Rekeyed Digital Signature Schemes: Damage-containment in the face of key exposure

Michel Abdalla* Mihir Bellare†

July 2001

Abstract

Motivated by the problem of delegating signing keys to vulnerable mobile devices, we define rekeyed digital signature schemes. We provide an adversary model and a strong notion of security for such schemes, and show that the classic self-certification paradigm, properly implemented, provably meets this notion of security. We then suggest altnerative solutions, based on identification schemes, and having certain performance benefits compared to self-certification.

Keywords: Digital signatures, key exposure, delegation, forward security, identification schemes, proofs of security.

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: mabdalla@cs.ucsd.edu. URL: http://www.michelabdalla.net. Supported by CAPES under Grant BEX3019/95-2.

†Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-mail: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir. Supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering.

Contents

1 Introduction

THE MULTI-DEVICE SETTING. The number of devices per person is proliferating: besides a primary desktop computer, a user may have a laptop computer, a palm-pilot and a cell phone, all with Internet access. Such a user may want a digital signature capability on all these devices which, for convenience and simplicity of key-management, is based on a single certified public verification key pk associated to the user. The user certainly has the option of downloading the secret signing key sk, associated to public key pk, from his or her primary device, onto secondary devices. However, the secondary devices, usually being mobile, have a greater vulnerability to capture and consequent exposure of the secret key.

CONSEQUENCES OF KEY EXPOSURE. Key-exposure for a digital signature scheme remains undesirable even if the public key can be quickly and effectively revoked, itself a non-trival task. The reason is that exposure of the signing key means that all signatures ever produced under this key, even those that were legitimately produced prior to key-exposure, become untrusted: a verifier confronted with signature cannot be sure of its authenticity, even if the document is dated.

SELF-CERTIFICATION. A standard paradigm, that can be employed in this situation, is the following. The user creates, for each secondary device j, a new public verification key pk_j with associated secret signing key sk_j. It certifies pk_j by signing it using the primary secret key sk. Rather than downloading sk onto device j, it provides the device with sk_j, together with pk_j and its certificate. The device will produce signatures under sk_j but include, as part of the signature, the secondary public key pk_j together with its certificate. So verification under pk remains possible. In this paper we call this *self-certification* because the user issues himself or herself a certificate.

If the device is captured, the adversary obtains the capabilty to create signatures under sk_j. In that case one might have to revoke pk. But the hoped for gain is that signatures produced by devices other than j, being under different secondary keys, can still be trusted. The damage caused by loss of sk_j has been "localized," or "contained," even if not eliminated.

OUR WORK. This paper suggests that underlying the specific solution above is a more general concept. We distill this to arrive at a notion of a *rekeyed digital signature scheme*. We provide an adversary model and a definition of security for such schemes.

We then cast the self-certification scheme outlined above as a rekeyed digital signature scheme and prove its security. Although simple, we believe this is important for several reasons. It provides information on details of how the scheme should be implemented that are important to security, and enables us to understand precisely what are the security attributes that the scheme provably possesses. It also sets the context for the alternative solutions we discuss next.

A benefit of the generalization is that it provides the perspective to see that self-certification is not the only solution. We provide a rekeyed digital signature scheme that we call the *rekeyed iterated-root scheme*. It is based on the Ong-Schnorr scheme [OS90]. The interest of this and similar alternative solutions is that they compare favorably with the self-certification scheme with regard to certain performance attributes, including signing time and signature sizes. (The self certification scheme results in an increase of the signature size due to inclusion of the secondary public key and its certificate in the signature, and our rekeyed iterated-root scheme avoids this.) The rekeyed iterated-root scheme is proved secure in the random oracle model (cf. [BR93a]) assuming factoring is hard.

We now look at some of these items in some more detail, and then conclude this Introduction with a discussion of related work.

REKEYED SIGNATURE SCHEMES. A user generates a public key and matching secret key, which in this context are called the *primary public key* and *primary secret key respectively.* The user obtains a certificate for the primary public key. Rather than use sk directly to produce signatures, the user uses it to derive sub-keys $sk_1, sk_2, \ldots$, where sk_j is viewed as associated to a session with number j, and is hence also called a session key. This is done via a *session-key generation algorithm* that on inputs sk, j returns sk_j. A signature can be produced using a session key, but remains a signature relative to the primary public key. (Signatures are verified under the single, fixed primary public key, regardless of the session key used to produce them, so that certification of the public key is done only once.) The verification algorithm will indicate not just whether a signature is valid or not, but the index of the session in which it believes the signature to have been produced. See Section 2.

TRUST ASSUMPTIONS AND SECURITY GOALS. We assume that the primary secret key is protected, but that session keys are vulnerable to exposure. We adopt a model in which the adversary is very powerful, being able to initiate sessions, expose session keys of its choice, and mount chosen-message attacks on un-exposed session keys. In the face of this adversary capability we ask for a very strong security requirement, namely that signatures produced under any un-exposed session key can be trusted. In other words, if sk_i has not been exposed, then it is computationally infeasible for the adversary to create a message M and candidate signature σ such that the verification algorithm, on input pk, M, σ, says that σ is valid for M with respect to session i. See Section 2.

CONTEXT. The multi-user setting we discussed above is one in which rekeyed encryption could be used, but there are others. For example the user could issue session keys valid for certain intervals of time rather than for different devices, or combine these ideas by issuing tickets, valid for certain intervals, to each device.

The model ideas are inspired by those for forward-secure signatures as formalized in [BM99], and those for session-key distribution as formalized in [BR94, BR96]. In particular the security requirement that loss of security of one session not compromise the security of other sessions mimics the same requirement in the session-key distribution context (cf. [DS81]), but "security" has to be re-interpreted here.

SELF-CERTIFICATION REVISITED. While the basic idea is the classic one discussed above, attaining the type of security we target requires that one be careful about implementation. In particular it is important to include, in a signature, the index of the session in which the signature was created, and to implement verification, as mentioned above, to return not just a decision bit, but the index of the session in which it imagines the signature was created. With these amendments, we prove the expected result that the self-certification scheme meets our notion of security for rekeyed schemes provided the underlying signature schemes (used to certify the secondary public keys and to produce signatures under the secondary secret keys) are secure against forgery under chosen-message attack according to the standard definition of [GMR88]. See Section 3.

THE REKEYED ITERATED-ROOT SCHEME. A special case of Ong-Schnorr signature scheme [OS90] is the following. The user has public key a modulus N and a value $U = S^{2^h} \bmod N$ where the user's secret key is $S \in \mathsf{Z}_N^*$ and $h \geq 1$ is a fixed, public integer paremeter. Signatures are created by applying the Fiat-Shamir transform [FS87] to a 2^h-th root based identification protocol. In the adaptation to rekeying that we propose, the user's primary public key is N, while the prime factors of N make up the secret key. Associated to session j is a (secondary) public key U_j. This is not specified explicitly, but rather implicitly, via the output of a hash function on fixed inputs that contain the value j. The corresponding (secret) session key S_j is a 2^h-th root of U_j that is computed

using the prime factors of N in the primary secret key. Signatures are computed under S_j as they would be in the Ong-Schnorr scheme, but include j. (There are several details left out of this description. In particular one has to deal with the fact that random outputs of the hash function may not be squares and thus not suitable as public keys. See Section 4 for a full description.)

Conceptually this borrows from the self-certification idea in that there is a secondary public key associated to each session. Reduction in the signature size is achieved because the secondary public key is implicit rather than explicit, and also because *it is not explicitly certified* so no certificate need be included in the signature. A detailed cost and efficiency comparision between an RSA-based version of the self-certification scheme and the rekeyed iterated-root scheme is provided in Section 5, and shows that the latter fares better with respect to all parameters, except perhaps verification time if one uses a small verification exponent for RSA.

We show that the rekeyed iterated root scheme meets our definition of a secure rekeyed signature scheme assuming the Ong-Schnorr scheme meets the standard notion of security for a digital signature scheme, namely unforgeability under chosen-message attack. Known analyses of the Ong-Schnorr scheme (cf. [OS90, MR02]) then imply that our scheme is secure if factoring is hard. All these results are in the random oracle model.

Related work. There are two lines of work dealing with key exposure. The first considers prevention of key exposure. One class of solutions is based on hardware tokens such as secure co-processors or smartcards. Another class of solutions, originating with [Bla79, Sha79], is based on the idea of distributing a secret key across multiple devices that. (See for example [DF90, Des94, HJJ+97], and in another vein, [Sin, MR01].) The second line of work assumes key exposure will happen and considers damage control. An instance of this is forward-security for signatures, initiated in [And00, BM99] and continued in [AR00, Kra00, IR01, MMM02]. Systems combining distribution and forward-security are considered in [AMN01, TT01].

Our setting builds on both lines of work. The primary secret key could be protected using a method from one of the first lines of work above. These methods however are typically not conducive to mobility, so we would not wish to directly extend them to secondary devices. Instead, rekeying is used, providing security properties related to the second line of work above. In particular the security property we require from our rekeyed schemes can be considered an extension of forward security in which one gets not just forward-security, but also backward-security, and in fact all-directions-security, since keys from sessions that may be concurrent and overlapping have security independent of each other. (One should remember however that the solutions for forward security from the above-mentioned works are software-only, meaning they assume that any information can can be compromised, while we assume that the primary secret key will not be compromised.)

Forward-secure schemes based on self-certification were suggested by Anderson [And00], and these ideas are extended in [BM99, Kra00, MMM02]. Forward-secure schemes based on identification were proposed in [BM99, AR00, IR01].

2 Definitions

Conventions. We will use both the standard model and the random oracle model [BR93a]. The computational model in the latter case enhances that in the former case by providing a new instruction, or call, $\mathsf{HO}(\cdot,\cdot)$, available to all algorithms including the adversary. The instruction takes input a (description of a) range set R and a string x and returns an element y of R, with the intended semantics that y is uniformly distributed over R subject to consistency, meaning the same query made twice results in the same answer.

Experiment $\mathbf{Exp}^{\text{uf-cma}}_{\text{DS},F}(k)$
Begin

1. $(pk, sk) \xleftarrow{R} \text{KG}(k)$; $SM \leftarrow \emptyset$

2. Run F on input pk, responding to induced queries as indicated in the table below:

Query	**Response**
$\mathsf{HO}(R, x)$	If $\text{HT}[R, x] = \perp$ then $\text{HT}[R, x] \xleftarrow{R} R$ Return $\text{HT}[R, x]$
$\mathsf{sign}(M)$	$\sigma \leftarrow \text{SIGN}(sk, M)$; $SM \leftarrow SM \cup \{M\}$ Return σ to F
$\mathsf{verify}(M, \sigma)$	If ($\text{VF}(pk, M, \sigma) = 1$ AND $M \notin SM$) Then $w \leftarrow 1$ else $w \leftarrow 0$

3. Return the bit w computed in the response to F's verify query above

End

Figure 1: The experiment measuring the success of uf-cma-adversary F in its attack on digital signature scheme $\text{DS} = (\text{KG}, \text{SIGN}, \text{VF})$. The table indicates the types of oracle queries that F can make and how the experiment responds to them.

All algorithms are randomized unless otherwise indicated. Each definition of security associates to a given scheme, adversary, and value of the security parameter an experiment that returns 1 if the adversary is successful and 0 otherwise. The *time-complexity* of the adversary is by definition the execution time of the associated experiment plus the size of the code of the adversary, in some fixed RAM model of computation, and measured as a function of the security parameter. In particular, the time-complexity of the adversary includes the time to reply to random oracle queries, which is done by maintaining a table $\text{HT}[\cdot, \cdot]$ with the intended semantics that $\text{HT}[R, x]$ is the response to $\mathsf{HO}(R, x)$. Table entries are chosen at random in response to queries. This means that if the adversary has polynomial time-complexity then, implicitly, the time to sample uniformly from a set R included in a $\mathsf{HO}(R, \cdot)$ query is polynomial in k. (If it is possible to sample uniformly except with an exponentially small failure probability, as in the case where R is Z^*_N for some integer N, we ignore the failure probability, and assume perfect sampling in polynomial time.)

Digital signature schemes. Recall that a (digital) signature scheme $\text{DS} = (\text{KG}, \text{SIGN}, \text{VF})$ is specified by three polynomial-time algorithms. The key generation algorithm KG takes input a security parameter k and returns a pair (pk, sk) consisting of a public key pk and matching secret key sk. The signing algorithm SIGN takes the secret key sk and a message M to produce a signature σ. We call σ produced in this way a *legal* signature of M with respect to pk. The (deterministic) verification algorithm VF takes the public key pk, a message M, and a candidate signature σ, and returns a bit. We say that σ is a *valid* signature of M with respect to pk if $\text{VF}(pk, M, \sigma) = 1$. We require that for all messages M, any legal signature of M is valid.

Security of a digital signature scheme. The notion of security is unforgeability under chosen-message attack, as per [GMR88] for the standard model, and [BR93b] for the random oracle model. Our presentation is unified and differs a little from the usual one for consistency

with the framework used in later definitions. We consider a *uf-cma-adversary* F who takes input pk and is allowed to make certain types of queries. Associated to F and a value k of the security is an experiment, shown in Figure 1, which returns a bit w indicating whether or not F succeeded. Within this experiment, F is executed, and replies to its queries are returned according to the rules rules in Figure 1. For simplicity we assume that F halts immediately after making a verify query, and that it makes at most one such query. We let

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

be the probability that the bit w returned by the experiment equals 1. We say that DS is *unforgeable under chosen-message attack* if the function $\mathbf{Adv}^{\text{uf-cma}}_{\text{DS},F}(\cdot)$ is negligible for all uf-cma-adversaries F whose time-complexity is $\text{poly}(k)$.

The experiment and definition hold either in the random oracle model or the standard one, the difference being whether or not the $\mathsf{HO}(\cdot,\cdot)$ instruction is invoked. In the experiment, a table $\text{HT}[\cdot,\cdot]$ is maintained with the intended semantics that $\text{HT}[R,x]$ is the response to $\mathsf{HO}(R,x)$. We note that in the experiment, $\mathsf{HO}(\cdot,\cdot)$ queries can be made either directly by the adversary, or indirectly by algorithms invoked to answer other adversary queries. The indirect queries are "under the rug," but it is understood that the experiment responds to them by executing the procedure indicated for a response to a $\mathsf{HO}(R,x)$ query in the table of Figure 1.

REKEYED DIGITAL SIGNATURE SCHEMES. A rekeyed (digital) signature scheme RKDS = (PKG, SKG, SIGN, VF) is specified by four polynomial-time algorithms. The *primary-key generation* algorithm PKG takes input a security parameter k and returns a pair (pk, sk) consisting of a *(primary) public key* pk and matching *primary secret key* sk. The *session-key generation* algorithm SKG takes input the primary secret key sk and an integer $j \geq 1$ representing a session number, and returns a session key $\overline{sk}$ for session j. The signing algorithm SIGN takes a session secret key $\overline{sk}$ and a message M to produce a signature σ. We call σ produced in this way a *legal* signature of M with respect to pk. The (deterministic) verification algorithm VF takes the public key pk, a message M, and a candidate signature σ, and returns a pair (d,i), where d is a bit and $i \geq 1$ is an integer. We say that σ is a *valid* signature of M with respect to pk and relative to session i if $\text{VF}(pk, M, \sigma) = (1, i)$. We require that for all messages M, any legal signature of M is valid.

SECURITY OF A REKEYED DIGITAL SIGNATURE SCHEME. We introduce a notion of unforgeability under breakin attack that is modeled on the notions and definitional ideas of [BM99, BR94, BR96]. We first present the formal definition and then discuss it.

We consider a *uf-ba-adversary* F who takes input pk and is allowed to make certain types of queries. Associated to F and a value k of the security is an experiment, shown in Figure 2, which returns a bit w indicating whether or not F succeeded. Within this experiment, F is executed, and replies to its queries are returned according to the rules prescribed in Figure 2. For simplicity we assume that F halts immediately after making a verify query, and that it makes at most one such query. We let

$$\mathbf{Adv}^{\text{uf-ba}}_{\text{RKDS},F}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\text{uf-ba}}_{\text{RKDS},F}(k) = 1\,\right]$$

be the probability that the bit w returned by the experiment equals 1. We say that RKDS is *unforgeable under breakin attack* if the function $\mathbf{Adv}^{\text{uf-ba}}_{\text{RKDS},F}(\cdot)$ is negligible for all uf-ba-adversaries F whose time-complexity is $\text{poly}(k)$.

The model and definition are for both the standard and the random oracle models. As above, in the experiment, a table $\text{HT}[\cdot,\cdot]$ is maintained with the intended semantics that $\text{HT}[R,x]$ is the response to $\mathsf{HO}(R,x)$, and it is understood that $\mathsf{HO}(\cdot,\cdot)$ queries made indirectly, by algorithms invoked to answer other adversary queries, are answered by executing the procedure indicated for

Experiment $\mathbf{Exp}^{\text{uf-ba}}_{\text{RKDS},F}(k)$
Begin

1. $(pk, sk) \xleftarrow{R} \text{PKG}(k)$; $j \leftarrow 0$

2. Run F on input pk, responding to induced queries as indicated in the table below:

Query	**Response**
$\mathsf{HO}(R, x)$	If $\text{HT}[R, x] = \perp$ then $\text{HT}[R, x] \xleftarrow{R} R$ Return $\text{HT}[R, x]$
start	$j \leftarrow j + 1$; $sk_j \leftarrow \text{SKG}(sk, j)$; $SM_j \leftarrow \emptyset$
$\mathsf{sign}(M, i)$	If $1 \le i \le j$ then $\sigma \leftarrow \text{SIGN}(sk_i, M)$; $SM_i \leftarrow SM_i \cup \{M\}$ Else $\sigma \leftarrow \perp$ Return σ to F
$\mathsf{breakin}(i)$	If $1 \le i \le j$ then $s \leftarrow sk_i$; $B \leftarrow B \cup \{i\}$ Else $s \leftarrow \perp$ Return s to F
$\mathsf{verify}(M, \sigma)$	$(d, i) \leftarrow \text{VF}(pk, M, \sigma)$ If ($d = 1$ AND $1 \le i \le j$ AND $i \notin B$ AND $M \notin SM_i$) Then $w \leftarrow 1$ else $w \leftarrow 0$

3. Return the bit w computed in the response to F's verify query above

End

Figure 2: The experiment measuring the success of uf-ba-adversary F in its attack on two-tier digital signature scheme RKDS = (PKG, SKG, SIGN, VF). The table indicates the types of oracle queries that F can make and how the experiment responds to them.

a response to a $\mathsf{HO}(R, x)$ query in the table of Figure 2.

DISCUSSION. At any point in the experiment, sessions $1, \ldots, j$ are active. The adversary can create a new session at any time by issuing a start query, the result of which is that j gets incremented by 1 and a session key sk_j is created for the new session. The sign query models a chosen-message attack: the adversary indicates a message M, and index i of an active session, and will be returned a signature of M under the session key corresponding to session i. The breakin query models compromise of a session: the adversary indicates the index i of an active session and is returned the corresponding session key sk_i. The verify query models an attempt at forgery: the adversary indicates a message M and candidate signature σ, and the experiment sets $w = 1$ if M, σ is a successful forgery, and $w = 0$ otherwise. Letting $(d, i) = \text{VF}(pk, M, \sigma)$, "successful" means that that σ is a valid signature of M with respect to pk, meaning $d = 1$, and also that it was not legitimately obtained, meaning that the session i was not compromised and no $\mathsf{sign}(M, i)$ query had been issued by the adversary.

Algorithm $\textsc{Pkg}(k)$ $(pk, sk) \leftarrow \textsc{kg}_1(k)$ Return (pk, sk)	Algorithm $\textsc{Skg}(sk, j)$ $(pk_2, sk_2) \stackrel{R}{\leftarrow} \textsc{kg}_2(k)$ $cert \leftarrow \textsc{sign}_1(sk, (j, pk_2))$ $\overline{sk}_j \leftarrow (j, pk_2, cert, sk_2)$ Return $\overline{sk}_j$
Algorithm $\textsc{Sign}(\overline{sk}, M)$ Parse $\overline{sk}$ as $(j, pk_2, cert, sk_2)$ $\sigma \leftarrow \textsc{sign}_2(sk_2, M)$ $\overline{\sigma} \leftarrow (j, pk_2, cert, \sigma)$ Return $\overline{\sigma}$	Algorithm $\textsc{Vf}(pk, M, \overline{\sigma})$ Parse $\overline{\sigma}$ as $(j, pk_2, cert, \sigma)$ $d_1 \leftarrow \textsc{vf}_1(pk, (j, pk_2), cert)$ $d_2 \leftarrow \textsc{vf}_2(pk_2, M, \sigma)$ $d \leftarrow d_1 \wedge d_2$ Return (d, j)

Figure 3: Self-certification scheme $\textsc{rkds} = (\textsc{Pkg}, \textsc{Skg}, \textsc{Sign}, \textsc{Vf})$ associated to digital signature schemes $\textsc{ds}_1 = (\textsc{kg}_1, \textsc{sign}_1, \textsc{vf}_1)$ and $\textsc{ds}_2 = (\textsc{kg}_2, \textsc{sign}_2, \textsc{vf}_2)$.

3 The self-certification scheme

Construction. Suppose $\textsc{ds}_1 = (\textsc{kg}_1, \textsc{sign}_1, \textsc{vf}_1)$ and $\textsc{ds}_2 = (\textsc{kg}_2, \textsc{sign}_2, \textsc{vf}_2)$ are digital signature schemes. The *self-certification scheme* associated to them is the rekeyed digital signature scheme $\textsc{rkds} = (\textsc{Pkg}, \textsc{Skg}, \textsc{Sign}, \textsc{Vf})$ whose constituent algorithms are described in Figure 3. If the constituent schemes $\textsc{ds}_1, \textsc{ds}_2$ do not use random oracles, neither does the associated self-certification scheme. If either of the constituent schemes does use its random oracle calls, however, the new scheme can implement them by issuing its own random oracle calls. Care must however be taken with regard to exactly how this is done. We will do it as follows. A $\mathsf{HO}(R, x)$ instruction issued by any algorithm X of $\textsc{ds}_1$ is replaced by a $\mathsf{HO}(R, 1\|x)$ instruction in any RKDS algorithm that calls X. Similarly a $\mathsf{HO}(R, x)$ instruction issued by any algorithm X of $\textsc{ds}_2$ is replaced by a $\mathsf{HO}(R, 2\|x)$ instruction in any RKDS algorithm that calls X.

Security. The following theorem says that the self-certification scheme meets our strong notion of security for rekeyed schemes assuming the constituent digital signature schemes are secure in the standard sense.

Theorem 3.1 Let RKDS be the self-certification rekeyed digital signature scheme associated to standard digital signature schemes $\textsc{ds}_1$ and $\textsc{ds}_2$. If $\textsc{ds}_1$ and $\textsc{ds}_2$ are unforgeable under chosen-message attack, then RKDS is unforgeable under breakin attack. ▮

The following lemma indicates the concrete security of the underlying reduction. Theorem 3.1 is an obvious corollary of this lemma.

Lemma 3.2 Let RKDS be the self-certification rekeyed digital signature scheme associated to digital signature schemes $\textsc{ds}_1$ and $\textsc{ds}_2$. To any uf-ba-adversary F we can associate uf-cma-adversaries F_1, F_2 such that for all k

$$\mathbf{Adv}^{\text{uf-ba}}_{\textsc{rkds},F}(k) \;\leq\; \mathbf{Adv}^{\text{uf-cma}}_{\textsc{ds}_1,F_1}(k) + \mathbf{Q}_{F,\mathsf{start}}(k) \cdot \mathbf{Adv}^{\text{uf-cma}}_{\textsc{ds}_2,F_2}(k)\ . \qquad (1)$$

Furthermore

$$\begin{array}{rcl|rcl} \mathbf{T}_{F_1}(k) & \leq & \mathbf{T}_F(k) & \mathbf{T}_{F_2}(k) & \leq & \mathbf{T}_F(k) \\ \mathbf{Q}_{F_1,\mathsf{sign}}(k) & \leq & \mathbf{Q}_{F,\mathsf{start}}(k) & \mathbf{Q}_{F_2,\mathsf{sign}}(k) & \leq & \mathbf{Q}_{F,\mathsf{sign}}(k) \end{array}$$

for all k. ∎

The basic intuition behind this lemma is standard, being the intuition underlying the idea of certification. Namely, success in attacking the self-certification scheme can come only via one of the following: forgery of a certificate for a pair (i, pk_2), or forgery of a signature of a message M under a key pk_2 already certified under pk. Lemma 3.2 attests to the fact that a reductionist proof based on this idea goes through in the strong adversarial models we have defined. Equation (1) shows, however, that the roles played by the two underlying signature schemes are not equal, meaning there is some loss of concrete security. A proof of Lemma 3.2 is in Appendix A.

4 The rekeyed iterated-root scheme

SPECIAL MODULII. We say that N is a $(3, 7; 8)$-modulus if $N = p_1 p_2$ is the product of odd primes $p_1 < p_2$ such that $p_1 \equiv 3 \pmod 8$ and $p_2 \equiv 7 \pmod 8$ (cf. [Wil80, Blu82, GMR88]). It is convenient to let

$$\mathrm{SQ}_N^i(v) \;=\; v^{2^i} \bmod N$$

for $i \geq 0$ and any $v \in \mathsf{Z}_N^*$. We let $\mathrm{ASQR}_N(u)$ denote the set of all square roots of a quadratic residue $u \in \mathsf{Z}_N^*$. This set has size exactly four, and contains exactly one quadratic residue, which is denoted $\mathrm{SQR}_N^1(u)$. For $i \geq 2$ we define $\mathrm{SQR}_N^i(\cdot)$ recursively, namely

$$\mathrm{SQR}_N^i(u) \;=\; \mathrm{SQR}_N^{i-1}(\mathrm{SQR}_N^1(u))$$

for any quadratic residue u.

FACTORING. An algorithm MKG is said to be a $(3, 7; 8)$-*modulus generator* if on input security parameter k it returns a triple (N, p_1, p_2) such that N is a $(3, 7; 8)$-modulus N satisfying $2^{k-1} \leq N < 2^k$, and $p_1 < p_2$ are primes such that $N = p_1 p_2$. (There are many such generators, differing in the distributions of their outputs. For example the canonical generator picks the primes p_1, p_2 at random and of approximately equal length subject to the necessary constraints and sets $N = p_1 p_2$. However, all our results are relative to an arbitrary generator.) Associated to a *factoring-adversary* A and value k of the security parameter is the experiment

$$\begin{array}{l} \texttt{Experiment } \mathbf{Exp}^{\mathrm{factor}}_{\mathrm{MKG},A}(k) \\ (N, p_1, p_2) \xleftarrow{R} \mathrm{MKG}(k)\ ;\ (\overline{p}_1, \overline{p}_2) \leftarrow A(N) \\ \text{If } (p_1 = \overline{p}_1 \text{ and } p_2 = \overline{p}_2) \text{ then return 1 else return 0} \end{array}$$

We let

$$\mathbf{Adv}^{\mathrm{factor}}_{\mathrm{MKG},A}(k) \;=\; \Pr\left[\mathbf{Exp}^{\mathrm{factor}}_{\mathrm{MKG},A}(k) = 1\right].$$

We say that *factoring is hard relative to* MKG if the function $\mathbf{Adv}^{\mathrm{factor}}_{\mathrm{MKG},A}(\cdot)$ is negligible for all factoring-adversaries A whose time-complexity is $\mathrm{poly}(k)$.

NUMBER THEORY. We will use this in the rekeyed scheme and in the proofs. If p is a prime and x is an integer relatively prime to p then we define the *quadratic character* of $x \bmod p$, denoted $\mathsf{QC}_p(x)$, to be the Legendre symbol of $x \bmod p$, meaning it is $+1$ if x is a quadratic residue mod

p and -1 otherwise. If $N = p_1 p_2$ is the product of primes $p_1 < p_2$, and x is an integer relatively prime to N, then we define the quadratic character of x mod N, denoted $\mathsf{QC}_N(x)$, to be the vector $[\mathsf{QC}_{p_1}(x), \mathsf{QC}_{p_2}(x)]$. For any $Q_1, Q_2 \in \{-1, +1\}$ we define the *quadratic character class* of $[Q_1, Q_2]$, denoted $\mathsf{QCC}_N[Q_1, Q_2]$, to be the set of integers modulo N with quadratic character $[Q_1, Q_2]$, namely

$$\mathsf{QCC}_N[Q_1, Q_2] \;=\; \{\, x \in \mathsf{Z}_N^* \;:\; \mathsf{QC}_N(x) = [Q_1, Q_2] \,\}\,.$$

The sets

$$\mathsf{QCC}_N[+1,+1],\;\; \mathsf{QCC}_N[+1,-1],\;\; \mathsf{QCC}_N[-1,+1],\;\; \mathsf{QCC}_N[-1,-1]$$

form a partition of Z_N^*, and each of these sets has size $\varphi(N)/4 = (p_1 - 1)(p_2 - 1)/4$. Note $\mathsf{QCC}_N[+1,+1]$ is the set of quadratic residues modulo N.

Some well-known facts are the following. For any $Q_1, Q_2 \in \{-1, +1\}$ and $y \in \mathsf{QCC}_N[Q_1, Q_2]$, the product yy mod N is in $\mathsf{QCC}_N[+1,+1]$. For any $Q_1, Q_2 \in \{-1, +1\}$, any $y \in \mathsf{QCC}_N[Q_1, Q_2]$ and $u \in \mathsf{QCC}_N[1,1]$, if we let $P_{N,y}(u) = yu \bmod N$, then the map $P_{N,y}$ is a bijection from $\mathsf{QCC}_N[1,1]$ to $\mathsf{QCC}_N[Q_1, Q_2]$.

We define the following, where N is a $(3, 7; 8)$-modulus:

$$\begin{array}{rclcrcl} \mathrm{QCR}_N[+1,+1] &=& 1 & \qquad & \mathrm{QCR}_N[-1,-1] &=& -1 \\ \mathrm{QCR}_N[-1,+1] &=& 2 & & \mathrm{QCR}_N[+1,-1] &=& -2\,. \end{array}$$

We call these the *quadratic character representatives* since for all $Q_1, Q_2 \in \{-1, +1\}$ we have

$$\mathsf{QC}_N(\mathrm{QCR}_N[Q_1, Q_2]) \;=\; [Q_1, Q_2]\,.$$

Moreover, for any element $y \in \mathsf{Z}_N^*$,

$$\mathsf{QC}_N(y \cdot \mathrm{QCR}_N[\mathsf{QC}_N(y)] \bmod N) \;=\; [1,1]\,.$$

The iterated-root scheme. Our rekeyed signature scheme is based on a special case of the Ong-Schnorr (OS) [OS90], which we call the iterated-root scheme IRDS. In the OS scheme, the number of quadratic residues in the public key varies, being one of the parameters of the scheme. In the iterated-root scheme IRDS, we fix this number to 1 in order to obtain shorter public and secret keys, while preserving the overall efficient of the scheme. The iterated-root scheme IRDS can also be seen as a generalization of the Micali signature scheme [Mic94], as done in [AR00] in their construction of a forward-secure signature scheme.

Let us recall the description of the iterated-root scheme IRDS $= (\mathrm{KG}, \mathrm{SIGN}, \mathrm{VF})$. It works as follows. The key generation algorithm, KG, gets as input a security parameter k and returns a pair (sk, pk). The secret key sk consists of a $(3, 7; 8)$-modulus N, a random element S in Z_N^* and the challenge length h. The public key pk consists of the $(3, 7; 8)$-modulus N and the quadratic residue $U = \mathrm{SQ}_N^h(S^{-1}) = S^{2^{-h}}$. The signing algorithm, SIGN, gets as input the secret key $sk = (N, S, h)$ and a message M and outpus the signature (C, Z) for M. It does so by first picking a random quadratic residue Y in Z_N^*, making the challenge $C = \mathsf{HO}(\{0,1\}^h, \langle Y, M\rangle)$, and then finally computing one of the four square roots Z in $\mathrm{SQR}_N^h(Y \cdot U^{-\langle C\rangle})$. The verifying algorithm, VF, gets as input a message M, the $(3, 7; 8)$-modulus N and signature (C, Z), and checks whether the latter is a valid signature for M by re-computing the challenge associated with the value Z, the message M and the public-key pk. Please refer to Figure 4 for a full description of the base scheme.

One observation worth making here is one made in [AR00] which states that the secret key S and the value Z need not be quadratic residues in Z_N^*. In fact, there is no way for the verifier to check that since telling square residues apart from non-square residues in Z_N^* is hard problem in itself if one does not know the prime factors of N.

Algorithm KG(k)	Algorithm SIGN(sk, M)	Algorithm VF(pk, M, σ)
$(N, p_1, p_2) \leftarrow$ MKG(k) $h \leftarrow H(k)$ $S \xleftarrow{R} \mathsf{Z}_N^*$ $U \leftarrow \mathrm{SQ}_N^h(S^{-1})$ $pk \leftarrow (N, U)$ $sk \leftarrow (N, S, h)$ Return (pk, sk)	Parse sk as (N, h, S) $R \xleftarrow{R} \mathsf{Z}_N^*$ $Y \leftarrow \mathrm{SQ}_N^h(R)$ $C \leftarrow \mathsf{HO}(\{0,1\}^h, \langle Y, M\rangle)$ Parse C as $C_1 \| \cdots \| C_h$ $e \leftarrow \sum_{l=1}^{h} C_l \cdot 2^{l-1}$ $Z \leftarrow R \cdot S^e \bmod N$ $\sigma \leftarrow (C, Z)$ Return σ	Parse pk as (N, U) $h \leftarrow H(k)$ Parse σ as (C, Z) Parse C as $C_1 \| \cdots \| C_h$ $Y \leftarrow \mathrm{SQ}_N^h(Z)$ $e \leftarrow \sum_{l=1}^{h} C_l \cdot 2^{l-1}$ $Y \leftarrow Y \cdot U^e \bmod N$ $\overline{C} \leftarrow \mathsf{HO}(\{0,1\}^h, \langle Y, M\rangle)$ If $(C = \overline{C})$ then return 1 Else return 0

Figure 4: The iterated-root signature scheme IRDS = (KG, SIGN, VF) associated to $(3, 7; 8)$-modulus generator MKG and parameter H.

The security of this scheme is captured by the following lemma, whose proof can be easily adapted from that given in [MR02] for the Micali signature scheme. It also follows as particular case of the proof of security for the Abdalla-Reyzin forward-secure signature scheme [AR00] by making $T = 1$ (in which case forward security implies normal security).

Lemma 4.1 Let IRDS be the iterated-root digital signature scheme associated to $(3, 7; 8)$-modulus generator MKG and parameter H. To any uf-cma-adversary F we can associate a factoring-adversary A such that for all k

$$\mathbf{Adv}^{\text{uf-cma}}_{\text{IRDS},F}(k) \;\leq\; 2^{-H(k)} + \sqrt{2\mathbf{Q}_{F,\mathsf{HO}}(k) \cdot \mathbf{Adv}^{\text{factor}}_{\text{MKG},A}(k)} + 4\mathbf{Q}_{F,\mathsf{HO}}(k)\mathbf{Q}_{F,\mathsf{sign}}(k)/|Z_N^*| \,. \quad (2)$$

Furthermore $\mathbf{T}_A(k) \leq 2 \cdot \mathbf{T}_F(k) + O(k^3 + k^2 H(k))$ for all k. ▌

THE REKEYED ITERATED-ROOT SCHEME. Let MKG be a $(3, 7; 8)$-modulus generator, and let $H(\cdot)$ be a positive-integer valued function of the security parameter. (Read H as "height".) The *rekeyed iterated-root signature scheme* associated to MKG, H is the rekeyed digital signature scheme RKIRDS = (PKG, SKG, SIGN, VF) whose constituent algorithms are described in Figure 5.

As the code indicates, the primary public key is a $(3, 7; 8)$-modulus N, while the primary secret key includes the primes $p_1 < p_2$ such that $N = p_1 p_2$. The session-key generation algorithm, given the primary secret key and a session index j, uses the random oracle to specify a quadratic residue U to play the role of a *secondary public key* for session j. Because the output V of the random oracle is not guaranteed to be a a quadratic residue in Z_N^*, we first compute the quadratic character of V and the factor f that, if multiplying V, results in a quadratic residue U. This is only possible due to the fact p_1 and p_2 are part of primary secret key. After this step, we then compute (using p_1 and p_2) one of 2^h-roots of U and set the secret key for session j to be (N, h, j, S, f). Signing is done as in the base iterated-root scheme IRDS. The only difference is that, in addition to C and Z, the signature also include the session index j and the factor f computed during the session key generation. The latter is only included in the signature for efficient purposes. Otherwise, the verifier would need to check the signature (j, C, Z) with respect to all values in $\{V, -V, 2V, -2V\}$,

Algorithm $\textsc{Pkg}(k)$ $(N, p_1, p_2) \leftarrow \textsc{mkg}(k)$ $pk \leftarrow N$; $sk \leftarrow (N, p_1, p_2)$ Return (pk, sk)	Algorithm $\textsc{Skg}(sk, j)$ Parse sk as (N, p_1, p_2) $h \leftarrow H(k)$ $V \leftarrow \mathsf{HO}(\mathsf{Z}_N^*, \langle j \rangle)$ $f \leftarrow \mathrm{QCR}_N[\mathsf{QC}_N(V)]$ $U \leftarrow V \cdot f \bmod N$ $T \leftarrow \mathrm{SQR}_N^h(U^{-1})$ $S \xleftarrow{R} \mathrm{ASQR}_N(T)$ $\overline{sk} \leftarrow (N, h, j, S, f)$ Return $\overline{sk}$
Algorithm $\textsc{Sign}(\overline{sk}, M)$ Parse $\overline{sk}$ as (N, h, j, S, f) $R \xleftarrow{R} \mathsf{Z}_N^*$; $Y \leftarrow \mathrm{SQ}_N^h(R)$ $C \leftarrow \mathsf{HO}(\{0,1\}^h, \langle j, Y, M \rangle)$ Parse C as $C_1 \| \cdots \| C_h$ $e \leftarrow \sum_{l=1}^{h} C_l \cdot 2^{l-1}$ $Z \leftarrow R \cdot S^e \bmod N$ $\overline{\sigma} \leftarrow (j, f, C, Z)$ Return $\overline{\sigma}$	Algorithm $\textsc{Vf}(pk, M, \overline{\sigma})$ Parse pk as N $h \leftarrow H(k)$ Parse $\overline{\sigma}$ as (j, f, C, Z) Parse C as $C_1 \| \cdots \| C_h$ $Y \leftarrow \mathrm{SQ}_N^h(Z)$ $V \leftarrow \mathsf{HO}(\mathsf{Z}_N^*, \langle j \rangle)$ $U \leftarrow V \cdot f \bmod N$ $e \leftarrow \sum_{l=1}^{h} C_l \cdot 2^{l-1}$ $Y \leftarrow Y \cdot U^e \bmod N$ $\overline{C} \leftarrow \mathsf{HO}(\{0,1\}^h, \langle j, Y, M \rangle)$ If $(C = \overline{C})$ then return 1 Else return 0

Figure 5: The rekeyed iterated-root signature scheme $\textsc{rkirds} = (\textsc{Pkg}, \textsc{Skg}, \textsc{Sign}, \textsc{Vf})$ associated to $(3, 7; 8)$-modulus generator $\textsc{mkg}$ and parameter H.

---

where $V = \mathsf{HO}(\mathsf{Z}_N^*, \langle j \rangle)$, since it does not know which one of them is the quadratic residue. f allows the verifier to know the factor multiplying the output of the random oracle. Aside from this, the verification is similar to that of the base iterated-root scheme $\textsc{irds}$.

Security. The following theorem states that the rekeyed iterated-root signature scheme meets the strong notion of security for rekeyed schemes assuming factoring $(3, 7; 8)$-modulus is hard.

Theorem 4.2 Let $\textsc{rkirds}$ be the rekeyed iterated-root digital signature scheme associated to $(3, 7; 8)$-modulus generator $\textsc{mkg}$ and parameter H. If factoring is hard relative to $\textsc{mkg}$ then $\textsc{rkirds}$ is unforgeable under breakin attack. ▌

The following lemma indicates the concrete security of the underlying reduction. Theorem 4.2 is an obvious corollary of this lemma.

Lemma 4.3 Let RKIRDS be the rekeyed iterated-root digital signature scheme associated to $(3, 7; 8)$-modulus generator MKG and parameter H. To any uf-ba-adversary F we can associate a factoring-adversary A such that for all k

$$\begin{aligned}&\mathbf{Adv}^{\text{uf-ba}}_{\text{RKIRDS},F}(k)\\ &\quad\leq\quad \mathbf{Q}_{F,\mathsf{start}}\left(2^{-H(k)}+\sqrt{2\mathbf{Q}_{F,\mathsf{HO}}(k)\cdot\mathbf{Adv}^{\text{factor}}_{\text{MKG},A}(k)}+4\mathbf{Q}_{F,\mathsf{HO}}(k)\mathbf{Q}_{F,\mathsf{sign}}(k)/|Z_N^*|\right)\ .\end{aligned}\tag{3}$$

Furthermore $\mathbf{T}_A(k)\leq 2\cdot\mathbf{T}_F(k)+O(k^3+k^2H(k))$ for all k. ▌

Rather than prove this result from scratch, we use Lemma 4.1. The following lemma says that it is possible to reduce the security of the rekeyed iterated-root scheme to the security of the iterated-root scheme, although there is a loss of a factor $\mathbf{Q}_{F,\mathsf{start}}(k)$ in the security. Lemma 4.3 follows directly from Lemma 4.4 and Lemma 4.1, so it suffices to prove Lemma 4.4.

Lemma 4.4 Let MKG be a $(3, 7; 8)$-modulus generator and let H be the height parameter. Let IRDS be the iterated-root digital signature scheme associated to MKG, H, and let RKIRDS be the rekeyed iterated-root digital signature scheme associated to MKG, H. To any uf-ba-adversary F we can associate a uf-cma-adversary F_1 such that for all k

$$\mathbf{Adv}^{\text{uf-ba}}_{\text{RKIRDS},F}(k)\quad\leq\quad\mathbf{Q}_{F,\mathsf{start}}(k)\mathbf{Adv}^{\text{uf-cma}}_{\text{IRDS},F_1}(k)\ .\tag{4}$$

Furthermore

$$\begin{aligned}\mathbf{T}_{F_1}(k)&\leq\mathbf{T}_F(k)\\ \mathbf{Q}_{F_1,\mathsf{sign}}(k)&\leq\mathbf{Q}_{F,\mathsf{sign}}(k)\\ \mathbf{Q}_{F_1,\mathsf{HO}}(k)&\leq\mathbf{Q}_{F,\mathsf{HO}}(k)\end{aligned}$$

for all k. ▌

5 Comparison of two rekeyed signature schemes

We provide in this section a detailed cost and efficiency comparison between an RSA-based version of the self-certification scheme in Section 3 and the rekeyed iterated-root scheme in Section 4. Our comparisons are presented in terms of the security parameter k of the rekeyed signature scheme (i.e., the size of modulus N), the length l_e of the RSA encryption exponent e and the number of challenge length h used in rekeyed iterated-root scheme. For security reasons, we assume the length of RSA decryption exponent parameter d to be in the order of k. The signing and verification time is expressed in terms of the total number of modular multiplications involved in these operations. We do not take into account, however, the size of session index as it equally affects both schemes. Table 1 describes in detail how these two rekeyed scheme compare to one another in terms of signing time, verification time, signature size, session key size, primary secret key size, and public key size.

Typical values used in practice for parameters k and h are 1024 and 128, respectively. The first choice of value makes the success probability of all well-known factoring algorithms negligible. The second choice of value makes the success probability of guessing correctly the challenge C in the signature of the rekeyed iterated-root scheme negligible. Having these values in mind, one can see that the rekeyed iterated-root scheme performs better than the RSA-based self-certification scheme in almost every aspect, except for the verification time, if one uses small verification exponents for RSA.

Parameter	Self-Certification scheme	rekeyed iterated-root scheme
Signing time	$2k$	$3h$
Verification time	$2l_e$	$3h+1$
Signature size	$3k+l_e$	$k+h+2$
Session key size	$3k+l_e$	$2k+\log h+2$
Primary secret key size	$4k+l_e$	$2k$
Public key size	$k+l_e$	k

Table 1: Comparison of an RSA-based self-certification scheme and the rekeyed iterated-root scheme.

References

[AMN01] Michel Abdalla, Sara K. Miner, and Chanathip Namprempre. Forward-secure threshold signature schemes. In David Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 441–456, San Francisco, CA, USA, April 8–12, 2001. Springer-Verlag, Berlin, Germany.

[And00] Ross Anderson. Two remarks on public-key cryptology. Manuscript. Relevant material presented by the author in an invited lecture at the 4th ACM Conference on Computer and Communications Security, CCS 1997, Zurich, Switzerland, April 1–4, 1997, September 2000.

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129, Kyoto, Japan, December 3–7, 2000. Springer-Verlag, Berlin, Germany.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979.

[Blu82] Manuel Blum. Coin flipping by telephone. In *Proc. IEEE Spring COMPCOM*, pages 133–137, 1982.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448, Santa Barbara, CA, USA, August 15–19, 1999. Springer-Verlag, Berlin, Germany.

[BR93a] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BR93b] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Santa Barbara, CA, USA, August 22–26, 1994. Springer-Verlag, Berlin, Germany.

[BR96] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution — the three party case. In *28th Annual ACM Symposium on Theory of Computing*, pages 57–66, Philadephia, Pennsylvania, USA, May 22–24, 1996. ACM Press.

[Des94] Yvo Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July/August 1994.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA, August 20–24, 1990. Springer-Verlag, Berlin, Germany.

[DS81] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the Association for Computing Machinery*, 24(8):533–536, 1981.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer-Verlag, Berlin, Germany.

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

[HJJ+97] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *ACM CCS 97: 4th Conference on Computer and Communications Security*, pages 100–110, Zurich, Switzerland, April 1–4, 1997. ACM Press.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.

[Kra00] Hugo Krawczyk. Simple forward-secure signatures from any signature scheme. In *ACM CCS 00: 7th Conference on Computer and Communications Security*, pages 108–115, Athens, Greece, November 1–4, 2000. ACM Press.

[Mic94] Silvio Micali. A secure and efficient digital signature algorithm. Technical Memo MIT/LCS/TM-501b, Massachusetts Institute of Technology, Laboratory for Computer Science, April 1994.

[MMM02] Tal Malkin, Daniele Micciancio, and Sara Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 400–417, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer-Verlag, Berlin, Germany.

[MR99] Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. In Rainer Baumgart, editor, *International Exhibition and Congress on Network Security – CQRE'99*, volume 1740 of *Lecture Notes in Computer Science*, pages 167–182, Dsseldorf, Germany, November 30 – December 2, 1999. Springer-Verlag, Berlin, Germany.

[MR01] Philip D. MacKenzie and Michael K. Reiter. Networked cryptographic devices resilient to capture. In *2001 IEEE Symposium on Security and Privacy*, pages 12–25, Oakland, CA, May 2001. IEEE Computer Society Press.

[MR02] Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. *Journal of Cryptology*, 15(1), 2002. Full version of [MR99].

[OS90] H. Ong and Claus-Peter Schnorr. Fast signature generation with a Fiat Shamir–like scheme. In Ivan Damgård, editor, *Advances in Cryptology – EUROCRYPT'90*, volume 473 of *Lecture Notes in Computer Science*, pages 432–440, Aarhus, Denmark, May 21–24, 1990. Springer-Verlag, Berlin, Germany.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[Sin] `http://www.singlesignon.net`.

[TT01] Wen-Guey Tzeng and Zhi-Jia Tzeng. Robust forward-secure signature schemes with proactive security. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 264–276, Cheju Island, South Korea, February 13–15, 2001. Springer-Verlag, Berlin, Germany.

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

A Proof of Lemma 3.2

The adversaries F_1 and F_2 are described, respectively, in Figures 6 and 7.

Adversary F_1 takes input a public key pk for digital signature scheme DS_1. It will run F on input pk until the latter makes a $\mathsf{verify}(M, \overline{\sigma})$ query. Parsing $\overline{\sigma}$ as $(i, pk_{2,i}, cert_i, \sigma)$, it bets that the forgery is due to forgery under pk of the certificate $cert_i$, and accordingly ends by making its own query $\mathsf{verify}((i, pk_2), cert)$. In order to run $F(pk)$, however, F_1 must be able to answer F's queries. It can do this based by choosing the session keys itself and certifying them via its own $\mathsf{sign}(\cdot)$ queries. Answers to $\mathsf{breakin}(\cdot)$ or $\mathsf{sign}(\cdot, \cdot)$ queries made by F are easily provided because F_1 is in possession of all session keys.

Adversary F_2 takes input a public key pk_2 for digital signature scheme DS_2. It picks a primary key pair (pk, sk) for DS_1, and will run F on input pk. It will answer F's queries in such a way that the secondary public-key $pk_{2,c}$ associated to a certain session c equals the public key pk_2 that it received as input. Parsing the signature $\overline{\sigma}$ from the $\mathsf{verify}(M, \overline{\sigma})$ query of F as $(i, pk_{2,i}, cert_i, \sigma)$, it hopes that $i = c$, in which case it bets that the forgery is due to forgery under pk_2 of the signature σ of M, and accordingly ends by making its own query $\mathsf{verify}(M, \sigma)$. The challenge for F_2 is to answer the queries of F and yet retain the possibility that $i = c$. Its strategy, standard in these types of situations, is to pick c at random, and give up if asked a $\mathsf{breakin}(c)$ query.

The analysis yielding Lemma 3.2 is standard and is omitted.

Adversary $F_1(pk)$
Begin

1. $j \leftarrow 0$

2. Run F on input pk, responding to induced queries as indicated in the table below:

Query	**Instructions for F_1**

$\mathsf{HO}(R, x)$	Parse x as $i \| y$ with $i \in \{1, 2\}$
	If $i = 1$ then make query $\mathsf{HO}(R, y)$ and obtain answer h
	Else
	  If $\mathrm{HT}_2[R, y] = \perp$ then $\mathrm{HT}_2[R, y] \xleftarrow{R} \{0,1\}^l$
	  $h \leftarrow \mathrm{HT}_2[R, y]$
	Return h to F

start	$j \leftarrow j + 1$; $(pk_{2,j}, sk_{2,j}) \xleftarrow{R} \mathrm{KG}_2(k)$ Make query $\mathsf{sign}((j, pk_{2,j}))$ and let $cert_j$ denote the answer obtained $\overline{sk}_j \leftarrow (j, pk_{2,j}, cert_j, sk_{2,j})$

$\mathsf{sign}(M, i)$	If (NOT $1 \le i \le j$) then return $\perp$ to F
	Else
	  $\sigma \leftarrow \mathrm{SIGN}_2(sk_{2,i}, M)$; $\overline{\sigma} \leftarrow (i, pk_{2,i}, cert_i, \sigma)$
	  Return $\overline{\sigma}$ to F

$\mathsf{breakin}(i)$	If (NOT $1 \le i \le j$) then return $\perp$ to F Else return sk_i to F
$\mathsf{verify}(M, \overline{\sigma})$	Parse $\overline{\sigma}$ as $(i, pk_2, cert, \sigma)$ Make query $\mathsf{verify}((j, pk_2), cert)$

End

Figure 6: Description of uf-cma-adversary F_1 attacking DS_1. It runs F as a subroutine. The table indicates the actions taken by F_1 in response to queries made (directly or indirectly) by F.

B Proof of Lemma 4.4

Proof Idea. We use a standard reduction argument. We first assume there exists an adversary F that breaks the security of the rekeyed iterated-root scheme RKIRDS. We then show how to construct an adversary F_1 that breaks the security of IRDS, using F as a sub-routine. F_1 starts by guessing a session c with respect to which F succeeds in outputting a forgery. It then sets the public key of that period to be related to its own so that a forgery for the rekeyed iterated-root scheme RKIRDS gets translated into a forgery for the iterated-root scheme IRDS. In doing so, F_1 has to respond the queries that F makes to its oracles. Since F_1 does not have access to these oracles, it has to simulate them in a way that does not differ from the real experiment defininig the security of the rekeyed iterated-root scheme RKIRDS.

Proof Details. The adversary F_1 against the iterated-root scheme IRDS is described inFigure 8. It takes as input a pair (N, U) consisting of a $(3, 7; 8)$-modulus N and random quadratic residue U

Adversary $F_2(\overline{pk}_2)$
Begin

1. $j \leftarrow 0$; $(pk, sk) \stackrel{R}{\leftarrow} \text{KG}_1(k)$

2. $c \stackrel{R}{\leftarrow} \{1, \ldots, \mathbf{Q}_{F,\mathsf{start}}(k)\}$; $pk_{2,c} \leftarrow \overline{pk}_2$

3. Run F on input pk, responding to induced queries as indicated in the table below:

Query	**Instructions for F_2**

$\mathsf{HO}(R, x)$	Parse x as $i \Vert y$ with $i \in \{1, 2\}$
	If $i = 2$ then make query $\mathsf{HO}(R, y)$ and obtain answer h
	Else
	  If $\text{HT}_1[R, y] = \perp$ then $\text{HT}_1[R, y] \stackrel{R}{\leftarrow} \{0,1\}^l$
	  $h \leftarrow \text{HT}_1[R, y]$
	Return h to F

start	$j \leftarrow j + 1$ If $(j \neq c)$ then $(pk_{2,j}, sk_{2,j}) \stackrel{R}{\leftarrow} \text{KG}_2(k)$ $cert_j \leftarrow \text{SIGN}_1(sk, (j, pk_{2,j}))$ If $(j \neq c)$ then $\overline{sk}_j \leftarrow (j, pk_{2,j}, cert_j, sk_{2,j})$

$\mathsf{sign}(M, i)$	If (NOT $1 \leq i \leq j$) then return $\perp$ to F
	Else
	  If $i = c$
	  Then make query $\mathsf{sign}(M)$ and let σ denote the answer obtained
	  Else $\sigma \leftarrow \text{SIGN}_2(sk_{2,i}, M)$
	  $\overline{\sigma} \leftarrow (i, pk_{2,i}, cert_i, \sigma)$
	  Return $\overline{\sigma}$ to F

$\mathsf{breakin}(i)$	If (NOT $1 \leq i \leq j$) then return $\perp$ to F If $(i = c)$ then ABORT Else return sk_i to F
$\mathsf{verify}(M, \overline{\sigma})$	Parse $\overline{\sigma}$ as $(i, pk_2, cert, \sigma)$ Make query $\mathsf{verify}(M, \sigma)$

End

Figure 7: Description of uf-cma-adversary F_2 attacking DS_2. It runs F as a subroutine. The table indicates the actions taken by F_2 in response to queries made (directly or indirectly) by F.

in Z_N^*. It starts by guessing the period with respect to which F will output a forgery and implicitly sets the public key pk_c for session c to be the pair (N, U_c), where $U_c = f_c U \bmod N$ and f_c is a random element in $\{1, -1, 2, -2\}$. It does so by returning U_c as the output to the hash query $\mathsf{HO}(Z_N^*, c)$. The factor f_c is due to the fact that the output of $\mathsf{HO}[Z_N^*, j]$ should be a random element in Z_N^*. For all other sessions j, it first picks random elements $S_j \in Z_N^*$ and $f_j \in \{1, -1, 2, -2\}$, and sets the secret key for that session to be $sk = (N, h, i, S_i, f_i)$. It also implicitly sets the public key pk_j to be the pair (N, U_j), where $U_j = \text{SQ}_N^h(S_i^{-1}) \cdot f_j$, by setting the output of the hash query

$\mathsf{HO}[Z_N^*, j]$ to be U_j.

When F makes queries of the type $\mathsf{HO}[\{0,1\}^h, x]$, F_1 first parses x to check the session to which this query refers. If it is for session c, it then forwards it to the hash oracle that was given to it. If the hash query is for a different session, it first checks whether the same has been asked before. If so, it returns the same answer that was given to the previous query. If not, then it first pick a random element in $\{0,1\}^h$, records it as the being the response to that query and returns it to F.

Signature queries are answered in a way similar to that used in the response of hash queries. That is, it first checks whether the signing query is with respect to session c. If so, then it uses its signing oracle to respond to this query. If not, then F_1 can use the secret key that it has generated for that period to generate the signature.

Whenever F makes a $\mathsf{breakin}(i)$ query, F_1 first checks that i is a valid index for a session. If i is not a valid index, then it returns $\perp$ to F. Otherwise, it either returns sk_i to F if $i \neq c$ or aborts the program since it does not know the value of the secret key sk_c.

Finally, when F makes a verification query $\mathsf{verify}(M, \overline{\sigma})$, F_1 parses the signature $\overline{\sigma}$ as (j, f_j, C, Z) and outputs σ as the forgery for M, by calling its verification oracle on input (M, σ).

Analysis. We first notice that F_1 succeeds whenever F does and when its guess for the time period with respect to which F outputs a forgery is correct. Moreover, F_1 makes at most $\mathbf{Q}_{F,\mathsf{sign}}(k)$ to its signing oracle and at most $\mathbf{Q}_{F,\mathsf{HO}}(k)$ to its hash oracle. The lemma follows easily by noticing that the running time $\mathbf{T}_{F_1}(k)$ is at most $\mathbf{T}_F(k)$.

`Adversary` $F_1(\overline{pk})$
`Begin`

1. Parse $\overline{pk}$ as (N, U) ; $pk = N$

2. $j \leftarrow 0$; $h \leftarrow H(k)$; $c \stackrel{R}{\leftarrow} \{1, \ldots, \mathbf{Q}_{F,\mathsf{start}}(k)\}$

3. Run F on input pk, responding to induced queries as indicated in the table below:

Query	**Instructions for** F_1

$\mathsf{HO}(\{0,1\}^h, x)$	Parse x as (j, Y, M)
	If $i = c$ then make query $\mathsf{HO}(\{0,1\}^h, (Y, M))$ and obtain answer C
	Else
	  If $\mathrm{HT}_1[\{0,1\}^h, x] = \perp$ then $\mathrm{HT}_1[\{0,1\}^h, x] \stackrel{R}{\leftarrow} \{0,1\}^h$
	  $C \leftarrow \mathrm{HT}_1[\{0,1\}^h, x]$
	Return C to F

$\mathsf{HO}(Z_N^*, i)$	If $\mathrm{HT}_2[Z_N^*, i] = \perp$ then
	  If $(i \neq c)$ then $S_i \stackrel{R}{\leftarrow} Z_N^*$; $V_i \leftarrow \mathrm{SQ}_N^h(S_i^{-1})$ else $V_i \leftarrow U$
	  $f_i \stackrel{R}{\leftarrow} \{1, -1, 2, -2\}$; $U_i \leftarrow f_i \cdot V_i \bmod N$; $sk_i \leftarrow (N, h, i, S_i, f_i)$
	  $\mathrm{HT}_2[Z_N^*, i] \leftarrow U_i$
	$U_i \leftarrow \mathrm{HT}_2[Z_N^*, i]$
	Return U_i to F

start	$j \leftarrow j + 1$; Make query $\mathsf{HO}(Z_N^*, j)$

$\mathsf{sign}(M, i)$	If (NOT $1 \leq i \leq j$) then return $\perp$ to F
	Else
	  If $i = c$ then
	  make query $\mathsf{sign}(M)$ and let σ denote the answer obtained
	  Parse σ as (C, Z) ; $\overline{\sigma} \leftarrow (j, f_j, C, Z)$
	  Else
	  $\overline{\sigma} \leftarrow \textsc{Sign}(sk_{2,i}, M)$
	  Return $\overline{\sigma}$ to F

$\mathsf{breakin}(i)$	If (NOT $1 \leq i \leq j$) then return $\perp$ to F If $(i = c)$ then ABORT else return sk_i to F
$\mathsf{verify}(M, \overline{\sigma})$	Parse $\overline{\sigma}$ as (j, f_j, C, Z) ; $\sigma \leftarrow (C, Z)$ Make query $\mathsf{verify}(M, \sigma)$

`End`

Figure 8: Description of uf-cma-adversary F_1 attacking IRDS. It runs F as a subroutine. The table indicates the actions taken by F_1 in response to queries made (directly or indirectly) by F.