SiBIR: Signer-Base Intrusion-Resilient Signatures

Gene Itkis and Leonid Reyzin

Boston University Computer Science Dept.
111 Cummington St.
Boston, MA 02215, USA
{itkis,reyzin}@bu.edu

Abstract. We propose a new notion of *signer-base intrusion-resilient (SiBIR) signatures*, which generalizes and improves upon both forward-secure [?,?] and key-insulated [?] signature schemes.
Specifically, as in the prior notions, time is divided into predefined time periods (e.g., days); each signature includes the number of the time period in which it was generated; while the public key remains the same, the secret keys evolve with time. Also, as in key-insulated schemes, the user has two modules, *signer* and *home base*: the signer generates signatures on his[1] own, and the base 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, she 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 discuss 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 [?,?] 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 his secret at

[1] We use masculine pronouns for signer, feminine for adversary, and neuter for base.

the end of each time period; the adversary is unable to forge signatures for past periods even if she 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, she has the same information as the signer.

Threshold and Proactive Security. An alternative approach explores the multi-party computation paradigm [?,?]: in threshold schemes [?], 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 [?,?] improve upon this model by allowing multiple corruptions of *all* signers, limiting only the number of *simultaneous* corruptions. Proactive forward-secure signatures considered in [?] combine this with the advantages of forward-security.

Key-Insulated Security. The recently proposed model of Dodis, Katz, Xu and Yung [?] 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[2]. This is accomplished via the use of two modules: a (possibly mobile) *signer*, and a (generally stationary) *home base*[3]. 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* his 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 [?,?,?,?,?,?].) 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

Model We define intrusion-resilient signature schemes to combine benefits of the above three approaches. Namely, while maintaining the efficiency of non-interactive computation of signatures (not provided by threshold and proactive schemes), intrusion-resilient schemes preserve security of past *and* future time

[2] [?] primarily addresses encryption schemes. Signature schemes are addressed in [?].
[3] The terms *user* and *secure device* are used in [?]; we find "signer" and "home base" to be more descriptive.

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[4] and most proactive schemes).

These points deserve some elaboration. To address potential compromise of the base key, [**?**] 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 [**?**] 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 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.

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

Construction In Section **??** we provide an efficient SiBIR signature scheme we call SiBIR1. Its signing and verifying are as efficient as in the Guillou-Quisquater (ordinary) signatures [**?**], requiring just two modular exponentiations with short (typically, 128-160 bits) exponents for both signing and verifying. This is as or more efficient than many of the ordinary signatures used in practice today. The construction is based on our forward-secure signature scheme [**?**].

As for that underlying scheme [**?**], our SiBIR1 security proof relies on the strong RSA assumption (see Section **??**) and is 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

[4] Because the focus of [**?**] is on encryption schemes, and no non-trivial forward-secure encryption schemes had been known until very recently [**?**], it was, in a sense, by necessity that key-insulated notion of [**?**] did not provide forward security when all the secrets are compromised.

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 (which could be made less than the time required for the certificate revocation process, which is typically one day). Then the users need not check whether the site's certificate is revoked or not: by the time the revocation information could be 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 [**?**,**?**], key-insulated signatures [**?**] and proactive signature [**?**,**?**] 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 [**?**], which, in turn, are based on the definitions of forward secure [**?**] and ordinary [**?**] signatures schemes. Before describing our model formally, we explain its differences from that of [**?**].

First, 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). Second, we also provide for a special refresh procedure (akin to proactivization): if a refresh is run after a compromise of one of the modules but before the compromise of the other, the information the adversary learned during the compromise becomes essentially useless, and the system remains secure (except, in the case of signer compromise, for the current time period). 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 her choice. Furthermore, the adversary can intercept update and refresh messages of her choice between the base and the signer. Like in [**?**], if the adversary only compromises the base (in fact, even if the base is continuously monitored by the adversary from the start), she still cannot forge signatures. Also like in [**?**], if the adversary compromises the signer, then she 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 some update and refresh messages). In contrast to [**?**], however, our model tolerates 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, the scheme still remains forward-secure, even if there is no such refresh between some compromises of the two modules.

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 [**?**], 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[5].

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, called *update* and *refresh*. Updates change the secrets from one time period to the next (e.g. from one day to the next), changing also the period number in the signatures. In contrast, refreshes affect only the internal secrets and messages of the system, and are transparent to the verifier.

Thus we use notation $SK_{t.r}$ for secret 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 say $t.r = t'.r'$ when $t = t'$ and $r = r'$. Similarly, we say $t.r < t'.r'$ when either $t < t'$ or $t = t'$ and $r < r'$. We follow the convention of [**?**], 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 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).

[5] With respect to key update information, [**?**] define a scheme as having "secure key updates" if key update information sent for time period i can be computed from the 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 the signer's keys, it is *no more useful* than the 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 [**?**], if the number of signer compromises that the scheme resists is limited to c, then number of update information exposures is limited to only $c/2$.

Definition 1. *A* ⟨signer-base⟩ key-evolving *signature scheme is a septuple of probabilistic polynomial-time algorithms* $(Gen, Sign, Ver; \mathcal{US}, \mathcal{UB}; \mathcal{RB}, \mathcal{RS})$[6]*:*

1. *Gen, the key generation algorithm.*[7]
   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: *new 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: *new 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: *current signer key $SKS_{t.r}$ and the key refresh message $SKR_{t.r}$*
   Out: *new signer key $SKS_{t.(r+1)}$ (corresponding to the base key $SKB_{t.(r+1)}$)*

Note that this definition implies that messages are processed by the signer in the same order in which they are generated by the base.

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 [**?**], to be more precise).

[6] 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.

[7] As opposed to the other algorithms below, which are meant to be run by a single module (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 trusted third party. For simplicity, we postpone this discussion until Section **??**, where we propose a practical intermediate solution.

Functional definition of a "key-insulated" signature scheme [?] is obtained by further relaxing the restrictions to allow the use of $\mathcal{UB}$ as well (and thus removing $SKU_t = 1$ restriction), 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 a notation for the number of refreshes in each period: Let $RN(t)$ denote the number of times the keys are refreshed in the period t: 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. RN is used only for notational convenience: it need not be known; security is defined below in terms of RN (among other parameters).

Now, consider all the keys generated during the entire run of the signature scheme. They can be generated by the following "thought experiment" (we do not need to *actually* run it — it is used just for definitions).

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** $RN(t)$
    $(SKB_{t.r}, SKR_{t.(r-1)}) \leftarrow \mathcal{RB}(SKB_{t.(r-1)})$
    $SKS_{t.r} \leftarrow \mathcal{RS}(SKS_{t.(r-1)}, SKR_{t.(r-1)})$

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. We want these sets to contain all the secrets that can be directly stolen (as opposed to computed) by the adversary. Thus, we omit from these sets the keys $SKS_{t.0}, SKB_{t.0}$ for $0 \le t \le T$, SKU_0 and $SKR_{1.0}$, which are never actually stored or sent (because key generation is immediately followed by update, and each update is immediately followed by refresh). 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 using 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 (based on the sets SKS^*, SKB^*, SKU^* and SKR^*), which

1. on input ("s", $t.r$) for $1 \leq t \leq T, 1 \leq r \leq RN(t)$ outputs $SKS_{t.r}$;
2. on input ("b", $t.r$) for $1 \leq t \leq T, 1 \leq r \leq RN(t)$ outputs $SKB_{t.r}$;
3. on input ("u", t) for $1 \leq t \leq T-1$ outputs SKU_t and $SKR_{t+1}.0$; and
4. on input ("r", $t.r$) for $1 \leq t \leq T, 1 \leq r < RN(t)$, outputs $SKR_{t.r}$.

Queries to *Osec* oracle correspond to intrusions, resulting in the corresponding secrets exposures. Exposure of an update key SKU_{t-1} automatically exposes the subsequent refresh key $SKR_{t.0}$, because they are sent together in one message.

Adversary's queries to *Osig* and *Osec* must have the $t.r$ values within the appropriate bounds. It may be reasonable to require the adversary to "respect erasures" and prohibit a value that should have been erased from being queried (or used for signature computation). However, this restriction is optional, and in fact we do not require it here. Note, that respecting erasures is local to signer and base and is different from requiring any kind of global synchronization.
For any set of valid 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 refresh**] if $r > 1$, ("r", $t.(r-1)$) $\in Q$, and $SKS_{t.(r-1)}$ is Q-exposed; or
[**via update**] if $r = 1$, ("u", $t-1$) $\in Q$, and $SKS_{(t-1).RN(t-1)}$ is Q-exposed.

Replacing SKS with SKB throughout the above definition yields the definition of base key exposure (or more precisely, of $SKB_{t.r}$ being *Q-exposed*). Both definitions are recursive, with direct exposure as the base case.

Clearly, exposure of a signer key $SKS_{t.r}$ for the given t and any r enables the adversary to generate legal signatures for this period t. Similarly, simultaneous exposure of both base and signer keys ($SKB_{t.r}, SKS_{t.r}$, for some t, r) allows the adversary to run the algorithms of definition **??** to generate valid signatures for any messages for all later periods $t' \geq t$.

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
- $SKS_{t'.r}$ and $SKB_{t'.r}$ are both Q-exposed for some $t' < t$.

In other words, a particular time period has been rendered insecure 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.[8] If the scheme is (j, Q)-compromised, then clearly adversary, possessing the secrets returned by *Osec* in response to queries in Q, can generate signatures for the period t.

[8] Interestingly, and perhaps counter-intuitively, a secret that has not been exposed might still be deduced by adversary. For example, it may be possible in some implementations to compute $SKB_{t.r}$ from $SKB_{t.r+1}$ and $SKR_{t.r}$ (similarly for update and for the signer). The only case we cannot allow —in order to stay consistent with our definition— is that of adversary computing $SKS_{t.r}$ without Q-exposing some $SKS_{t.r'}$. So, it may be possible to compute $SKS_{t.r}$ from $SKS_{t.r+1}$ and $SKR_{t.r}$, but not from $SKS_{t+1.0}$ and $SKU_{t.r}$ (even if $r = RN(t)$).

The following experiment captures adversary's functionality. Intuitively, adversary succeeds if she generates a valid signature without "cheating": not obtaining this signature from *Osig*, asking only legal queries (e.g. no out of bounds queries), and not compromising the scheme for the given time period. We call this adversary "adaptive" because she is allowed to decide which keys and signatures to query based on previous answers she receives.

Experiment `Run-Adaptive-Adversary`(F, k, T, RN)
  `Generate-Keys`(k, T, RN)
  $(m, j, sig) \leftarrow F^{Osig,Osec}(1^k, T, PK, RN)$
  Let Q be the set of key exposure queries F made to *Osec*;
  `if` $Ver(m, j, sig) =$"invalid" `or` (m, j) was queried by F to *Osig* `or` there
        was an illegal query `or` the scheme is (j, Q)-compromised
    `then return` 0
    `else return` 1

We now define security for the intrusion resilient signature schemes.

Definition 2. *Let* $\mathsf{SiBIR}[k, T, RN]$ *be a* $\langle$*signer-base*$\rangle$ *key-evolving scheme with security parameter* k*, number of time periods* T*, and table* RN *of* T *refresh numbers. For adversary* F*, define adversary success function as*

$$\mathbf{Succ}^{\mathrm{IR}}(F, \mathsf{SiBIR}[k, T, RN]) \stackrel{def}{=} \Pr[\texttt{Run-Adaptive-Adversary}(F, k, T, RN) = 1] .$$

Let insecurity function $\mathbf{InSec}^{\mathrm{IR-adaptive}}(\mathsf{SiBIR}[k, T, RN], \tau, q_{\mathrm{sig}})$ *be the maximum of* $\mathbf{Succ}^{\mathrm{IR}}(F, \mathsf{SiBIR}[k, T, RN])$ *over all adaptive adversaries* F *that run in time at most* τ *and ask at most* q_{sig} *signature queries.*

Finally, $\mathsf{SiBIR}[k, T, RN]$ *is* $(\tau, \epsilon, q_{\mathrm{sig}})$-intrusion-resilient *if*

$$\mathbf{InSec}^{\mathrm{IR-adaptive}}(\mathsf{SiBIR}[k, T, RN], \tau, q_{\mathrm{sig}}) < \epsilon .$$

Although the notion of (j, Q)-compromise depends only the *set* Q, it is important how Q is generated by the adversary. Allowing the adversary to decide her queries based on previous answers gives her potentially more power.

While we do not see an attack on our scheme in Sec. **??** by a fully adaptive adversary, the proof for such a strong adversary seems elusive. Instead, we consider two types of slightly restricted adversaries. The first type, which we call *partially adaptive*, is allowed to adaptively choose queries to *Osig*, but not to *Osec*. To be precise, the experiment `Run-Adaptive-Adversary` is modified `Run-Partially-Adaptive-Adversary` by requiring F to output the set Q of *all* her *Osec* queries before she makes any of them. The rest of the definitions remain the same, giving us $\mathbf{InSec}^{\mathrm{IR-partially-adaptive}}$ instead of $\mathbf{InSec}^{\mathrm{IR-adaptive}}$.

The second type of restricted adversary is *partially synchronous*. She may select all of her queries adaptively, but is not allowed to go back in time "too far." Specifically, upon querying any key in time period t, she is not allowed to query keys in time period $t - 2$ (note that the choice of 2 is arbitrary and can be replaced with another constant). This is a reasonable assumption in practice, essentially saying that the base, the network and the signer can be at most one time period apart at any given time. In order to formally define security against such an adversary, we simply expand the definition of "illegal queries" to encompass the above restriction. The rest of the definitions remain the same, giving us $\mathbf{InSec}^{\mathrm{IR-partially-synchronous}}$ instead of $\mathbf{InSec}^{\mathrm{IR-adaptive}}$.

3 Intrusion-Resilient Scheme: Construction

Our scheme, which we call SiBIR1, is based on the [?] forward-secure signature scheme (which will call $\mathsf{FSIG}^{\mathsf{IR}}$). In turn, $\mathsf{FSIG}^{\mathsf{IR}}$ is based on the Guillou-Quisquater [?] ordinary signature scheme. In fact, the [?] 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). In the interests of conciseness, we do not present the rationale behind $\mathsf{FSIG}^{\mathsf{IR}}$ here. We do, however, recall how keys are generated and updated in the $\mathsf{FSIG}^{\mathsf{IR}}$ scheme (utilizing our own notation, rather than notation used in [?], in the interests of clarity)

Keys in $\mathsf{FSIG}^{\mathsf{IR}}$. Both the public and the secret keys contain a modulus $n = p_1 p_2$, where p_1 and p_2 are $(k/2)$-bit *safe* primes: $p_i = 2q_i + 1$ such that q_i are odd primes (such q_i, satisfying $2q_i+1$ is prime, are known as *Sophie Germain primes*).[9] The public key also contains a value $v \in Z_n^*$. For each time period t, there is a corresponding $(l+1)$-bit exponent e_t that the signer can easily compute (we require all the exponents to be relatively prime; then they need not be stored in the public key, but rather the appropriate e_t is included in each signature—see [?] for further details).

Messages during time period t are signed using the secret value $\widehat{s}_t$, such that $\widehat{s}_t^{e_t} \equiv 1/v \pmod{n}$.

The factorization of n must be erased after key generation in order to achieve forward security. Knowledge of secrets $\widehat{s}_t, \widehat{s}_{t+1}, \ldots, \widehat{s}_T$, is equivalent (by the so-called "Shamir's trick" [?]—see Proposition1 in [?]) to knowledge of the 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 $\widehat{s}_{[t,T]}$: then $\widehat{s}_{[t,T]}^{e_{[t,T]}} \equiv 1/v \pmod{n}$.

At key generation we actually select $\widehat{s}_{[1,T]}$ at random, and compute v as $v \leftarrow 1/\widehat{s}_{[1,T]}^{e_{[1,T]}} \bmod n$.

Subsequently (just before each time period t), $\widehat{s}_{[t,T]}$ is updated as follows: $\widehat{s}_{[t+1,T]} \leftarrow \widehat{s}_{[t,T]}^{e_t} \bmod n$; and the "current" signing secret is computed as $\widehat{s}_t \leftarrow \widehat{s}_{[t,T]}^{e_{[t+1,T]}} \bmod n$.

Key generation and update in SiBIR1. We use essentially the same keys in SiBIR1. However, in order to achieve intrusion-resilience, $\widehat{s}_{[t,T]}$ is never stored

[9] See [?] for an excellent discussion on efficient generation of safe primes and short primes.

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 $\widehat{s}_{[t,T]} = s_{[t,T]} b_{[t,T]}$. 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 $\widehat{s}_t$.

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

Also, following the convention that the only "storage" available to the base and the signer is the secret key, we store some values in the secret key that need not really be secret, such as the current time period and the information needed to regenerate the e_t values. The only values that the signer needs to keep secret are $s_{t,T}$ and $\widehat{s}_t$; the only values that the base needs to keep secret are $b_{t,T}$.

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

```
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
      (Some seed ℰ can be used with H to generate these e_1, ..., e_T.
       This ℰ might need to be stored for later regeneration of e_1, ..., e_T.)
  s_[1,T] ←R Z*_n; SKS_0 ← (0, T, n, ∅, s_[1,T], ∅, ℰ)  % ∅'s will be filled in in 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)
```

Fig. 1. *Key generation. Refresh index on the keys is omitted to simplify notation.*

Distributing 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 (without factoring), 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 amount 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.

$$
\begin{array}{l}
\texttt{algorithm}\ \mathsf{SiBIR1}.\mathcal{UB}(SKB_t)\\
\quad \text{Let } SKB_t = (t < T, T, n, b_{[t+1,T]}, \mathcal{E})\\
\quad \text{Regenerate } e_{t+1}, \ldots, e_T \text{ using } \mathcal{E}\\
\quad b_{t+1} \leftarrow b_{[t+1,T]}^{e_{t+2}\cdot\ldots\cdot e_T} \bmod n;\ b_{[t+2,T]} \leftarrow b_{[t+1,T]}^{e_{t+1}} \bmod n\\
\quad \texttt{return}\ (SKB_{t+1} = (t+1, T, n, b_{[t+2,T]}, \mathcal{E}), SKU_t = b_{t+1})
\end{array}
$$

$$
\begin{array}{l}
\texttt{algorithm}\ \mathsf{SiBIR1}.\mathcal{US}(SKS_t, SKU_t)\\
\quad \text{Let } SKS_t = (t < T, T, n, \widehat{s}_t, s_{[t+1,T]}, e_t, \mathcal{E});\ SKU_t = b_{t+1}\\
\quad \text{Regenerate } e_{t+1}, \ldots, e_T \text{ using } \mathcal{E}\\
\quad s_{t+1} \leftarrow s_{[t+1,T]}^{e_{t+2}\cdot\ldots\cdot e_T} \bmod n;\ s_{[t+2,T]} \leftarrow s_{[t+1,T]}^{e_{t+1}} \bmod n\\
\quad \widehat{s}_{t+1} \leftarrow s_{t+1} b_{t+1} \bmod n\\
\quad \texttt{return}\ SKS_{t+1} = (t+1, T, n, \widehat{s}_{t+1}, s_{[t+2,T]}, e_{t+1}, \mathcal{E})
\end{array}
$$

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

Refresh. Because the signer and the base share a single value multiplicatively, the refresh algorithm presented in Figure **??** 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).

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

$$
\begin{array}{l}
\texttt{algorithm}\ \mathsf{SiBIR1}.\mathcal{RS}(SKS_{t.r}, SKR_{t.r})\\
\quad \text{Let } SKS_{t.r} = (t, T, n, \widehat{s}_t, s_{[t+1,T]}, e_t, \mathcal{E});\ SKR_{t.r} = R_{t.r}\\
\quad s_{[t+1,T]} \leftarrow s_{[t+1,T]} \cdot R_{t.r}\\
\quad \texttt{return}\ SKS_{t.r+1} = (t, T, n, \widehat{s}_t, s_{[t+1,T]}, e_t, \mathcal{E})
\end{array}
$$

Fig. 3. *Key refresh algorithms*

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

```
algorithm SiBIR1.Sign(M, SK_t) %same as IR.Sign in [?]
  Let SK_t = (t, T, n, ŝ_t, s_[t+1,T], e_t, ℰ)
  x ←R Z*_n
  y ← x^{e_t} mod n
  σ ← H(t, e_t, y, M)
  z ← x ŝ_t^σ mod n
  return (z, σ, t, e_t)
```

```
algorithm SiBIR1.Ver(M, PK, (z, σ, t, e)) %same as IR.Ver in [?]
  Let PK = (n, v, T)
  if e ≥ 2^l (1 + t/T) or e < 2^l or e is even then return 0
  if z ≡ 0 (mod n) then return 0
  y' ← z^e v^σ mod n
  if σ = H(t, e, y', M) then return 1 else return 0
```

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

at a time. To enable the signer to compute $\widehat{s}_{t_1}, \widehat{s}_{t_1+1}, \ldots \widehat{s}_{t_2}$, 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 [**?**]). 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

We use a variant of the strong RSA assumption (introduced in [**?**] and [**?**], our variant is identical to the one in [**?**]), 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 $\beta^r = \alpha$.

In our version, 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 $\mathbf{Succ}^{\mathrm{SRSA}}(A, k, l) = \Pr[\text{Break-Strong-RSA}(k, l, A) = 1]$. Let the "insecurity function" $\mathbf{InSec}^{\mathrm{SRSA}}(k, l, \tau)$ be the maximum of $\mathbf{Succ}^{\mathrm{SRSA}}(A, k, l)$ 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 [**?**], although the two proofs are quite similar. Both are based on the forking lemma of [**?**].

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

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

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

A similar theorem also holds for partially synchronous adversary. The proofs of these theorems will be given in the full version of the paper.

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 **??**, 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.

Acknowledgements We thank Nenad Dedic, Scott Russell, Yael Tauman and the anonymous referees for helpful comments.

References

[AMN01] Michel Abdalla, Sara Miner, and Chanathip Namprempre. Forward-secure threshold signature schemes. In David Naccache, editor, *Progress in Cryptology — CT-RSA 2001*, *Lecture Notes in Computer Science* 2020, 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.

[CS00] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000.

[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.

[DKXY] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong key-insulated signature schemes. Unpublished Manuscript.

[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 Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 153–168. Springer-Verlag, 23–27 August 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.

[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*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354. Springer-Verlag, 19–23 August 2001.

[Kat02] Jonathan Katz. A forward-secure public-key encryption scheme. Cryptology ePrint Archive, Report 2002/60, 2002. `http://eprint.iacr.org/`.

[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.

[MR01a] Philip D. MacKenzie and Michael K. Reiter. Delegation of cryptographic servers for capture-resilient devices. In *Eighth ACM Conference on Computer and Communication Security*, pages 10–19. ACM, November 5–8 2001.

[MR01b] Philip D. MacKenzie and Michael K. Reiter. Networked cryptographic devices resilient to capture. In *IEEE Symposium on Security and Privacy*, pages 12–25, 2001.

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

[Sha83] Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems*, 1(1):38–44, 1983.

[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.