

A Hierarchical Key-Insulated Signature Scheme in The CA Trust Model

Zhengyi Le, Yi Ouyang, James Ford, and Fyllia Makedon

Department of Computer Science, Dartmouth College, USA
{zhengyi.le, yi.ouyang, James.Ford, Fyllia.Makdon}@dartmouth.edu

Abstract. In key-insulated cryptography, there are many private keys with different indexes and a single, fixed public key. When the trust model includes multiple Certification Authorities (CAs), it can be used to shorten the verification path and mitigate the damage caused by the compromise of a CA's private key. Existing work requires that the total number of CAs be fixed and that a trusted keystore store all private keys. This paper presents a hierarchical key-insulated signature scheme, called HKI, which converts existing key-insulated methods to a hierarchical scheme. Our scheme allows the system to repeatedly generate a new private key for a new CA and also provides two important features, namely a shortened verification path and mitigated damage. By basing our approach on a general key-insulated scheme, we have made it possible to take advantage of any future improvements in computation complexity, key length, or robustness in current key-insulated methods.

1 Introduction

A CA is a trusted third-party organization that issues digital certificates which are signed messages specifying a name and the corresponding public key. The role of the CA is to guarantee that an individual granted a unique certificate is who he or she claims to be. The individual's certificate is signed by the CA's private key; a verifier can verify it by using the CA's public key. In the real world, a CA is trusted by a subset of people, called the *trust domain* of this CA, instead of by everyone. If a verifier and a certificate holder are in different subsets, the verifier cannot verify the certificate issued by a CA who he doesn't trust. When creating the trust relationships among CAs, the traditional cryptography requires every CA have its own key pair. This means that a verifier must go through the *trust path* to verify a certificate issued in a different trust domain and that the compromise of a CA's private key will affect some trust paths.

This paper suggests a solution to this problem, the *Hierarchical Key-Insulated Signature scheme* called HKI. It shortens the verification path in multiple-CA models and mitigates the damage caused by the compromise of a CA's private key. The basis of our scheme is *Key-Insulated Signature (KIS)* cryptography. In KIS, there are many private keys with different indexes and a single, fixed public key. This property makes it possible that all the CAs share one public key but hold their own private keys. The most closely related work in the literature is

[13]; however, this scheme uses KIS straightforwardly, assuming a fixed number of CAs, and it needs a trusted keystore to store private keys (which must all be generated beforehand). Our scheme can achieve the same goal but has none of these deficiencies.

Since there are variants of KIS, such as [6] and [11], we limit ourselves to general constructions based on KIS so as to make our work independent of KIS progress. This should ensure that any improvement to the underlying KIS cryptography will benefit our scheme.

Section 2 summarizes existing CA trust models with the traditional cryptography and also surveys related cryptography. Section 3 describes our goal, presents our hierarchical key-insulated signature scheme with a correctness proof, and discusses issues related to primes and revocation. Section 4 presents the conclusion and future work.

2 Related Work

2.1 CA Trust Models

There are three fundamental categories of CA trust models (the oligarchy model and the fully connected mesh model can be grouped together as extreme variants of the anarchy model):

Monopoly model. There is only one CA globally. This single CA is self-certified. A verifier can use the monopoly CA’s public key to verify any user’s certificate. But this is impractical—the number of global users is huge and keeps increasing. The workload is too heavy for a single CA. Furthermore, in case the CA’s private key is compromised, all the certificates should be revoked and re-signed.

Anarchy model. There are multiple CAs. Each CA has its own key pair and is self-certified. The workload of global certificate issuance is distributed. Any two CAs can perform *cross-certification*¹ to create a bilateral trust relationship. A verification procedure needs a *trust path*, also called *verification path* or *CA chain*, from a verifier to a certificate holder through certain cross-certifications if they are in different trust domains. The length of the verification path is the total of cross-certifications between them. This model eliminates monopoly pricing, but increases the burden on verifiers. The verifier needs to verify each CA’s certificate and cross-certificate on the trust path. When a CA’s private key is compromised, only those certificates issued by this CA and those related cross-certificates need to be revoked. In addition, it introduces another problem: the existence of a trust path doesn’t necessarily imply that the trust relationship is always transitive.

Hierarchical model. This model organizes CAs hierarchically. Every CA has its own key pair. The root CA is self-certified and every subCA’s certificate is signed by its parent CA (each CA has two kinds of users: common users and subCAs.). The root CA’s trust domain is global. A subCA’s trust domain is a

¹ A cross-certification is defined in the X.509 specification [1].

subdomain of its parent. The workload of global certificate issuance is distributed to subCAs. The length of the verification path is the distance from a certificate holder to the nearest common ancestor of the certificate holder and verifier.

In this model the registration of a new CA should be permitted by a higher-level CA. Each CA self-governs its trust domain and is the supervisor of all its subdomains. The drawbacks of this model are (1) the increased burden on verifiers; (2) revocation of the certificates of a CA's common users, as well as those of all its subCAs, when a CA's private key is compromised.

2.2 Related Cryptography

Forward secure public-key cryptography. The first Forward Secure Signature (FSS) scheme was designed by Bellare and Miner [3]. Their idea is as follows. Time is divided into periods: $0, 1, 2, \dots, T$. With a fixed public key PK , the private key evolves every period: $SK_0, SK_1, SK_2, \dots, SK_T$. The key-evolving algorithm is a one-way function so that forward security is provided. In period i , ($0 \leq i \leq T$), a message M is signed by SK_i . When someone wants to verify the signature S of M , he inputs $\langle PK, i, S, M \rangle$. Following that work, many improvements and similar forward secure schemes have been published (e.g., [2, 10, 14, 15, 12, 5, 4]). However, FSS cannot provide backward security; the cryptographic algorithms are public, and once an attacker successfully compromises SK_i he can evolve the key by himself so that he can forge future signatures.

Key-insulated cryptography. A strong key-insulated signature scheme was designed by Dodis *et al.* [6], which can provide both forward and backward security. It was improved with forward secure cryptography and provable security by Itkis and Reyzin [11]. The general idea of KIS is similar to that of FSS—the difference is that KIS uses a physically secure device to store a master key MK . When a user needs to update his private key, the device generates a partial secret from its master key and then sends the partial secret to the user. The user generates the next private key from the partial secret and his current private key. Thus, the key update cannot be performed without the interaction of the device. Since we assume the device is physically secure, the exposure of the current private key doesn't allow the attacker to derive the private key in any other time period.

Here we give an overview of a general KIS, which is the basis of our proposed scheme. KIS is a 5-tuple of poly-time algorithms ($Gen, Upd^*, Upd, Sign, Vrfy$):

- Gen : the key generation algorithm. $Gen(k, t) \rightarrow (PK, MK, SK_0)$, where k and t are two security parameters, SK_0 is the private key of period 0.
- Upd^* : the device key update algorithm. $Upd^*(i, j, MK) \rightarrow SK'_{i,j}$, where i and j are the time period indexes. This algorithm returns a partial secret $SK'_{i,j}$ which will be sent to the user to update the user's current private key SK_i to SK_j .
- Upd : the user private key update algorithm. $Upd(SK_i, SK'_{i,j}) \rightarrow SK_j$. The user uses the partial secret received from the device and his current private key of period i to generate a new private key of period j .

- *Sign*: the signing algorithm. $Sign(M, i, SK_i) \rightarrow \langle i, S \rangle$. S is the signature. i indicates the time period in which S is signed.
- *Vrfy*: the verification algorithm. $Vrfy(PK, M, \langle i, S \rangle) \rightarrow true/false$.

3 The Hierarchical Key-Insulated Signature Scheme

While the hierarchy model is the most desirable of its properties, it unfortunately still has deficiencies. Thus, a worthwhile goal is to use recent cryptographic advances to address weaknesses in the hierarchical model by shortening the verification path and mitigating the damage caused by the compromise of a CA's private key.

Koga *et al.* noted that one public key has many matched private keys in FSS and KIS. Therefore, they suggested two constructions [13] to achieve this goal based on an interweaving FSS and a plain KIS, respectively. In their constructions, all CAs share one fixed public key but each CA holds a different private key. Thus, the verification path is short (a verifier uses the fixed public key to verify certificates issued by any CA) and an attacker cannot derive other CAs' private keys with a compromised CA's private key. However, their constructions require that the number of CAs be fixed and that all the private keys be generated during a system initialization phase. Because of these limitations, dynamically adding a subCA is impossible in their models. Furthermore, their constructions are linear, instead of being hierarchical.

Our work focuses on removing these limitations, and allowing the dynamic addition of a new CA within a hierarchical model while also achieving the overall goals of shortened verification path and damage limitation. In contrast to FSS, KIS supports an unbounded number of periods and a compromised private key cannot be used to derive any other private key without the help of a device. We therefore have selected KIS as the basis for our approach. The principal challenge addressed in this paper is the construction of a hierarchical scheme using KIS, since each layer may have an unbounded number of CAs and the number of layers is also unbounded.

3.1 Constructions And Proofs

In the desired hierarchical tree, the size of layer zero is 1; the size of layer one is N , where N is the size of the set of the natural numbers; the size of layer two is $N * N$; and the size of layer i is N^i . It seems that the hierarchical structure has the size of $\lim_{N \rightarrow +\infty} \sum_{i=0}^N N^i$. This number is equal to $\lim_{N \rightarrow +\infty} \frac{N^N - 1}{N - 1}$, which is close to N^{N-1} . However, since the total number we can assign in KIS is N , our design challenge becomes in essence how to map the N private keys to $\sum_{i=0}^N N^i$ CAs such that no two CAs share the same private key.

Since all CAs share one public key, the length of the verification path is always one. Furthermore, in KIS there is only one physically secure device. If our hierarchy has only one such device, it may be very inconvenient to add new CAs, since each addition will require use of the same device. Therefore, we give

each subCA a subdevice. A desired property of these subdevices is that even if a subCA's private key is compromised and its device captured, an attacker should not be able to generate any other existing CA's private key.

Our constructions are based on two well-known number theorems:

Theorem 1. *The number of primes is infinite.*

Theorem 2. *Every positive integer (greater than 1) is a product of prime numbers, and its factorization into primes is unique up to the order of the factors.*

We give some definitions that are helpful in explaining our constructions.

Definition 1. $L = \{[x_1 \cdot x_2 \cdots x_n] : n \in \mathbb{N} \text{ and } \forall i(1 \leq i \leq n)x_i \in \mathbb{N}\} \cup \{\phi\}$

L is the union of the set of all lists of natural numbers with $\{\phi\}$.

Definition 2. *A CA's position in the hierarchical tree is defined by a $[x_1 \cdot x_2 \cdots x_t] \in L$. The $CA_{[x_1 \cdot x_2 \cdots x_t]}$ is in layer t ; its ancestors are $\{[x_1 \cdot x_2 \cdots x_i] : 1 \leq i < t\}$; it is the x_t th child of its father $CA_{[x_1 \cdot x_2 \cdots x_{t-1}]}$. The position of the root CA is an empty list, denoted by CA_ϕ .*

For instance, the CAs in layer one are $CA_{[1]}$, $CA_{[2]}$, \dots , $CA_{[i]}$, \dots . The subCAs of $CA_{[i]}$ are $CA_{[i,1]}$, $CA_{[i,2]}$, \dots , $CA_{[i,j]}$, \dots . Since each CA has its own device, the devices could be also denoted by the same list of numbers, *i.e.* $Dev_{[x_1 \cdot x_2 \cdots x_t]}$.

Definition 3. p_i is the i th prime number.

For example, $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$, \dots

Definition 4. ϖ_k is the total of k - bit primes.

In prime number theory there is a well-known quantity $\pi(x)$ that specifies the number of primes less than some integer x . We can derive

$$\varpi_k = \pi(2^k - 1) - \pi(2^{k-1} - 1) = \pi(2^k) - \pi(2^{k-1}) \quad (1)$$

since 2^k and 2^{k-1} are not primes (for $k > 2$).

Definition 5. f is defined as $f : \forall X \in L \rightarrow f(X) \in \mathbb{N}$.

f maps each position in the hierarchy to a unique natural number.

Definition 6. *Dev is a forward secure device of KIS. First, it is physically secure, i.e., it is a functional black box, unbreakable but capturable. Second, it is forward secure, i.e., it is stateful, and at each invocation it outputs a certain partial secret as a function of the current state and then deletes the old state, never repeating previous partial secrets.*

The device described in [11], which is protected by forward security technique, can provide these properties.

Now, the remaining work is to (1) create an f and (2) design the matched Dev to achieve the aforesaid goals .

Construction I We construct the f function as follows (see Figure 1):

$$f_1([x_1 \cdot x_2 \cdots x_n]) = \prod_{i=1}^n p_{(\sum_{j=1}^i x_j) - i + 1} \quad (2)$$

For instance, $f_1([i]) = p_i$. This means the $CA_{[i]}$ ($i \geq 1$) holds the p_i th private key of KIS. $f_1([i \cdot j]) = p_i * p_{i+j-1}$. This means the $CA_{[i \cdot j]}$ holds the $p_i * p_{i+j-1}$ th private key of KIS. In this way, each CA is assigned a natural number so that it holds the corresponding KIS private key. In addition, we define $f_1(\phi) = 1$. This means the root CA_ϕ holds SK_1 .

For example, $f_1([4]) = p_4 = 7$ so that $CA_{[4]}$ holds SK_7 , the 7th KIS private key; $f_1([4 \cdot 2]) = p_4 * p_{4+2-1} = p_4 * p_5 = 7 * 11 = 77$ so that $CA_{[4 \cdot 2]}$, which is the 2nd child of $CA_{[4]}$, holds SK_{77} .

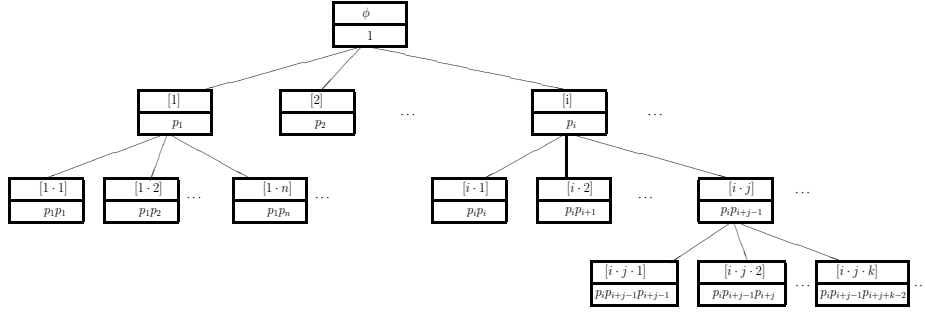


Fig. 1. Construction I

Theorem 3. $f_1 : L \rightarrow N$ is a one-to-one function, i.e., no two CAs share the same private key.

Proof. It is obvious that every member in L has a unique f_1 value in N , i.e. $\forall [x_1 \cdot x_2 \cdots x_n] \in L$, there is a unique product $\prod_{i=1}^n p_{(\sum_{j=1}^i x_j) - i + 1} \in N$.

Then we prove that every member in N has a unique f_1^{-1} value in L , $f_1^{-1} : N \rightarrow L$. Theorem 2 says that $\forall y \in N$, $y \geq 2$, y can be factored uniquely:

$$y = p_{a_1}^{b_1} \cdot p_{a_2}^{b_2} \cdots p_{a_n}^{b_n} \quad (3)$$

where $n \in N$, $a_1 < a_2 < \cdots < a_n$ and $b_i \in N (1 \leq i \leq n)$. From (2) we have:

$$f_1^{-1}(y) = [a_1 \cdot \underbrace{1 \cdot 1 \cdots 1}_{b_1 - 1} \cdot (a_2 - a_1 + 1) \cdot \underbrace{1 \cdot 1 \cdots 1}_{b_2 - 1} \cdots (a_n - a_{n-1} + 1) \cdot \underbrace{1 \cdot 1 \cdots 1}_{b_n - 1}] \quad (4)$$

For example, $f_1^{-1}(189) = f_1^{-1}(3^3 * 7) = f_1^{-1}(p_2^3 * p_4) = [2 \cdot 1 \cdot 1 \cdot 3]$. We can see every y must have a unique f_1^{-1} value since its $\{(a_1, b_1), (a_2, b_2), \dots, (a_n, b_n)\}$ is unique. In addition, since $f_1(\phi) = 1$, $f_1^{-1}(1) = \phi$. \square

It follows that any node of this tree (*i.e.*, $CA_{[x_1 \cdot x_2 \cdots x_n]}$) is mapped to a unique natural number (*i.e.*, a unique private key of KIS) and that every private key of KIS (except SK_0) has a unique position in this tree. In this way, we convert the linear structure of KIS to a hierarchical one. In addition, Theorem 1 guarantees that this tree could have infinite layers and that each node could create unlimited children as required.

A remaining problem is specifying the function of the devices used in KIS. Assume every CA has a Dev . $Dev_{[x_1 \cdot x_2 \cdots x_n]}$ stores three parameters: MK , $[x_1 \cdot x_2 \cdots x_n]$, and i , where i indicates how many children this CA has created. At each invocation, $Dev_{[x_1 \cdot x_2 \cdots x_n]}$ does the following:

- replace i with $i + 1$.
- run $Upd^*(f_1([x_1 \cdot x_2 \cdots x_n]), f_1([x_1 \cdot x_2 \cdots x_n \cdot i]), MK)$. (The description of Upd^* is in Section 2.2.)
- output a partial secret $SK'_{f_1([x_1 \cdot x_2 \cdots x_n]), f_1([x_1 \cdot x_2 \cdots x_n \cdot i])}$.

$CA_{[x_1 \cdot x_2 \cdots x_n]}$ derives $SK_{f_{[x_1 \cdot x_2 \cdots x_n \cdot i]}}$ from this partial secret and its private key $SK_{f_{[x_1 \cdot x_2 \cdots x_n]}}$ and then gives this new private key to its i th child $CA_{[x_1 \cdot x_2 \cdots x_n \cdot i]}$. Furthermore, as noted in Definition 6, every Dev is both physically secure and forward secure: it is infeasible to either break it and reconstruct MK or to invoke it to output a past partial secret. It only outputs the next partial secret, as intended, sequentially at each invocation.

In brief, this construction allows all the CAs to share one public key so that the verification path is shortened, and also allows every CA to create infinite children so that the hierarchy is dynamically scalable. Section 3.2 will discuss the issue of revocation.

In addition, we also can use Gödel numbers [9] to construct f :

$$f_G([x_1 \cdot x_2 \cdots x_n]) = \prod_{i=1}^n p_i^{x_i} \quad (5)$$

This f_G can also map any list of numbers to a unique natural number—for example, $f_G([2 \cdot 3 \cdot 2]) = p_1^2 * p_2^3 * p_3^2 = 2^2 * 3^3 * 5^2 = 2700$, but there are some numbers in N for which we cannot find corresponding number lists—for example, those natural numbers whose factors are not consecutive primes, say $33 (= 3 * 11 = p_2 * p_5)$, we can never find a list with a matching f_G value (here, 33). Thus, while f_G would assign every CA a unique KIS private key, not every private key would be assigned to a CA (that is, some private keys are unused).

Construction II We first introduce a pseudo random prime generator, which is embedded in every Dev .

Definition 7. *Prpg(k) is a pseudo random prime generator. At each invocation, it generates a random k – bit prime. It is not feasible either to make it regenerate any previous random prime or to predict any future prime.*

VIII

We construct function f as follows:

$$f_2([x_1 \cdot x_2 \cdots x_n]) = f_2([x_1 \cdot x_2 \cdots x_{n-1}]) \cdot Prpg(k) \quad (6)$$

where $f_2(\phi) = 1, \forall x_1 \in N, f_2([x_1]) = Prpg(k)$, and k could be 1024.

The root CA_ϕ holds SK_1 of KIS. Its child, $CA_{[i]}$, holds $SK_{f_2([i])} \cdot CA_{[i,j]}$, the child of $CA_{[i]}$, holds $SK_{f_2([i]).Prpg(k)}$. Note that f_2 is not a one-to-one function from L to N . The following Theorem 4 guarantees that f_2 nonetheless works.

Lemma 1. *Let Pr' be the probability that a new CA is assigned the same private key as any other existing CA by f_2 . When the depth of the hierarchy is not greater than ϖ_{1024} , $Pr' \leq \frac{1}{\varpi_{1024}}$.*

Proof. In our hierarchy, f_2 guarantees that CAs in different layers never share the same private key. Therefore, a new CA may only share a private key with CAs in the same layer.

In the i th layer, when a new CA is being added, let Pr'_i be the probability that this new CA shares the same private key with any given CA. We can derive

$$Pr'_1 = \frac{1}{\varpi_{1024}}, Pr'_2 = \frac{2!}{\varpi_{1024}^2} - \frac{1}{\varpi_{1024}^3}, Pr'_3 = \frac{3!}{\varpi_{1024}^3} - \frac{9}{\varpi_{1024}^4} + \frac{4}{\varpi_{1024}^5}, \dots$$

Note that Pr'_i becomes more complicated with increasing i since $Prpg(1024)$ might choose any ϖ_{1024} -bit prime more than once. However, we can find an upper bound: $Pr'_i \leq \frac{i!}{\varpi_{1024}^i}$. The numerator is not greater than $i!$ since the value of a product cannot be affected by the order of its factors.

Furthermore, $\forall i, 1 \leq i \leq \varpi_{1024}$, we have

$$\frac{i!}{\varpi_{1024}^i} \leq \frac{(i-1)!}{\varpi_{1024}^{i-1}} < \dots < \frac{1}{\varpi_{1024}} \quad (7)$$

So, we have $\forall i, 1 \leq i \leq \varpi_{1024}, Pr'_i \leq \frac{1}{\varpi_{1024}}$, i.e. $Pr' \leq \frac{1}{\varpi_{1024}}$. \square

In traditional cryptography with a hierarchical model, each CA has its own public/private key pair, such as an RSA key pair. A parent vouches for the private keys of its children. We know that a RSA private key is decided by its modulus and public key, and that RSA is no less secure if public keys are always chosen to be the same number (in the real world the default public key is always 65537 in OpenSSL). Assuming all RSA users use the default number, it is possible that two of them share the same private key and modulus.

Theorem 4. *In the traditional method, let Pr be the probability that a new CA generates a RSA prime couple identical to that in use by another CA. When the depth of the hierarchy is not greater than ϖ_k , we have $Pr' < Pr$.*

Proof. Assume the modulus of RSA is 1024-bit so that the two big primes in RSA are 512 bit each. In our hierarchy, we assume $k = 1024$.

Step 1: Compute ϖ_{512} and ϖ_{1024} . (1) suggests that we need to obtain $\pi(2^{512})$, $\pi(2^{511})$, $\pi(2^{1024})$ and $\pi(2^{1023})$ first². Dusart [7] showed:

$$\frac{x}{\ln x} \left(1 + \frac{0.992}{\ln x}\right) \leq \pi(x) \leq \frac{x}{\ln x} \left(1 + \frac{1.2762}{\ln x}\right) \quad (9)$$

where the lower bound holds for $x > 598$ and the upper bound holds for $x > 1$. The results of using (9) are

$$2^{502.5288} \leq \varpi_{512} \leq 2^{502.5323} \quad (10)$$

$$2^{1013.5288} \leq \varpi_{1024} \leq 2^{1013.5305} \quad (11)$$

Step 2: In the traditional method, for any existing CA, its two primes could be identical with the probability of $\frac{1}{\varpi_{512}}$ and be different with the probability of $\frac{\varpi_{512}-1}{\varpi_{512}}$. So, we have

$$Pr = \frac{1}{\varpi_{512}} \left(\frac{1}{\varpi_{512}} \cdot \frac{1}{\varpi_{512}}\right) + \frac{\varpi_{512}-1}{\varpi_{512}} \left(\frac{1}{\varpi_{512}} \cdot \frac{1}{\varpi_{512}} \cdot 2\right) = \frac{2}{\varpi_{512}^2} - \frac{1}{\varpi_{512}^3} \quad (12)$$

From (10) and (12), we can derive $Pr > \frac{1}{\varpi_{512}^2} \geq \frac{1}{2^{1005.0646}}$. From (11) and Lemma 1, we can likewise derive $Pr' \leq \frac{1}{\varpi_{1024}} \leq \frac{1}{2^{1023.5288}}$. So, $Pr' < Pr$. \square

Thus, any node of this tree is mapped to a natural number by f_2 , and this tree is dynamically scalable. However, it is not the case either that every KIS private key is assigned to a CA or that every private key is assigned to a unique CA. Theorem 4 guarantees that the probability that any two CAs share a KIS private key is less than the probability that any two RSA users share a prime couple.

To summarize the function of the KIS device, each $Dev_{[x_1 \cdot x_2 \cdots x_n]}$ stores three parameters, MK , $[x_1 \cdot x_2 \cdots x_n]$ and child counter i , and also embeds a $Prpg(k)$. At each invocation, $Dev_{[x_1 \cdot x_2 \cdots x_n]}$ accomplishes the same steps as in Construction I, but by calling f_2 instead of f_1 .

Construction III We can construct function f using $Prpg(k)$ in another way:

$$f_3([x_1 \cdot x_2 \cdots x_n]) = Prpg(1023 + n) \quad (13)$$

The supporting theorem is as follows:

² The latest research [16] gives $\pi(x) = \int_2^x \frac{dx}{\ln x} + O(xe^{-\frac{A(\ln x)^{3/5}}{(\ln \ln x)^{1/5}}})$ for some constant A . But generally we can approximate

$$\pi(x) \sim \frac{x}{\ln x - 1} \quad (8)$$

The results of using this formulas are $\varpi_{512} \sim 2^{502.5300}$ and $\varpi_{1024} \sim 2^{1013.5293}$.

Theorem 5. *If each CA is assigned a private key of KIS by f_3 , let Pr'' be the probability that a new CA is assigned the same private key as another CA. We have $Pr'' < Pr$, where Pr is the same as in Theorem 4.*

Proof. Step 1: It is easy to see that CAs in different layers can never share the same private key. Let Pr''_i be the probability that a new CA of layer i shares the same private key with any other CA in the same layer, $Pr''_i = \frac{1}{\varpi_{1023+i}}$.

Step 2: We begin by proving Pr''_i is nonincreasing. If ϖ_k is nondecreasing, Pr''_i is nonincreasing. Thus, we prove that ϖ_k is nondecreasing, where $k \geq 1024$.

$$\begin{aligned}
& \varpi_k \geq \varpi_{k-1} \\
& \stackrel{(1)}{\Leftrightarrow} \pi(2^k) - \pi(2^{k-1}) \geq \pi(2^{k-1}) - \pi(2^{k-2}) \\
& \stackrel{(8)}{\Leftrightarrow} \frac{2^k}{k \ln 2 - 1} - \frac{2^{k-1}}{(k-1) \ln 2 - 1} \geq \frac{2^{k-1}}{(k-1) \ln 2 - 1} - \frac{2^{k-2}}{(k-2) \ln 2 - 1} \\
& \Leftrightarrow \frac{4}{k \ln 2 - 1} + \frac{1}{(k-2) \ln 2 - 1} \geq \frac{3}{(k-1) \ln 2 - 1} + \frac{1}{(k-1) \ln 2 - 1} \\
& \Leftarrow \frac{4}{k \ln 2 - 1} > \frac{3}{(k-1) \ln 2 - 1} \\
& \Leftrightarrow k > 4 + \frac{1}{\ln 2}
\end{aligned}$$

So, we have when $i \geq 1, k \geq 1024 \Rightarrow k > 4 + \frac{1}{\ln 2} \Rightarrow \varpi_k \geq \varpi_{k-1} \Rightarrow Pr''_i \leq Pr''_{i-1}$.

Step 3: We already know that $\frac{1}{\varpi_{1024}} < Pr$, i.e., $Pr''_1 < Pr$ (see the proof of Theorem 4). So, $\forall i, i > 1, Pr''_i < Pr''_1 < Pr$, i.e., $Pr'' < Pr$. \square

The function and discussion of *Dev* for Construction III is identical to that in Construction II.

3.2 Discussion

Primes. Construction I can infinitely expand but its actual scalability depends on how many available primes we have. It requires that the device used generate the next prime when needed. If the scale of the hierarchy is very large, recent research shows that Construction I will have trouble. The current world record³ found the largest known value of $\pi(n)$: $\pi(4 * 10^{22})$ which is 783,964,159,847,056,303,858. Since $4 * 10^{22} = 2^{75.0824}$, it is computationally challenging that the device is required to compute the next prime for a 75-bit prime.

Construction II and III cannot infinitely expand but we have proved that the probability that a new CA might share a same private key with an existing CA is smaller than the probability that two CAs share a same prime couple in the traditional RSA construction. Judging whether a number is prime can be done much faster than counting the number of primes less than that number or generating the next prime. The largest known prime⁴ is $2^{20996011} - 1$ which has 6320430 digits. So, generating a larger prime is also a challenge.

³ <http://numbers.computation.free.fr/Constants/Primes/countingPrimes.html>

⁴ http://www.utm.edu/research/primes/notes/by_year.html

Revocation We classify the types of compromises that are possible. Note that the security proof to KIS is outside the scope of our work.

First, an attacker only compromises a CA's private key. In the traditional hierarchical model, if a CA's private key is compromised, we need to revoke (1) this compromised private key, (2) all of its subCAs' private keys, and (3) all the users' certificates issued by this CA and its subCAs. However, in our scheme since the device is physically secure, the exposure of any CA's private key will not allow the attacker to derive any other private key. We only revoke (1) this compromised private key and (2) the common users' certificates signed by this CA's private key. Its subCAs' private keys are all still valid, as are any user certificates signed by its subCAs. This CA will be added as a new CA by its parent and these revoked certificates will be re-signed using its new private key.

Second, the attacker only captures a CA's device. Since the device only generates partial secrets, the attacker cannot derive any private key. So, we needn't revoke anything—what we need to do is to give this CA a new device.

Third, the attacker compromises a CA's private key and captures its device. Since the *Dev* is both physically secure and forward secure, the attacker only can derive future private keys which haven't been assigned to any existing subCA. So, the private keys of existing CAs are still secure and valid. We need to revoke (1) this CA's private key and (2) all the common users' certificates issued by this CA. This CA will be re-added as a new CA by its parent and will re-issue these revoked certificates.

If this compromised CA is the root CA, we can re-initialize a HKI scheme for it and its old childrens' private keys are still safe and valid. If so, we will have two HKI systems, *i.e.*, two HKI public keys; therefore, we use another method. Construction I only has one unused private key, SK_0 (because $f_1(\phi) = 1$ and $f_1([1]) = p_1 = 2$). Therefore, the root CA only has one backup private key, which is SK_0 . We could assign $SK_{f_1([1])}$ to the root CA as its initial private key so that $SK_{f_1([i])}$, $i \geq 2$, is the backup of $SK_{f_1([1])}$; then every layer in Construction I is shifted one-layer lower. Construction II and III have many unused private keys which could be the backups of the root CA's private key.

Note that in *Hierarchical ID-based cryptography* [8] if any of those private key generators is broken, the attacker can generate any private key of its domain. However, in our constructions if the physically secure device is captured, the attacker cannot generate any private key without an exposed private key. Even if he also has an exposed private key, the newly generated private keys are different from any existing CA's.

4 Conclusion And Future Work

This hierarchical key-insulated signature scheme shortens the verification path using the KIS property that many private keys with different indexes share one fixed public key; it mitigates the damage of the exposure of a CA's private key because our *Dev* is forward secure; and it allows the hierarchy to add new CAs dynamically and without bound, which is the main problem not addressed by

existing work on the previous points. In addition, our constructions are built on a general KIS scheme, allowing improvements in key length, computation complexity, or robustness for KIS to directly benefit our scheme as well.

This paper has focused on providing a functional description based on a new forward secure *Dev*. The implementation of *Dev*, in hardware or software, is our next work. Furthermore, there are several aspects of our scheme that we hope to improve; for example, Construction II and III have many unused private keys, and the key length in Construction II grows quickly with increasing depth in the hierarchy. The scheme does not effectively scale. It would be desirable to come up with a scheme that computes the i -th private key with time complexity independent of i .

References

1. Internet X.509 Public Key Infrastructure Certificate and CRL profile. *RFC3280*, April 2002.
2. Michel Abdalla and Leonid Reyzin. A New Forward-Secure Digital Signature Scheme. *Advances in Cryptology-ASIACRYPT'00*, pages 116–129, Dec 2000.
3. Mihir Bellare and Sara K. Miner. A Forward-Secure Digital Signature Scheme. *Advances in Cryptology-CRYPTO'99*.
4. Mihir Bellare and Bennet Yee. Forward-Security in Private-Key Cryptography. *Topics in Cryptology-CT-RSA*, 2003.
5. Ran Canetti, Shai Halevi, and Jonathan Katz. A Forward-Secure Public-Key Encryption Scheme. *Advances in Cryptology - Eurocrypt'02*.
6. Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong Key-Insulated Public-Key Schemes. *Workshop on Public Key Cryptography (PKC)*, Jan 2003.
7. Pierre Dusart. The k^{th} prime is greater than $k(\ln k + \ln \ln k - 1)$ for $k \geq 2$. *mc*, 68(225):411–415, January 1999.
8. Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. *Proceedings of Asiacrypt'02*.
9. Kurt Gödel. *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*. Dover Publications, INC., 1992.
10. Gene Itkis and Leonid Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. *Advances in Cryptology-CRYPTO'01*.
11. Gene Itkis and Leonid Reyzin. SiBIR: Signer-Base Intrusion-Resilient Signatures. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.
12. Jonathan Katz. A Forward-Secure Public-Key Encryption Scheme. *Cryptology eprint archive Report 2002/060*, May 2002. <http://eprint.iacr.org/2002/060/>.
13. Satoshi Koga and Kouichi Sakurai. Decentralization Methods of Certification Authority Using the Digital Signature Schemes. In *Proceedings of 2nd Annual PKI Research Workshop*, 2003.
14. Anton Kozlov and Leonid Reyzin. Forward-Secure Signatures with Fast Key Update. *3rd Conference on Security in Communication Networks*, 2002.
15. Hugo Krawczyk. Simple Forward-Secure Signatures From Any Signature Scheme. *7th ACM Conference on Computer and Communication Security*, 2000.
16. Hans Riesel. The Remainder Term in the Prime Number Theorem. *Prime Numbers and Computer Methods for Factorization (Progress in Mathematics, Vol 126)*, 1994.