An extended abstract of this paper appeared as "Transitive Signatures based on Factoring and RSA" in *Advances in Cryptology – ASIACRYPT '02*, Lecture Notes in Computer Science Vol. 2501, Y. Zheng ed., Springer-Verlag, 2002. This is a slightly revised version of the full paper that appeared in *IEEE Transactions on Information Theory*, Vol. 51, No. 6, pp. 2133–2151, June 2005.

Transitive Signatures: New Schemes and Proofs

MIHIR BELLARE[*] GREGORY NEVEN[†]

March 2005

Abstract

We present novel realizations of the transitive signature primitive introduced by Micali and Rivest [MR02], enlarging the set of assumptions on which this primitive can be based, and also providing performance improvements over existing schemes. More specifically, we propose new schemes based on factoring, the hardness of the one-more discrete logarithm problem, and gap Diffie-Hellman groups. All these schemes are proven transitively unforgeable under adaptive chosen-message attack in the standard (not random-oracle) model. We also provide an answer to an open question raised in [MR02] regarding the security of their RSA-based scheme, showing that it is transitively unforgeable under adaptive chosen-message attack assuming the security of RSA under one-more-inversion. We then present hash-based modifications of the RSA, factoring and gap Diffie-Hellman based schemes that eliminate the need for "node certificates" and thereby yield shorter signatures. These modifications remain provably secure under the same assumptions as the starting scheme, in the random oracle model.

Keywords: Signatures, RSA, transitive signatures.

[*] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir. Supported in part by NSF grant CCR-0098123, NSF grant ANR-0129617, and an IBM Faculty Partnership Development Award.

[†] Dept. of Electrical Engineering, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium. E-Mail: Gregory.Neven@esat.kuleuven.be. URL: http://www.neven.org. Work done while at Dept. of Computer Science and Engineering, University of California San Diego. Supported in part by a Research Assistantship and a travel credit from the Fund for Scientific Research, Flanders (Belgium), in part by the Concerted Research Action (GOA) Mefisto 2000/06 and Ambiorix 2005/11 of the Flemish Government, and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

Contents

1 Introduction

We present novel realizations of the transitive signature primitive introduced by Micali and Rivest [MR02], and also provide an answer to an open question they raise regarding the security of an RSA based scheme.

1.1 Background

The concept. The context envisioned by Micali and Rivest [MR02] 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. With regard to applications, Micali and Rivest [MR02] suggest administrative domains, where nodes represent machines and an edge between i and j means that i and j are in the same domain. It seems that a truly compelling application, however, remains to be found.

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 [MR02] to exclude this trivial solution: the growth in signature size, and the loss of privacy incurred by having signatures carry information about their history. The main result of [MR02] is a (non-trivial) transitive signature scheme, here denoted $\mathcal{DLTS}$, that is proven to be (transitively) unforgeable under adaptive chosen-message attack (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, here denoted $\mathcal{RSATS}$*-1*, but point out that even though it seems secure, and a proof of transitive unforgeability under non-adaptive chosen-message attacks exists, there is no known proof of transitive 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.

This work. In summary, transitive signatures (transitively unforgeable under adaptive chosen-message attacks) at this point have just a single realization, namely the $\mathcal{DLTS}$ 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 $\mathcal{DLTS}$ 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 $\mathcal{DLTS}$ 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 on the concatenation of i and $L(i)$ under the standard scheme. The signature of an edge contains the

Scheme	Signing cost	Verification cost	Composition cost	Signature size
DLTS	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 $\mathbb{Z}_q$
RSATS-1	2 stand. sigs 2 RSA encs	2 stand. verifs 1 RSA enc.	$O(\|N\|^2)$ ops	2 stand. sigs 3 points in $\mathbb{Z}_N^*$
FactTS-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 $\mathbb{Z}_N^*$
DLTS-1M	2 stand. sigs 1 exp. in $\mathbb{G}$	2 stand. verifs 1 exp. in $\mathbb{G}$	1 add in $\mathbb{Z}_q$	2 stand. sigs 2 points in $\mathbb{G}$ 1 point in $\mathbb{Z}_q$
GapTS-1	2 stand. sigs 2 exp. in $\hat{\mathbb{G}}$	2 stand. verifs 1 $\mathsf{S}_{\rm ddh}$	$O(\|N\|^2)$ ops	2 stand. sigs 3 points in $\hat{\mathbb{G}}$
RSATS-2	1 RSA dec.	1 RSA enc.	$O(\|N\|^2)$ ops	1 point in $\mathbb{Z}_N^*$
FactTS-2	2 sq. roots in $\mathbb{Z}_N^*$	$O(\|N\|^2)$ ops	$O(\|N\|^2)$ ops	1 point in $\mathbb{Z}_N^*$
GapTS-2	1 exp. in $\hat{\mathbb{G}}$	1 $\mathsf{S}_{\rm ddh}$	$O(\|N\|^2)$ ops	1 point in $\hat{\mathbb{G}}$

Figure 1: Cost comparisons amongst transitive signature schemes. The word "stand." refers to operations of the underlying standard signature scheme, which are eliminated for *RSATS-2*, *FactTS-2* and *GapTS-2*. $\mathbb{G}$ denotes the group of prime order q used in *DLTS*, and N denotes a modulus product of two primes as used in the RSA and factoring-based schemes. $\hat{\mathbb{G}}$ is a gap Diffie-Hellman group and $\mathsf{S}_{\rm ddh}$ refers to the decision Diffie-Hellman algorithm in $\hat{\mathbb{G}}$. 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.

certificates of its endpoints plus an *edge label* δ. Verification of an edge signature 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.

The paradigm is useful, but brings an associated cost. Producing a signature for an edge can involve computing two standard 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.

1.2 Transitive signatures based on RSA

THE *RSATS-1* SCHEME. This scheme, briefly mentioned in [MR02], 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 transitively 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 transitively unforgeable under adaptive chosen-message attack.

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.

Our result. We prove that $\mathcal{RSATS}$-1 is transitively 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 [BNPS03], who used it to prove the security of Chaum's blind signature scheme [Cha83]. It was also used in [BP02] to prove security of the GQ identification scheme against impersonation under active attack, which had been open from [GQ89].

1.3 New transitive signature schemes

The $\mathcal{FactTS}$-1 scheme. After seeing the $\mathcal{RSATS}$-1 scheme, one might wonder whether there exists a transitive signature scheme that is provably secure (transitively unforgeable under adaptive chosen-message attack) under the standard one-wayness assumption on RSA. We answer this question positively by presenting the $\mathcal{FactTS}$-1 scheme that is provably secure under the (even weaker) assumption that factoring is hard.

In our $\mathcal{FactTS}$-1 scheme, 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 $\mathcal{FactTS}$-1 is transitively 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.

One might wonder why proofs under standard assumptions exist for $\mathcal{DLTS}$ and $\mathcal{FactTS}$-1 but remain elusive for $\mathcal{RSATS}$-1 in spite of the obvious similarities between these schemes. The proofs for $\mathcal{DLTS}$ and $\mathcal{FactTS}$-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 $\mathcal{RSATS}$-1, the edge label is uniquely determined by the two node certificates, and this paradigm fails.

Since $\mathcal{FactTS}$-1 continues to employ the node certification paradigm, it incurs the same costs as $\mathcal{DLTS}$ and $\mathcal{RSATS}$-1 from the use of the standard signature scheme. However, as Figure 1 indicates, it is otherwise computationally cheaper than $\mathcal{DLTS}$ and $\mathcal{RSATS}$-1 for signing and verifying, reducing the extra cost from cubic (exponentiation) to quadratic (a couple of multiplications and an inverse).

The $\mathcal{DLTS}$-1$\mathcal{M}$ scheme. The $\mathcal{DLTS}$ scheme [MR02] uses two generators. We briefly note a simpler and perhaps more natural discrete-log based scheme called $\mathcal{DLTS}$-1$\mathcal{M}$ that uses a single generator. This scheme is a discrete-log based analog of $\mathcal{RSATS}$-1. As Figure 2 indicates, it offers some slight performance improvements over $\mathcal{DLTS}$. However, while the security of $\mathcal{DLTS}$ is proven under the standard discrete-logarithm assumption [MR02], our proof of security of $\mathcal{DLTS}$-1$\mathcal{M}$ requires a stronger assumption, namely the hardness of the one-more discrete logarithm problem as defined in [BNPS03].

The tradeoff here is analogous to one arising for discrete-logarithm based identification (ID)

Scheme	Proven to be transitively unforgeable under *adaptive* chosen-message attack assuming	RO Model?
DLTS	Security of standard signature scheme Hardness of discrete logarithm in prime-order group	No
RSATS-1	Security of standard signature scheme RSA is secure against one-more-inversion attack	No
FactTS-1	Security of standard signature scheme Hardness of factoring	No
DLTS-1M	Security of standard signature scheme Hardness of one-more discrete logarithm in prime-order group	No
GapTS-1	Security of standard signature scheme One-more gap Diffie-Hellman assumption	No
RSATS-2	RSA is secure against one-more-inversion attack	Yes
FactTS-2	Hardness of factoring	Yes
GapTS-2	One-more gap Diffie-Hellman assumption	Yes

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

schemes. *DLTS* is similar to Okamoto's two-generator using ID scheme [Oka93] while *DLTS-1M* is similar to Schnorr's one-generator using ID scheme [Sch90]. Schnorr's scheme is simpler, more natural and slightly more efficient. However, while Okamoto proved his scheme secure (against impersonation under active attack) under the standard discrete-logarithm assumption, the proof of security for Schnorr's scheme (which remained elusive for a while) is based on the hardness of the one-more discrete logarithm problem [BP02].

The *GapTS-1* scheme. Gap DH groups are groups where the CDH (Computational Diffie-Hellman) problem is hard but the DDH (Decision Diffie-Hellman) problem is easy. They have been used to yield short signatures [BLS01] and also simple, efficient schemes for threshold, blind and multi-signatures [Bol03].

We present a transitive signature scheme *GapTS-1* using these groups as well. It is proven transitively unforgeable under adaptive chosen-message attack assuming hardness of the one-more CDH problem introduced in [Bol03].

This scheme is actually not of direct interest, because it is inferior to *DLTS-1M* both with regard to assumptions made to prove security and with regard to performance. (In any group where one may implement *GapTS-1*, one may also implement *DLTS-1M*, and obtain security under weaker assumptions and with lower cost.) The value of *GapTS-1* is that, unlike *DLTS-1M* or *DLTS*, it is amenable to the hash-based modification described next, resulting in *GapTS-2*, a scheme that has the shortest signatures amongst all schemes we have discussed.

1.4 Eliminating node certificates via hashing

The *RSATS-2* scheme. The *RSATS-1* scheme is amenable to a hash-based modification which eliminates the need for node certificates and thereby removes the standard signature scheme, and all its associated costs, from the picture. In the *RSATS-2* scheme, the public label of a node i is not chosen by the signer but rather implicitly specified as the output of a public hash function applied to i, and RSA decryption is used to compute edge labels. We prove that *RSATS-2* is transitively unforgeable under adaptive chosen-message attack, assuming the hardness of one-more RSA-inversion in a model where the hash function is a random oracle [BR93].

The *FactTS-2* scheme. The fact that squaring modulo a composite is a trapdoor one-way function makes *FactTS-1* amenable to a similar elimination of node certificates via hashing. We present the *FactTS-2* transitive signature scheme where the public label of a node i is not chosen by the signer but rather specified via the output of a public hash function applied to i. (A difficulty, addressed in Section 5.2, is that the hash output might not be a quadratic residue.) We prove that *FactTS-2* is transitively unforgeable under adaptive chosen-message attacks in the random oracle model assuming factoring the underlying modulus is hard.

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.

The *GapTS-2* scheme. The *GapTS-1* scheme is also amenable to a similar hash-based modification, resulting in a scheme, *GapTS-2*, whose parameters are depicted in Figure 1. The signature here is simply a group element, and by the nature of gap DH groups, this means the *GapTS-2* scheme has the shortest signatures of all.

DLTS and *DLTS-1M*. The *DLTS* and *DLTS-1M* schemes are not amenable to the hash-based modification since the discrete exponentiation function is not trapdoor over the groups used for these schemes.

Stateful versus stateless schemes. The five basic schemes *DLTS*, *DLTS-1M*, *RSATS-1*, *FactTS-1*, *GapTS-1* are stateful. (Meaning, the signing algorithm maintains state across invocations.) As discussed in Section 6, there is a simple, general way to modify these schemes to result in stateless ones. It may be interesting to note, however, that the *RSATS-2* and *GapTS-2* schemes are naturally stateless. (*FactTS-2* is not, and needs to be modified according to Section 6 if we want a stateless version.)

1.5 Definitional contributions

Regarding the composability property, Micali and Rivest [MR02, 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, while surely they meant 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 [JMSW02], who require a condition that implies this.) But this requirement is too strong in the current context. Indeed, as we show in Appendix A, the *DLTS* 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 *DLTS* 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 2.1, we follow [JMSW02] 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 [JMSW02], this guarantees privacy.) The $\mathcal{DLTS}$ scheme, as well as all our schemes, meet this strong definition.

1.6 Related work, and versions of this paper

Transitive signatures are one case of a more general concept promulgated by Rivest [Riv00] 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 [JMSW02] formalize a notion of homomorphic signature schemes that captures this. Context Extraction Signatures, as introduced earlier by [SBZ02], as well as redactable signatures and set-homomorphic signatures [JMSW02], fall in this framework. A signature scheme that is homomorphic with respect to the prefix operation is presented by Chari, Rabin and Rivest [CRR02].

The preliminary (i.e. conference proceedings) version of this paper [BN02] contained the results pertaining to $\mathcal{RSATS}$-*1*, and presented the new schemes $\mathcal{F}act\mathcal{TS}$-*1* and $\mathcal{F}act\mathcal{TS}$-*2*. The current, full version, besides including proofs omitted in the preliminary version, also adds the new schemes $\mathcal{DLTS}$-*1*$\mathcal{M}$, $\mathcal{G}ap\mathcal{TS}$-*1*, $\mathcal{G}ap\mathcal{TS}$-*2*.

We now mention some work on transitive signatures done subsequent to the appearance of the conference version of our work [BN02]. First is the work of Hohenberger [Hoh03], who presents a general framework for the design and analysis of transitive signature schemes, as well as some results on the difficulty of constructing transitive signature schemes for directed graphs. Second is the work of Shahandashti, Salmasizadeh and Mohajeri [SSM05]. These authors independently discovered our $\mathcal{G}ap\mathcal{TS}$-*1* scheme. (The scheme in their paper is the same as $\mathcal{G}ap\mathcal{TS}$-*1* barring notation and some minor implementation details.) They however claim to prove a stronger security result, namely that the scheme is secure under the CDH assumption (and the security of an underlying standard signature scheme), rather than under the stronger *one-more* CDH assumption that we rely on. Unfortunately, we found their proof to be flawed, invalidating their result. We provide details of the flaw in Appendix C. As a consequence, our result about $\mathcal{G}ap\mathcal{TS}$-*1* remains the best known so far about the security of this scheme.

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 $x \stackrel{\$}{\leftarrow} S$ denotes that x is selected randomly from set S. If A is a possibly randomized algorithm then the notation $x \stackrel{\$}{\leftarrow} \mathsf{A}(a_1, a_2, \ldots, a_n)$ denotes that x is assigned the outcome of the experiment of running A on inputs $a_1, a_2, \ldots, a_n$.

Graphs. All graphs in this paper are undirected. 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. 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

```
$(tpk, tsk) \overset{\$}{\leftarrow} \mathsf{TKG}(1^k)$
$S \leftarrow \emptyset$ ; $Legit \leftarrow \mathtt{true}$ ; $NotOK \leftarrow \mathtt{false}$
Run $\mathsf{A}$ with its oracles until it halts, replying to its oracle queries as follows:
    If $\mathsf{A}$ makes $\mathsf{TSign}$ query $i, j$ then
        If $i = j$ then $Legit \leftarrow \mathtt{false}$
        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 $NotOK \leftarrow \mathtt{true}$
    If $\mathsf{A}$ 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
            $Legit \leftarrow \mathtt{false}$
        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 $NotOK \leftarrow \mathtt{true}$
When $\mathsf{A}$ halts, output $(Legit \wedge NotOK)$ and halt
```

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

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^*. 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* $\mathcal{TS} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ 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 1.5, formulating a correctness requirement for the composition algorithm is more delicate. Micali and Rivest [MR02] seem to suggest that composition is only required to work when the given signatures were explicitly produced by the signer, while surely they meant composition to work even if the given signatures were themselves obtained via composition. Furthermore the indistinguishability requirement is not formalized in [MR02].

Definitions taking these issues into account are however provided in [JMSW02]. They ask that whenever the composition algorithm is invoked on valid signatures (valid meaning 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 $\mathcal{DLTS}$, $\mathcal{RSATS}$-*1*, and also not true for our new schemes. For all 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, as we illustrate in Appendix A. (Roughly, this happens 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. (While computational indistinguishability would be sufficient to guarantee privacy, we continue to follow [JMSW02] here in capturing indistinguishability by the strong requirement that composed signatures are identical to original ones, because all schemes seem to satisfy this simpler definition anyway. We weaken their requirement though 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 2.1 We say that the transitive signature scheme $\mathcal{TS}$ 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 *Legit* = `true`) but violate correctness (meaning *NotOK* = `true`). The experiment returns `true` iff A wins. The definition requires that this happen with probability zero.

Our definition asks that real and composed signatures be the same. This could be relaxed to ask that they are, say, statistically indistinguishable. Such a definition is however more complicated, and not needed in this paper since the schemes all meet our definition, so we have not pursued it.

Security of transitive signature schemes. We recall the notion of security of [MR02]. Associated to transitive signature scheme $\mathcal{TS} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$, algorithm F (called a tu-cma *adversary*) and security parameter $k \in \mathbb{N}$ parameter $k \in \mathbb{N}$ is an experiment, denoted

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

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). 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. Let E be the set of all edges $\{i, j\}$ such that F made oracle query i, j, and let V be the set of all nodes involved in edges in E. Eventually, F will output $i', j' \in \mathbb{N}$ and a forgery σ'. 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 of graph $G = (V, E)$. The experiment returns 1 if F wins and 0 otherwise. The *advantage* of F in its attack on $\mathcal{TS}$ is the function $\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{TS},F}(\cdot)$ defined for $k \in \mathbb{N}$ by

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

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

Random oracle model. Some of our schemes will be defined in the random oracle model [BR93], 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 $\mathbf{Exp}^{\text{tu-cma}}_{\mathcal{TS},\mathsf{F}}(k)$ are augmented to choose a random function mapping $\{0,1\}^*$ to an appropriate range, possibly depending on the public key, and the adversary as well as the $\mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp}$ algorithms then get oracle access to this function. In Definition 2.1, the probability includes the choice of these functions, and so does the $\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{TS},\mathsf{F}}(k)$.

Standard signature schemes. Some of our schemes use an underlying standard digital signature scheme $\mathcal{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$, described as usual via its polynomial-time key generation, signing and verification algorithms. We use the security definition of unforgeability under chosen-message attack [GMR88], where the forger B is given adaptive oracle access to the signing algorithm, and its advantage $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(k)$ in breaking $\mathcal{SDS}$ 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 $\mathcal{SDS}$ is said to be unforgeable under adaptive chosen-message attack if $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(\cdot)$ is negligible for every polynomial-time forger B.

3 Transitive Signatures based on RSA

The $\mathcal{RSATS}$*-1* scheme was noted in [MR02] as a simple alternative to $\mathcal{DLTS}$ which can be shown to be transitively unforgeable under non-adaptive chosen-message attacks assuming RSA is one-way. We do not know whether the same assumption suffices to prove it is transitively unforgeable under adaptive chosen-message attacks, but here we will show that this is true under a stronger assumption.

RSA generators and the assumption. A *RSA key generator* $\mathsf{K}_{\mathrm{rsa}}$ is a randomized, polynomial-time algorithm that on input 1^k outputs a tuple (N, e, d) where $2^{k-1} \leq N < 2^k$, $ed \equiv 1 \mod \varphi(N)$ and N is the product of two distinct, odd primes. We do not attempt to pin down exactly how the generator operates, for example with regard to the distribution on the primes it chooses, or the choice of encryption exponent. (The latter may be chosen to be a small number, like 3, for efficiency, or a large number if desired.) All we ask is that the one-more RSA-inversion problem associated to the generator be hard, as defined below. This makes our results more general.

We recall the notion of security under one more inversion [BNPS03]. An om-rsa *adversary* is a randomized, polynomial-time algorithm A that gets input N, e and has access to two oracles. The first is an RSA-inversion oracle $\textsc{Inv}(\cdot)$ that given $Y \in \mathbb{Z}_N^*$ returns $Y^d \bmod N$. The second is a challenge oracle $\textsc{Chall}()$ that, each time it is invoked (it takes no inputs), returns a random challenge point from $\mathbb{Z}_N^*$. Adversary A wins the game if it succeeds in inverting all n points output by the challenge oracle, using strictly less than n queries to the RSA-inversion oracle. More formally, A is run in the following experiment:

```
Experiment $\mathbf{Exp}^{\text{om-rsa}}_{\mathsf{K}_{\mathrm{rsa}},\mathsf{A}}(k)$:
    $(N, e, d) \xleftarrow{\$} \mathsf{K}_{\mathrm{rsa}}(1^k)$ ; $n \leftarrow 0$ ; $m \leftarrow 0$
    $(w_1, \ldots, w_{n'}) \xleftarrow{\$} \mathsf{A}^{\textsc{Inv},\textsc{Chall}}(N, e)$
    If $n' = n$ and $m < n$ and $\forall\, i = 1 \ldots n \ : \ w_i^e \equiv W_i \bmod N$
    Then return 1 else return 0
```

```
Oracle $\textsc{Inv}(Y)$:
    $m \leftarrow m + 1$ ; Return $Y^d \bmod N$
Oracle $\textsc{Chall}()$ :
    $i \leftarrow i + 1$ ; $W_i \xleftarrow{\$} \mathbb{Z}_N^*$
    Return $W_i$ .
```

The om-rsa *advantage* of A, denoted $\mathbf{Adv}^{\text{om-rsa}}_{\mathsf{K}_{\mathrm{rsa}},\mathsf{A}}(k)$, is the probability that the above experiment returns 1, taken over the coins of $\mathsf{K}_{\mathrm{rsa}}$, the coins of A, and the coins used by the challenge oracle across its invocations. We say that the one-more RSA-inversion problem associated to $\mathsf{K}_{\mathrm{rsa}}$ is hard if the function $\mathbf{Adv}^{\text{om-rsa}}_{\mathsf{K}_{\mathrm{rsa}},\mathsf{A}}(\cdot)$ is negligible for all polynomial-time adversaries A.

Note that the standard one-wayness assumption on $\mathsf{K}_{\mathrm{rsa}}$ can be formulated as requiring that the function $\mathbf{Adv}^{\text{om-rsa}}_{\mathsf{K}_{\mathrm{rsa}},\mathsf{A}}(\cdot)$ is negligible for all polynomial-time adversaries A that make one challenge query and no inversion queries. Thus, hardness of the one-more RSA-inversion problem is an extension of one-wayness.

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

- $\mathsf{TKG}(1^k)$ does the following

  (1.1) Run $\mathsf{SKG}(1^k)$ to generate a key pair $(\mathit{spk}, \mathit{ssk})$ for $\mathcal{SDS}$

  (1.2) Run $\mathsf{K}_{\mathrm{rsa}}(1^k)$ to get a triple (N, e, d)

  (1.3) Output $\mathit{tpk} = (N, e, \mathit{spk})$ as the public key and $\mathit{tsk} = (N, e, \mathit{ssk})$ as the secret key.

  Note that the exponent d is 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 ℓ: $V \to \mathbb{Z}_N^*$ assigns to each node $i \in V$ a *secret label* $\ell(i) \in \mathbb{Z}_N^*$, while the function L: $V \to \mathbb{Z}_N^*$ assigns to each node $i \in V$ a *public label* $L(i)$, and the function Σ: $V \to \{0,1\}^*$ 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:

```
(2.1)  If $i > j$ then $swap(i, j)$
(2.2)  If $i \notin V$ then
(2.3)      $V \leftarrow V \cup \{i\}$
(2.4)      $\ell(i) \xleftarrow{\$} \mathbb{Z}_N^*$ ; $L(i) \leftarrow \ell(i)^e \bmod N$
(2.5)      $\Sigma(i) \leftarrow \mathsf{SSign}(ssk, i \| L(i))$
(2.6)  If $j \notin V$ then
(2.7)      $V \leftarrow V \cup \{j\}$
(2.8)      $\ell(j) \xleftarrow{\$} \mathbb{Z}_N^*$ ; $L(j) \leftarrow \ell(j)^e \bmod N$
(2.9)      $\Sigma(j) \leftarrow \mathsf{SSign}(ssk, j \| L(j))$
(2.10) $\delta \leftarrow \ell(i)\ell(j)^{-1} \bmod N$
(2.11) $C_i \leftarrow (i, L(i), \Sigma(i))$ ; $C_j \leftarrow (j, L(j), \Sigma(j))$
(2.12) Return $(C_i, C_j, \delta)$ as the signature of $\{i, j\}$.
```

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

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

```
(3.1) If $i > j$ then $swap(i, j)$
(3.2) Parse $\sigma$ as $(C_i, C_j, \delta)$, parse $C_i$ as $(i, L_i, \Sigma_i)$, parse $C_j$ as $(j, L_j, \Sigma_j)$
(3.3) If $\mathsf{SVf}(spk, i \| L_i, \Sigma_i) = 0$ or $\mathsf{SVf}(spk, j \| L_j, \Sigma_j) = 0$ then return 0
(3.4) If $\delta^e \equiv L_i L_j^{-1} \bmod N$ then return 1 else return 0.
```

- The composition algorithm Comp takes tpk, nodes i, j, k and signatures σ_1 and σ_2, and computes a composed signature for edge $\{i, k\}$ as follows:

```
(4.1) If $i > k$ then $swap(i, k)$ ; $swap(\sigma_1, \sigma_2)$
(4.2) Parse $\sigma_1$ as $(C_1, C_2, \delta_1)$ ; Parse $\sigma_2$ as $(C_3, C_4, \delta_2)$
(4.3) If $i > j$ then $swap(C_1, C_2)$ ; $\delta_1 \leftarrow \delta_1^{-1} \bmod N$
(4.4) If $j > k$ then $swap(C_3, C_4)$ ; $\delta_2 \leftarrow \delta_2^{-1} \bmod N$
(4.5) $\delta \leftarrow \delta_1 \delta_2 \bmod N$
Return $(C_1, C_4, \delta)$ as the signature for $\{i, k\}$.
```

Proposition 3.1 The $\mathcal{RSATS}$-*1* transitive signature scheme described above satisfies the correctness requirement of Definition 2.1. ▌

The proof is provided in Appendix A. 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.

Computational costs. As Figure 1 indicates, over and above costs associated to the standard signature scheme, signing and verifying require RSA encryptions, whose cost dominates that of quadratic-time operations such as multiplications and inverses mod N. The cost of the RSA encryptions depends on the choice of encryption exponent made by the RSA key generator, and can be small for a small exponent. Composition is efficient, involving only quadratic-time operations.

Security of $\mathcal{RSATS}$-*1*. The following theorem says that as long as the RSA one-more-inversion problem is hard for the associated generator, and as long as the standard signature scheme is secure, the $\mathcal{RSATS}$-*1* transitive signature scheme is transitively unforgeable under adaptive chosen-message attack.

Theorem 3.2 Let $\mathsf{K}_{\mathrm{rsa}}$ be an RSA key generator and let $\mathcal{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ be a standard digital signature scheme. Let $\mathcal{RSATS}$-*1* be the transitive signature scheme associated to $\mathsf{K}_{\mathrm{rsa}}$ and

$\mathcal{SDS}$ as defined above. If the one-more RSA-inversion problem associated to $\mathsf{K}_{\mathrm{rsa}}$ is hard and $\mathcal{SDS}$ is unforgeable under adaptive chosen-message attack, then $\mathcal{RSATS}$-*1* is transitively unforgeable under adaptive chosen-message attack. ▌

Proof of Theorem 3.2. Suppose we are given a polynomial-time adversary F for $\mathcal{RSATS}$-*1*. We construct a om-rsa adversary A, and a forger B attacking $\mathcal{SDS}$, both polynomial-time, such that for all k

$$\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{RSATS}\text{-}\mathit{1},\mathsf{F}}(k) \le \mathbf{Adv}^{\text{om-rsa}}_{\mathsf{K}_{\mathrm{rsa}},\mathsf{A}}(k) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(k)\ . \tag{1}$$

The assumptions, namely that the one-more RSA-inversion problem associated to $\mathsf{K}_{\mathrm{rsa}}$ is hard and $\mathcal{SDS}$ is unforgeable under adaptive chosen-message attack, imply that the advantage functions on the right-hand-side of Equation (1) are negligible. The equation then says that $\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{RSATS}\text{-}\mathit{1},\mathsf{F}}(\cdot)$ is also negligible, which completes the proof. It remains to describe A and B.

The om-rsa adversary A, as per the definitions above, gets inputs N, e and, has access to an inversion oracle $\textsc{Inv}(\cdot)$ and a challenge oracle $\textsc{Chall}$. It wins if it outputs the inverses of all points returned by $\textsc{Chall}$, using strictly less queries to the inversion oracle than it makes to the challenge oracle. Let us now describe how it operates. It first generates a fresh key pair $(\mathit{spk}, \mathit{ssk})$ for $\mathcal{SDS}$ by running $\mathsf{SKG}(1^k)$. It then runs F on input $\mathit{tpk} = (N, e, \mathit{spk})$. The idea is that when answering F's signature queries, A uses target points generated by the challenge oracle as public labels. Running the TSign algorithm in order to create signatures is not possible because A doesn't know the corresponding secret labels; instead, it sparingly uses the inversion oracle to compute edge labels, calling the oracle only when the requested signature cannot be computed by composing previously signed edges. For this purpose, A maintains state information (V, L, Σ, Δ), where V, L and Σ are defined as in the TSign algorithm, and $\Delta : V \times V \to \mathbb{Z}_N^*$ is a function storing known edge labels. Now, in detail, when asked for a signature on edge $\{i, j\}$, A proceeds as follows:

```
(5.1)  If $i > j$ then $\mathit{swap}(i,j)$
(5.2)  If $i \notin V$ then
(5.3)      $V \leftarrow V \cup \{i\}$ ; $L(i) \leftarrow \textsc{Chall}()$
(5.4)      $\Sigma(i) \leftarrow \mathsf{SSign}(\mathit{ssk}, i \| L(i))$
(5.5)      $\Delta(i,i) \leftarrow 1$
(5.6)  If $j \notin V$ then
(5.7)      $V \leftarrow V \cup \{j\}$ ; $L(j) \leftarrow \textsc{Chall}()$
(5.8)      $\Sigma(j) \leftarrow \mathsf{SSign}(\mathit{ssk}, j \| L(j))$
(5.9)      $\Delta(j,j) \leftarrow 1$
(5.10) If $\Delta(i,j)$ is not defined then
(5.11)     $\Delta(i,j) \leftarrow \textsc{Inv}(L(i) \cdot L(j)^{-1} \bmod N)$
(5.12)     $\Delta(j,i) \leftarrow \Delta(i,j)^{-1} \bmod N$
(5.13)     For all $v \in V \setminus \{i,j\}$ do
(5.14)         If $\Delta(v,i)$ is defined then
(5.15)             $\Delta(v,j) \leftarrow \Delta(v,i) \cdot \Delta(i,j) \bmod N$
(5.16)             $\Delta(j,v) \leftarrow \Delta(v,j)^{-1} \bmod N$
(5.17)         If $\Delta(v,j)$ is defined then
(5.18)             $\Delta(v,i) \leftarrow \Delta(v,j) \cdot \Delta(j,i) \bmod N$
(5.19)             $\Delta(i,v) \leftarrow \Delta(v,i)^{-1} \bmod N$
(5.20) $\delta \leftarrow \Delta(i,j)$
(5.21) Return $((i \| L(i), \Sigma(i)), (j \| L(j), \Sigma(j)), \delta)$ to $\mathsf{F}$.
```

At the end of its execution, F outputs a forgery $\sigma' = ((i', L_{i'}, \Sigma_{i'}), (j', L_{j'}, \Sigma_{j'}), \delta')$ for edge $\{i', j'\}$. (During this analysis, we assume without loss of generality that $i' < j'$. If this is not the case, one can swap i' and j'.) Let $G = (V, E)$ be the graph defined by F's signature queries, and let $\widetilde{G} = (V, \widetilde{E})$ be its transitive closure. If σ' is not a valid forgery, meaning that $\mathsf{TVf}(\mathit{tpk}, i', j', \sigma') = 0$ or $\{i', j'\} \in \widetilde{E}$, then A aborts. Let $\mathbf{E}$ be the event that F's forgery contains recycled node certificates, i.e. $L_{i'} = L(i')$ and $L_{j'} = L(j')$. In case $\overline{\mathbf{E}}$ happens, A aborts. Else it computes solutions for all challenges that it received from the challenge oracle, as follows. The transitively closed graph $\widetilde{G}$ is partitioned into c disjoint components $V_k \subset V$ for $k = 1 \ldots c$. Let $V_{k'}$ be the component containing node i'. For all $k = 1 \ldots c$, $k \neq k'$, algorithm A chooses a *reference node* $r_k \in V_k$ and computes the secret labels of all nodes in V_k as

```
(5.22)  $\ell(r_k) \leftarrow \textsc{Inv}(L(r_k))$
(5.23)  For all $v \in V_k \setminus \{r_k\}$ do
(5.24)      $\ell(v) \leftarrow \Delta(v, r_k) \cdot \ell(r_k) \bmod N$
```

while the secret labels of all nodes in component $V_{k'}$ are computed as

```
(5.25)  $\ell(i') \leftarrow \delta' \cdot \ell(j') \bmod N$
(5.26)  For all $v \in V_{k'} \setminus \{i'\}$ do
(5.27)      $\ell(v) \leftarrow \Delta(v, i') \cdot \ell(i') \bmod N$ .
```

From the way A answers F's signature queries, one can see that $\Delta(i, j)$ is defined for all nodes i, j that belong to the same component, and hence that the values of Δ needed in the computations above are also defined. Algorithm A can now output the solutions for all its target points: for each $i \in V$, the public label $L(i)$ was obtained as a result of a query to $\textsc{Chall}()$, so the algorithm outputs $\ell(i)$ for all $i \in V$.

Now we need to check that A actually won the game. To do this we have to count the number of inversion queries. For each component V_k, $k \neq k'$, algorithm A needed $|V_k| - 1$ inversion queries to answer F's signature queries (the number of edges in a minimal spanning tree of V_k) plus one additional query at the end of the game to compute the secret label of r_k, summing up to $|V_k|$ inversion queries for each component. The component $V_{k'}$ only needed $|V_{k'}| - 1$ queries, because it did not need the additional query. So in summary, A inverted $|V|$ target points using $\sum_{k \neq k'} |V_k| + (|V_{k'}| - 1) = |V| - 1$ inversion queries, and hence wins the game.

The description of the forger B is rather straightforward: when run on input spk, it generates RSA parameters N, e, d using $\mathsf{K}_{\mathrm{rsa}}$. It then runs F on input $\mathit{tpk} = (N, e, \mathit{spk})$, answering F's signature queries using the real TSign algorithm but consulting its own $\mathsf{SSign}(\mathit{ssk}, \cdot)$ oracle to create node certificates. In the event $\mathbf{E}$ that F's forgery recycles old node certificates, B gives up, but otherwise (in the event $\overline{\mathbf{E}}$) at least one of the node certificates contains a signature on a new message, and this can be used to output a forgery.

It is clear that A's simulation of F's environment is perfect. Accordingly we have

$$\begin{aligned}
\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{RSATS}\text{-}\mathit{1},\mathsf{F}}(k) &= \Pr\left[\mathbf{Exp}^{\text{tu-cma}}_{\mathcal{RSATS}\text{-}\mathit{1},\mathsf{F}}(k) = 1\right] \\
&= \Pr\left[\mathbf{Exp}^{\text{tu-cma}}_{\mathcal{RSATS}\text{-}\mathit{1},\mathsf{F}}(k) = 1 \wedge \mathbf{E}\right] + \Pr\left[\mathbf{Exp}^{\text{tu-cma}}_{\mathcal{RSATS}\text{-}\mathit{1},\mathsf{F}}(k) = 1 \wedge \overline{\mathbf{E}}\right] \\
&\leq \mathbf{Adv}^{\text{om-rsa}}_{\mathsf{K}_{\mathrm{rsa}},\mathsf{A}}(k) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(k) \; .
\end{aligned}$$

This yields Equation (1), as required.

4 New Schemes

We describe three new transitive signature schemes, all proven transitively unforgeable under adaptive chosen-message attack.

4.1 The *FactTS-1* scheme

Our factoring-based transitive signature (*FactTS-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 *DLTS* and *RSATS-1*.

FACTORING PROBLEM. A *modulus generator* K_{fact} is a randomized, polynomial-time algorithm that on input 1^k returns a triple (N, p, q) where p, q are distinct, odd primes, $N = pq$ and $2^{k-1} \leq N < 2^k$. For any factoring adversary A and $k \in \mathbb{N}$ we let

$$\mathbf{Adv}^{\text{fact}}_{\mathsf{K}_{\text{fact}},\mathsf{A}}(k) \;=\; \Pr\left[\, r \in \{p, q\} \;:\; (N, p, q) \xleftarrow{\$} \mathsf{K}_{\text{fact}}(1^k)\,;\; r \xleftarrow{\$} \mathsf{A}(N) \,\right] .$$

We say that *factoring is hard relative to* K_{fact} if the function $\mathbf{Adv}^{\text{fact}}_{\mathsf{K}_{\text{fact}},\mathsf{A}}(\cdot)$ is negligible for every polynomial-time algorithm A.

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.

THE SCHEME. We associate to any modulus generator K_{fact} and any standard digital signature scheme $\mathcal{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ a transitive signature scheme *FactTS-1* $= (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ defined as follows:

- $\mathsf{TKG}(1^k)$ proceeds as in *RSATS-1* with the following changes:

  (1.2) Run $\mathsf{K}_{\text{fact}}(1^k)$ to get a triple (N, p, q)

  (1.3) Output $tpk = (N, spk)$ as the public key and $tsk = (N, ssk)$ as the secret key.

  Note that the primes p, q are discarded and in particular are not part of the secret key.
- The signing algorithm TSign maintains state as in *RSATS-1*. On inputs tsk, i, j it proceeds as the TSign algorithm of *RSATS-1* except that rather than computing $L(\cdot)$ as $\ell(\cdot)^e$ it computes $L(\cdot)$ as $\ell(\cdot)^2$ in lines (2.4) and (2.8).
- TVf, on input $tpk = (N, spk)$, nodes i, j and a candidate signature σ, proceeds exactly as the TVf algorithm of the *RSATS-1* scheme, except that δ^e is replaced with δ^2 in line (3.4).
- The composition algorithm is identical to that of *RSATS-1*.

A proof by induction similar to the one in Appendix A can be used to show the *FactTS-1* transitive signature scheme described above satisfies the correctness requirement of Definition 2.1.

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 $\mathbb{Z}_N^*$. See Figure 1 for the cost summary.

SECURITY. The following is the formal statement of our result about the security of *FactTS-1*.

Theorem 4.1 Let K_{fact} be a modulus generator and let $\mathcal{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ be a standard digital signature scheme. Let *FactTS-1* be the transitive signature scheme associated to them as defined above. If the factoring problem associated to K_{fact} is hard and $\mathcal{SDS}$ is unforgeable under adaptive chosen-message attack, then *FactTS-1* is transitively unforgeable under adaptive chosen-message attack. ▌

Although *RSATS-1* and *FactTS-1* are very similar, the security of the latter is based on a weaker assumption. Intuitively, the reason is that RSA induces a permutation on $\mathbb{Z}_N^*$, whereas squaring maps 4 different elements of $\mathbb{Z}_N^*$ to the same square and two different roots of the same square reveal the factorization of the modulus.

Proof of Theorem 4.1. Suppose we are given a polynomial-time adversary F for *FactTS-1*. We construct a factoring adversary A, and a forger B attacking $\mathcal{SDS}$, both polynomial-time, such that for all k

$$\mathbf{Adv}^{\text{tu-cma}}_{\mathit{FactTS\text{-}1},\mathsf{F}}(k) \leq 2 \cdot \mathbf{Adv}^{\text{fact}}_{\mathsf{K}_{\text{fact}},\mathsf{A}}(k) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(k) \ . \tag{2}$$

The assumptions made in the theorem conclude the proof. It remains to describe A and B.

The factoring algorithm A gets as input a modulus N generated by K_{fact} and begins by picking keys for the standard signature scheme via $(\mathit{spk}, \mathit{ssk}) \xleftarrow{\$} \mathsf{SKG}(1^k)$. It then lets $\mathit{tpk} = (N, \mathit{spk})$ be a public key for the transitive signature scheme and starts running F on input tpk. To reply to F's sign-oracle queries, algorithm A simply runs the TSign procedure of the transitive signature scheme, which it can because it possesses the secret key $\mathit{tsk} = (N, \mathit{ssk})$ corresponding to tpk. (We use here the fact that signing does not require knowledge of the prime factors of N.) A maintains the state information (V, ℓ, L, Σ) of the TSign procedure. Once F is done querying its oracle, it will output its forgery $\sigma' = ((i', L_{i'}, \Sigma_{i'}), (j', L_{j'}, \Sigma_{j'}), \delta')$ for edge $\{i', j'\}$. (We again assume without loss of generality that $i' < j'$.) Let $G = (V, E)$ be the graph defined by F's signature queries, and let $\widetilde{G} = (V, \widetilde{E})$ denote its transitive closure. Let $\mathbf{E}_1$ denote the event that σ' is a certificate-recycling forgery (i.e. $L_{i'} = L(i')$ and $L_{j'} = L(j')$), and let $\mathbf{E}_2$ be the event that $\delta' \equiv \pm\delta$ where $\delta \equiv \ell(i')\ell(j')^{-1} \bmod N$. If σ' is not a valid forgery, or in the event that $\overline{\mathbf{E}_1} \vee \mathbf{E}_2$, algorithm A gives up. Otherwise, it computes and returns $r = \gcd(\delta + \delta', N)$, which is a factor of N because δ and δ' are different square roots of $L(i')L(j')^{-1} \bmod N$. This completes the description of factoring algorithm A.

With regard to the analysis, it is tempting to say that since $\ell(i')$ and $\ell(j')$ were chosen at random, with probability 1/2 A now has two square roots δ and δ' such that $\delta \not\equiv \pm\delta' \bmod N$, enabling it to factor N. This argument would be correct if the forger were only given $L(i')$ and $L(j')$, without having any further information on exactly which root A knows. However, by signing edges involving nodes i' or j', algorithm A might have given away some additional information about its 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 A'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. The crucial fact, which we will justify later, is that

$$\Pr\left[\overline{\mathbf{E}_2} \mid \mathbf{E}_1 \wedge \mathbf{Exp}^{\text{tu-cma}}_{\mathit{FactTS\text{-}1},\mathsf{F}}(k) = 1\right] \;=\; \frac{1}{2} \ . \tag{3}$$

Given this, we have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{fact}}_{\mathsf{K}_{\mathrm{fact}},\mathsf{A}}(k) &\geq \Pr\left[\mathbf{Exp}^{\text{tu-cma}}_{\mathcal{F}\mathit{act}\mathcal{IS}\text{-}\mathit{1},\mathsf{F}}(k) = 1 \wedge \mathbf{E}_1 \wedge \overline{\mathbf{E}_2}\right] \\
&= \Pr\left[\overline{\mathbf{E}_2} \mid \mathbf{E}_1 \wedge \mathbf{Exp}^{\text{tu-cma}}_{\mathcal{F}\mathit{act}\mathcal{IS}\text{-}\mathit{1},\mathsf{F}}(k) = 1\right] \cdot \Pr\left[\mathbf{E}_1 \wedge \mathbf{Exp}^{\text{tu-cma}}_{\mathcal{F}\mathit{act}\mathcal{IS}\text{-}\mathit{1},\mathsf{F}}(k) = 1\right] \\
&= \frac{1}{2} \cdot \Pr\left[\mathbf{E}_1 \wedge \mathbf{Exp}^{\text{tu-cma}}_{\mathcal{F}\mathit{act}\mathcal{IS}\text{-}\mathit{1},\mathsf{F}}(k) = 1\right] .
\end{aligned}$$

The description of algorithm B breaking $\mathcal{SDS}$ is very similar to the description of algorithm B in the proof of Theorem 3.2. Details are omitted. As in that proof we will have

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(k) \geq \Pr\left[\overline{\mathbf{E}}_1 \wedge \mathbf{Exp}^{\text{tu-cma}}_{\mathcal{F}\mathit{act}\mathcal{IS}\text{-}\mathit{1},\mathsf{F}}(k) = 1\right] .$$

Putting the above together we have

$$\begin{aligned}
\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{F}\mathit{act}\mathcal{IS}\text{-}\mathit{1},\mathsf{F}}(k) &= \Pr\left[\mathbf{E}_1 \wedge \mathbf{Exp}^{\text{tu-cma}}_{\mathcal{F}\mathit{act}\mathcal{IS}\text{-}\mathit{1},\mathsf{F}}(k) = 1\right] + \Pr\left[\overline{\mathbf{E}}_1 \wedge \mathbf{Exp}^{\text{tu-cma}}_{\mathcal{F}\mathit{act}\mathcal{IS}\text{-}\mathit{1},\mathsf{F}}(k) = 1\right] \\
&\leq 2 \cdot \mathbf{Adv}^{\mathrm{fact}}_{\mathsf{K}_{\mathrm{fact}},\mathsf{A}}(k) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(k)
\end{aligned}$$

as desired. It remains to justify Equation (3).

Let $G = (V, E)$ be the graph defined by the forger's signature queries, and let $\widetilde{G} = (V, \widetilde{E})$ be the transitive closure of G. We represent A's secret information by a random variable $\boldsymbol{\ell}$ distributed uniformly over *Secrets*, the set of all functions from V to $\mathbb{Z}^*_N$. The forger's view consists of a function L assigning a square modulo 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 ignore 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 $\ell \in \mathit{Secrets}$ (and vice versa that ℓ is consistent with $\langle L, \Delta\rangle$) if and only if

$$L(i) \equiv \ell(i)^2 \bmod N \quad \text{for all } i \in V \tag{4}$$

$$\Delta(i,j) \equiv \ell(i)\ell(j)^{-1} \bmod N \quad \text{for all } \{i,j\} \in \widetilde{E}\ ,\ i < j \tag{5}$$

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 $\ell \in \mathit{Secrets}$. The actual view of the forger is a random variable $\boldsymbol{\mathit{View}}$ distributed over *Views* as induced by $\boldsymbol{\ell}$. The following lemma states that for every $\langle L, \Delta\rangle \in \mathit{Views}$ 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 $\boldsymbol{\mathit{View}} = \langle L, \Delta\rangle$, and hence that no forger, on input only $\boldsymbol{\mathit{View}}$, can predict $\boldsymbol{\delta}$ with higher probability of success than random guessing.

Lemma 4.2 For any $\langle L, \Delta\rangle \in \mathit{Views}$, for any $\{i', j'\} \notin \widetilde{E}$ and for any $\delta \in \mathbb{Z}^*_N$ with $\delta^2 \equiv L(i')L(j')^{-1} \bmod N$:

$$\Pr\left[\boldsymbol{\delta} \equiv \delta \bmod N \mid \boldsymbol{\mathit{View}} = \langle L, \Delta\rangle\right] = \frac{1}{4} . \blacksquare \tag{6}$$

Equation (3) follows easily from this. To complete the proof of Theorem 4.1 we prove the lemma.

Proof of Lemma 4.2: Since the outcome of all random variables is uniquely determined by the signer's choice for $\boldsymbol{\ell}$, we can reduce all probabilities on random variables to the probability of making some particular choice for $\boldsymbol{\ell}$. For example, if we define $\mathit{Cons}(\langle L, \Delta\rangle) \subseteq \mathit{Secrets}$ to be the set of all $\ell \in \mathit{Secrets}$ consistent with $\langle L, \Delta\rangle$, then we can replace $\Pr[\boldsymbol{View} = \langle L, \Delta\rangle]$ with $\Pr[\boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)]$. Using this fact and some basic probability theory, we can write

$$\begin{aligned}
&\Pr[\boldsymbol{\delta} = \delta \mid \boldsymbol{View} = \langle L, \Delta\rangle] \\
&= \Pr[\boldsymbol{\delta} = \delta \ \wedge\ \boldsymbol{View} = \langle L, \Delta\rangle \mid \boldsymbol{View} = \langle L, \Delta\rangle] \\
&= \Pr[\boldsymbol{\delta} = \delta \ \wedge\ \boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle) \mid \boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)] \\
&= \Pr[\boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle) \mid \boldsymbol{\delta} = \delta \ \wedge\ \boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)] \cdot \frac{\Pr[\boldsymbol{\delta} = \delta \ \wedge\ \boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)]}{\Pr[\boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)]} \\
&= \frac{\Pr[\boldsymbol{\delta} = \delta \ \wedge\ \boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)]}{\Pr[\boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)]}\ . \qquad (7)
\end{aligned}$$

We want to find a numerical expression for the last two factors in Equation (7). Because $\boldsymbol{\ell}$ is uniformly distributed over *Secrets*, the probability that $\boldsymbol{\ell} \in S \subseteq \mathit{Secrets}$ is simply the number of elements in S divided by $|\mathit{Secrets}| = \varphi(N)^{|V|}$.

We first try to find an expression for the number of elements in $\mathit{Cons}(\langle L, \Delta\rangle)$. For ℓ to be consistent with forger view $\langle L, \Delta\rangle$, it has to satisfy the system of equations given by (4) and (5). Considering only equations (4), there are four possibilities for $\ell(i)$ left for every $i \in V$, namely the four square roots of $L(i)$. Equations (5) impose additional restrictions on ℓ. Many of these are linearly dependent, though. In order to count the actual number of possible solutions, we'd like to replace (5) with an equivalent but linearly independent set of equations.

Let c be the number of disjoint components $V_k \subset V$ in the transitively closed graph $\widetilde{G} = (V, \widetilde{E})$. If we define one node r_k in each component V_k to be the *reference node* for that component, and denote the reference node in the component of node i as $R(i)$, then the equations in the system given by

$$\Delta(i, R(i)) \equiv \ell(i)\ell(R(i))^{-1} \bmod N \qquad (8)$$

for all $i \in V \setminus \{r_k \mid k = 1 \ldots c\}$ are clearly independent. At the same time, they also form a system equivalent to (5), because every equation in (5) is either contained in (8), or can be written as the quotient of two equations in (8). The equations in (8) imply that once ℓ is fixed for the c reference nodes, ℓ is completely defined. Together with Equation (4), that leaves c entries of ℓ to be chosen freely from four values, so

$$\Pr[\boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)] = \frac{4^c}{\varphi(N)^{|V|}} \qquad (9)$$

To what amount does the addition of the requirement $\boldsymbol{\delta} = \delta$ restrict our choices for $\boldsymbol{\ell}$? This comes down to adding

$$\ell(i')\ell(j')^{-1} \equiv \delta \bmod N$$

to the systems given by (4) and (8), or equivalently, adding the equation

$$\ell(R(i')) \equiv \delta \cdot \Delta(i', R(i'))^{-1} \cdot \Delta(j', R(j')) \cdot \ell(R(j')) \bmod N$$

which directly links $\ell(R(i'))$ to the choice for $\ell(R(j'))$. So now there are only $c - 1$ entries of ℓ left to choose, giving

$$\Pr[\boldsymbol{\delta} = \delta \ \wedge\ \boldsymbol{\ell} \in \mathit{Cons}(\langle L, \Delta\rangle)] = \frac{4^{c-1}}{\varphi(N)^{|V|}} \qquad (10)$$

Substituting the factors in Equation (7) with Equation (9) and Equation (10) yields Equation (6), thereby proving the lemma. ▌

4.2 The $\mathcal{DLTS}$-$1\mathcal{M}$ scheme

Micali and Rivest's $\mathcal{DLTS}$ scheme [MR02] uses two generators, which is important to their security proof. The underlying ideas trace back to Okamoto's two-generator-based identification scheme and its proof of security (against impersonation under active attack). However, Schnorr's identification scheme [Sch90], which uses only a single generator, is simpler, more natural and has lower cost, particularly in size of secret keys. We ask whether there is an analogous single-generator-based transitive signature scheme. We answer this in the affirmative, presenting $\mathcal{DLTS}$-$1\mathcal{M}$, which is a simpler and somewhat more natural single-generator based version of $\mathcal{DLTS}$, offering some performance improvements. However, while the proof of security of $\mathcal{DLTS}$ relied only on the standard hardness of discrete logarithms assumption, the proof of security of $\mathcal{DLTS}$-$1\mathcal{M}$ relies on the hardness of the one-more discrete logarithm problem [BNPS03]. This is again analogous to the situation for identification schemes. Okamoto proved his scheme secure under the standard hardness of discrete logarithm assumption, while the proof of security of Schnorr's scheme (which remained an open problem for a while) is based on the hardness of the one-more discrete logarithm problem [BP02]. The $\mathcal{DLTS}$-$1\mathcal{M}$ scheme is similar to $\mathcal{RSATS}$-1.

DL GENERATORS AND THE ASSUMPTION. A *cyclic group generator* K_{cg} is a randomized polynomial-time algorithm that on input 1^k outputs a tuple $(\langle\mathbb{G}\rangle, g, q)$, where q is an odd prime with $2^{k-1} \leq q < 2^k$, $\langle\mathbb{G}\rangle$ is the description of a cyclic group $\mathbb{G}$ of order q, and g is a generator of $\mathbb{G}$.

We recall the one-more discrete logarithm problem [BNPS03]. A om-dlog adversary is a randomized, polynomial-time algorithm A that gets input $\langle\mathbb{G}\rangle, g, q$ and has access to two oracles. The first is a discrete logarithm oracle $\mathrm{DLog}(\cdot)$ that given $Y \in \mathbb{G}$ returns $y \in \mathbb{Z}_q$ such that $Y = g^y$. The second is a challenge oracle that, each time it is invoked (it takes no inputs), returns a random challenge point from $\mathbb{G}$. We say that A wins the game if it succeeds in computing the discrete logarithm of all n points output by the challenge oracle, using strictly less than n queries to the DLog oracle. More formally, we define the following experiment:

```
Experiment $\mathbf{Exp}^{\text{om-dlog}}_{\mathsf{K}_{\mathrm{cg}},\mathsf{A}}(k)$:
    $(\langle\mathbb{G}\rangle, g, q) \xleftarrow{\$} \mathsf{K}_{\mathrm{cg}}(1^k)$ ; $n \leftarrow 0$ ; $m \leftarrow 0$
    $(y_1, \ldots, y_{n'}) \xleftarrow{\$} \mathsf{A}^{\mathrm{DLog},\mathrm{Chall}}(\langle\mathbb{G}\rangle, g, q)$
    If $n' = n$ and $m < n$ and $\forall\, i = 1 \ldots n \;:\; g^{y_i} \equiv Y_i$
    Then return 1 else return 0

Oracle $\mathrm{Inv}(Y)$:
    $m \leftarrow m + 1$ ; Return $y \in \mathbb{Z}_q$ s.t. $g^y \equiv Y$
Oracle $\mathrm{Chall}()$ :
    $n \leftarrow n + 1$ ; $Y_n \xleftarrow{\$} \mathbb{G}$
    Return $Y_n$ .
```

The om-dlog *advantage* of A, denoted $\mathbf{Adv}^{\text{om-dlog}}_{\mathsf{K}_{\mathrm{cg}},\mathsf{A}}(k)$, is the probability that the above experiment returns 1, taken over the coins of K_{cg}, the coins of A, and the coins used by the challenge oracle across its invocations. We say that the one-more discrete-logarithm problem associated to K_{cg} is hard if the function $\mathbf{Adv}^{\text{om-dlog}}_{\mathsf{K}_{\mathrm{cg}},\mathsf{A}}(\cdot)$ is negligible for all polynomial-time adversaries A.

Note that the standard one-wayness assumption on K_{cg} can be formulated as requiring that the function $\mathbf{Adv}^{\text{om-dlog}}_{\mathsf{K}_{\mathrm{cg}},\mathsf{A}}(\cdot)$ is negligible for all polynomial-time adversaries A that make one challenge query and no discrete-logarithm queries. Thus, hardness of the one-more discrete-logarithm problem is an extension of one-wayness.

We do not attempt to pin down exactly how the generator operates. In particular there are many classes of groups (of prime order) which could be used. One example is that $2q+1$ is a prime and $\mathbb{G}$ is the subgroup of quadratic residues of $\mathbb{Z}^*_{2q+1}$. Another possibility is elliptic curve groups. This makes our results more general.

THE SCHEME. We associate to any cyclic group generator K_{cg} and any standard digital signature scheme $\mathcal{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ a transitive signature scheme $\mathcal{DLTS}$-$1\mathcal{M} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ defined as follows:

- $\mathsf{TKG}(1^k)$ proceeds as in $\mathcal{RSATS}$-*1* with the following changes:

  (1.2) Run $\mathsf{K}_{\rm cg}(1^k)$ to get a triple $(\langle\mathbb{G}\rangle, g, q)$
  (1.3) Output $tpk = (\langle\mathbb{G}\rangle, g, q, spk)$ as the public key and $tsk = (\langle\mathbb{G}\rangle, g, q, ssk)$ as the secret key.

- The signing algorithm TSign maintains state (V, ℓ, L, Σ) where V, Σ are as in $\mathcal{RSATS}$-*1*, ℓ: $V \rightarrow Z_q$ and L: $V \rightarrow \mathbb{G}$. When invoked on inputs tsk, i, j, it proceeds as the TSign algorithm of $\mathcal{RSATS}$-*1* except for the following changes:

  (2.4) $\quad \ell(i) \stackrel{\$}{\leftarrow} \mathbb{Z}_q$; $L(i) \leftarrow g^{\ell(i)}$
  (2.8) $\quad \ell(j) \stackrel{\$}{\leftarrow} \mathbb{Z}_q$; $L(j) \leftarrow g^{\ell(j)}$
  (2.10) $\delta \leftarrow \ell(i) - \ell(j) \bmod q$

- TVf, on input $tpk = (\langle\mathbb{G}\rangle, g, q, spk)$, nodes i, j and a candidate signature σ, proceeds as the TVf algorithm of the $\mathcal{RSATS}$-*1* scheme, except for the following change:

  (3.4) If $g^\delta \equiv L_i L_j^{-1}$ then return 1 else return 0.

- The composition algorithm Comp takes tpk, nodes i, j, k and signatures σ_1 and σ_2, and proceeds as the Comp algorithm of $\mathcal{RSATS}$-*1* except for the following changes. In line (4.3) the computation of δ_1^{-1} is in $\mathbb{G}$ rather than being modulo N, and similarly for line (4.4). Also line (4.6) is replaced with

  (4.6) $\delta \leftarrow \delta_1 + \delta_2 \bmod q$

This scheme offers some perfomance benefits compared to $\mathcal{DLTS}$, as indicated in Figure 1, namely a reduced signature size and composition time.

As the above indicates, this scheme is very similar to $\mathcal{RSATS}$-*1*, replacing "$x^e \bmod N$" (with $x \in \mathbb{Z}_N^*$) by "g^x" (with $x \in \mathbb{Z}_q$). Accordingly a proof similar to the one in Appendix A can be used to show that $\mathcal{DLTS}$-*1*$\mathcal{M}$ satisfies the correctness requirement of Definition 2.1, and the following security result is established by a proof analogous to that of Theorem 3.2. Rather than repeating the entire proof here, we only give a proof sketch and highlight the differences with the proof of Theorem 3.2.

Theorem 4.3 Let $\mathsf{K}_{\rm cg}$ be a cyclic group generator and let $\mathcal{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ be a standard digital signature scheme. Let $\mathcal{DLTS}$-*1*$\mathcal{M}$ be the transitive signature scheme associated to them as defined above. If the one-more discrete logarithm problem associated to $\mathsf{K}_{\rm cg}$ is hard and $\mathcal{SDS}$ is unforgeable under adaptive chosen-message attack, then $\mathcal{DLTS}$-*1*$\mathcal{M}$ is transitively unforgeable under adaptive chosen-message attack. ▌

Proof: Given a polynomial-time tu-cma adversary F attacking $\mathcal{DLTS}$-*1*$\mathcal{M}$, we construct polynomial-time om-dlog and uf-cma adversaries A and B, respectively, such that

$$\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{DLTS}\text{-}1\mathcal{M},\mathsf{F}}(k) \leq \mathbf{Adv}^{\text{om-dlog}}_{\mathsf{K}_{\rm cg},\mathsf{A}}(k) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(k)\ . \tag{11}$$

Algorithm A, on inputs $\langle\mathbb{G}\rangle, g, q$, generates a new key pair $(spk, ssk) \stackrel{\$}{\leftarrow} \mathsf{SKG}(1^k)$ and runs algorithm F on input $tpk = (\langle\mathbb{G}\rangle, g, q, spk)$. Algorithm A then proceeds exactly as the om-rsa algorithm A in the proof of Theorem 3.2, except that all calls to the INV oracle are replaced with calls to the DLOG oracle, that A maintains a function $\Delta : V \times V \rightarrow \mathbb{Z}_q$ in its state information, that all multiplications mod N are replaced with multiplications in $\mathbb{G}$, and that the following changes are made to the code:

(5.5) $\quad \Delta(i, i) \leftarrow 0$
(5.9) $\quad \Delta(j, j) \leftarrow 0$
(5.11) $\quad \Delta(i, j) \leftarrow \text{DLOG}(L(i) \cdot L(j)^{-1} \bmod N)$

```
(5.12)     $\Delta(j,i) \leftarrow -\Delta(i,j) \bmod q$
(5.15)             $\Delta(v,j) \leftarrow \Delta(v,i) + \Delta(i,j) \bmod q$
(5.16)             $\Delta(j,v) \leftarrow -\Delta(v,j) \bmod q$
(5.18)             $\Delta(v,i) \leftarrow \Delta(v,j) + \Delta(j,i) \bmod q$
(5.19)             $\Delta(i,v) \leftarrow -\Delta(v,i) \bmod q$
(5.24)     $\ell(v) \leftarrow \Delta(v,r_k) + \ell(r_k) \bmod q$
(5.25)  $\ell(i') \leftarrow \delta' + \ell(j') \bmod q$
(5.27)     $\ell(v) \leftarrow \Delta(v,i') + \ell(i') \bmod q$ .
```

From the analysis in the proof of Theorem 3.2, it follows that A is always successful in the event $\mathbf{E}$ that F's forgery contains recycled node certificates.

Algorithm B gets a public key *spk* as input, generates $(\langle\mathbb{G}\rangle, g, q) \xleftarrow{\$} \mathsf{K}_{\mathrm{cg}}(1^k)$ and runs the forger F on input $(\langle\mathbb{G}\rangle, g, q, \mathit{spk})$. It simulates F's environment exactly as algorithm B in the proof of Theorem 3.2, and by the same reasoning it can be seen to succeed in the event $\overline{\mathbf{E}}$ that F's forgery contains a new node certificate. Combining the results for algorithms A and B yields Equation (11).

4.3 The $\mathcal{G}ap\mathcal{TS}$-*1* scheme

Gap Diffie-Hellman groups and the assumption. A *gap DH group specifier* is a pair $(\mathsf{K}_{\mathrm{cg}}, \mathsf{S}_{\mathrm{ddh}})$ where K_{cg} is a cyclic group generator and $\mathsf{S}_{\mathrm{ddh}}$ is a polynomial-time algorithm called the *decision Diffie-Hellman (DDH) solver*. To describe what it does, let us introduce some notation. Let $(\langle\hat{\mathbb{G}}\rangle, g, q)$ be an output of $\mathsf{K}_{\mathrm{cg}}(1^k)$. For $X \in \hat{\mathbb{G}}$ we let $\mathit{dlog}_{\hat{\mathbb{G}},g}(X)$ be the discrete logarithm of X to base g, namely the (unique) $x \in \mathbb{Z}_q$ such that $g^x = X$. For $X, Y \in \mathbb{G}$ we let

$$\mathit{CDH}_{\hat{\mathbb{G}},g}(X,Y) \;=\; g^{\mathit{dlog}_{\hat{\mathbb{G}},g}(X)\cdot \mathit{dlog}_{\hat{\mathbb{G}},g}(Y)} \; .$$

Now, $\mathsf{S}_{\mathrm{ddh}}$, given inputs $\langle\hat{\mathbb{G}}\rangle, g, q, X, Y, Z$, where $X, Y, Z \in \hat{\mathbb{G}}$, returns 1 if $Z = \mathit{CDH}_{\hat{\mathbb{G}},g}(X,Y)$ and 0 otherwise. Namely, it solves the DDH problem in $\hat{\mathbb{G}}$.

In analogy with the one-more RSA-inversion and one-more discrete logarithm problems introduced by [BNPS03], Boldyreva [Bol03] defined the one-more CDH problem (called the *chosen-target CDH problem* in her work). Let us recall it. A om-cdh adversary is a randomized, polynomial-time algorithm A that gets input $\langle\hat{\mathbb{G}}\rangle, g, q$ as well as an element X of $\hat{\mathbb{G}}$. A has access to two oracles. The first is the oracle $\textsc{Cdh}(\cdot)$ that given $Y \in \hat{\mathbb{G}}$ returns $\mathit{CDH}_{\hat{\mathbb{G}},g}(X,Y)$. The second is a challenge oracle $\textsc{Chall}()$ that, each time it is invoked (it takes no inputs), returns a random challenge point from $\hat{\mathbb{G}}$. We say that A wins the game if it outputs the CDH solution with respect to X of all n points output by the challenge oracle, using strictly less than n calls to its $\textsc{Cdh}$ oracle. More formally, A is run in the following experiment:

```
Experiment $\mathbf{Exp}^{\text{om-cdh}}_{\mathsf{K}_{\mathrm{cg}},\mathsf{A}}(k)$:
  $(\langle\hat{\mathbb{G}}\rangle, g, q) \xleftarrow{\$} \mathsf{K}_{\mathrm{cg}}(1^k)$ ; $X \xleftarrow{\$} \hat{\mathbb{G}}$ ; $n \leftarrow 0$ ; $m \leftarrow 0$
  $(Z_1, \ldots, Z_{n'}) \xleftarrow{\$} \mathsf{A}^{\textsc{Cdh},\textsc{Chall}}(\langle\hat{\mathbb{G}}\rangle, g, q, X)$
  If $n' = n$ and $m < n$
  and $\forall\, i = 1 \ldots n \;:\; Z_i \equiv \mathit{CDH}_{\hat{\mathbb{G}},g}(X, Y_i)$
  Then return 1 else return 0

Oracle $\textsc{Cdh}(Y)$:
  $m \leftarrow m+1$ ; Return $\mathit{CDH}_{\hat{\mathbb{G}},g}(X,Y)$
Oracle $\textsc{Chall}()$ :
  $n \leftarrow n+1$ ; $Y_n \xleftarrow{\$} \hat{\mathbb{G}}$ ; Return $Y_n$ .
```

The om-cdh *advantage* of A, denoted $\mathbf{Adv}^{\text{om-cdh}}_{\mathsf{K}_{\mathrm{cg}},\mathsf{A}}(k)$, is the probability that the above experiment returns 1, taken over the coins of K_{cg}, the random choice of X, the coins of A, and the coins used by the challenge oracle across its invocations. We say that the one-more CDH problem associated to K_{cg} is hard if the function $\mathbf{Adv}^{\text{om-cdh}}_{\mathsf{K}_{\mathrm{cg}},\mathsf{A}}(\cdot)$ is negligible for all polynomial-time adversaries A.

Gap Diffie-Hellman groups were originally used by Boneh, Lynn and Shacham [BLS01] to construct a signature scheme with very short signature length (≈ 160 bits), and later by Boldyreva [Bol03] to construct threshold, blind and multi-signatures. The only currently known examples of gap Diffie-Hellman groups are based on bilinear maps on elliptic curves. If $\hat{e} : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is a non-degenerate bilinear map, and $\psi : \hat{\mathbb{G}} \to \mathbb{G}$ is a group isomorphism from $\hat{\mathbb{G}}$ to $\mathbb{G}$, then the DDH instance (g, X, Y, Z) in $\hat{\mathbb{G}}$ can be solved by testing whether $\hat{e}(\psi(X), Y) = \hat{e}(\psi(Z), g)$. We refer to [BLS01] for more details. Since we do not need pairings per se, but merely the more general concept of gap Diffie-Hellman groups, we continue to use the latter notation throughout the rest of this paper.

The scheme. We associate to any gap DH group specifier $(\mathsf{K}_{\mathrm{cg}}, \mathsf{S}_{\mathrm{ddh}})$ and any standard digital signature scheme $\mathcal{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ a transitive signature scheme $\mathcal{GapTS}$-$\mathit{1} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ defined as follows:

- $\mathsf{TKG}(1^k)$ proceeds as in $\mathcal{RSATS}$-$\mathit{1}$ with the following changes:

  (1.2) Run $\mathsf{K}_{\mathrm{cg}}(1^k)$ to get a triple $(\langle\hat{\mathbb{G}}\rangle, g, q)$. Then let $x \xleftarrow{\$} \mathbb{Z}_q$ and $X \leftarrow g^x$.

  (1.3) Output $tpk = (\langle\hat{\mathbb{G}}\rangle, g, q, X, spk)$ as the public key and $tsk = (\langle\hat{\mathbb{G}}\rangle, g, q, X, ssk)$ as the secret key.

  Note that x is discarded.

- The signing algorithm TSign maintains state (V, ℓ, L, Σ) where V, Σ are as in $\mathcal{RSATS}$-$\mathit{1}$ and ℓ, L: $V \to \mathbb{G}$. When invoked on inputs tsk, i, j, it proceeds as the TSign algorithm of $\mathcal{RSATS}$-$\mathit{1}$ except for the following changes:

  (2.4) $\quad y_i \xleftarrow{\$} \mathbb{Z}_q$; $\ell(i) \leftarrow X^{y_i}$; $L(i) \leftarrow g^{y_i}$

  (2.8) $\quad y_j \xleftarrow{\$} \mathbb{Z}_q$; $\ell(j) \leftarrow X^{y_j}$; $L(j) \leftarrow g^{y_j}$

  (2.10) $\delta \leftarrow \ell(i)\ell(j)^{-1}$

- TVf, on input $tpk = (\langle\hat{\mathbb{G}}\rangle, g, q, U, spk)$, nodes i, j and a candidate signature σ, proceeds as the TVf algorithm of the $\mathcal{RSATS}$-$\mathit{1}$ scheme, except for the following change:

  (3.4) If $\mathsf{S}_{\mathrm{ddh}}(\langle\hat{\mathbb{G}}\rangle, g, q, X, L_i L_j^{-1}, \delta) = 1$ then return 1 else return 0.

  That is, it checks that $\delta = CDH_{\hat{\mathbb{G}},g}(X, L_i L_j^{-1})$.

- The composition algorithm Comp takes tpk, nodes i, j, k and signatures σ_1 and σ_2, and proceeds as the Comp algorithm of $\mathcal{RSATS}$-$\mathit{1}$ except for the following changes. In line (4.3) the computation of δ_1^{-1} is in $\hat{\mathbb{G}}$ rather than being modulo N, and similarly for line (4.4). Also line (4.5) is replaced with

  (4.5) $\delta \leftarrow \delta_1 \delta_2$

As usual, this scheme can be shown to meet Definition 2.1. The security result is the following.

Theorem 4.4 Let $(\mathsf{K}_{\mathrm{cg}}, \mathsf{S}_{\mathrm{ddh}})$ be a gap DH group specifier and let $\mathcal{SDS} = (\mathsf{SKG}, \mathsf{SSign}, \mathsf{SVf})$ be a standard digital signature scheme. Let $\mathcal{GapTS}$-$\mathit{1}$ be the transitive signature scheme associated to them as defined above. If the one-more CDH problem associated to K_{cg} is hard and $\mathcal{SDS}$ is unforgeable under adaptive chosen-message attack, then $\mathcal{GapTS}$-$\mathit{1}$ is transitively unforgeable under adaptive chosen-message attack. ▌

Proof: Given a polynomial-time tu-cma adversary F attacking $\mathcal{GapTS}$-$\mathit{1}$, we construct a polynomial-time om-cdh adversary A and a uf-cma adversary B such that

$$\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{GapTS}\text{-}1,\mathsf{F}}(k) \leq \mathbf{Adv}^{\text{om-cdh}}_{\mathsf{K}_{\mathrm{cg}},\mathsf{A}}(k) + \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{SDS},\mathsf{B}}(k)\ . \tag{12}$$

Algorithm A gets input $\langle\hat{\mathbb{G}}\rangle, g, q, X$ and has access to a CDH oracle $\textsc{Cdh}_{\hat{\mathbb{G}},g}(X,\cdot)$ and a challenge oracle $\textsc{Chall}$. Its task is to compute $\mathit{CDH}_{\hat{\mathbb{G}},g}(X,Y_i)$ for all target points Y_i returned by the challenge oracle using a number of CDH queries that is strictly smaller than the number of target points solved. To this end, algorithm A runs the forger F on input $\mathit{tpk} = (\langle\hat{\mathbb{G}}\rangle, g, q, X, \mathit{spk})$, where $(\mathit{spk}, \mathit{ssk})$ is generated via $\mathsf{SKG}(1^k)$. It then proceeds exactly as the om-rsa adversary A in the proof of Theorem 3.2, except that whenever the om-rsa adversary calls $\textsc{Inv}(\cdot)$, the om-cdh adversary calls its $\textsc{Cdh}_{\hat{\mathbb{G}},g}(X,\cdot)$ oracle, that operations modulo N become operations in $\hat{\mathbb{G}}$, and that the function Δ maps to $\hat{\mathbb{G}}$ instead of $\mathbb{Z}_N^*$. From the same analysis as shown in the proof of Theorem 3.2, one can see that A is always successful in the event $\mathbf{E}$ that F uses recycled node certificates in its forgery.

The uf-cma adversary B gets a public key spk as input, generates $(\langle\hat{\mathbb{G}}\rangle, g, q) \xleftarrow{\$} \mathsf{K}_{\mathrm{cg}}(1^k)$ and computes $X \leftarrow g^x$ for a randomly chosen $x \xleftarrow{\$} \mathbb{Z}_q$. It then runs F on input $\mathit{tpk} = (\langle\hat{\mathbb{G}}\rangle, g, q, X, \mathit{spk})$ and proceeds exactly as algorithm B in the proof of Theorem 3.2. By the same analysis, B is always successful in the event $\overline{\mathbf{E}}$ that F's forgery contains a new node certificate, and Equation (12) follows.

5 Eliminating Node Certificates via Hashing

The above schemes use the node certification paradigm, and the standard signatures involved are a significant factor in the cost of the scheme. Here we show how, for some of the above schemes, one can eliminate node certificates by specifying the public label of a node i as the output of a hash function applied to 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, in a model where the hash function is a random oracle. Let us now illustrate this by presenting the schemes.

5.1 The *RSATS-2* scheme

We associate to any RSA key generator $\mathsf{K}_{\mathrm{rsa}}$ a transitive signature scheme *RSATS-2* = $(\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ defined as follows:

- $\mathsf{TKG}(1^k)$ does the following

  (1.1) Run $\mathsf{K}_{\mathrm{rsa}}(1^k)$ to get a triple (N, e, d)

  (1.2) Output $\mathit{tpk} = (N, e)$ as the public key and $\mathit{tsk} = (N, d)$ as the secret key.

  Now the following algorithms all have oracle access to a random function $\mathrm{H}_N\colon\ \mathbb{N} \to \mathbb{Z}_N^*$.

- The (stateless) signing algorithm TSign, when invoked on inputs tsk, i, j, meaning when asked to produce a signature on edge $\{i, j\}$, does the following:

  (2.1) If $i > j$ then $\mathit{swap}(i, j)$

  (2.2) $\delta \leftarrow [\mathrm{H}_N(i)\mathrm{H}_N(j)^{-1}]^d \bmod N$

  (2.3) Return δ as the signature of $\{i, j\}$.

- TVf, on input $\mathit{tpk} = (N, e)$, nodes i, j and a candidate signature δ, proceeds as follows:

  (3.1) If $i > j$ then $\mathit{swap}(i, j)$

  (3.2) If $\delta^e \equiv \mathrm{H}_N(i)\mathrm{H}_N(j)^{-1} \bmod N$ then return 1 else return 0.

- The composition algorithm Comp takes tpk, nodes i, j, k and signatures δ_1 and δ_2, and computes a composed signature for edge $\{i, k\}$ as follows:

```
(4.1) If $i > k$ then $swap(i, k)$ ; $swap(\delta_1, \delta_2)$
(4.2) If $i > j$ then $\delta_1 \leftarrow \delta_1^{-1} \bmod N$
(4.3) If $j > k$ then $\delta_2 \leftarrow \delta_2^{-1} \bmod N$
(4.4) $\delta \leftarrow \delta_1 \delta_2 \bmod N$
(4.5) Return $\delta$ as the signature for $\{i, k\}$.
```

As illustrated by Figure 1, this brings some significant performance gains over $\mathcal{RSATS}$-*1*, particularly with regard to signature size. Regarding security, in $\mathbf{Exp}^{\text{tu-cma}}_{\mathcal{RSATS}\text{-}2,\mathsf{F}}(k)$, we consider H_N: $\mathbb{N} \rightarrow \mathbb{Z}_N^*$ to be chosen at random after the public and secret keys (defining N) have been chosen. The TSign, TVf, and Comp algorithms, as well as the adversary, then get oracle access to H_N. In this random oracle model, we have the following.

Theorem 5.1 Let K_{rsa} be an RSA key generator and let $\mathcal{RSATS}$-*2* be the transitive signature scheme associated to K_{rsa} as defined above. If the one-more RSA-inversion problem associated to K_{rsa} is hard, then $\mathcal{RSATS}$-*2* is transitively unforgeable under adaptive chosen-message attack in the random oracle model. ▮

Proof of Theorem 5.1. Suppose we are given a polynomial-time tu-cma adversary F for $\mathcal{RSATS}$-*2*. We construct a polynomial-time om-rsa adversary A such that for all $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{RSATS}\text{-}2,\mathsf{F}}(k) \leq \mathbf{Adv}^{\text{om-rsa}}_{\mathsf{K}_{\text{rsa}},\mathsf{A}}(k) . \tag{13}$$

The theorem follows from the assumption that the one-more RSA-inversion problem associated to K_{rsa} is hard. It remains to describe A.

The om-rsa adversary A gets inputs N, e and has access to an inversion oracle $\textsc{Inv}(\cdot)$ and a challenge oracle $\textsc{Chall}$. It lets $tpk = (N, e)$ and runs F on input tpk. It will itself provide answers, both to F's queries to its random oracle H and to F's signature queries.

Adversary A maintains a table that represents H_N. When a query $\mathrm{H}_N(i)$ is made by F, adversary A does the following:

```
If $i \notin V$ then $\mathrm{H}_N(i) \xleftarrow{\$} \textsc{Chall}()$ ; $\Delta(i, i) \leftarrow 1$ ; $V \leftarrow V \cup \{i\}$
Return $\mathrm{H}_N(i)$ to $\mathsf{F}$
```

A answers F's signature queries similar to the way the om-rsa adversary from the proof of Theorem 3.2 did, but letting $\mathrm{H}_N(i)$ play the role of $L(i)$. More specifically, lines (5.4) and (5.8) are removed, and lines (5.1–13) are modified as follows:

```
(5.3)     $V \leftarrow V \cup \{i\}$ ; $\mathrm{H}_N(i) \xleftarrow{\$} \textsc{Chall}()$
(5.7)     $V \leftarrow V \cup \{j\}$ ; $\mathrm{H}_N(j) \xleftarrow{\$} \textsc{Chall}()$
(5.11)        $\Delta(i, j) \leftarrow \textsc{Inv}(\mathrm{H}_N(i) \cdot \mathrm{H}_N(j)^{-1} \bmod N)$
(5.21)    Return $\delta$ to $\mathsf{F}$.
```

At the end of its execution, F outputs a forgery δ' for edge $\{i', j'\}$. We can assume without loss of generality that F queried the hash oracle on i' and j' (and hence that $i', j' \in V$), because if it didn't, then A can query the hash oracle itself after F outputs its forgery. Algorithm A proceeds exactly as the om-rsa adversary in the proof of Theorem 3.2, treating the elements of V that were not involved in signature queries (but were queried to the random oracle only) as singleton components. Following the same reasoning, A can be seen to be successful whenever F is, and Equation (13) follows. (Note that there is no need for a uf-cma adversary B since the scheme does not rely on an underlying standard signature scheme.)

5.2 The *FactTS-2* scheme

To eliminate node certificates from the *FactTS-1* scheme, it is natural to want to let $L(i) = \mathrm{H}_N(i)$, where H_N is some public hash function. The question is, what is the range of this hash function? It would depend on N, the modulus in the signer's public key, which is why we have the subscript of N. However, unlike in *RSATS-2*, it would not suffice for this range to be $\mathbb{Z}_N^*$, because $L(i)$ needs to be a quadratic residue in $\mathbb{Z}_N^*$. So H_N would have to have range the set QR_N of quadratic residues modulo N. This is not a problem in the abstract random oracle model, where one can simply mandate that H_N be chosen with domain $\mathbb{N}$ and range QR_N, but the resulting scheme would be difficult to instantiate. In practice, one would like to build H_N out of a cryptographic hash function like SHA-1 that has range $\{0,1\}^{160}$. Given N, there are standard techniques that yield a hash function with range $\mathbb{Z}_N^*$ [BR93]. This is possible because membership in $\mathbb{Z}_N^*$ is decidable in polynomial time given N, and also $\mathbb{Z}_N^*$ is a "dense" subset of $\{0,1\}^k$ where k is the bit-length of N. However, there is no known way to build a function, computable in polynomial time given the input and N alone, that has range QR_N, because membership in the latter is not (known to be) decidable in polynomial time given N alone.

We could consider setting $L(i) = \mathrm{H}_N(i)^2 \bmod N$ where H_N has range $\mathbb{Z}_N^*$, but this reveals a square root of $L(i)$ which makes the scheme insecure. Instead, we let the signer choose N to be a Blum-Williams integer [Blu82, Wil80] (i.e. $N = pq$ with p and q primes such that $p \equiv q \equiv 3 \bmod 4$). Then it is well-known that -1 is a non-square modulo both p and q, and hence is a non-square modulo N with Jacobi symbol $+1$. As a consequence, for every element $x \in \mathbb{Z}_N^*[+1]$ (the elements of $\mathbb{Z}_N^*$ with Jacobi symbol $+1$), either x or $-x$ is a square modulo N. Now we will use as $\ell(i)$ a random square root of either $\mathrm{H}_N(i)$ or $-\mathrm{H}_N(i)$, whichever is a square, where H_N is a hash function with range $\mathbb{Z}_N^*[+1]$. 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 *FactTS-2* SCHEME. A modulus generator K_{BW} (as defined in Section 4.1), is said to be a *Blum-Williams modulus generator* if the primes p, q satisfy $p \equiv q \equiv 3 \bmod 4$. We associate to any given Blum-Williams modulus generator K_{BW} a transitive signature scheme *FactTS-2* = $(\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ defined as follows:

- TKG, on input 1^k, runs $\mathsf{K}_{\mathrm{BW}}(1^k)$ 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 $\mathrm{H}_N\colon\ \mathbb{N}^* \to \mathbb{Z}_N^*[+1]$.
- 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 $i > j$ then $swap(i, j)$
    If $i \notin V$ then $V \leftarrow V \cup \{i\}$ ; $\ell(i) \overset{\$}{\leftarrow} \sqrt{\pm \mathrm{H}_N(i)} \bmod N$
    If $j \notin V$ then $V \leftarrow V \cup \{j\}$ ; $\ell(j) \overset{\$}{\leftarrow} \sqrt{\pm \mathrm{H}_N(j)} \bmod N$
    $\delta \leftarrow \ell(i)\ell(j)^{-1} \bmod N$
```

  where the notation $x \overset{\$}{\leftarrow} \sqrt{\pm y} \bmod N$ means that x is chosen at random from the four square roots of y or $-y \bmod N$, whichever is a square modulo 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 $i > j$. It returns 1 if $\mathrm{H}_N(i) \cdot \mathrm{H}_N(j)^{-1} \equiv \pm\delta^2 \bmod N$ and returns 0 otherwise.
- The composition algorithm Comp is identical to that of *RSATS-2*.

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

Proposition 5.2 The *FactTS-2* transitive signature scheme described above satisfies the correctness requirement of Definition 2.1. ▌

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 this 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. We prove breaking the *FactTS-2* scheme equivalent to factoring in the random oracle model. This means that in the experiment $\mathbf{Exp}^{\text{tu-cma}}_{\mathit{FactTS\text{-}2},\mathsf{F}}(k)$ 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 5.3 Let K_{BW} be a Blum-Williams modulus generator. Let *FactTS-2* be the transitive signature scheme as defined above. If the factoring problem associated to K_{BW} is hard, then *FactTS-2* is transitively unforgeable under adaptive chosen-message attack in the random oracle model. ▌

Proof: Suppose we have a polynomial-time tu-cma forger F for *FactTS-2*. We will give a factoring algorithm A that uses F as a subroutine to factor composite numbers generated by K_{BW}. On input Blum-Williams integer N, A runs F on input N, answering its random oracle queries for node i as

```
If $i \notin V$ then
    $\ell(i) \xleftarrow{\$} \mathbb{Z}_N^*$; $s(i) \xleftarrow{\$} \{-1,+1\}$; $V \leftarrow V \cup \{i\}$
Return $s(i) \cdot \ell(i)^2 \bmod N$ to $\mathsf{F}$
```

Half of the elements in $\mathbb{Z}_N^*[+1]$ are squares with Legendre symbols $+1$ modulo both p and q, while the other half are non-squares with Legendre symbols -1 modulo both p and q. For a Blum-Williams integer N, -1 belongs to the latter subset, and every non-square in $\mathbb{Z}_N^*[+1]$ can be written as the product of -1 times a square mod N. Consequently, the output of the above algorithm follows the same distribution as a truly random function from $\mathbb{N}$ to $\mathbb{Z}_N^*[+1]$, as required.

A answers F's signature queries as follows:

```
If $i \notin V$ then
    $\ell(i) \xleftarrow{\$} \mathbb{Z}_N^*$; $s(i) \xleftarrow{\$} \{-1,+1\}$; $V \leftarrow V \cup \{i\}$
If $j \notin V$ then
    $\ell(j) \xleftarrow{\$} \mathbb{Z}_N^*$; $s(j) \xleftarrow{\$} \{-1,+1\}$; $V \leftarrow V \cup \{j\}$
If $i < j$ then return $\ell(i) \cdot \ell(j)^{-1} \bmod N$
else return $\ell(j) \cdot \ell(i)^{-1} \bmod N$.
```

Let F's forgery be (i', j', δ'). Again, we assume without loss of generality that F queried the random oracle on i' and j' before halting. Let E be the set of edges for which F queried a signature and let $\widetilde{G} = (V, \widetilde{E})$ be the transitive closure of the graph $G = (V, E)$. If F's output is not a successful forgery, meaning that either $\mathsf{TVf}(N, i', j', \delta') \neq 1$ or $\{i', j'\} \in \widetilde{E}$, A aborts. If $i' < j'$ let

$\delta \leftarrow \ell(i') \cdot \ell(j')^{-1} \bmod N$, otherwise let $\delta \leftarrow \ell(j') \cdot \ell(i')^{-1} \bmod N$. If $\delta \equiv \pm\delta' \bmod N$, then A aborts, otherwise it outputs $\gcd(\delta + \delta', N)$.

By arguments analogous to those in the proof of Theorem 4.1, A is successful whenever it doesn't abort. The advantage of the forger F is bounded by

$$\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{F}act\mathcal{TS}\text{-}2,\mathsf{F}}(k) \quad \leq \quad 2 \cdot \mathbf{Adv}^{\text{fact}}_{\mathsf{K}_{\text{BW}},\mathsf{A}}(k) \tag{14}$$

by a similar information-theoretic reasoning as presented in the proof of Theorem 4.1. ▌

5.3 The $\mathcal{G}ap\mathcal{TS}$-2 scheme

The discrete logarithm-based $\mathcal{DLTS}$ and $\mathcal{DLTS}$-$1\mathcal{M}$ schemes are not amenable to a hash-based improvement because the discrete exponentiation function is not trapdoor. For the $\mathcal{G}ap\mathcal{TS}$-1 scheme on the other hand, one can view $x = dlog_{\hat{\mathbb{G}},g}(X)$ as trapdoor information allowing to compute secret labels from public labels, giving rise to the stateless and very compact (in terms of signature size) $\mathcal{G}ap\mathcal{TS}$-2 scheme described below.

- The key generation algorithm $\mathsf{TKG}(1^k)$ calls $\mathsf{K}_{\text{cg}}(1^k)$ to generate the description $\langle\hat{\mathbb{G}}\rangle$ of a cyclic group $\hat{\mathbb{G}}$, a generator g and its order q. It chooses $x \xleftarrow{\$} \mathbb{Z}_q$ and computes $X \leftarrow g^x$. It outputs the public key $tpk = (\langle\hat{\mathbb{G}}\rangle, g, q, X)$ and the corresponding secret key $tsk = (\langle\hat{\mathbb{G}}\rangle, g, q, x)$. All algorithms have oracle access to a random function $\text{H}_{\hat{\mathbb{G}}} \colon \mathbb{N} \to \hat{\mathbb{G}}$.
- The (stateless) signing algorithm TSign, on input nodes i, j and secret key $tsk = (\langle\hat{\mathbb{G}}\rangle, g, q, x)$, proceeds exactly as the TSign algorithm of $\mathcal{RSATS}$-2 but replacing line (2.2) by

  (2.2) $\quad \delta \leftarrow \left[\text{H}_{\hat{\mathbb{G}}}(i) \cdot \text{H}_{\hat{\mathbb{G}}}(j)^{-1}\right]^x$
- TVf, on input $tpk = (\langle\hat{\mathbb{G}}\rangle, g, q)$, nodes i, j and candidate signature δ, first swaps i and j if $i > j$. It outputs 1 if $\mathsf{S}_{\text{ddh}}(\langle\hat{\mathbb{G}}\rangle, g, q, X, \text{H}_{\hat{\mathbb{G}}}(i) \cdot \text{H}_{\hat{\mathbb{G}}}(j)^{-1}, \delta) = 1$, or returns 0 otherwise.
- The Comp algorithm is the same as that of $\mathcal{RSATS}$-2, except that the operations in lines (4.2), (4.3) and (4.4) are performed in $\hat{\mathbb{G}}$, rather than modulo N.

Note that just like the short signature scheme of [BLS01], an edge signature under $\mathcal{G}ap\mathcal{TS}$-2 contains only a single element of $\hat{\mathbb{G}}$, which can be represented in roughly 160 bits when using elliptic curves to achieve the same security as a 1024-bit RSA modulus.

Proposition 5.4 The $\mathcal{G}ap\mathcal{TS}$-2 transitive signature scheme described above is correct according to Definition 2.1. ▌

Theorem 5.5 Let K_{cg} be a gap Diffie-Hellman group generator and let $\text{H}_{\hat{\mathbb{G}}} : \mathbb{N} \to \hat{\mathbb{G}}$ be a random oracle. The associated $\mathcal{G}ap\mathcal{TS}$-2 transitive signature scheme described above is transitively unforgeable under adaptive chosen-message attack under the one-more gap Diffie-Hellman assumption associated to K_{cg}. ▌

Proof: Suppose there exists a polynomial-time tu-cma forger F that breaks the $\mathcal{G}ap\mathcal{TS}$-2 scheme. We construct a om-cdh adversary A that uses F as subroutine such that for all $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{tu-cma}}_{\mathcal{G}ap\mathcal{TS}\text{-}2,\mathsf{F}}(k) \leq \mathbf{Adv}^{\text{om-cdh}}_{\mathsf{K}_{\text{rsa}},\mathsf{A}}(k) \; . \tag{15}$$

The theorem then follows from the one-more gap Diffie-Hellman assumption. Algorithm A takes input $\langle\hat{\mathbb{G}}\rangle, g, q, X$ and has access to a CDH oracle $\textsc{Cdh}_{\hat{\mathbb{G}},g}(X,\cdot)$ and a challenge oracle $\textsc{Chall}$. It runs the forger F on input $tpk = (\langle\hat{\mathbb{G}}\rangle, g, q, X)$ and proceeds exactly as the om-rsa adversary in the proof of Theorem 5.1, replacing multiplications mod N and $\textsc{Inv}(\cdot)$ queries with multiplications in $\hat{\mathbb{G}}$ and $\textsc{Cdh}_{\hat{\mathbb{G}},g}(X,\cdot)$ queries, respectively, and maintaining a function $\Delta : V \times V \to \hat{\mathbb{G}}$ as part of its state information. The same analysis shows that algorithm A is successful whenever the forger F is, yielding Equation (15).

6 From Stateful to Stateless Schemes

The signing algorithms of many transitive signature schemes are stateful. (For example this is true for the *RSATS-1* scheme, where it is important for composition that the signer associates a single public label to node i.) Statefulness can also be important for security in that it associates to a public label a single secret label $\ell(i)$. (The *FactTS-1* and *FactTS-2* schemes for example would otherwise soon give away two different square roots of $L(i)$, allowing an attacker to factor the modulus.) The *DLTS*, *DLTS-1M* and *GapTS-1* schemes also have stateful signing algorithms.

In case one would like a stateless scheme, we note here a simple transformation that can be used to make the signer stateless, while preserving security and incurring a low computational overhead. The transform does not apply to all stateful transitive signature schemes, but to a large class that includes the stateful schemes of this paper.

The class of schemes we consider is those where the state consists of a function St with domain the set V of all nodes that were an input to some execution of the signing algorithm so far. (The set V starts out empty.) We assume there are polynomial time, deterministic, stateless algorithms $\mathsf{TSignN}, \mathsf{TSignE}$ such that the algorithm TSign, given inputs tsk, i, j, coins ω, and the current state St, does the following. First it splits ω into two halves, $\omega = \omega_i \| \omega_j$. Then it proceeds as follows:

```
If $i \notin V$ then $V \leftarrow V \cup \{i\}$ ; $\mathrm{St}(i) \leftarrow \mathsf{TSignN}(tsk, i, \omega_i)$
If $j \notin V$ then $V \leftarrow V \cup \{j\}$ ; $\mathrm{St}(j) \leftarrow \mathsf{TSignN}(tsk, j, \omega_j)$
$\sigma \leftarrow \mathsf{TSignE}(tsk, i, j, \mathrm{St}(i), \mathrm{St}(j))$
Return $\sigma$
```

We modify this scheme to be stateless as follows. Augment the signer's secret key to also include a key K specifying an instance F_K from a pseudorandom function (PRF) family F [GGM86]. The new signing algorithm is both deterministic and stateless. Given inputs $(tsk, K), i, j$, it proceeds as follows:

```
$\omega_i \leftarrow F_K(i)$ ; $\omega_j \leftarrow F_K(j)$
$\mathrm{St}_i \leftarrow \mathsf{TSignN}(tsk, i, \omega_i)$
$\mathrm{St}_j \leftarrow \mathsf{TSignN}(tsk, j, \omega_j)$
$\sigma \leftarrow \mathsf{TSignE}(tsk, i, j, \mathrm{St}_i, \mathrm{St}_j)$
Return $\sigma$
```

In practice one can implement the pseudorandom function family via a block cipher. (Direct application of the block cipher may not be enough since the number of bits output by the block cipher may be too small, but there are many ways to turn a block cipher, assumed to be a PRF family, into a PRF family providing outputs of some larger desired length.) Since operation of a block cipher is significantly cheaper than the number-theoretic operations already being used in the transitive signature schemes, the stateless scheme will have a cost close to that of the original stateful one.

We note that, interestingly, the $\mathcal{RSATS}$-2 and $\mathcal{GapTS}$-2 schemes are naturally stateless, as the public label associated to node i is defined by the output of a random oracle $\mathrm{H}(i)$, and the secret label is the unique "inverse" of this value.

References

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001. (Cited on page 6, 23, 28.)

[Blu82] Manuel Blum. Coin flipping by telephone. In A. Gersho, editor, *Advances in Cryptology: A Report on CRYPTO 81*, University of California, Santa Barbara, Department of ECE Report No 82-04, pages 11–15, 1982. (Cited on page 26.)

[BN02] Mihir Bellare and Gregory Neven. Transitive signatures based on factoring and RSA. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 397–414. Springer-Verlag, 2002. (Cited on page 8.)

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003. (Cited on page 5, 12, 20, 22.)

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *Advances in Cryptology – Public-Key Cryptography 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, 2003. (Cited on page 6, 22, 23.)

[BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attack. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, August 2002. (Cited on page 5, 6, 20.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In ACM, editor, *Proceedings of the 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993. (Cited on page 7, 11, 26.)

[Cha83] David Chaum. Blind signatures for untraceable payments. In D. Chaum, R. Rivest, and A. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum Press, 1983. (Cited on page 5.)

[Cor00] Jean-Sbastien Coron. On the exact security of full domain hash. In M. Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, 2000. (Cited on page 35.)

[CRR02] Suresh Chari, Tal Rabin, and Ronald Rivest. An efficient signature scheme for route aggregation. Manuscript, available from `http://theory.lcs.mit.edu/~rivest/publications.html`, 2002. (Cited on page 8.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986. (Cited on page 29.)

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 11.)

[GQ89] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO 1988*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, August 1989. (Cited on page 5.)

[Hoh03] Susan Hohenberger. The cryptographic impact of groups with infeasible inversion. Master's thesis, Massachusetts Institute of Technology, available from `http://theory.lcs.mit.edu/~cis/cis-theses.html`, 2003. (Cited on page 8.)

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In B. Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262. Springer-Verlag, 2002. (Cited on page 7, 8, 10, 32.)

[MR02] Silvio Micali and Ronald Rivest. Transitive signature schemes. In B. Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 236–243. Springer-Verlag, 2002. (Cited on page 1, 3, 4, 5, 7, 9, 10, 11, 20, 32.)

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In E. Brickell, editor, *Advances in Cryptology – CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer-Verlag, August 1993. (Cited on page 6.)

[Riv00] Ronald Rivest. Two signature schemes. Slides from talk given at Cambridge University, October 17, 2000. Available from `http://theory.lcs.mit.edu/~rivest/publications.html`, 2000. (Cited on page 8.)

[SBZ02] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In Kwangjo Kim, editor, *Information Security and Cryptology - ICISC 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 285–304. Springer-Verlag, 2002. (Cited on page 8.)

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smartcards. In G. Brassard, editor, *Advances in Cryptology – CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer-Verlag, August 1990. (Cited on page 6, 20.)

[SSM05] Siamak Fayyaz Shahandashti, Mahmoud Salmasizadeh, and Javad Mohajeri. A provably secure short transitive signature scheme from bilinear group pairs. In C. Blundo and S. Cimato, editors, *Security in Communication Networks: 4th International Conference, SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 60–76. Springer-Verlag, 2005. (Cited on page 2, 8, 34, 35.)

[Wil80] Hugh C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, 26(6):726–729, 1980. (Cited on page 26.)

A Definitional Issues in Composition of Signatures

It is natural to consider the following alternative definition of correctness for composition. Say that a transitive signature scheme $\mathcal{TS} = (\mathsf{TKG}, \mathsf{TSign}, \mathsf{TVf}, \mathsf{Comp})$ is *strongly-correct* if for every k, and every (tpk, tsk) that might be returned by $\mathsf{TKG}(1^k)$, we have:

If σ_1 is a valid signature of edge $\{i, j\}$ relative to tpk, meaning $\mathsf{TVf}(tpk, i, j, \sigma_1) = 1$
And σ_2 is a valid signature of edge $\{j, k\}$ relative to tpk, meaning $\mathsf{TVf}(tpk, j, k, \sigma_2) = 1$
And $\sigma = \mathsf{Comp}(tpk, i, j, k, \sigma_1, \sigma_2)$
Then: σ is a valid signature of edge $\{i, k\}$ relative to tpk, meaning $\mathsf{TVf}(tpk, i, k, \sigma) = 1$

The purpose of this section is to point out that the $\mathcal{DLTS}$ scheme does not meet this definition. Similar examples can be used to see that none of our schemes meet this definition either. Note that the definition of [JMSW02] implies strong-correctness as we have formulated it above (it requires more) and thus neither the $\mathcal{DLTS}$ scheme nor our schemes meet their definition.

We note that the "bad" inputs of our example can only be created by an adversary capable of forging standard signatures. However, we feel that composition is a "correctness" rather than a security requirement and should not rely on computational restrictions on adversaries.

The rest of this section assumes that the reader is familiar with the $\mathcal{DLTS}$ scheme [MR02] that we sketched in Section 4.2. We present a counter-example to show that this scheme does not meet the above strong-correctness requirement. The signer's public key tpk in the $\mathcal{DLTS}$ scheme contains two random generators g_1, g_2 of an underlying group $\mathbb{G}$ of prime order q, as well as a public key spk for the underlying standard signature scheme. Suppose i, j, k are distinct nodes such that $i < j < k$. Suppose $\sigma_1 = (C_i, C_j, \delta_{11}, \delta_{12})$ is a valid signature of $\{i, j\}$ relative to tpk, meaning

- $C_i = (i, L_i, \Sigma_i)$
- $C_j = (j, L_j, \Sigma_j)$
- $L_i L_j^{-1} \equiv g_1^{\delta_{11}} g_2^{\delta_{12}}$
- Σ_i a valid signature of $i \| L_i$ relative to spk
- Σ_j a valid signature of $j \| L_j$ relative to spk.

Also suppose $\sigma_2 = (C'_j, C_k, \delta_{21}, \delta_{22})$ is a valid signature of $\{j, k\}$ relative to tpk, meaning

- $C'_j = (j, L'_j, \Sigma'_j)$
- $C_k = (k, L_k, \Sigma_k)$
- $L'_j L_k^{-1} = g_1^{\delta_{21}} g_2^{\delta_{22}}$
- Σ'_j a valid signature of $j \| L'_j$ relative to spk
- Σ_k a valid signature of $k \| L_k$ relative to spk.

On inputs $tpk, i, j, k, \sigma_1, \sigma_2$, the composition algorithm of the $\mathcal{DLTS}$ scheme is defined to return $(C_i, C_k, \delta_{31}, \delta_{32})$ where

$$\delta_{31} \;=\; \delta_{11} + \delta_{21} \bmod q \qquad \delta_{32} \;=\; \delta_{12} + \delta_{22} \bmod q\ .$$

Now, using the above, we have

$$\begin{aligned} g_1^{\delta_{31}} g_2^{\delta_{32}} &= g_1^{\delta_{11}+\delta_{21}} g_2^{\delta_{12}+\delta_{22}} \\ &= g_1^{\delta_{11}} g_2^{\delta_{12}} \cdot g_1^{\delta_{21}} g_2^{\delta_{22}} \\ &= L_i L_j^{-1} \cdot L'_j L_k^{-1} \ . \end{aligned}$$

For the verification algorithm to accept, the above should equal $L_i L_k^{-1}$, meaning the verification algorithm would only accept $(C_i, C_k, \delta_{31}, \delta 32)$ as a valid signature of $\{i, k\}$ relative to *tpk* if $L_j = L'_j$. However, the validity of the given signatures σ_1, σ_2 does not imply that $L_j = L'_j$. Accordingly, we have an example of valid signatures yielding, via composition, an invalid signature. This shows that the $\mathcal{DLTS}$ scheme is not strongly-correct.

Note that creating the valid signatures σ_1, σ_2, even given oracle access to the signer, would require forging relative to the standard scheme, so in practice we do not expect the composition algorithm to receive these inputs. However, it is inconvenient for formulate a correctness requirement that hinges on security.

The $\mathcal{DLTS}$ scheme is, however, correct as per Definition 2.1. Even though an algorithm A in the experiment of Figure 3 is not computationally restricted and could create σ_1, σ_2 as above and invoke the composition algorithm, examination of the experiment shows that the flag *Legit* would be set to `false`, and thus A would not win, so our definition is not violated.

B Correctness Proof for $\mathcal{RSATS}$-*1*

Claim B.1 If (ℓ, L, Σ, V) is the internal state of the TSign algorithm in $\mathcal{RSATS}$-*1*, then at any time during the experiment in Figure 3, the following invariant holds true:

$$\mathit{Legit} = \texttt{false} \quad \vee \quad \forall\, (\{i,j\}, \sigma) \in S :$$

$$i \neq j \quad \wedge \quad \sigma = \begin{cases} (C_i,\ C_j,\ \delta_{ij}) & \text{if } i < j \\ (C_j,\ C_i,\ \delta_{ji}) & \text{if } j < i \end{cases} \tag{16}$$

where $C_i = (i, L(i), \Sigma(i))$, $C_j = (j, L(j), \Sigma(j))$, $\delta_{ij} = \ell(i)\ell(j)^{-1} \bmod N$ and $\delta_{ji} = \ell(j)\ell(i)^{-1} \bmod N$.

Proof: We will prove the claim by induction on the number of TSign oracle queries q. In the initial state, $S = \emptyset$ and the claim is trivial. Suppose that the claim is true after $q-1$ oracle queries. We will prove that it still holds after the qth oracle query.

If *Legit* $=$ `false` before the qth query, then it will still be `false` after the qth query, directly proving the claim. We now concentrate on the case that *Legit* $=$ `true`.

If the qth query is a TSign query i, j with $i = j$, *Legit* is set to `false`, again easily proving the claim. Otherwise, a new element $(\{i,j\}, \sigma)$ is added to S, where σ is the output of $\mathsf{TSign}(tsk, i, j)$. All elements of S that satisfied Equation (16) in the previous state of TSign, will still do so in the new state, because TSign only adds new entries to ℓ, L and Σ, but never changes existing entries. Therefor, it suffices to show that the newly added element $(\{i,j\}, \sigma)$ satisfies Equation (16). This can be seen from the description of the TSign algorithm. If $i < j$, it outputs a signature $\sigma = ((i, L(i), \Sigma(i)), (j, L(j), \Sigma(j)), \delta)$ with $\delta = \ell(i)\ell(j)^{-1} \bmod N$, as required. If $j < i$, TSign first swaps the values of i and j in line (2.1), such that the output of the algorithm is actually $\sigma = ((j, L(j), \Sigma(j)), (i, L(i), \Sigma(i)), \delta)$ with $\delta = \ell(j)\ell(i)^{-1} \bmod N$, again as required by Equation (16).

If the qth query is a Comp query $i, j, k, \sigma_1, \sigma_2$, we prove the claim as follows. If $(\{i, j\}, \sigma_1) \notin S$ or $(\{j, k\}, \sigma_2) \notin S$ or i, j, k are not all distinct, then *Legit* is set to `false` and the claim holds true. Otherwise, the composition algorithm is run to create $\sigma = \mathsf{Comp}(tpk, i, j, k, \sigma_1, \sigma_2)$, and the element $(\{i, k\}, \sigma)$ is added to S. As the internal state of the TSign algorithm is not affected by the composition algorithm, all elements that previously satisfied Equation (16) will still do so. We only have to check that the newly added element also satisfies Equation (16). If $i > k$ then i and k are swapped in line (4.1), as are the signatures σ_1 and σ_2. At this point we have signatures σ_1 and σ_2 for edges $\{i, j\}$ and $\{j, k\}$ satisfying equation Equation (16) with $i < k$. Let $\sigma_1 = (C_1, C_2, \delta_1)$, and let $\sigma_2 = (C_3, C_4, \delta_2)$. Line (4.3) of the Comp algorithm swaps C_1 and C_2 and inverts δ_1 if $i > j$, ensuring that after this step $C_1 = (i, L(i), \Sigma(i))$, $C_2 = (j, L(j), \Sigma(j))$ and $\delta_1 \equiv \ell(i)\ell(j)^{-1} \bmod N$. The same is done with C_3, C_4 and δ_2 if $j > k$ in line (4.4), ensuring that $C_3 = (j, L(j), \Sigma(j))$, $C_4 = (k, L(k), \Sigma(k))$ and $\delta_2 \equiv \ell(j)\ell(k)^{-1} \bmod N$. The signature that is finally returned is (C_1, C_4, δ), which indeed satisfies Equation (16) since $i < k$ and δ is computed as $\delta_1\delta_2 \equiv \ell(i)\ell(j)^{-1} \cdot \ell(j)\ell(k)^{-1} \equiv \ell(i)\ell(k)^{-1} \bmod N$. ▌

A corollary of the previous claim is that at any time during the experiment, $\mathsf{TVf}(tpk, i, j, \sigma) = 1$ for all $(\{i, j\}, \sigma) \in S$. From the description of the TSign algorithm, we can see that $L(i) \equiv \ell(i)^e \bmod N$ and $\Sigma(i)$ is a valid standard signature under *spk* for $i \| L(i)$. Given these facts and Equation (16), we can go through the description of TVf and check that it always returns 1.

Claim B.2 The variable *NotOK* in the experiment in Figure 3 can never become `true`.

Proof: By the corollary above, the verification of a signature in S always succeeds, so the only way left for *NotOK* to become true during the experiment is when $\sigma \neq \tau$ in a Comp query. The output of the signature algorithm for nodes i, k is $\tau = ((i, L(i), \Sigma(i)), (k, L(k), \Sigma(k)), \ell(i)\ell(k)^{-1})$ when $i < k$, and is $\tau = ((k, L(k), \Sigma(k)), (i, L(i), \Sigma(i)), \ell(k)\ell(i)^{-1})$ if $k < i$. We now prove that this is identical to the output of the composition algorithm when applied to nodes i, j, k and signatures σ_1, σ_2 such that $(\{i, j\}, \sigma_1), (\{j, k\}, \sigma_2) \in S$. In the proof of Claim B.1, we already argued that the variables C_1 and C_4 by the end of the Comp algorithm are always assigned values $(i, L(i), \Sigma(i))$ and $(k, L(k), \Sigma(k))$, respectively, and that $\delta \equiv \ell(i)\ell(k)^{-1} \bmod N$. The values for i and k, however, might have been swapped in the first line of the Comp algorithm, so the returned signature is actually $\sigma = ((i, L(i), \Sigma(i)), (k, L(k), \Sigma(k)), \ell(i)\ell(k)^{-1})$ if $i < k$ and $\sigma = ((k, L(k), \Sigma(k)), (i, L(i), \Sigma(i)), \ell(k)\ell(i)^{-1})$ if $k < i$, exactly like τ. ▌

Since the experiment outputs $(Legit \wedge NotOK)$ at the end of its execution, the previous theorem implies that it returns `false` for every adversary A, thereby proving the correctness of $\mathcal{RSATS}$*-1*. ▌

C Flaw in Proof of [SSM05]

Recently, Shahandashti et al. [SSM05] published a transitive signature scheme based on bilinear maps that (barring notation and some minor implementation details) is the same as our $\mathcal{G}ap\mathcal{TS}$*-1* scheme. They however claim to prove a stronger security result, namely that the scheme is secure under the CDH assumption (and the security of an underlying standard signature scheme), rather than under the stronger *one-more* CDH assumption under which we prove it secure. However, we show here that their proof is flawed, invalidating their result.

To point out the error in the proof of [SSM05], we briefly sketch the proof here, adapting their notation to be consistent with ours. Given an adversary A breaking the tu-cma-security of $\mathcal{G}ap\mathcal{TS}$*-1*, they construct a CDH algorithm B as follows. On input a CDH instance $(\hat{\mathbb{G}}, g, q, X, Y)$, algorithm

B generates a fresh key pair (spk, ssk) for the underlying standard signature scheme, and runs A on input $tpk = (\hat{\mathbb{G}}, g, q, X, spk)$. (Shahandashti et al. actually use $g^r X$ instead of X for a randomly chosen $r \in \mathbb{Z}_q$, but this is irrelevant to the proof since X is already uniformly distributed over $\hat{\mathbb{G}}$.) The proof employs a technique due to Coron [Cor00] to answer A's TSign queries: when a node i is queried for the first time, B chooses $b_i \overset{\$}{\leftarrow} \mathbb{Z}_q$ and flips a biased coin c_i that turns out heads $(c_i = 0)$ with probability p_0 and turns out tails $(c_i = 1)$ with probability $1 - p_0$. If $c_i = 0$ (in which case node i is referred to as a *Y-node*), it sets the public label $L(i) \leftarrow Y \cdot g^{b_i}$; if $c_i = 1$ (node i is a *non-Y-node*), it sets $L(i) \leftarrow g^{b_i}$. If A asks for a signature on edge $\{i, j\}$ such that $c_i = c_j$ (meaning i and j are both Y-nodes or both non-Y-nodes), B computes the edge label as $X^{b_i - b_j}$. If $c_i \neq c_j$, then B gives up. Eventually, A outputs its forgery on an edge $\{i', j'\}$ with edge label Z. If $c_{i'} = 0$ and $c_{j'} = 1$, then B outputs $Z \cdot g^{b_j - b_i}$ as its CDH solution; if $c_{i'} = 1$ and $c_{j'} = 0$, then B outputs $Z^{-1} \cdot g^{b_i - b_j}$; if $c_{i'} = c_{j'}$, then B gives up. It is clear that B's output is equal to $CDH_{\hat{\mathbb{G}},g}(X, Y)$ if it doesn't give up.

The critical point of the proof is the probability that B doesn't give up during the game. Let q be the number of A's TSign queries. Shahandashti et al. argue [SSM05, p. 74] that in each query, the nodes involved are of the same type (i.e. both Y-nodes or both non-Y-nodes) with probability $p_0^2 + (1 - p_0)^2$, and that the nodes i', j' involved in the forgery are of different types with probability $2p_0(1 - p_0)$, yielding a success probability for B of $\beta(q, p_0) = 2p_0(1 - p_0)[p_0^2 + (1 - p_0)^2]^q$ times that of A. By maximizing this function over $p_0 \in [0, 1]$, one obtains

$$\max_{p_0 \in [0,1]} (\beta(q, p_o)) = \frac{\left(1 - \frac{1}{q+1}\right)^{q+1}}{q} = O(1/q) \ ,$$

losing only a factor linear in q in the reduction.

The above reasoning, however, implicitly assumes that the probability that two nodes are of the same type is independent of previously signed edges. This is not the case, as multiple edges may be incident to one node.

While this already shows that the analysis of the advantage of adversary B is wrong, and is enough to invalidate the proof, one might ask a further question, namely that perhaps their construction (of B) actually works, and a different analysis would show this. This turns out not to be true, meaning that the reduction itself is flawed. We demonstrate this by constructing a polynomial-time adversary A such that the success probability of adversary B is exponentially small, regardless of A's success probability. Consider an adversary A that, right before outputting its forgery on edge $\{i', j'\}$, first performs additional TSign queries to organize the graph as two disjoint cliques of about equal size, creating new nodes if necessary. Let q' be the total number of nodes in the graph. (Without loss of generality, we can assume that $q' = O(q)$.) For B to succeed, the clique of node i' has to consist entirely of Y-nodes, while the clique of node j' only contains non-Y-nodes, or vice versa. Since each node is chosen to be a Y-node with probability p_0, independently of other nodes, this happens with probability

$$2 \cdot p_0^{q'/2} \cdot (1 - p_0)^{q'/2} \ = \ 2[p_0(1 - p_0)]^{q'/2} \ \leq \ 2^{-q'+1} \ ,$$

which is exponentially small in the security parameter k if q (and hence q') is polynomial in k.