Intrusion-Resilient Signatures, or Towards Obsoletion of Certificate Revocation

Gene Itkis Leonid Reyzin
Boston University
Computer Science
111 Cummington St.
Boston, MA 02215, USA
{itkis,reyzin}@bu.edu

April 30, 2002

Abstract

We propose a new notion of *intrusion-resilient signature schemes*, which generalizes and improves upon both forward-secure [And97, BM99] and key-insulated [DKXY02] signature schemes.

Specifically, similarly to the prior notions, time is divided into predefined time periods (e.g., days) so that a signature includes the time period number, changing which invalidates the signature. Also, similarly to key-insulated schemes, the user has two modules, *signer* and *home base*: the former generates signatures on its own, and the latter is needed only to help update the signer's key from one period to the next.

The main strength of intrusion-resilient schemes, as opposed to prior notions, is that they remain secure even after *arbitrarily many* compromises of *both* modules, as long as the compromises are not simultaneous. Moreover, even if the intruder does compromise both modules simultaneously, it will still be unable to generate any signatures for the previous time periods.

We provide an efficient intrusion-resilient signature scheme, provably secure in the random oracle model based on the strong RSA assumption.

We also show how such schemes can eliminate the need for certificate revocation in the case of on-line authentication.

1 Introduction

Key exposures appear to be unavoidable. Thus, limiting their impact is extremely important and is the focus of active research. While this issue applies to a wide range of security protocols, here we focus on digital signatures.

1.1 Previous Work

Forward Security. Forward-secure signature schemes [And97, BM99] preserve the security of past signatures even after the secret signing key has been exposed: time is divided into predefined time periods, with the signer updating its secret at the end of each time period; the adversary is unable to forge signatures for past periods even if it learns the key for the current one. In this model, nothing can be done about the future periods: once the adversary exposes the current secret, it has the same information as the signer.

Threshold and Proactive Security. Alternative approach explores the multi-party computation paradigm [Yao82, GMW87]: in threshold schemes [DF89], the signing key is somehow shared among a number of signers, and signature generation requires a distributed computation involving some subset of them. The adversary, however, cannot generate valid signatures as long as the number of compromised signers is less than some predetermined security parameter (smaller than the number of signers needed to generate a valid signature). Proactive schemes [OY91, HJJ+97] improve upon this model by allowing multiple corruptions of *all* signers, limiting only the number of *simultaneous* corruptions. Proactive forward-secure signatures considered in [AMN01] combine this with the advantages of forward-security.

Key-Insulated Security. The recently proposed model of Dodis, Katz, Xu and Yung [DKXY02] addresses the limitation of forward security: the adversary cannot generate signatures for the future (as well as past) time periods even after learning the current signing key[1]. This is accomplished via the use of two modules: a (possibly mobile) *signer*, and a (generally stationary) *home base*[2]. The signer has the secret signing key, and can generate signatures on its own. At the end of each time period, the signing key expires and the signer needs to *update* its keys by communicating with the home base and performing some local computations (the communication with the base is, in fact, limited to a single message from the base to the signer). Thus, although the signer's keys are vulnerable (because they are frequently accessed, and, moreover, because the signer may be mobile), key exposure is less valuable to the adversary, as it reveals only short-term keys. Perhaps the most compelling application of such a model is the example of a frequently traveling user, whose laptop (or handheld) is the signer, and office computer is the home base. (Alternative approaches with such applications in mind were proposed by [Mic96, Riv98, GPR98, LR00].) This model enables security that is not possible in ordinary or even forward-secure schemes: even if the signing key is compromised (for up to k time periods, for predetermined security parameter k), the adversary will be unable to forge signatures for *any* other time periods. (Notice that in forward-secure schemes model, signatures for any time period following a compromise are *necessarily* forgeable.)

1.2 Our Results: Intrusion-Resilient Security

1.2.1 Model

We define intrusion-resilient signature schemes to combine the benefits of the above three approaches. Namely, while maintaining the efficiency of non-interactive signature generation (not provided by threshold and proactive schemes), intrusion-resilient schemes preserve security of past *and* future time periods when *both* signer and base are compromised, though not simultaneously (not preserved by key-insulated and forward-secure schemes), and security of past time periods in the case of simultaneous compromise (not preserved by key-insulated[3] and most proactive schemes).

These points deserve some elaboration. To address potential compromise of the base key, [DKXY02] introduce a stronger version of key-insulated security, which requires that the base cannot generate signatures on its own. However, no security is guaranteed in [DKXY02] if the adversary manages to compromise *both* the base and the signer, even during different time periods. (In fact, the encryption scheme becomes completely insecure in such a case.) This is a serious limitation. If the user's key is compromised even just once, then the prudent thing to do would

[1] [DKXY02] primarily addresses encryption schemes, but includes definitions for signature schemes as well.

[2] The terms *user* and *secure device* are used in [DKXY02]; we find "signer" and "home base" to be more descriptive.

[3] Because the focus of [DKXY02] is on encryption schemes, and no non-trivial forward-secure encryption schemes are known today, it is, in a sense, by necessity that key-insulated notion of [DKXY02] does not provide forward security when all the secrets are compromised.

be to revoke the entire public key and erase the secrets of the home base. Otherwise, a single compromise of the home base would expose not only the future, but also all the past, messages.

In contrast, the salient feature of our new model is the guarantee that a compromise of the home base is *entirely inconsequential* as long as the signer's secret is not exposed at the same time. It thus has the benefits of proactive security. Moreover, our model retains the benefits of forward security even when all the secrets are compromised simultaneously.

To sum up, our intrusion-resilient model appears to provide the *maximum possible* security in the face of corruptions that occur.

1.2.2 Construction

In Section 3 we provide an efficient scheme that satisfies intrusion-resilience. This scheme is as efficient for signing and verifying as the signature scheme of Guillou-Quisquater [GQ88], requiring just two modular exponentiations with short exponents for both signing and verifying. This is as or more efficient than many of the ordinary signatures used in practice today. It is based on the forward-secure signature scheme of Itkis and Reyzin [IR01].

Like the forward-secure signature scheme of [IR01], the security of our intrusion-resilient scheme is based on the strong RSA assumption (precisely defined in Section 4.1) in the random oracle model.

1.3 Towards Obsoletion of Certificate Revocation

On-line authentication is a common application of signatures. For example, a user establishing an authenticated connection to a web site (e.g., over SSL), must verify the web site's signature on a protocol message, as well as the web site's certificate that attests to the authenticity of the web site's public key. If the web site's secret key is compromised, the certificate needs to be *revoked*.

Certificate revocation, however, is a complex logistical problem that results in some of the most cumbersome aspects of public key infrastructures. The most common, though perhaps not the most efficient, mechanism is to consult a certificate revocation list (CRL), which would most likely be stored at a remote location (certificates usually include a pointer to the corresponding CRL site).

However, if the web site uses our signature scheme, then an exposed secret key would compromise the authenticity of the web site only for a limited time (less than the frequency with which certificate revocation information is updated, which is typically one day). Then the user would not need to check whether the certificate is revoked or not: by the time the revocation information is updated, the web site would be authentic again, anyway.

Note that forward-secure signatures do not help address this problem: the web site's certificate would still have to be revoked in case of compromise. In contrast, if the web site uses intrusion-resilient signatures, the certificate would have to be revoked only in the unlikely case that the web site and its (presumably, separately protected) home base are compromised *simultaneously*. (We note that short-lived certificates [Mic96, GGM00], key-insulated signatures [DKXY02] and proactive signature [OY91, HJJ$^+$97] can also be used to address certificate revocation; our solution, however, seems to provide the most security if one is interested in abandoning certificate revocation/reissuing entirely and having truly off-line certification authorities.)

2 Intrusion-Resilient Security Model

Our definitions are based on the definitions of key-insulated security of [DKXY02], which, in turn, are based on the definitions of secure signatures of [GMR88] and forward secure signatures of [BM99]. Before describing our model formally, we explain its differences from that of [DKXY02].

In our model the home base updates its internal state at the end of each time period (in addition to sending the update information to the signer). We also provide for a special refresh procedure (akin to proactivization) which allows recovery from compromise procedure. If a refresh is run after a compromise of one of the parties but before the compromise of the other, the information the adversary learned during the compromise becomes essentially useless, and the system remains secure. Moreover, because our refresh involves just one message from the home base to the signer, it can be combined with update and thus run at least every time period.

The adversary in our model is allowed the usual adaptive-chosen-message-and-time-period attack, and, additionally, can obtain the secrets from the home base and the signer for time periods of its choice. Furthermore, the adversary can intercept update and refresh messages of its choice between the base and the signer. Like in [DKXY02], if the adversary only compromises the base (in fact, even if the base is continuously monitored by the adversary from the start), it still cannot forge signatures. Also like in [DKXY02], if the adversary compromises the signer, then it can forge signatures only for the periods for which the secrets were obtained (either directly via signer compromise, or by combination of signer compromise and interception of consecutive update messages). In contrast to [DKXY02], however, our model tolerates even multiple compromises of both base and signer (in arbitrary order), as long as there is a refresh between any compromise of the different modules. Moreover, even if there is no refresh, then the scheme still remains forward-secure.

We treat all compromises in one definition, as opposed to separately defining security against different kinds of compromises. This allows us to precisely specify the security requirements when different types of compromises (base, signer, update messages) are combined. This is in contrast to the key-insulated definitions of [DKXY02], where compromises of the base key are considered in isolation, and compromises of key update messages are reduced to compromises of pairs of consecutive time periods[4].

The definitions below are given in the standard model, but can easily incorporate random oracles (used in our proofs).

2.1 Functional definition

We first define the functionality of the various components of the system; the security definition is given in the subsequent section. Recall that the system's secret keys may be modified in two different ways, which we call *update* and *refresh*. Updates lead to changes that are visible to the verifier and change the secrets from one time period to the next (e.g. from one day to the next). In contrast, refreshes are transparent to outsiders (including the verifier).

Thus we use notation $SK_{t.r}$ for key SK, where t is the time period (the number of times the key has been updated) and r is the "refresh number" (the number of times the key has been refreshed since the last update). We follow the convention of [BM99], which requires key update immediately after key generation in order to obtain the keys for $t = 1$ (this is done merely for notational convenience, in order to make the number of time periods T equal to the number of updates, and need not affect the efficiency of an actual implementation). We also require key refresh immediately after key update in order to obtain keys for $r = 1$ (this is also done for convenience, and need not

[4]With respect to key update information, [DKXY02] define a scheme as having "secure key updates" if key update information sent for time period i can be computed from signer's keys for time period i and $i - 1$. We find this requirement to be both too strong and too weak. It is too strong because it is quite possible that, while key update information cannot be *computed* from signer's keys, it is *no more useful* two consecutive signer's keys. It is too weak, because it does not rule out the possibility for the adversary to forge signatures for two consecutive time periods if key update information is compromised. In fact, in [DKXY02], if the number of signer compromises that the scheme resists is limited to t, then number of update information exposures is limited to only $t/2$.

affect efficiency of an actual implementation; in particular, the update and refresh information that the base sends to the signer can be combined into a single message).

A $\langle$*signer-base*$\rangle$ *key-evolving* signature scheme is a septuple of probabilistic polynomial-time algorithms $(Gen, Sign, Ver; \mathcal{US}, \mathcal{UB}; \mathcal{RB}, \mathcal{RS})$[5]:

1. *Gen*, the key generation algorithm.[6]
   In: security parameter(s) (in unary), the total number T of time periods
   Out: initial signer key $SKS_{0.0}$, initial home base key $SKB_{0.0}$, and the public key PK.

2. *Sign*, the signing algorithm.
   In: current signer key $SKS_{t.r}$, message m
   Out: signature (t, sig) on m for time period t

3. *Ver*, the verifying algorithm
   In: message m, signature (t, sig) and public key PK
   Out: "valid" or "invalid" *(as usual, signatures generated by Sign must verify as "valid")*

4. $\mathcal{UB}$, the base key update algorithm
   In: current base key $SKB_{t.r}$
   Out: the next base key $SKB_{(t+1).0}$ and the key update message SKU_t

5. $\mathcal{US}$, the signer update algorithm
   In: current signer secret key $SKS_{t.r}$ and the key update message SKU_t
   Out: the next signer secret key $SKS_{(t+1).0}$

6. $\mathcal{RB}$, the base key refresh algorithm
   In: current base key $SKB_{t.r}$
   Out: new base key $SKB_{t.(r+1)}$ and the corresponding key refresh message $SKR_{t.r}$

7. $\mathcal{RS}$, the signer refresh algorithm
   In: signer's current secret key $SKS_{t.r}$ and the key refresh message $SKR_{t.r}$
   Out: signer's secret key $SKS_{t.(r+1)}$ corresponding to the base's key $SKB_{t.(r+1)}$

Differences from Prior Notions. If only $Gen, Sign, Ver$ are used, then $t.r$ and SKB can be ignored in these algorithms, and the above functional definition becomes that of an ordinary signature scheme.

Relaxing the above restrictions to also allow the use of $\mathcal{US}$ (while setting $SKU_t = 1$ for all t), extends the definition to that of forward-secure signatures (or a "key-evolving" scheme [BM99], to be more precise).

Functional definition of a "key-insulated" signature scheme [DKXY02] is obtained by further relaxing the restrictions to allow the use of $\mathcal{UB}$ as well (and thus removing $SKU_t = 1$ restriction),

[5] Intuitively (and quite roughly), the first three correspond to the ordinary signatures; the first four correspond to forward-secure ones, the first five (with some restrictions) correspond to key-insulated ones; and all seven are needed to provide the full power of the intrusion-resilient signatures.

[6] As opposed to the other algorithms below, which are meant to be run by a single party (signer, verifier or base), it may be useful to implement the key generation algorithm as distributed between the signer and the home base modules, in such a way that corruption even during key generation does not fully compromise the scheme. Alternatively, key generation may be run by a third party (avoiding the need the need for the signer and the base to run the expensive and sensitive prime generation). For simplicity of definitions, however, we postpone consideration of these scenarios until Section 3.

but restricting $SKB_t = \langle SKB, t\rangle$ for some secret SKB and for every period t (i.e. the base secret does not change).

Finally, our model is obtained by removing the remaining restrictions: allowing the base secret to vary and using $\mathcal{RB}, \mathcal{RS}$.

2.2 Security Definition

In order to formalize security, we need to specify, for each time period, how many times the base and the signer refresh their keys. Let $RN : \mathcal{N} \to \mathcal{N}$ be the function specifying this. That is, in time period t, the keys will be refreshed $RN(t)$ times–i.e., there will be $RN(t)+1$ instances of signer and base keys. (Recall that each update is immediately followed by a refresh; thus, keys with refresh index 0 are never actually used.) Then consider all the keys generated during the entire run of the signature scheme. They are generated by the following procedure.

Experiment `Generate-Keys`(k, T, RN)
  $t \leftarrow 0$; $r \leftarrow 0$
  $(SKS_{t.r}, SKB_{t.r}, PK) \leftarrow Gen(1^k, T)$
  `for` $t = 1$ `to` T
    $(SKB_{t.0}, SKU_{t-1}) \leftarrow \mathcal{UB}(SKB_{(t-1).r})$
    $SKS_{t.0} \leftarrow \mathcal{US}(SKS_{(t-1).r}, SKU_{t-1})$
    `for` $r = 1$ `to` $RN(i)$
      $(SKB_{t.r}, SKR_{t.(r-1)}) \leftarrow \mathcal{RB}(SKB_{t.(r-1)})$
      $SKS_{t.r} \leftarrow \mathcal{RS}(SKS_{t.(r-1)}, SKR_{t.(r-1)})$

Let SKS^*, SKB^*, SKU^* and SKR^* be the sets consisting of, respectively, signer and base keys and update and refresh messages, generated during the above experiment. Because the keys $SKS_{t.0}, SKB_{t.0}$ for $0 \le t \le T$, SKU_0 and $SKR_{1.0}$ are generated only by our notational convention, but are never actually stored or sent (because key generation is immediately followed by update, and each update is immediately followed by refresh), we omit them from the sets SKS^*, SKB^* and SKU^*, respectively. Note that $SKR_{t.0}$ for $t > 1$ is used (it is sent together with SKU_{t-1} to the signer), and thus is included into SKR^*.

To define security, let F, the adversary (or "forger"), be a probabilistic polynomial-time oracle Turing machine with the following oracles:

- *Osig*, the signing oracle (constructed based on SKS^*), which on input (m, t, r) $(1 \le t \le T$, $1 \le r \le RN(t))$ outputs $Sign(SKS_{t.r}, m)$
- *Osec*, the key exposure oracle (constructed based on the sets SKS^*, SKB^*, SKU^* and SKR^*), which
  1. on input ("s", $t.r$) for $1 \le t \le T, 1 \le r \le RN(t)$ outputs $SKS_{t.r}$;
  2. on input ("b", $t.r$) for $1 \le t \le T, 1 \le r \le RN(t)$ outputs $SKB_{t.r}$;
  3. on input ("u", t) for $1 \le t \le T-1$ outputs SKU_t and $SKR_{t+1}.0$; and
  4. on input ("r", $t.r$) for $1 \le t \le T, 1 \le r < RN(t)$, outputs $SKR_{t.r}$.

Note that exposure of the update key automatically exposes the following refresh key, because they are sent together in one message.

The only restriction we put on adversary's queries to *Osig* is that the values t and r be within the appropriate bounds, and that the order of exposures "respect erasures" (in particular, we do not assume any synchrony of the network that presumably carries the update and refresh messages). In other words, if a value has been queried, another value that has to have been erased prior to that query cannot be queried. Formally,

- ("b", $t.r$) has to be queried *before* ("b", $t'.r'$) if $t' > t$ or $t' = t$ and $r' > r$;
- ("b", $t.r$) has to be queried *before* ("r", $t'.r'$) if $t' > t$ or $t' = t$ and $r' \geq r$;
- ("b", $t.r$) has to be queried *before* ("u", t') if $t' \geq t$;
- ("s", $t.r$) has to be queried *before* ("s", $t'.r'$) if $t' > t$ or $t' = t$ and $r' > r$.

The following experiment captures adversary's functionality.

Experiment `Run-Adversary`(F, k, T, RN)
  `Generate-Keys`(k, T, RN)
  $(m, j, sig) \leftarrow F^{Osig, Osec}(1^k, T, PK)$
  `if` $Ver(m, j, sig) =$"invalid" `or` (m, j) was queried by F to *Osig* `or` a query to *Osec* was out of bounds or did not respect erasures
    `then return` "adversary failed"
    `else` let Q be the set of key exposure queries F made to *Osec*; `return` (j, Q)

For a set of key exposure queries Q, time period $t \geq 1$ and refresh number r, $1 \leq r \leq RN(t)$, we say that key $SKS_{t.r}$ is Q-*exposed*:

[directly] if ("s", $t.r$) $\in Q$; or

[via update] if $r = 1$, ("u", $t-1$) $\in Q$, and $SKS_{(t-1).RN(t-1)}$ is Q-exposed; or

[via previous refresh] if $r > 1$, ("r", $t.(r-1)$) $\in Q$, and $SKS_{t.(r-1)}$ is Q-exposed; or

[via next refresh] if $r < RN(t)$, ("r", $t.r$) $\in Q$, and $SKS_{t.(r+1)}$ is Q-exposed.

Replacing SKS with SKB throughout the above definition results in the definition of base key exposure (or more precisely $SKB_{t.r}$ being Q-*exposed*).

Note that the above definitions are recursive, with direct exposure as the base case. Also note a difference between update and refresh messages: intercepting the former might help F to expose only future keys, while intercepting the latter might help F to expose both future and past (down to the previous update) keys.

Clearly, exposure of signer key ($SKS_{t.r}$ for the given t and any r) enables the adversary to generate legal signatures for that period (t). Similarly, simultaneous exposure of both base and signer keys ($SKB_{t.r}, SKS_{t.r}$, for some t, r) exposes all information in the system to the adversary. Therefore, we cannot hope to prevent the adversary from generating valid signatures for the current and future time periods.

Thus, we say that the scheme is (t, Q)-*compromised*, if either

- $SKS_{t.r}$ is Q-exposed for some r, $1 \leq r \leq RN(t)$; or
- $SKB_{t_1.r}$ and $SKS_{t_1.r}$ are both Q-exposed for some $t_1 < t$ and r, $1 \leq r \leq RN(t_1)$.

In other words, a particular time period has been rendered useless if either the signer was broken into during that time period, or, during a previous time period, the signer and the base were compromised without a refresh in between. Note that update and refresh messages by themselves do not help the adversary in our model—they only help when combined, in unbroken chains, with signer or base keys.

Definition 1. Let $\mathsf{SiBIR}[k, T]$ be a ⟨signer-base⟩ key-evolving scheme with security parameter k and number of time periods T. Let F be an adversary for it and $RN : \mathcal{N} \to \mathcal{N}$ be a function. Define

$$\mathbf{Succ}^{\text{SiBIR}}(\mathsf{SiBIR}[k, T], F, RN) \stackrel{def}{=}$$
$$\Pr[\texttt{Run-Adversary}(F, k, T, RN) = (j, Q) \,:\, F \text{ has not } (j, Q)\text{-compromised the scheme}] \leq \epsilon(k, T)\,.$$

Let $\mathbf{InSec}^{\text{SiBIR}}(\mathsf{SiBIR}[k, T], \tau, q_{\text{sig}})$ (the *insecurity function for intrusion resilience*) be the maximum of $\mathbf{Succ}^{\text{SiBIR}}(\mathsf{SiBIR}[k, T], F, RN)$ over all functions RN and all F that run in time at most τ and ask at most q_{sig} signature queries. The scheme $\mathsf{SiBIR}[k, T]$ is $(\tau, \epsilon, q_{\text{sig}})$-*intrusion-resilient* if $\mathbf{InSec}^{\text{SiBIR}}(\mathsf{SiBIR}[k, T], \tau, qsig) < \epsilon$.

3 Intrusion-Resilient Scheme: Construction

Our scheme, which we call $\mathsf{SiBIR1}$, is based on the [IR01] forward-secure signature scheme, which, in turn, is based on the Guillou-Quisquater [GQ88] ordinary signature scheme. (In the interests of conciseness, we do not present the rationale behind the [IR01] forward-secure scheme.) In fact, the [IR01] forward-secure scheme can be obtained from our scheme by simply eliminating the base, and setting all the messages that the signer expects equal to 1.

The scheme utilizes two security parameters, l and k. Let $H : \{0,1\}^* \to \{0,1\}^l$ be a hash function (modeled in the security proof as a random oracle).

```
algorithm SiBIR1.Gen(k, l, T)
    Generate a modulus n:
        Generate random (⌈k/2⌉ − 1)-bit primes q_1, q_2 s.t. p_i = 2q_i + 1 are both prime
        n ← p_1 p_2
    Generate exponents:
        Generate primes e_i s.t. 2^l(1 + (i − 1)/T) ≤ e_i < 2^l(1 + i/T) for i = 1, 2, ..., T.
            (This generation is done either deterministically or using a small seed seed and H as a
            pseudorandom function; let E be the information, if any, necessary to regenerate e_1, ..., e_T)
    s_[1,T] ←R Z*_n; SKS_0 ← (0, T, n, ∅, s_[1,T], ∅, E) //the two ∅ are placeholders to be filled in during US
    b_[1,T] ←R Z*_n; SKB_0 ← (0, T, n, b_[1,T], E)
    v ← 1/(s_[1,T] b_[1,T])^{e_1 · ... · e_T} mod n; PK ← (n, v, T)
    return (SKS_0, SKB_0, PK)
```

Figure 1: *Key generation algorithm. Refresh index on the keys is omitted to simplify notation.*

KEY GENERATION AND UPDATE. Like in the forward-secure scheme of [IR01] and the ordinary signature scheme of [GQ88], the public key consists of an RSA modulus n and a value $v \in Z_n^*$. For

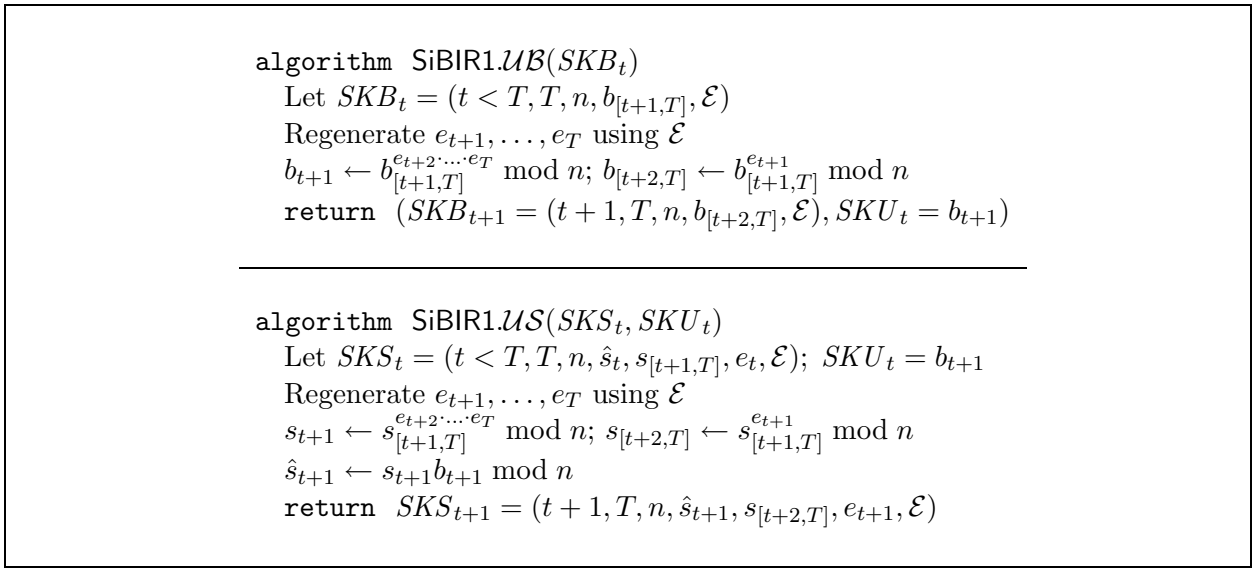

Figure 2: *Update algorithms. Refresh index on the keys is omitted to simplify notation.*

each time period t, there is a corresponding exponent e_t (all the exponents have to be relatively prime). What is needed to generate signatures in time period t is $\hat{s}_t$ such that $\hat{s}_t^{e_t} \equiv 1/v \pmod{n}$.

Because the factorization of n has to be erased after key generation in order to achieve forward security, in order be able to generate $\hat{s}_t, \hat{s}_{t+1}, \ldots, \hat{s}_T$, one needs to know a root of $1/v$ of degree $e_{[t,T]} \stackrel{def}{=} e_t \cdot e_{t+1} \cdot \ldots \cdot e_T$. Call this root $\hat{s}_{[t,T]}$: then $\hat{s}_{[t,T]}^{e_{[t,T]}} \equiv 1/v \pmod{n}$. Note that this value can be easily updated: $\hat{s}_{[t+1,T]} = \hat{s}_{[t,T]}^{e_t}$; also note that the "current" signing key can be easily obtained via $\hat{s}_t = \hat{s}_{[t,T]}^{e_{[t+1,T]}}$.

In order to achieve intrusion-resilience, $\hat{s}_{[t,T]}$ is never stored explicitly. Rather, it is shared multiplicatively between the signer and the base. The signer stores $s_{[t,T]}$ and the base stores $b_{[t,T]}$, such that $\hat{s}_{[t,T]} = s_{[t,T]} b_{[t,T]}$. This multiplication is never explicitly performed: instead, the signer computes $s_t = s_{[t,T]}^{e_{[t+1,T]}}$, the base computes $b_t = b_{[t,T]}^{e_{[t+1,T]}}$, and the two values are multiplied together to obtain $\hat{s}_t$.

Following the conventions that key generation is immediately followed by key update, the first signer secret key contains blanks for $\hat{s}_0$ and e_0. We note that, in actual implementation, it will be more efficient to combine the first key generation and update.

Finally, note that key generation and update algorithms do not affect the refresh index, so we omit it in Figures 1 and 2 in order to simplify notation.

Variations on Key Generation. Most of the key generation algorithm can be easily split between the signer and the base. Namely, once the shared modulus n is generated and given to both parties, the base can generate $b_{[1,T]}$ on its own, and the signer can generate $s_{[1,T]}$ on its own, as well. Both parties can then generate "shares" $b_{[1,T]}^{e_{[1,T]}}$ and $s_{[1,T]}^{e_{[1,T]}}$ that can be combined to compute the public key. The shares themselves can be made public without adversely affecting security. Thus, the amoung of cooperation required during key generation is minimal

The same modulus n can be used by multiple signature schemes. In particular, our signature scheme can be made identity-based if a third party is trusted to take roots modulo n of the identity v.

Refresh. Because the signer and the base share a single value multiplicatively, the refresh

algorithm presented in Figure 3 is quite simple: the base divides its share by a random value, and signer multiplies its share by the same value. Recall that each update is immediately followed by refresh (and, in fact, update and refresh information can be sent by the base to the signer in one message).

algorithm SiBIR1.$\mathcal{RB}(SKB_{t.r})$
  Let $SKB_{t.r} = (t, T, n, b_{[t+1,T]}, \mathcal{E})$
  $R_{t.r} \stackrel{R}{\leftarrow} Z_n^*$
  $b_{[t+1,T]} \leftarrow b_{[t+1,T]}/R_{t.r}$
  return $(SKB_{t.r+1} = (t, T, n, b_{[t+1,T]}, \mathcal{E}), SKR_{t.r} = R_{t.r})$

---

algorithm SiBIR1.$\mathcal{RS}(SKS_{t.r}, SKR_{t.r})$
  Let $SKS_{t.r} = (t, T, n, \hat{s}_t, s_{[t+1,T]}, e_t, \mathcal{E})$; $SKR_{t.r} = R_{t.r}$
  $s_{[t+1,T]} \leftarrow s_{[t+1,T]} \cdot R_{t.r}$
  return $SKS_{t.r+1} = (t, T, n, \hat{s}_t, s_{[t+1,T]}, e_t, \mathcal{E})$

Figure 3: *Key refresh algorithms*

Signing and Verifying. Figure 4 describes our signature and verification algorithms. They are exactly the same as in the forward-secure signature scheme of [IR01]. Again, we omit the refresh index on the signer's key for ease of notation.

algorithm SiBIR1.*Sign*(M, SK_t) %*same as* IR.*Sign in [IR01]*
  Let $SK_t = (t, T, n, \hat{s}_t, s_{[t+1,T]}, e_t, \mathcal{E})$
  $x \stackrel{R}{\leftarrow} Z_n^*$
  $y \leftarrow x^{e_t} \bmod n$
  $\sigma \leftarrow H(t, e_t, y, M)$
  $z \leftarrow x\hat{s}_t^{\sigma} \bmod n$
  return (z, σ, t, e_t)

---

algorithm SiBIR1.*Ver*$(M, PK, (z, \sigma, t, e))$ %*same as* IR.*Ver* in [IR01]
  Let $PK = (n, v, T)$
  if $e \geq 2^l(1 + t/T)$ **or** $e < 2^l$ **or** e is even **then return** 0
  if $z \equiv 0 \pmod{n}$ **then return** 0
  $y' \leftarrow z^e v^{\sigma} \bmod n$
  if $\sigma = H(t, e, y, M)$ **then return** 1 **else return** 0

Figure 4: *Signing and verifying algorithms. Refresh index on the keys is omitted to simplify notation.*

Variations. Our scheme can be easily modified (with no or minimum increase in storage requirement!) to "re-charge" the signer for more than one time period at a time. To enable the signer

to compute $\hat{s}_{t_1}, \hat{s}_{t_1+1}, \ldots \hat{s}_{t_2}$, the base simply needs to send the signer $b_{[t_1,t_2]} = b_{[t_1,T]}^{e_{[t_2+1,T]}}$. In fact, it is easy to extend this method to non-contiguous time periods. This feature may have interesting applications for delegation (including self-delegation).

Another simple modification of our scheme can yield forward-secure threshold and proactive scheme (similar to, but more efficient than, the scheme of [AMN01]). Efficiency for the verifier and for each of the modules participating in the signing will be essentially the same as for the regular Guillou-Quisquater scheme.

4 Security

4.1 Complexity Assumption

(This section is heavily based on [IR01].) We use a variant of the strong RSA assumption (introduced in [BP97] and [FO97]), which postulates that it is hard to compute *any* root of a fixed value modulo a composite integer. More precisely, the strong RSA assumption states that it is intractable, given n that is a product of two primes and a value α in Z_n^*, to find $\beta \in Z_n^*$ and $r > 1$ such that $\beta^r = \alpha$.

However, we modify the assumption in two ways. First, we restrict ourselves to the moduli that are products of so-called "safe" primes (a safe prime is one of the form $2q + 1$, where q itself is a prime). Note that, assuming safe primes are frequent, this restriction does not strengthen the assumption. Second, we upperbound the permissible values or r by 2^{l+1}, where l is a security parameter for our scheme (in an implementation, l will be significantly shorter than the length k of the modulus n).

More formally, let A be an algorithm. Consider the following experiment.

Experiment Break-Strong-RSA(k, l, A)
Randomly choose two primes q_1 and q_2 of length $\lceil k/2 \rceil - 1$ each
such that $2q_1 + 1$ and $2q_2 + 1$ are both prime.
$p_1 \leftarrow 2q_1 + 1$; $p_2 \leftarrow 2q_2 + 1$; $n \leftarrow p_1 p_2$
Randomly choose $\alpha \in Z_n^*$.
$(\beta, r) \leftarrow A(n, \alpha)$
`If` $1 < r \leq 2^{l+1}$ `and` $\beta^r \equiv \alpha \pmod{n}$ `then return` 1 `else return` 0

Let $\mathsf{Succ}^{\mathrm{SRSA}}(A, k, l) = \Pr[\text{Break-Strong-RSA(k,l,A)} = 1]$. Let $\mathbf{InSec}^{\mathrm{SRSA}}(k, l, \tau)$ be the maximum of $\mathsf{Succ}^{\mathrm{SRSA}}(A, k, l)$ over all the adversaries A who run in expected time at most τ. Our assumption is that $\mathbf{InSec}^{\mathrm{SRSA}}(k, l, \tau)$, for τ polynomial in k, is negligible in k. The smaller the value of l, of course, the weaker the assumption.

In fact, for a sufficiently small l, our assumption follows from a variant of the fixed-exponent RSA assumption. Namely, assume that there exists a constant ϵ such that, for every r, the probability of computing, in expected time τ, an r-th root of a random integer modulo a k-bit product of two safe primes, is at most 2^{-k^ϵ}. Then, $\mathbf{InSec}^{\mathrm{SRSA}}(k, l, \tau) < 2^{l+1-k^\epsilon}$, which is negligible if $l = o(k^\epsilon)$.

4.2 Security Proof

Our security proof is more complex that the one of [IR01], although the two proofs are quite similar. Both are based on the forking lemma of [PS96].

Theorem 1 *For any τ, q_{sig}, and q_{hash},*

$$\mathbf{InSec}^{\mathrm{SiBIR}}(\mathsf{SiBIR1}[k,l,T];t,q_{\mathrm{sig}},q_{\mathrm{hash}}) \leq$$
$$T\sqrt{(q_{\mathrm{hash}}+1)\mathbf{InSec}^{\mathrm{SRSA}}(k,l,\tau')} + 2^{-l+1}T(q_{\mathrm{hash}}+1) + 2^{2-k}q_{\mathrm{sig}}(q_{\mathrm{hash}}+1)\ ,$$

where $\tau' = 4\tau + O(lT(l^2T^2+k^2))$.

The proof of the theorem is given in Appendix A

4.3 Active Attacks

Because information flows only from the base to the signer, the adversary's only possible active attack is to send a bad *SKR* or *SKU* value to the signer. An active attacker can thus always prevent signatures from being issued. While our definition does not consider active attacks for the sake of simplicity, in our implementation in Section 3, the active adversary cannot do anything worse that merely sabotage the system. It is easy to show that, in terms of forging new signatures, its powers are no greater than those of a passive attacker who merely obtains *SKR* and *SKU* values.

References

[AMN01] Michel Abdalla, Sara Miner, and Chanathip Namprempre. Forward-secure threshold signature schemes. In David Naccache, editor, *Progress in Cryptology — CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer-Verlag, April 8-12 2001.

[And97] Ross Anderson. Invited lecture. Fourth Annual Conference on Computer and Communications Security, ACM, 1997.

[BM99] Mihir Bellare and Sara Miner. A forward-secure digital signature scheme. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer-Verlag, 15–19 August 1999. Revised version is available from `http://www.cs.ucsd.edu/~mihir/`.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology—EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer-Verlag, 11–15 May 1997.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer-Verlag, 1990, 20–24 August 1989.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars Knudsen, editor, *Advances in Cryptology—EUROCRYPT 2002*, Lecture Notes in Computer Science. Springer-Verlag, 28 April–2 May 2002.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 17–21 August 1997.

[GGM00] Irene Gassko, Peter Gemmell, and Philip MacKenzie. Efficient and fresh certication, 2000.

[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.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, 25–27 May 1987.

[GPR98] Oded Goldreich, Birgit Pfitzmann, and Ronald L. Rivest. Self-delegation with controlled propagation - or - what if you lose your laptop. In *CRYPTO*, pages 153–168, 1998.

[GQ88] Louis Claude Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 1990, 21–25 August 1988.

[HJJ+97] Amir Herzberg, Markus Jakobsson, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *Fourth ACM Conference on Computer and Communication Security*, pages 100–110. ACM, April 1–4 1997.

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

[LR00] Anna Lysyanskaya and Ron Rivest. Bepper-based signatures. Presented by Rivest at the CIS seminar at MIT, 27 October 2000.

[Mic96] Silvio Micali. Efficient certificate revocation. Technical Report MIT/LCS/TM-542b, Massachusetts Institute of Technology, Cambridge, MA, March 1996.

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks. In *10-th Annual ACM Symp. on Principles of Distributed Computing*, pages 51–59, 1991.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology—EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 12–16 May 1996.

[Riv98] Ronald L. Rivest. Can we eliminate certificate revocation lists? In Rafael Hirschfeld, editor, *Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[Yao82] A.C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, 3–5 November 1982. IEEE.

A Security Proof

Below we present the outline of the proof of Theorem 1, much of which is borrowed from [IR01]; we provide details only where the proof differs from that of [IR01].

First, we state the following theorem that will allow us to upper-bound the insecurity function.

Theorem 2 *Given a forger F for $\mathsf{SiBIR1}.[k,l,T]$ that runs in expected time at most τ, asking q_{hash} hash queries and q_{sig} signing queries, such that $\mathbf{Succ}^{\text{SiBIR}}(\mathsf{SiBIR1}.[k,l,T],F,RN) \geq \varepsilon$, we can construct an algorithm A that, on input n (a product of two safe primes), $\alpha \in Z_n^*$ and l, runs in expected time τ' and outputs (β, r) such that $1 < r \leq 2^{l+1}$ and $\beta^r \equiv \alpha \pmod{n}$ with probability ε', where*

$$\begin{aligned}\tau' &= 4\tau + O(lT(l^2T^2 + k^2)) \\ \varepsilon' &= \frac{\left(\varepsilon - 2^{2-k}q_{\text{sig}}(q_{\text{hash}}+1)\right)^2}{T^2(q_{\text{hash}}+1)} - \frac{\varepsilon - 2^{2-k}q_{\text{sig}}(q_{\text{hash}}+1)}{2^l T}.\end{aligned}$$

Proof Outline. A will use F as a subroutine. (Note that A gets to provide the public key for F and to answer its signing, hashing and key exposure queries.)

A randomly guesses j between 1 and T, hoping that F's eventual forgery will be for the j-th time period. It then generates $e_1, \ldots, e_T$ just like the real signer, and computes $v = 1/\alpha^{e_1 \cdot \ldots \cdot e_{j-1} e_{j+1} \cdot \ldots \cdot e_T} \bmod n$ (note that this is different from [IR01] proof).

Then A runs F on input v. Answering F's hash and signature queries is easy, because A fully controls the random oracle H. Answering key exposure queries is a little trickier, and is discussed below. If A's guess for j was correct, and F indeed will output a forgery for the j-th time period,(z, σ, j, e) on some message m. We will assume that F asked a hash query on (j, e, y, m) where $y = z^e v^\sigma \bmod n$ (F can always be modified to do so.)

Then, A resets F and runs it a second time with the same random tape, giving the same answers to all the oracle queries before the query (j, e, y, m). For (j, e, y, m), A gives a new answer σ'. If F again forges a signature (z', σ', j, e) using the same hash query, we will have that $y \equiv z^e v^\sigma \equiv z'^e v^{\sigma'} \pmod{n}$, so $(z/z')^e \equiv v^{\sigma'-\sigma} \equiv \alpha^{e_{[j+1,T]}(\sigma-\sigma')} \pmod{n}$. Note that because e is guaranteed to be relatively prime with $e_{[j+1,T]}$, and $\sigma - \sigma'$ has at least one fewer bit than e, $\gcd(e_{[j+1,T]}(\sigma - \sigma'), e) = \gcd(\sigma - \sigma', e) < e$ (as long as $\sigma \neq \sigma'$). Thus, Lemma 1 of [IR01], A will be able to efficiently compute the a root of α of degree $e/\gcd(f_{j+1}(\sigma' - \sigma), e) > 1$.

What remains is to show how A answers key exposure queries. First, A generates a random $b_{[1,T].0} \in Z_n^*$. Then, A performs all the base update and refresh operations using $\mathsf{SiBIR1}.\mathcal{UB}$ and $\mathsf{SiBIR1}.\mathcal{RB}$ to generate $b_{[t+1,T].r}$, $SKU_t = b_{t+1}$ and $SKR_{t.r} = R_{t.r}$ for $1 \leq t \leq T$ and $0 \leq r < RN(t)$. A sets $s_{[1,T].0} = 1/b_{[1,T].0} \bmod n$ and performs updates and refreshes (using $\mathsf{SiBIR1}.\mathcal{US}$ and $\mathsf{SiBIR1}.\mathcal{RS}$ and the already generated $SKR_{t.r}$ and SKU_t) until time period j (exclusive) to generate $s_{[t+1,T].r}$ for $1 \leq t < j$ and $0 \leq r < RN(t)$. A sets $s_{[j+1,T].0} = \alpha^{e_1 \cdot \ldots \cdot \ldots \cdot e_{j-1}}/b_{[j+1,T].0} \bmod n$, and performs all the remaining updates and refreshes to generate $s_{[t+1,T].r}$ for $j \leq t < T$ and $0 \leq r < RN(t)$.

A then throws away the above-generated values $\hat{s}_t$ and $SKU_t = b_{t+1}$ for $t < j$. Instead, for $t < j$, A sets $\hat{s}_t = \alpha^{e_1 \cdot \ldots \cdot e_{t-1} e_{t+1} \cdot \ldots \cdot e_{j-1} e_{j+1} \cdot \ldots \cdot e_T} \bmod n$.

Thus, now A has all the information needed to answer "s", "b" and "r" queries, as well as ("u", t) queries for $t \geq j$, except $\hat{s}_j$, which will not be asked if F forges for time period j. Note that $\hat{s}_t^{e_t} = 1/v$, and thus is the same as the true signer would give. In addition, for $t \geq j$, $(b_{[t+1,T].r} s_{[t+1,T].r})^{e_{[t+1,T]}} = 1/v$, and thus is also the same as the true signer would give. The only

"inconsistency" with the true signer is for $t < j$, when $b_{[t+1,T].r} s_{[t+1,T].r} = 1$. However, because F is prohibited from exposing both $b_{[t+1,T].r}$ and $s_{[t+1,T].r}$ for $t < j$, F will never notice this inconsistency. This will be proven more formally by Lemma 1 below.

We have not yet specified, however, how A answers ("u", t) queries for $t < j$. This is non-trivial, because it seems that the answer must be consistent with both $SKB_{(t+1).1}$ and $SKS_{(t+1).1}$, which are themselves inconsistent. However, note that only one of $SKB_{(t+1).1}$ and $SKS_{(t+1).1}$ can be exposed by F. Thus, whenever A receives the query ("u", t), A simply guesses which of the two will be exposed, and makes SKU_t consistent with that one. If A guesses that $SKB_{(t+1).1}$ will be exposed, then A sets $SKU_t = b_{t+1} = b_{[t+1,T].RN(t)}^{e_{[t+2,T]}}$. Otherwise, A sets $SKU_t = \hat{s}_{t+1}/s_{t+1} = \hat{s}_{t+1}/s_{[t+1,T].RN(t)}^{e_{[t+2,T]}}$. If it later turns out that A's guess is incorrect, A rewinds F to the update query, and tries again. A will not need to rewind past an update query, because A will learn whether $SKS_{(t+1).1}$ or $SKB_{(t+1).1}$ is exposed before the next update, because queries respect the order of erasures. Therefore, the expected number of tries per update query is 2, thus slowing A down by a factor of 2. Note that this is the only place in the proof where we use that A's queries come in order—otherwise, if A were able to ask all the update queries first, and then decide which base or signer keys to expose, we would have to rewind exponentially many times.

To complete the proof, we need the following Lemma.

Lemma 1 *Let Q be a set of key exposure queries that F asked, such that the scheme is not (j, Q)-compromised. Then the probability that A returns a particular set of answers to F's key exposure queries is equal to the probability that the true signer/base pair returns this set of answers to F.*

Proof The public key v and the exponents $e_1, \ldots, e_T$ have the same distribution for A as for the true signer. Once we fix v, $e_1, \ldots, e_T$ and $b_{[1,T]}$, then the random choices made by the true signer/base pair consist of values $R_{t.r}$ for $0 \leq r < RN(t)$ ($s_{[1,T]}$ is no longer random, but rather uniquely determined by v, $e_1, \ldots, e_T$ and $b_{[1,T]}$). We will call them "honest choices" and superscript them and the corresponding $s_{[t+1,T].r}$, $b_{[t+1,T].r}$ and SKU_t values with H for "honest": $R^H_{t.r}$, $s^H_{[t+1,T].r}$, $b^H_{[t+1,T].r}$ and SKU^H_t. The random choices made by A also consist of $R_{t.r}$ for $0 \leq r < RN(t)$. We will call these "fake choices" and superscript them and the corresponding $s_{[t+1,T].r}$, $b_{[t+1,T].r}$ and SKU_t values with F for "fake": $R^F_{t.r}$, $s^F_{[t+1,T].0}$, $b^F_{[t+1,T].r}$ and SKU^F_t.

We claim that there exists a bijection B that maps fake choices to honest choices and preserves the answers to F's queries. This claim suffices to establish that F will see answers with the right distribution.

To prove the claim, consider the signer's keys that are Q-exposed. For each pair $t.r$, with $t < j$, such that $SKS_{t.r}$ is Q-exposed, but the previous key $SKS_{t.(r-1)}$ (or $SKS_{(t-1).RN(t-1)}$, if $r = 1$) is not Q-exposed, let $R^H_{t.(r-1)} = B(R^F_{t.(r-1)}) = R^F_{t.(r-1)}/v^{1/(e_{[t+1,T]}}$. Similarly, for each pair $t.r$, with $t < j$, such that $SKS_{t.r}$ is not Q-exposed, but the previous key $SKS_{t.(r-1)}$ (or $SKS_{(t-1).RN(t-1)}$, if $r = 1$) is Q-exposed, let $R^H_{t.(r-1)} = B(R^F_{t.(r-1)}) = R^F_{t.(r-1)} \cdot v^{1/(e_{[t+1,T]}}$.

It is clear that the above map B is a bijection. We now need to verify that it results in the same answers, i.e., $R^F_{t.r} = R^H_{t.r}$ whenever ("r", $t.r$) $\in Q$ (or ("u", $t-1$) $\in Q$ and $r = 0$), $s^F_{[t+1,T].0} = s^H_{[t+1,T].0}$ whenever ("s", $t.r$) $\in Q$, $b^F_{[t+1,T].0} = b^H_{[t+1,T].0}$ whenever ("b", $t.r$) $\in Q$, and $SKU^F_t = SKU^H_t$ whenever ("u", t) $\in Q$.

This is done by induction: one proves that, under the correspondence provided by B, if $SKS_{t.r}$ is Q-exposed for $t < j$, then $s^F_{[t+1,T].r} = s^H_{[t+1,T].r}$, and otherwise (in particular, if $SKB_{t.r}$ is Q-exposed) $b^F_{[t+1,T].r} = b^H_{[t+1,T].r}$ (the base case relies on the fact that $SKS_{0.0}$ is not Q-exposed by definition,

and the inductive case relies on the fact that $SKS_{t.r}$ and $SKB_{t.r}$ cannot be simultaneously Q-exposed). Because $SKS_{j.r}$ is not Q-exposed for any r, and the map B does not modify anything past the j-th time period, this guarantees that $b^F_{[t+1,T].r} = b^H_{[t+1,T].r}$ for $t \geq j$. By construction, $s^F_{[t+1,T].r} = s^H_{[t+1,T].r}$ for $t \geq j$. Also, $R^F_{t.r} = R^H_{t.r}$, unless $SKS_{t.r-1}$ is Q-exposed and $SKS_{t.r}$ is not (or vice versa), in which case $SKR_{t.r}$ cannot be queried by the adversary (otherwise, both $SKS_{t.r-1}$ and $SKR_{t.r}$ would be Q-exposed). Finally, $SKU^F_t = SKU^H_t$ by the construction above. ■

This allows us to prove Theorem 1

Proof of Theorem 1 To compute the insecurity function, simply solve for $(\varepsilon - 2^{2-k} q_{\text{sig}}(q_{\text{hash}} + 1))/T$ the quadratic equation in Theorem 2 that expresses ε' in terms of ε to get

$$\begin{aligned}
&(\varepsilon - 2^{2-k} q_{\text{sig}}(q_{\text{hash}} + 1))/T \\
&\qquad = \quad 2^{-l}(q_{\text{hash}} + 1) + \sqrt{2^{-2l}(q_{\text{hash}} + 1)^2 + \varepsilon'(q_{\text{hash}} + 1)} \\
&\qquad \leq \quad 2^{-l}(q_{\text{hash}} + 1) + \sqrt{2^{-2l}(q_{\text{hash}} + 1)^2} + \sqrt{\varepsilon'(q_{\text{hash}} + 1)} \\
&\qquad = \quad 2^{-l+1}(q_{\text{hash}} + 1) + \sqrt{\varepsilon'(q_{\text{hash}} + 1)},
\end{aligned}$$

and then solve the resulting inequality for ε. ■