.

3 A Transitive Signature Scheme based on Factoring

FACTORING PROBLEM. A *modulus generator* is a randomized, polynomial-time algorithm that on input 1^k returns a triple (N, p, q) where $N = pq$, $2^{k-1} \leq N < 2^k$, and p, q are distinct, odd primes. There are numerous possible modulus generators which differ in the structure of the primes chosen or the distribution under which they are chosen. We do not restrict the type of generator, but only assume that the associated factoring problem is hard. Formally, for any modulus generator MG, adversary A and $k \in \mathbb{N}$ we let

$$\mathbf{Adv}^{\text{fac}}_{\mathsf{MG},A}(k) \;=\; \Pr\left[\, r \in \{p,q\} \;\;:\;\; (N,p,q) \overset{R}{\leftarrow} \mathsf{MG}(1^k)\,;\, r \overset{R}{\leftarrow} A(k,N) \,\right] .$$

We say that *factoring is hard relative to* MG if the function $\mathbf{Adv}^{\text{fac}}_{\mathsf{MG},A}(\cdot)$ is negligible for every A whose running time is polynomial in k.

THE SCHEME. We are given a modulus generator MG and a standard digital signature scheme $\mathsf{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$. We associate to them a transitive signature scheme $\mathsf{FBTS\text{-}1} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ defined as follows:

- Given input 1^k, the key generation algorithm TKG first runs SKG on input 1^k to generate a key pair (spk, ssk) for the standard signature scheme SDS. It then runs the modulus generator MG on input 1^k to get a triple (N, p, q). It outputs $tpk = (N, spk)$ as the public key of the transitive signature scheme and $tsk = (N, ssk)$ as the matching secret key. Note that the primes p, q are discarded and in particular not part of the secret key.

- The signing algorithm TSign maintains state (V, ℓ, L, Σ) where $V \subseteq \mathbb{N}$ is the set of all queried nodes, the function $\ell\colon\ V \to \mathbb{Z}_N^*$ assigns to each node $i \in V$ a *secret label* $\ell(i) \in \mathbb{Z}_N^*$, while the function $L\colon\ V \to \mathbb{Z}_N^*$ assigns to each node $i \in V$ a *public label* $L(i)$, and the function $\Sigma\colon\ V \to \{0,1\}^*$ assigns to each node i a standard signature on $i\|L(i)$ under ssk. When invoked on inputs tsk, i, j, meaning when asked to produce a signature on edge $\{i, j\}$, it does the following:

```
If $j < i$ then $l \leftarrow j$ ; $j \leftarrow i$ ; $i \leftarrow l$    // swap $i$ and $j$ if necessary
If $i \notin V$ then $V \leftarrow V \cup \{i\}$ ; $\ell(i) \xleftarrow{R} \mathbb{Z}_N^*$ ; $L(i) \leftarrow \ell(i)^2 \bmod N$ ;
        $\Sigma(i) \leftarrow \mathsf{SSign}(ssk, i\|L(i))$
If $j \notin V$ then $V \leftarrow V \cup \{j\}$ ; $\ell(j) \xleftarrow{R} \mathbb{Z}_N^*$ ; $L(j) \leftarrow \ell(j)^2 \bmod N$ ;
        $\Sigma(j) \leftarrow \mathsf{SSign}(ssk, j\|L(j))$
$\delta \leftarrow \ell(i)\ell(j)^{-1} \bmod N$ ; $C_i \leftarrow (i, L(i), \Sigma(i))$ ; $C_j \leftarrow (j, L(j), \Sigma(j))$
Return $(C_i, C_j, \delta)$ as the signature of $\{i, j\}$.
```

  We refer to $(l, L(l), \Sigma(l))$ as a *certificate* of node l.

- The verification algorithm TVf, on input $tpk = (N, spk)$, nodes i, j and a candidate signature σ, proceeds as follows:

```
If $j < i$ then $l \leftarrow j$ ; $j \leftarrow i$ ; $i \leftarrow l$    // swap $i$ and $j$ if necessary
Parse $\sigma$ as $(C_i, C_j, \delta)$, parse $C_i$ as $(i, L_i, \Sigma_i)$, parse $C_j$ as $(j, L_j, \Sigma_j)$
If $\mathsf{SVf}(spk, i\|L_i, \Sigma_i) = 0$ or $\mathsf{SVf}(spk, j\|L_j, \Sigma_j) = 0$ then return 0
If $L_i \equiv \delta^2 L_j \bmod N$ then return 1 else return 0.
```

- The composition algorithm Comp takes nodes i, j, k, a signature $\sigma_1 = (C_1, C_2, \delta_1)$ of $\{i, j\}$ and a signature $\sigma_2 = (C_3, C_4, \delta_2)$ of $\{j, k\}$, and proceeds as follows:

```
Let $C_i \in \{C_1, C_2\}$ be such that $C_i$ parses as $(i, L_i, \Sigma_i)$[4]
Let $C_j \in \{C_1, C_2\}$ be such that $C_j$ parses as $(j, L_j, \Sigma_j)$
If $C_j \notin \{C_3, C_4\}$ then return $\perp$
Let $C_k \in \{C_3, C_4\}$ be such that $C_k$ parses as $(k, L_k, \Sigma_k)$
If $j < i < k$ then $\delta \leftarrow \delta_1^{-1}\delta_2 \bmod N$ ; Return $(C_i, C_k, \delta)$
If $i < j < k$ then $\delta \leftarrow \delta_1\delta_2 \bmod N$    ; Return $(C_i, C_k, \delta)$
If $i < k < j$ then $\delta \leftarrow \delta_1\delta_2^{-1} \bmod N$ ; Return $(C_i, C_k, \delta)$
If $j < k < i$ then $\delta \leftarrow \delta_1\delta_2^{-1} \bmod N$ ; Return $(C_k, C_i, \delta)$
If $k < j < i$ then $\delta \leftarrow \delta_1\delta_2 \bmod N$    ; Return $(C_k, C_i, \delta)$
If $k < i < j$ then $\delta \leftarrow \delta_1^{-1}\delta_2 \bmod N$ ; Return $(C_k, C_i, \delta)$
```

A proof by induction can be used to show the following.

Proposition 1. *The* FBTS-1 *transitive signature scheme described above satisfies the correctness requirement of Definition 1.*

The proof is provided in [3]. We note that it was to ensure that this correctness requirement is met that we have been detailed regarding the specification of the composition algorithm above. We point in particular to the fact that the composition algorithm checks that the certificate C_j in the given signature of $\{i, j\}$ exactly matches the one in the given signature of $\{j, k\}$. This ensures that the public labels in these two certificates match, which is important in the proof of Proposition 1.

Eliminating state. The signing algorithm of the FBTS-1 scheme is stateful. It is important for composition that the signer associates a single public label to node i, and it is important for security that it associates to this a single secret label $\ell(i)$. (Else it would soon give away two different square roots of $L(i)$.) The MRTS, FBTS-2 and RSATS-1 schemes also have stateful signing algorithms, pointing to the fact that the natural formulation of many transitive signature schemes is in terms of a stateful signing algorithm. However, we note here that a simple transformation can be used to make the signer stateless, if so desired, without loss of security. Namely, let the signer's secret key include a key K specifying an instance F_K from a pseudorandom function family F [8], and use $F_K(i)$ as the underlying coins (randomness) for all choices made by the signer related to node i. This enables the signer to recompute quantities as it needs rather than store them and yet be consistent, always creating the same quantities for a given node. Having pointed this out, however, in the rest of the paper we continue to work with stateful signing algorithms, since they are more natural and convenient in this context.

Computational costs. The cost for the signature algorithm is dominated by multiplications and inversions modulo N, for both of which there exist algorithms quadratic in $|N|$, and the cost of generating two standard signatures, which depends on the choice of underlying standard signature scheme. It is not strictly necessary to test membership in $\mathbb{Z}_N^*$, because it is very unlikely that a randomly generated value is not coprime with N. Verification takes a couple of multiplications mod N and two standard signature verifications. The composition of two signatures involves one multiplication and possibly an inversion in $\mathbb{Z}_N^*$.

Security. Forging a signature for FBTS-1 is trivial if an insecure instance is used for the modulus generator MG or the standard signature scheme SDS. The following theorem, however, states that the construction of FBTS-1 contains no weaknesses other than those induced by the underlying primitives.

[4] This means that we assign the name C_i to whichever of C_1, C_2 has its first component equal to the integer i. It is understood that if this is not possible then the algorithm halts and returns $\perp$. The same is true for the other similar steps that follow.

Theorem 1. *Let* MG *be a modulus generator and let* $\mathsf{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ *be a standard digital signature scheme. Let* $\mathsf{FBTS}\text{-}\mathsf{1}$ *be the transitive signature scheme associated to them as defined above. If the factoring problem associated to* MG *is hard and* SDS *is secure against forgery under adaptive chosen-message attack, then* $\mathsf{FBTS}\text{-}\mathsf{1}$ *is transitively unforgeable under adaptive chosen-message attack.*

We briefly sketch how a forgery for $\mathsf{FBTS}\text{-}\mathsf{1}$ can be used to either factor numbers generated by MG or break the underlying standard signature scheme SDS, and highlight a small but crucial detail for the analysis in a lemma. We refer to [3] for the full proof, that involves relatively standard reduction techniques and probability theory.

Signatures for $\mathsf{FBTS}\text{-}\mathsf{1}$ can be forged in only two ways: either there is the forgery that recycles node certificates from previously issued signatures, or there is the forgery that includes at least one new node certificate. The latter can be easily transformed into an attack on SDS: the new node certificate is a successful forgery for SDS, because it contains a standard signature on a message that was not signed before. A forgery of the first type provides the signer with an edge label δ' that is valid relative to the same public labels $L(i')$ and $L(j')$ he once issued for nodes i' and j' himself. (During this analysis, we assume wlog that $i' < j'$. If this is not the case, one can swap i' and j'.) Because these were computed as the squares of private labels $\ell(i')$ and $\ell(j')$, he now knows two square roots of $L(i') \cdot L(j')^{-1} \bmod N$, namely δ' and $\delta \equiv \ell(i') \cdot \ell(j')^{-1} \bmod N$.

It is tempting to say that since $\ell(i')$ and $\ell(j')$ were chosen at random, with probability one in two the signer now has two square roots δ and δ' such that $\delta \not\equiv \pm\delta' \bmod N$, enabling him to factor N. This argument would be correct if the forger only knew $L(i')$ and $L(j')$, without having any further information on exactly which root the signer knows. However, by signing edges involving nodes i' or j', the signer might have given away some additional information about his choices for $\ell(i')$ and $\ell(j')$. It is crucial to the security of the scheme that this information doesn't help the forger in creating a forgery with edge label $\delta' \equiv \pm\delta$, as this would annihilate the signer's advantage in factoring N. Fortunately, it turns out that the exact value of δ remains information-theoretically hidden from the forger as long as $\{i', j'\}$ is not in the transitive closure of the signed edges.

We will prove this fact using the information-theoretical argument that for every possible square-root δ of $L(i')L(j')^{-1} \bmod N$ there are exactly as many choices for the signer's private information ℓ that generate the given forger view and have $\ell(i')\ell(j')^{-1} \equiv \delta \bmod N$. As the secret labels are chosen uniformly at random from $\mathbb{Z}_N^*$, this implies that the issued signatures don't leak any useful information about which root the signer has in mind.

We represent the signer's secret information by a random variable $\boldsymbol{\ell}$ distributed uniformly over $\mathsf{Secrets} = \{\ell \mid \ell : V \to \mathbb{Z}_N^*\}$. The forger's view consists of a function L assigning a square mod N to each node in V, and a function Δ assigning an edge label in $\mathbb{Z}_N^*$ to each edge in $\widetilde{E}$. (We discard the standard digital signatures on the node certificates, as they are irrelevant for this analysis.) However, not just any pair of functions $\langle L, \Delta \rangle$ can occur as the forger's view. We say that forger view $\langle L, \Delta \rangle$ is *consistent* with $\ell \in \mathsf{Secrets}$ (and vice versa that ℓ is consistent with $\langle L, \Delta \rangle$) if and only if

$$L(i) \equiv \ell(i)^2 \bmod N \quad \text{for all } i \in V \tag{1}$$

$$\Delta(i,j) \equiv \ell(i)\ell(j)^{-1} \bmod N \quad \text{for all } \{i,j\} \in \widetilde{E}\ ,\ i < j \tag{2}$$

The set of all possible forger views Views can then be defined as the set of all pairs $\langle L, \Delta \rangle$ that are consistent with some $\ell \in \mathsf{Secrets}$. The actual view of the forger is a random variable $\mathbf{View}$ distributed over Views as induced by $\boldsymbol{\ell}$. The following lemma states that for every $\langle L, \Delta \rangle \in \mathsf{Views}$ and for every $\{i', j'\} \notin \widetilde{E}$, any square root δ of $L(i')L(j')^{-1} \bmod N$ is equally likely to be $\boldsymbol{\delta} \equiv \boldsymbol{\ell}(i')\boldsymbol{\ell}(j')^{-1} \bmod N$ when given only $\mathbf{View} = \langle L, \Delta \rangle$, and hence that no forger, on input only $\mathbf{View}$, can predict $\boldsymbol{\delta}$ with higher probability of success than random guessing. The following lemma formalizes this idea and is proven in the full version of this paper [3].

Lemma 1. *For any $\langle L, \Delta \rangle \in \mathsf{Views}$, for any $\{i', j'\} \notin \widetilde{E}$ and for any $\delta \in \mathbb{Z}_N^*$ with $\delta^2 \equiv L(i')L(j')^{-1} \bmod N$:*

$$\Pr\left[\boldsymbol{\delta} \equiv \delta \bmod N \mid \mathbf{View} = \langle L, \Delta \rangle\right] = \frac{1}{4}\ . \ \blacksquare$$

4 Eliminating Node Certificates via Hashing

The idea. The MRTS and $\mathsf{FBTS\text{-}1}$ schemes rely on an underlying standard digital signature scheme to convince the verifier that the public label $L(i)$ was associated to node i by the signer, and was not generated by some fraudulent third party. The disadvantage of this approach is that the signer has to provide the verifier with all necessary node certificates, thereby increasing the signature size as well as the computational cost for signing and verifying. In this section we show how the need for node certificates can be eliminated by specifying the public labels $L(i)$ via the output of a hash function on input i. No explicit certification is attached to this value. Rather, we will be able to show that the edge label provides an "implicit authentication" of the associated node label that suffices to be able to prove that the scheme is transitively unforgeable under adaptive chosen-message attack assuming the hardness of factoring, in a model where the hash function is based on a random oracle.

The hash function. The first thought regarding transforming $\mathsf{FBTS\text{-}1}$ based on this idea is to simply let $L(i) = H(i)$ where H is some public hash function. However, $L(i)$ needs to be a quadratic residue in $\mathbb{Z}_N^*$, where N is the signer's modulus, and this needs to be verifiable given N alone. In practice H must be built via a cryptographic hash function like SHA-1, which returns 160 bits. Standard techniques [5] can be used to build H from h so that it has range $\mathbb{Z}_N^*$, exploiting the fact that $\mathbb{Z}_N^*$ is dense in $\{0,1\}^k$ where $2^{k-1} \leq N < 2^k$ and that membership in $\mathbb{Z}_N^*$ can be tested in poly(k) time given N. However, given that no polynomial-time algorithm to test quadratic residuosity is known, there is no practical way to ensure that $H(i)$ is a quadratic residue while being able to verify this given N.

We could set $L(i) = H(i)^2 \bmod N$ but this reveals a square-root of $L(i)$ which makes the scheme insecure. Instead, reusing ideas from [1] and [13], we let the signer choose N to be a Blum integer (i.e. $N = pq$ with p and q primes such that $p \equiv q \equiv 3 \bmod 4$). Then it is well-known that exactly one square-root (called the principal one) of each square is itself a square mod N. As a consequence, every square mod N is also a fourth power mod N, and has exactly four fourth roots. Now we will choose $L(i), \ell(i)$ such that $L(i) \equiv H(i)^2 \equiv \ell(i)^4 \mod N$ where H is a hash function with range $\mathbb{Z}_N^*[+1]$, the elements of $\mathbb{Z}_N^*$ with Jacobi symbol $+1$. Since the Jacobi symbol can be computed in polynomial time given N, such a hash function can be easily built starting from a cryptographic hash function.

The FBTS-2 scheme. A modulus generator BG (as defined in Section 3), is said to be a *Blum modulus generator* if the primes p, q satisfy $p \equiv q \equiv 3 \pmod 4$. We associate to any given Blum modulus generator BG a transitive signature scheme FBTS-2 = (TKG, TSign, TVf, Comp) defined as follows:

- TKG, on input 1^k, runs $\mathsf{BG}(1^k)$ to obtain (N, p, q) and outputs $tpk = N$ as the public key and $tsk = (N, p, q)$ as the matching secret key. All the following algorithms are now assumed to have oracle access to a function $H_N\colon \{0,1\}^* \to \mathbb{Z}_N^*[+1]$.
- TSign maintains state (V, ℓ) where $V \subseteq \mathbb{N}$ is the set of all queried nodes and the function $\ell\colon V \to \mathbb{Z}_N^*$ assigns to each node $i \in V$ a secret label $\ell(i) \in \mathbb{Z}_N^*$. When invoked on inputs tsk, i, j, meaning when asked to produce a signature on edge $\{i, j\}$, it does the following:

```
If $j < i$ then $l \leftarrow j$ ; $j \leftarrow i$ ; $i \leftarrow l$     // swap $i$ and $j$ if necessary
If $i \notin V$ then
    $V \leftarrow V \cup \{i\}$ ; $L(i) \leftarrow H_N(i)^2 \bmod N$ ; $\ell(i) \stackrel{R}{\leftarrow} L(i)^{\frac{1}{4}} \bmod N$
If $j \notin V$ then
    $V \leftarrow V \cup \{j\}$ ; $L(j) \leftarrow H_N(j)^2 \bmod N$ ; $\ell(j) \stackrel{R}{\leftarrow} L(j)^{\frac{1}{4}} \bmod N$
$\delta \leftarrow \ell(i)\ell(j)^{-1} \bmod N$
```

  where the notation $x \stackrel{R}{\leftarrow} (y)^{\frac{1}{4}} \bmod N$ means that x is chosen at random from all fourth roots of $y \bmod N$. (These roots can be efficiently computed using the prime factors p and q.) Return δ as the signature on $\{i, j\}$.
- TVf, on input $tpk = N$, nodes i, j and a signature δ, first swaps i and j if $j < i$. It returns 1 if $H_N(i)^2 \equiv \delta^4 H_N(j)^2 \mod N$ and returns 0 otherwise.
- Comp on input nodes i, j, k and signatures δ_1, δ_2, proceeds as follows:

```
If $j < i$ then $\delta_1 \leftarrow \delta_1^{-1} \bmod N$   ;   If $k < j$ then $\delta_2 \leftarrow \delta_2^{-1} \bmod N$
$\delta \leftarrow \delta_1 \cdot \delta_2 \bmod N$
```

  and outputs δ as the transitive composition.

A proof by induction can be used to show the following.

Proposition 2. *The* FBTS-2 *transitive signature scheme described above satisfies the correctness requirement of Definition 1.*

Computational costs. Since half of the elements in $\mathbb{Z}_N^*$ have Jacobi symbol $+1$, a hash function evaluation requires the computation of two Jacobi symbols on average, which takes time quadratic in $|N|$. Computing square roots, however, is cubic in $|N|$, so the computation of the fourth roots (by extracting square roots twice) will dominate the cost of generating signatures. Verification and composition of signatures involve multiplications, inverses and Jacobi symbols mod N, all of which are operations quadratic in $|N|$.

Security. In [3], we prove breaking the FBTS-2 scheme equivalent to factoring in the random oracle model. This means that in the experiment $\mathbf{Exp}^{\text{tu-cma}}_{\mathsf{FBTS\text{-}2},F}(k)$ used to define the advantage of an adversary F, the function H_N is assumed to be chosen at random from the space of all functions mapping $\{0,1\}^*$ to $\mathbb{Z}_N^*[+1]$. The result is stated as a theorem below.

Theorem 2. *Let* BG *be a Blum modulus generator. Let* FBTS-2 *be the transitive signature scheme as defined above. If the factoring problem associated to* BG *is hard, then* FBTS-2 *is transitively secure against forgery under adaptive chosen-message attack in the random oracle model.*

5 Transitive Signatures based on RSA

In [12], Micali and Rivest mentioned the following scheme as a simpler scheme that can only be proven secure against a static forger, meaning that the forger must commit to all of his oracle queries before seeing the responses to any of them. While we still don't know how to prove security against an adaptive forger assuming only the one-wayness of RSA (and whether this can be done at all), we revisit the scheme here to prove it secure under the assumption that the one-more RSA-inversion problem, as described in the introduction, is hard.

In analogy with the modulus generator of the previous section, we define an *RSA key generator* RG as a randomized, polynomial-time algorithm that on input 1^k outputs a tuple (N, e, d) where $2^{k-1} \leq N < 2^k$ and $ed \equiv 1 \mod \varphi(N)$. We do not restrict the type of generator, but only assume that its associated one-more RSA-inversion problem is hard.

The RSATS-1 scheme. We associate to any RSA key generator RG and to any standard digital signature scheme $\mathsf{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ a transitive signature scheme $\mathsf{RSATS\text{-}1} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ defined as follows:

- TKG runs $\mathsf{SKG}(1^k)$ to generate a key pair (spk, ssk) for SDS and runs $\mathsf{RG}(1^k)$ to generate an RSA key (N, e, d). It outputs $tpk = (N, e, spk)$ as the public key and $tsk = (N, d, ssk)$ as the matching secret key.
- The signing algorithm TSign is identical to that of the FBTS-1 scheme, except that now the public label $L(i)$ for node i is computed as $L(i) \equiv \ell(i)^e \bmod N$. The state information kept, the way of creating node certificates and the way of constructing the signature remain unchanged.
- The verification algorithm TVf is also very similar to that of FBTS-1: the only difference is the test on the edge label, which now consists of checking that $L_i \equiv \delta^e L_j \bmod N$.

- The Comp algorithm is perfectly identical to the composition algorithm of $\mathsf{FBTS\text{-}1}$.

The scheme described above can be shown to satisfy the correctness requirement of Definition 1 using a proof by induction.

Computational costs. Depending on the actual implementation of RSA, its computational overhead probably dominates over quadratic-time operations such as multiplications and inverses mod N. The generation of a transitive signature needs in the worst case two RSA encryptions, and two standard signatures for the node certificates. Signature verification takes one RSA encryption and two standard signature verifications, while quadratic operations are predominant in the composition algorithm.

Security of $\mathsf{RSATS\text{-}1}$. The security analysis for this scheme against an adaptive forger is very similar to the analysis of $\mathsf{FBTS\text{-}1}$, except that this time the certificate-recycling type of forgery can be proven equivalent to solving the one-more RSA-inversion problem associated to RG. The proof for the following theorem is given in [3].

Theorem 3. *Let* RG *be an RSA key generator and let* $\mathsf{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ *be a standard digital signature scheme. Let* $\mathsf{RSATS\text{-}1}$ *be the transitive signature scheme as defined above. If the one-more RSA-inversion problem associated to* RG *is hard and* SDS *is secure against forgery under adaptive chosen-message attack, then* $\mathsf{RSATS\text{-}1}$ *is transitively secure against forgery under adaptive chosen-message attack.*

The $\mathsf{RSATS\text{-}2}$ scheme. The idea of replacing node certificates by a suitable hash function can also be applied to the $\mathsf{RSATS\text{-}1}$ scheme. Since this time the public labels are uniformly distributed over the whole of $\mathbb{Z}_N^*$, we can use a hash function $H_N : \mathbb{N} \rightarrow \mathbb{Z}_N^*$ to directly map node i to its public label $L(i) = H_N(i)$. The unambiguous invertibility of RSA encryption allows for the first completely stateless signature algorithm: the signature for edge $\{i, j\}$ (swapping i and j if $j < i$) is computed as $\delta \equiv \left(H_N(i) \cdot H_N(j)^{-1}\right)^d \bmod N$. The verification of signature δ for edge $\{i, j\}$, $i < j$, is done by checking that $H_N(i) \equiv \delta^e H_N(j) \bmod N$. Composition of signatures works as in the $\mathsf{FBTS\text{-}2}$ scheme by multiplying the (if necessary inverted) edge labels. The proofs of correctness and security (in the random oracle model, assuming that the one-more RSA-inversion problem associated to RG is hard) are very similar to those given for $\mathsf{RSATS\text{-}1}$ and were hence omitted.

Acknowledgments

Mihir Bellare is supported in part by NSF grant CCR-0098123, NSF grant ANR-0129617, and an IBM Faculty Partnership Development Award. Gregory Neven is supported by a Research Assistantship and a travel credit from the Fund for Scientific Research, Flanders (Belgium) (F.W.O.–Vlaanderen). Work done while Gregory was visiting UCSD.

References

1. M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. *Advances in Cryptology – ASIACRYPT '00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.
2. M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. The One-More-RSA-Inversion problems and the security of Chaum's blind signature scheme. Cryptology ePrint Archive: Report 2001/002, `http://eprint.iacr.org/2001/002/`. Preliminary version, entitled "The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme," in *Financial Cryptography '01*, Lecture Notes in Computer Science Vol. 2339, P. Syverson ed., Springer-Verlag, 2001.
3. M. Bellare and G. Neven. Transitive signatures based on factoring and RSA. Full version of this abstract, available via `http://www-cse.ucsd.edu/users/mihir`.
4. M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. *Advances in Cryptology – CRYPTO '02*, Lecture Notes in Computer Science Vol. 2442 , M. Yung ed., Springer-Verlag, 2002.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *Proceedings of the 1st Annual Conference on Computer and Communications Security*, ACM, 1993.
6. S. Chari, T. Rabin and R. Rivest. An efficient signature scheme for route aggregation. Manuscript, February 2002. `http://theory.lcs.mit.edu/~rivest/publications.html`.
7. D. Chaum. Blind signatures for untraceable payments. *Advances in Cryptology - CRYTPO 82 Proceedings*, D. Chaum, R. Rivest and A. Sherman eds., Plenum Press.
8. O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217.
9. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, Vol. 17, No. 2, April 1988, pp. 281–308.
10. L. Guillou and J. J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. *Advances in Cryptology – CRYPTO '88*, Lecture Notes in Computer Science Vol. 403, S. Goldwasser ed., Springer-Verlag, 1988.
11. R. Johnson, D. Molnar, D. Song and D. Wagner. Homomorphic signature schemes. *Topics in Cryptology – CT-RSA '02*, Lecture Notes in Computer Science Vol. 2271 , B. Preneel ed., Springer-Verlag, 2002.
12. S. Micali and R. Rivest. Transitive signature schemes. *Topics in Cryptology – CT-RSA '02*, Lecture Notes in Computer Science Vol. 2271 , B. Preneel ed., Springer-Verlag, 2002.
13. S. Micali and L. Reyzin. Improving the exact security of digital signature schemes. *Journal of Cryptology*, Vol. 15, Number 1, 2002, pp. 1–18.
14. R. Rivest. Two signature schemes. Slides from talk given at Cambridge University, October 17, 2000. `http://theory.lcs.mit.edu/~rivest/publications.html`.
15. R. Steinfeld, L. Bull and Y. Zheng. Content Extraction Signatures. *Information Security and Cryptology - ICISC 2001*, Lecture Notes in Computer Science Vol. 2288, K. Kim ed., Springer-Verlag, 2002.