Strong Mobile Device Protection from Loss and Capture

Zhengyi Le[1] Matt Bishop[2] Fillia Makedon[1]
[1]Heracleia Human-Centered Computing Lab
Department of Computer Science and Engineering, University of Texas at Arlington, USA
{zyle, makedon}@uta.edu
[2]Department of Computer Science, University of California, Davis, USA
{bishop}@cs.ucdavis.edu

ABSTRACT

Assistive environments employ multiple types of devices to monitor human actions and identify critical events for physical safety. Some of the devices must be wireless in order to be nonintrusive. This introduces the problem of authenticating these devices and building secure communication channels among them. The traditional way is to assign a private key to a device for digital identification. In this paper, we present an approach to protect the private key by introducing a third party and bilaterally and proactively generating a random number to refresh key shares based on Bellare and Miner's forward secure signature scheme. This improves the resilient mediated RSA solution because the entire private key is also updated periodically. In this way, if an attacker steals one key share, he only can use it for a limited period of time because it will be obsolete immediately after the next refresh operation. Even if he compromises both key shares simultaneously, the digital signatures generated by previous private keys are still secure. Our scheme is proven to be intrusion resilient based on the CDH assumption in the random oracle model. The construction is also quite efficient.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public key cryptosystems; D.4.6 [**Security and Protection**]: Cryptographic controls; H.3.5 [**Online Information Services**]: Data Sharing; J.3 [**Life and Medical Sciences**]: Medical Information Systems

General Terms

Authentication, Digital Signature

Keywords

Assistive Environment, Mobile Device, Forward Security

1. INTRODUCTION

Much attention in the literature of assistive environments focuses on building secure communications channels under the assumption that a cryptographic key is secure. The basis for using public key cryptography for authentication, for example, assumes that only the individual being identified knows, or has access to, her public key. In theory, this is correct. But in practice, many other entities, including the system being used, have access. What happens if one of these other entities steals the key?

This *key exposure problem*—that, in practice, stealing a private key is often easier than breaking the cryptosystem behind it—motivates our work. If the method of attack is theft, a private key's safety relies on the security of both the underlying operating system and the storage of the private key. For example, in GnuPG and S/MIME, a private key is encrypted using a user password and stored as a file in a hard drive. Berger *et al.* demonstrated an efficient password attack based on keyboard acoustic emanations in their paper [5]. In addition, the insider threat has attracted increased attention [6]; many of these attacks take aim at trusted users and steal their private keys.

Existing approaches to protecting these secrets are (1) using physically secure devices for storage and computation, such as the secure coprocessor, and other chips like IBM TCPA, (2) splitting a secret, *e.g.*, by using threshold cryptography, and distributing the parts throughout one or more systems, (3) updating secrets periodically through proactive cryptography or forward security, and (4) some hybrid techniques. Our scheme falls into the hybrid category and uses secret sharing and refreshing to solve this problem.

The *Forward Secure signature scheme (FS)* by Bellare and Miner [3] is the basis for our scheme. They divide time into periods: $0, 1, 2, \ldots, T$. The public key PK is fixed, and the corresponding private key is changed every period by applying a one-way function: SK_0, SK_1, SK_2,..., SK_T. In period i ($0 \leq i \leq T$), a message m is signed by the current private key SK_i and the current time period index i. To verify the signature σ of m, a receiver must use the fixed PK and the time period index i with which the message was signed. If a private key SK_i is compromised, the previous signatures signed by SK_j ($0 \leq j < i$) are still valid. So, this scheme mitigates the damage caused by private key exposure. However, because the key changing algorithm is one-way and public, it is computationally hard to reverse the process to obtain the previous keys from the compromised key SK_i but it is easy to derive the private keys for future periods. So, the scheme is compromised and future signatures should be disabled after the exposure period. The difficulty is how to identify the exposure period and how to protect messages signed between exposure and detection. Thus the challenge

is whether there is a method to recover the security in time when an intrusion succeeds.

This paper suggests a solution, the *Intrusion-Resilient Two-Party Signature scheme (I2S)*, that protects the FS private keys against the above problem. A semi-trusted third party, called *base* (possibly a portable physical device or a server), stores a partial secret and generates partial signatures. This is actually a 2-out-of-2 variant of an existing threshold forward secure signature scheme [1]. However, in order to provide the intrusion resilient property, we interactively refresh the key shares with the help of bilateral random number generation. This prevents eavesdropping and spoofing: after refresh, the compromised key share immediately becomes invalid and the refreshed key share remains secure. In addition, because we preserve the forward secure property—each key share is updated periodically, so an attacker cannot derive the previous key shares from the exposed one—exposure of the current key share will not compromise past or future secrets. This improves the resilient mediated RSA solution [22]. If an attacker succeeds in stealing the user's share, he can impersonate the user only for a limited time before the next key refresh.

The next section, section 2 presents related work. The function definitions and the security models for our I2S scheme are described in Section 3. Section 4 gives the I2S algorithms. In Section 5 we prove that I2S is forward secure and intrusion resilient. Section 6 discuss other benefits, such as fast revocation and server witness, the selection of refresh frequency, and promising applications.

2. RELATED WORK

Forward security (FS) means that a compromise of the current private key does not enable an attacker to forge signatures pertaining to the past. The goal of forward security is to mitigate the damage caused by the exposure of a secret. After the first practical scheme was introduced [3], many improvements and derivative FS schemes have been published (e.g., [2, 17, 19, 20, 10, 4, 8]). The problem is that a user controls the entire private key, so the compromise of the current secret will disable the future use of the system. Furthermore, the revocation mechanism is still necessary for the FS schemes when exposure happens. So, this approach does not alleviate the burden of public key management.

A desired property might be to provide both forward security and backward security. Burmester first formally called this property *Strong Forward Security (SFS)* [9]. One current approach to providing SFS is to use a new public/private key pair for each period [9]. In this method, the public key also needs to be updated. This introduces an additional cost of issuing and revoking public keys that grows with the update frequency. In order to guarantee security, the key pairs must be updated frequently, which makes the cost problematic. Another approach is to use *threshold cryptography*, and distribute the secret among multiple trusted agents or servers. Updating the private key is then a distributed process that requires collaboration from all existing participants [28, 1]. However, once an attacker compromises a key share, he can impersonate that share holder until the public key expires or this intrusion is detected. A third approach to SFS is *key insulation*, which uses a physically secure device to store a master key. In this method, a private key update cannot be performed without the help of the master key [12, 13]. This technology is not likely to be adopted soon in practical settings due to concerns about efficiency.

Proactive cryptography is another approach to protecting secrets. Key shares in each party are refreshed periodically, but the entire private key is unchanged during the life time of the public key, as for example in [15], [26], and [24]. A proactive cryptosystem remains secure as long as the adversary does not corrupt more than t parties in each time period. The shares of corrupted parties become useless when time enters the next time period. The advantage of this method is that an adversary has only a short period of time to break into any t out of the n servers, while in the long-lived threshold systems the adversary has a long time to break into any t servers. This is the case even if the adversary obtained any $t'(t' < t)$ shares in the past time periods that are invalid in the new time periods. Thus, the proactive mechanism enhances the security of the threshold scheme.

The notion of *intrusion resilient security* combines the features of forward, key-insulated and proactive security paradigms. Ikis and Reyzin proposed the first intrusion resilient signature in [18]. Soon after, Itkis gave a generic construction of intrusion signatures without random oracles [16]. Like key-insulated schemes, they involve a device to store a master secret. This device, called a *base*, is assumed to be physically secure. A user holds the entire private key and is able to update it independently, whereas the key refresh operation needs the base to send the user a partial secret. Libert *et. al.* [23] proposed another signature construction based on Water's signature scheme [27], a hierarchical key derivation technique [7], and a generic conversion method [11].

3. SECURITY MODEL

Our definitions are based on forward security [3] and its signer-base follow-up [18]. Its differences from the previous related work are as follows. First, this is a two-party signature scheme, so signing a message needs the collaboration of both parties; in [3] and [18], a user holds the entire private key so he can sign by himself. Secondly, a user or the base can perform periodic key share updates independently but synchronously ; but in [18], the user needs a secret update message from the base. Third, our new method allows key share refreshes to be performed during an arbitrary period. A compromised secret will become useless after a refresh and the system remains safe. This is a desirable feature that [3] cannot provide.

3.1 Functional Definition

Here we give the functional definition of the components of the systems. A public key, denoted PK, remains the same during the lifetime of its certificate. The corresponding private key is composed of two shares, $SKB_{0,0}$ and $SKU_{0,0}$, held by the base and the user respectively. The life time is divided into T periods. At the end of each period, *e.g.,* t where $0 \leq t < T$, the base and the user independently update their shares, $SKB_{t,r}$ to $SKB_{t+1,0}$ and $SKU_{t,r}$ to $SKU_{t+1,0}$, where r is the number of times the share has been refreshed since the last update. After every signing operation (which could happen at any time), the user and the base interactively refresh both shares, so $SKB_{t,r}$ is changed to $SKB_{t,r+1}$ and $SKU_{t,r}$ to $SKU_{t,r+1}$.

DEFINITION 3.1.1. *An intrusion-resilient two-party signature scheme is an octuple of probabilistic polynomial-time*

($\mathcal{PPT}$) algorithms (Gen, BS.Sgn, UR.Sgn, Vrf, BS.Upd, UR.Upd, BS.Rfs, UR.Rfs):

1. *Gen, the key generation algorithm.*

   In: *security parameters and the total number T of time periods*

   Out: *the public key PK, the initial user private key $SKU_{0,0}$, and the initial base private key $SKB_{0,0}$*

2. *BS.Sgn, the base signing algorithm.*

   In: *current base key $SKB_{t,r}$, and message m*

   Out: *partial signature $\langle t, z_1\rangle$ on message m for time period t*

3. *UR.Sgn, the user signing algorithm.*

   In: *current user key $SKU_{t,r}$, and message m*

   Out: *entire signature $\langle t, \sigma\rangle$ on m for time period t*

4. *Vrf, the verifying algorithm.*

   In: *message m, signature $\langle t, \sigma\rangle$, and the public key PK*

   Out: *"valid" or "invalid"*

5. *BS.Upd, the base update algorithm.*

   In: *current base key $SKB_{t,r}$*

   Out: *new base key $SKB_{t+1,0}$*

6. *UR.Upd, the user update algorithm.*

   In: *current user key $SKU_{t,r}$*

   Out: *new user key $SKU_{t+1,0}$*

7. *BS.Rfs, the base refresh algorithm.*

   In: *current base key $SKB_{t,r}$, $\langle h_2, v_2\rangle$ and later g^b from the user*

   Out: *new base key $SKB_{t,r+1}$*

8. *UR.Rfs, the user refresh algorithm.*

   In: *current user key $SKU_{t,r}$, $\langle h_1, v_1\rangle$, and later g^a from the base*

   Out: *new user key $SKU_{t,r+1}$*

Note that when a message m needs a signature, both parties generate partial signatures z_1 and z_2 and then the user combines them together to make a complete signature, σ. Signature verifiers must be aware of the updates because they need the correct period index i as input to verify a signature while the refreshes are transparent to them.

3.2 Security Definition

We assume the number of times the key shares are refreshed in period t is R. Actually, R need be neither given or fixed; it is used only for notational convenience. We follow the notations in [18] and [3] to define security. Let $\mathcal{F}$, the adversary, be a $\mathcal{PPT}$ oracle Turing machine with the following oracles:

- *Osig*, the signing oracle, which
  1. on input (m, t) for $0 \le t \le T$ outputs σ.
  2. on input ("b", $m, t.r$) for $0 \le t \le T, 1 \le r \le R$ outputs z_1;
  3. on input ("u", $m, t.r$) for $0 \le t \le T, 1 \le r \le R$ outputs z_2.
- *Osec*, the secret exposure oracle, which
  1. on input ("b", $t.r$) for $1 \le t \le T, 1 \le r \le R$ outputs $SKB_{t,r}$;
  2. on input ("u", $t.r$) for $1 \le t \le T, 1 \le r \le R$ outputs $SKU_{t,r}$;
  3. on input ("rfs", $t.r$) for $1 \le t \le T, 1 \le r \le R$ output γ.

First, a restricted adversary $\mathcal{F}_1$ is defined. She asks only legal queries of *Osig* for the current period and she is able to choose a point of time j, r to break the key share of one party. Then she will try to forge signatures using $SKU_{j,a}$ for some $a > r$, and succeed if the signature is valid and the message is new. The following experiment captures the adversary's functionality.

Experiment Run-Intrusion($\mathcal{F}_1, k, l, T$)
Select $H : \{0,1\}^* \to \{0,1\}^l$ at random;
$Gen(k, l, T)$;
Choose an exposure point $j.r$ by $\mathcal{F}_1$ to *Osec*;
$(m, j, \sigma) \leftarrow \mathcal{F}_1^{H, Osec}$(forge);
If $Vrf(m, j, \sigma) = valid$ and $j.a > j.r$
and $(m, j.a)$ was not queried by $\mathcal{F}_1$ to *Osig*
then return 1
else return 0

In order to define the security against the above adversary, a security function called $\mathbf{Succ}^{\mathrm{ir}}$ was introduced in [3]. The $\mathbf{Succ}^{\mathrm{ir}}(\mathsf{I2S}[k, l, T], \mathcal{F}_1)$ is defined the probability that the above adversary who knows one key share succeeds in forging signatures belonging to other periods. Then, the value of the insecurity function is defined to be the maximum probability of success over all $\mathcal{PPT}$ adversaries. We say that our scheme is secure if the success probability of any $\mathcal{PPT}$ adversary is negligible.

DEFINITION 3.2.1. *Let $\mathsf{I2S}[k, l, T]$ be our intrusion-resilient two-party signature scheme with security parameter k, hash function output length l, and number of time periods T. For adversary $\mathcal{F}_1$, define the adversary success function as*

$$\begin{aligned} &\mathbf{Succ}^{\mathrm{ir}}(\mathsf{I2S}[k, l, T], \mathcal{F}_1) \\ &\stackrel{def}{=} \Pr[\mathrm{Run-Intrusion}(\mathcal{F}_1, k, l, T) = 1]. \end{aligned} \tag{1}$$

Then, the insecurity function $\mathbf{InSec}^{\mathrm{ir}}(\mathsf{I2S}[k, l, T]; \tau, q_{sig}, q_{hash})$ was the maximum of $\mathbf{Succ}^{\mathrm{ir}}(\mathcal{F}_1, \mathsf{I2S}[k, l, T])$ over all adaptive adversaries $\mathcal{F}_1$ that run in time at most τ and ask at most q_{sig} queries.

$$\begin{aligned} &\mathbf{InSec}^{\mathrm{ir}}(\mathsf{I2S}[k, l, T]; \tau, q_{sig}, q_{hash}) \\ &= \max_{\mathcal{F}_1}\{\mathbf{Succ}^{\mathrm{ir}}(\mathcal{F}_1, \mathsf{I2S}[k, l, T]). \end{aligned} \tag{2}$$

Finally, $\mathsf{I2S}[k, l, T]$ is intrusion resilient if

$$\mathbf{InSec}^{\mathrm{ir}}(\mathsf{I2S}[k, l, T]; \tau, q_{sig}, q_{hash}) < \epsilon_1. \tag{3}$$

where ϵ_1 is negligible.

Next, a stronger adversary $\mathcal{F}_2$ was defined and it also follows the definition in [3]. The adversary is allowed a chosen-message attack (cma) before she breaks into both parties simultaneously. After the break-in, she cannot access $Osig$ again and will try to forge a signature belonging to a previous time period. The adversary's functionality is described by the following experiment.

Experiment Run-Forge($\mathcal{F}_2, k, l, T$)
Select $H : \{0,1\}^* \to \{0,1\}^l$ at random;
$Gen(k, l, T)$;
Repeat
mount attacks by $\mathcal{F}_2^{H,Osig}$(cma);
Until the exposure time point j, r;
$(m, b, \sigma) \leftarrow \mathcal{F}_2^{H,Osec}$(forge);
If $Vrf(m, b, \sigma) = valid$ and $b < j$
and (m, b) was not queried by $\mathcal{F}_2$ to $Osig$
then return 1
else return 0

The corresponding adversary success function and system insecurity function are as follows:

DEFINITION 3.2.2. *The adversary success function is*

$$\begin{aligned} &\mathbf{Succ}^{\text{fs}}(\mathsf{I2S}\,[k, l, T], \mathcal{F}_2) \\ &\stackrel{def}{=} \Pr\left[\text{Run} - \text{Forge}(\mathcal{F}_2, k, l, T) = 1\right]. \end{aligned} \tag{4}$$

and the insecurity function is

$$\begin{aligned} &\mathbf{InSec}^{\text{fs}}(\mathsf{I2S}\,[k, l, T]; \tau, q_{sig}, q_{hash}) \\ &= \max_{\mathcal{F}_2}\{\mathbf{Succ}^{\text{fs}}(\mathcal{F}_2, \mathsf{I2S}\,[k, l, T]). \end{aligned} \tag{5}$$

Finally, $\mathsf{I2S}\,[k, l, T]$ *is forward secure if*

$$\mathbf{InSec}^{\text{fs}}(\mathsf{I2S}\,[k, l, T]; \tau, q_{sig}, q_{hash}) < \epsilon_2. \tag{6}$$

where ϵ_2 *is negligible.*

4. INTRUSION-RESILIENT TWO-PARTY SIGNATURE SCHEME

This section first describes our main scheme I2S and then specifies its basic property: validity. Its security will be analyzed in the next section.

I2S is based on the first practical forward secure signature scheme [3] (which we refer to as FSS), which in turn is based on the Fiat-Shamir [14] and Ong-Schnorr [25] identification and signature schemes. Intuitively, I2S first uses multiplicative secret sharing to extend FSS to a two-party paradigm, then a random number negotiation is introduced to refresh the key shares. In order to protect our system against random number spoofing, the user and the base use the Diffie-Hellman (DH) algorithm with bit commitment to collaboratively determine a new secret.

In the system setup phase (Algorithm 1) two distinct primes p and q that are congruent to 3 mod 4 are chosen at random. Their product N, called the Blum-Williams integer, serves as the modulus. A user's key share is a random series $x_{1,0.0}, x_{2,0.0}..., x_{l,0.0}$ in $\mathbb{Z}_N^*$, and likewise for a base's key share $y_{1,0.0}, y_{2,0.0}, ..., y_{l,0.0}$. The public key contains the modulus N, the total number of time periods T, and a series $u_1, u_2..., u_l$. Each u_i is generated by raising the product of $x_{i,0}$ and $y_{i,0}$ to the power of 2^{T+1}.

When there is a request to sign a message in time period j, the user and the base each generate a random number, r_1 and r_2 respectively, in $\mathbb{Z}_N^*$, raise them to the power of 2^{T+1-j}, and then exchange the results. The user multiplies the two values to get w and inserts w as a component of the final signature to commit it. $H : \{0,1\}^* \to \{0,1\}^l$ is a public hash function that generates the l-bit series $c_1c_2...c_l$ from inputs j, w and m. The user raises every unit of his key share $x_{i,j.r}$ to the power of c_i and multiplies all of them with the r_1 value she previously committed in w to get a partial signature z_1. The base does the same. Then the two parties exchange their results and the user generates z by multiplying z_1 and z_2 as another component of the final signature σ. Refer to Algorithm 2.1 and 2.2 for details.

When a verifier wants to verify a signature generated in period j, she re-extracts $c_1c_2...c_l$ from j, w and m, and raises every unit of the public key u_i to the power of c_i. Let τ refer to the product of all of them and w. If z is the 2^{T+1-j}-th root of τ, it is a valid signature. Algorithm 3 provides the details.

Fig. 2 gives algorithms for updating and refreshing. Key updates are executed at the end of every time period. Each share holder simply squares every unit of its key share and increases the current period index by one. Key refreshes are required to be executed immediately following each key update and signing operation. Note that key updates are periodic while signing could happen at any time and with no time limits. Both parties use the DH number g^{ab} as the new secret to refresh two key shares. Prior to this, the base hashes g^a and a random number v_1 to get h_1, which functions as the commitments for g^a; likewise, the user obtains h_2. One party multiplies her share by the new secret number and the other party multiplies her share by the inverse of that number in the multiplicative group of integers modulo $\hat{p}$. The Extended Euclidean Algorithm can take γ and $\hat{p}$ as inputs to calculate $\gamma^{-1} \bmod \hat{p}$.

The following proposition proves the validity of genuine signatures.

PROPOSITION 4.0.3. *Let* $PK = (N, T, u_1, \ldots, u_l)$, $SKB_{0,0} = (N, T, x_{1,0.0}, ..., x_{l,0.0})$, *and* $SKU_{0,0} = (N, T, y_{1,0.0}, ..., y_{l,0.0})$ *be keys generated by the key generation algorithm,* $Gen(k, l, T)$. *Let* $\sigma = \langle j, (w, z)\rangle$ *be an output of signing algorithm* $UR.Sgn(m, j, SKU_{j,r})$. *Then* $Vrf(m, j, \sigma, PK) = 1$.

PROOF.

$$\begin{aligned} & z^{2^{T+1-j}} (\text{mod} N) \\ &= (z_1 z_2)^{2^{T+1-j}} \\ &= (r_1 \Pi_{i=1}^{l} (x_{i,j.r})^{c_i} \cdot r_2 \Pi_{i=1}^{l} (y_{i,j.r})^{c_i})^{2^{T+1-j}} \\ &= r_1^{2^{T+1-j}} \cdot r_2^{2^{T+1-j}} \cdot \Pi_{i=1}^{l} (x_{i,j.r} \cdot y_{i,j.r})^{c_i \cdot 2^{T+1-j}} \\ &= w_1 w_2 \Pi_{i=1}^{l} ((x_{i,j-1} \cdot \gamma \cdot y_{i,j-1} \cdot \gamma^{-1})^{2^{T+1-j}})^{c_i} \\ &= w \Pi_{i=1}^{l} ((x_{i,0.0}^{2^j} \cdot y_{i,0.0}^{2^j})^{2^{T+1-j}})^{c_i} \\ &= w \Pi_{i=1}^{l} u_i^{c_i} \end{aligned} \tag{7}$$

as desired. □

1. Algorithm: $Gen(k,l,T)$

$p,q \leftarrow k/2$ bit random prime, such that $p,q \equiv 3 \bmod 4$;
$N \leftarrow pq$;
for $i = 1$ to l do
  $x_{i,0.0} \overset{R}{\leftarrow} \mathbb{Z}_N^*$;
  $y_{i,0.0} \overset{R}{\leftarrow} \mathbb{Z}_N^*$;
  $u_i \leftarrow (x_{i,0.0} \cdot y_{i,0.0})^{2^{T+1}} \bmod N$
end for
$SKU_{0,0} \leftarrow (N, T, 0, x_{1,0.0}, ..., x_{l,0.0})$;
$SKB_{0,0} \leftarrow (N, T, 0, y_{1,0.0}, ..., y_{l,0.0})$;
$PK \leftarrow (N, T, u_1, ..., u_l)$;
return $\langle PK, SKB_{0,0}, SKU_{0,0}\rangle$

2.1 Algorithm: $UR.Sgn(m,j,SKU_{j,r})$

$r_1 \leftarrow_r \mathbb{Z}_N^*$;
$w_1 \leftarrow r_1^{2^{T+1-j}} \bmod N$;
send w_1;
receive w_2;
$w \leftarrow w_1 w_2$;
$c_1...c_l \leftarrow H(j,w,m)$;
$z_1 \leftarrow r_1 \Pi_{i=1}^{l} x_{i,j.r}^{c_i} \bmod N$;
receive z_2;
$z \leftarrow z_1 z_2$;
$\sigma \leftarrow \langle j, (w,z)\rangle$;
return σ

2.2 Algorithm: $BS.Sgn(m,j,SKB_{j,r})$

$r_2 \leftarrow_r \mathbb{Z}_N^*$;
$w_2 \leftarrow r_2^{2^{T+1-j}} \bmod N$;
send w_2;
receive w_1;
$w \leftarrow w_1 w_2$;
$c_1...c_l \leftarrow H(j,w,m)$;
$z_2 \leftarrow r_2 \Pi_{i=1}^{l} y_{i,j.r}^{c_i} \bmod N$;
send z_2;
return

3. Algorithm: $Vrf(m,j,\sigma,PK)$

$c_1...c_l \leftarrow H(j,w,m)$;
$\tau = w \Pi_{i=1}^{l} u_i^{c_i} \bmod N$;
if $\tau = z^{2^{T+1-j}}$ then
  return *valid*
else
  return *invalid*
end if

Figure 1: Algorithm 1 - 3 of our I2S scheme

5. SECURITY

In this section, two assumptions are formally described in order to prove security.

5.1 Complexity Assumption

ASSUMPTION 5.1.1. *The Computational Diffie-Hellman (CDH) Assumption. Given a cyclic group* G *of order $\hat{p}$ with a randomly-chosen generator g, on input (g, g^a, g^b) where a and b are random numbers chosen from $\mathbb{Z}_{\hat{p}}$, for any $\mathcal{PPT}$ algorithm $\mathcal{A}$ that runs in time at most t, it is computationally infeasible for $\mathcal{A}$ to computes the value g^{ab}.*

ASSUMPTION 5.1.2. *The Factoring Assumption. Given two distinct randomly-chosen primes p and q, each $k/2$-bits long and congruent to* 3 mod 4*, it is computationally infeasible for a $\mathcal{PPT}$ algorithm to factor the product of p and q in time at most t.*

As far as we know, the running time of the best known factoring algorithm is about $2^{1.9k^{1/3}\lg(k)^{2/3}}$ [3].

5.2 Security Of Our Scheme

THEOREM 5.2.1. *Let* $\mathsf{I2S}[k,l,T]$ *represent our intrusion-resilient two-party signature scheme with parameters a modulus of size k, a hash function output of length l, and a number of time periods T. Assuming that an adversary $\mathcal{F}_1$ can break the scheme with probability ϵ_1, there is an algorithm $\mathcal{F}'$ that breaks the CDH assumption with probability ϵ', where*

$$\epsilon' \geq \epsilon_1 \frac{l}{2^l} \tag{8}$$

Therefore, $\mathbf{InSec}^{\mathrm{ir}}(\mathsf{I2S}[k,l,T]; \tau, q_{sig}, q_{hash}) \leq 2^l \epsilon'/l$.

4.1 Algorithm: $UR.Upd(SKU_{j-1,r})$

if $j < T+1$ then
  for $i = 1$ to l do
    $x_{i,j.0} \leftarrow x_{i,j-1.r}^2 \bmod N$
  end for
  $SKU_{j.0} \leftarrow (N, T, j, x_{1,j.0}, ..., x_{l,j.0}))$
end if
return $SKU_{j.0}$

4.2 Algorithm: $BS.Upd(SKB_{j-1,r})$

if $j < T+1$ then
  for $i = 1$ to l do
    $y_{i,j.0} \leftarrow y_{i,j-1.r}^2 \bmod N$
  end for
  $SKB_{j.0} \leftarrow (N, T, j, y_{1,j.0}, ..., y_{l,j.0}))$
end if
return $SKB_{j.0}$

5.1 Algorithm: $UR.Rfs(SKU_{j,r-1})$

if $j < T+1$ then
  $a, v_1 \leftarrow_r \{0,1\}^\lambda$;
  $h_1 \leftarrow H(g^a \bmod \hat{p}, v_1)$;
  send $\langle h_1, v_1\rangle$;
  receive $\langle h_2, v_2\rangle$;
  send $\langle g^a \bmod \hat{p}\rangle$;
  receive $\langle g^b \bmod \hat{p}\rangle$;
  verify $H(g^b \bmod \hat{p}, v_2) = h_2$;
  $\gamma \leftarrow (g^b)^a \bmod \hat{p}$;
  for $i = 1$ to l do
    $x_{i,j.r} \leftarrow x_{i,j.r-1} \cdot \gamma \bmod N$
  end for
end if
$SKU_{j,r} \leftarrow (N, T, j, x_{1,j.r}, ..., x_{l,j.r})$;
return $SKU_{j,r}$

5.2 Algorithm: $BS.Rfs(SKB_{j,r-1})$

if $j < T+1$ then
  $b, v_2 \leftarrow_r \{0,1\}^\lambda$;
  $h_2 \leftarrow H(g^y \bmod \hat{p}, v_2)$;
  send $\langle h_2, v_2\rangle$;
  receive $\langle h_1, v_1\rangle$;
  send $\langle g^y \bmod \hat{p}\rangle$;
  receive $\langle g^a \bmod \hat{p}\rangle$;
  verify $H(g^a \bmod m, v_1) = h_1$;
  $\gamma \leftarrow (g^a)^b \bmod \hat{p}$;
  for $i = 1$ to l do
    $y_{i,j.r} \leftarrow y_{i,j.r-1} \cdot \gamma^{-1} \bmod N$
  end for
end if
$SKB_{j,r} \leftarrow (N, T, j, y_{1,j.r}, ..., y_{l,j.r})$;
return $SKB_{j,r}$

Figure 2: Algorithm 4 and 5 of our I2S scheme

PROOF. Assume at period t and the r-th refresh, $\mathcal{F}_1$ chooses to access $Osec(\text{“}u\text{”}, t.r)$, so she has $SKU_{t,r}$, *i.e.,* $x_{1,t.r}, \ldots, x_{l,t.r}$. The goal of $\mathcal{F}_1$ is to generate

$$r_1 \Pi_{i=1}^{l} (x_{i,j.r}\gamma)^{c_i} \bmod N, \tag{9}$$

where $c_1...c_l = H(j, r_1^{2^{T+1-j}}, m)$. Since r_1 can be decided by the share holder alone, $\mathcal{F}_1$'s goal actually is to generate $\Pi_{i=1}^{l}(g^{ab})^{c_i}$. We define a function $f : \{0,1\}^l \rightarrow \mathbb{Z}_{l+1}$ that counts the number of 1's through every digit for the input value. So, it suffices to generate $g^{abf(c_1...c_l)}$.

We assume $H : \{0,1\}^* \rightarrow \{0,1\}^l$ is a random oracle, so its output $c_1...c_l$ is uniformly distributed in $\mathbb{Z}_{2^l-1}$. The output of the composite function $f \circ H(\cdot)$ follows the following distribution

$$\Pr[f \circ H(\cdot) = i] = \binom{l}{i} \Big/ 2^l = \frac{l!}{i!(l-i)!2^l}. \tag{10}$$

If adversary $\mathcal{F}_1$ successfully forges $g^{abf(c_1...c_l)}$, the probability that she can have g^{ab} is

$$\Pr[f \circ H(\cdot) = 1] = \binom{l}{1} / 2^l = l/2^l. \tag{11}$$

Since $\mathcal{F}_1$ is assumed to be able to forge $g^{abf(c_1...c_l)}$ with probability ϵ_1 , the overall probability that $\mathcal{F}_1$ can have g^{ab} is at least $\epsilon_1 \frac{l}{2^l}$. □

The CDH assumption implies that even when g^a and g^b are known, the value g^{ab} appears to be a “freshly chosen” random number for any computationally bounded attacker, so $g^{ab} \bmod \hat{p}$ could be considered uniformly distributed in $\mathbb{Z}_{\hat{p}}$, *i.e.,* $\epsilon' \simeq 1/2^\lambda$. Generally $\lambda = 1024$ is considered secure enough for CDH and $l = 160$ for a typical hash function such as SHA-1 and RIPEMD-160. Using these values, $\mathbf{InSec}^{\mathrm{ir}}(\mathsf{I2S}[k,l,T]; \tau, q_{sig}, q_{hash}) \leq 2^l \cdot 2^{-\lambda}/l = \frac{1}{160 \cdot 2^{864}}$.

The I2S is based on FSS, which was proved to have the upper bound of the insecurity function [3]. Since the I2S

uses multiplicative secret sharing, the difference is that the base and the user each has its own hash function. So, the actual total length of the "entire" hash function output is $2 \cdot l$ instead of l. Then for any τ, any q_{sig}, and any $q_{hash} \geq 1$, so the insecurity function for I2S is

$$\begin{aligned} &\mathbf{InSec}^{\mathrm{fs}}(\mathsf{I2S}\,[k,l,T]\,;\tau,q_{sig},q_{hash}) \\ &\leq q_{hash} \cdot T \cdot \left[2^{-2l} + \sqrt{4lT \cdot \mathbf{InSec}^{\mathrm{fac}}(k,\tau')}\right] \\ &\quad + \tfrac{q_{sig} \cdot q_{hash}}{2^k} \end{aligned} \tag{12}$$

where $\tau' = 2\tau + O(k^3 + 2k^2 l \lg(T))$.

6. OTHER BENEFITS AND APPLICATIONS

Fast Certificate Revocation. Certificate revocation is one of the hardest problems in public key infrastructures (PKI), and consequently one of the major costs. Furthermore there is generally a delay between a certification authority (CA) receiving a revocation request and publishing it through Certificate Revocation Lists (CRL) or an Online Certificate Status Protocol (OCSP). Such a delay could be critical for time-sensitive applications. However, in I2S, it suffices to notify a base to cancel its key share. Immediately, signing can no longer be performed, so the user's public key is revoked. A verifier need not validate the signer's certificate by checking a CRL or acquiring an OCSP response.

No Authentication. When signing a document, an I2S user and its base need not authenticate each other, since the correct key share already identifies its holder. Neither does I2S require mutual authentication for key refreshes. It is because any intrusion is assumed to be detectable with minimal delay. If an adversary already obtained one key share, she cannot extend the validity of her share by initiating a refresh with the other party. If she does, there will be three parties to refresh two key shares so that the entire private key is damaged.

Third Party Witness. In I2S, every private key operation requires the aid of a third party. This allows the third party to witness the user's relevant behavior. In some sense it prevents the user from abusing its private key. By providing optional security services, the third party can produce a record of the user's activity. In contrast, existing intrusion resilient signature schemes are all aimed at outside attackers compromising a user's private key. Users control their own private keys, computing new keys and deleting old keys. If a user is malicious, she can forge his old or future signature and leak her private key in order to repudiate a previously signed document. Similar situations could happen in other cryptographic algorithms where the user holds the entire private key.

Applications. The motivation of I2S is to provide forward and backward security simultaneously for private keys. However, the applications of I2S go beyond this. For example, its fixed public key and fast revocation implies that I2S could be used for Short-Lived Certificates (SLC). Existing approaches to SLCs involve traditional pair-wise keys; this requires issuing numerous ephemeral public keys, which carries a high cost. I2S has the potential to reduce these costs and make SLCs more practical [21]. As another example, I2S could be used for threshold role-based trust management. Assume a certain type of document becomes valid only after it is signed by both role A and role B. If the user of role B resigns or is removed and there is a new user to replace him, the new user just holds the refreshed share and no more changes are needed for the system. The key share of the removed user automatically becomes invalid while the public key remains the same, and this change of personnel could be transparent to verifiers.

7. CONCLUSION AND FUTURE WORK

In a real-world assistive environment, security depends on how the private key is stored and how its usage is authenticated. I2S protects private keys and mitigates the damage of key exposure. It uses random number negotiation to refresh the two key shares, which provides advantages relative to existing approaches. In this way, the exposure of a user's key share cannot break the cryptosystem and the compromised key share will become invalid instantly after the next refresh. In order to mount a successful attack, an attacker must break both parties simultaneously—thus it offers both forward security and intrusion resilience. I2S also provides fast revocation that existing schemes do not provide.

Since I2S requires additional work to refresh the key shares, and the private key operation needs another party's cooperation, one area of future work is to adopt an appropriate portable secure chip to store the other key share. The benefit that I2S brings is that users do not need to worry about the loss or compromise of the chip because it stores only a key share tha cannot work independently. Another area is migrating I2S to other forward secure signature schemes.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported in part by the National Science Foundation under award numbers CT-ISG 0716261 and CT-CND-0716827. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] Michel Abdalla, Sara K. Miner, and Chanathip Namprempre. Forward-secure threshold signature schemes. In *CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology*, pages 441–456, London, UK, 2001. Springer-Verlag.

[2] Michel Abdalla and Leonid Reyzin. A New Forward-Secure Digital Signature Scheme. In *Advances in Cryptology-ASIACRYPT'00*, pages 116–129, 2000.

[3] Mihir Bellare and Sara K. Miner. A Forward-Secure Digital Signature Scheme. In *Proc. of Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, pages 431–448, 1999.

[4] Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In *Proc. of Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003*, pages 1–18, 2003.

[5] Yigael Berger, Avishai Wool, and Arie Yeredor. Dictionary attacks using keyboard acoustic emanations. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 245–254, New York, NY, USA, 2006. ACM Press.

[6] Matt Bishop and Carrie Gates. Defining the insider threat. In *Proc. of the Cyber Security and Information Intelligence Research Workshop*, 2008.

[7] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Proc. of Advances in Cryptology - EUROCRYPT 2005*, pages 440–456, 2005.

[8] Xavier Boyen, Hovav Shacham, Emily Shen, and Brent Waters. Forward-secure signatures with untrusted update. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 191–200, New York, NY, USA, 2006. ACM Press.

[9] Mike Burmester, Vassilios Chrissikopoulos, Panayiotis Kotzanikolaou, and Emmanouil Magkos. Strong forward security. In *Proc.of the 16th international conference on Information security: Trusted information*, pages 109–121, 2001.

[10] Ran Canetti, Shai Halevi, and Jonathan Katz. A Forward-Secure Public-Key Encryption Scheme. In *Proc. of Advances in Cryptology - EUROCRYPT 2003*, pages 255–271, 2003.

[11] Yevgeniy Dodis, Matthew K. Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. A Generic Construction for Intrusion-Resilient Public-Key Encryption. In *Proc. of Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004*, pages 81–98, 2004.

[12] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-Insulated Public-Key Cryptosystems. In *Proc. of Advances in Cryptology - EUROCRYPT 2002*, pages 65–82, 2002.

[13] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong Key-Insulated Public-Key Schemes. In *Proc. of Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, pages 130–144, 2003.

[14] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proc. of Advances in Cryptology - CRYPTO 86, 6th Annual International Cryptology Conference*, pages 186–194, 1986.

[15] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Proactive RSA. In *Proc. of Advances in Cryptology - CRYPTO '97*, pages 440–454, 1997.

[16] Gene Itkis. Intrusion-resilient signatures: Generic constructions, or defeating strong adversary with minimal assumptions. In *Proc. of Security in Communication Networks, Third International Conference, SCN 2002*, pages 102–118, 2002.

[17] Gene Itkis and Leonid Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. In *Advances in Cryptology-CRYPTO'01.*, pages 332–354, 2001.

[18] Gene Itkis and Leonid Reyzin. Sibir: Signer-base intrusion-resilient signatures. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 499–514, London, UK, 2002. Springer-Verlag.

[19] Anton Kozlov and Leonid Reyzin. Forward-Secure Signatures with Fast Key Update. In *3rd Conference on Security in Communication Networks*, pages 241–256, 2002.

[20] Hugo Krawczyk. Simple Forward-Secure Signatures From Any Signature Scheme. In *7th ACM Conference on Computer and Communication Security*, pages 108–115, 2000.

[21] Zhengyi Le, Yi Ouyang, Yurong Xu, and Fillia Makedon. Preventing unofficial information propagation. In *Proc. of the 9th International Conference on Information and Communication Security (ICICS'07)*, pages 113–125, 2007.

[22] Zhengyi Le, Yi Ouyang, Yurong Xu, and Fillia Makedon. Mobile device protection against loss and capture. In *Proc. of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA'08)*, 2008.

[23] Benoît Libert, Jean-Jacques Quisquater, and Moti Yung. Efficient intrusion-resilient signatures without random oracles. In *Prof. of Information Security and Cryptology, Second SKLOIS Conference, Inscrypt 2006*, pages 27–41, 2006.

[24] Philip D. MacKenzie and Michael K. Reiter. Delegation of cryptographic servers for capture-resilient devices. *Distributed Computing*, 16(4):307–327, 2003.

[25] H. Ong and C.P. Schnorr. Fast Signature Generation with a Fiat Shamir—Like Scheme. In *Proc. of Advances in Cryptology - EUROCRYPT 1990, International Conference on the Theory and Applications of Cryptographic Techniques*, pages 432–440, 1990.

[26] Tal Rabin. A simplified approach to threshold and proactive rsa. In *Proc. of Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference*, pages 89–104, 1998.

[27] Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Proc. of Advances in Cryptology - EUROCRYPT 2005*, pages 114–127, 2005.

[28] Zhi-Jia Tzeng Wen-Guey Tzeng. Robust Key-Evolving Public Key Encryption Schemes. In *Proc. of Information and Communications Security, 4th International Conference, ICICS 2002*, pages 61–72, 2002.