Key Revocation with Interval Cover Families

J. Blömer, A. May

Department of Mathematics and Computer Science
University of Paderborn
33095 Paderborn, Germany
{*bloemer, alexx*}*@uni-paderborn.de*

Abstract. We present data structures for complement covering with intervals and their application for digital identity revocation. We give lower bounds showing the structures to be nearly optimal. Our method improves upon the schemes proposed by S. Micali [5, 6] and Aiello, Lodha, Ostrovsky [1] by reducing the communication between a Certificate Authority and public directories while keeping the number of tokens per user in the public key certificate small.

1 Introduction

Digital identities play an essential role in many cryptographic applications. Infrastructures for digital identities are built by means of public-key cryptography and Certification Authorities. The schemes differ in how digital identities can checked to be valid and how the identities can be revoked.

A digital identity is validated by a certificate issued by a Certification Authority (CA). The CA initially uses a public key generation process to create a public key/secret key pair. The public key together with a fingerprint is published. A user u who wants to establish his own digital identity creates a new public key/secret key pair and sends the public key together with identifying information to the CA. The Certificate Authority checks u's identity to ensure that the user is really the person he/she claims to be. After that, the CA signs with its secret key a certificate containing u's public key, the identifying information and an expiration date of this certification. Hence, anyone is able to check the certificate issued by the CA with the CA's public key. For accepting u's public key, one must not trust the user u himself but the CA. To establish higher levels of trust, one can use a hierarchy of CAs.

A digital identity is valid as long as its certificate has not expired. In contrast to this, we must also have a mean for revoking users. Assume u's identity is stolen or compromised before the certificate expiration date. The thief can sign arbitrary messages with u's secret key. Hence, as in the case of credit cards, one must establish an immediate identity revocation.

There are many solutions proposed in literature how to revoke digital identities. The first one is a centralized online solution where a trusted database holds the status of each public key. The database answers queries about public keys. However, these answers must be authenticated by the database to avoid man-in-the-middle attacks. In many cases the method is impractical because an online access is required.

Another solution, the Certificate Revocation List (CRL), is widely used in practice. In this offline approach, the CA makes a list of all users revoked thus far and signs it. This list is distributed at regular intervals – for example during a daily update period – to many public directories. A public directory is untrusted but one insists that it cannot cheat and must return a user's revocation status when queried. The main drawback of this scheme is the time it takes to check a key's validity. One must first check the CA signature and then look at the whole list of revoked users. Consider a fixed update time and let r be the total number of revoked users up to this point. The CA has to communicate a CRL of size $O(r)$ to each public directory in order to update the status of the keys. The time to check a key's validity using the CRL is also $O(r)$.

There are two other offline schemes proposed by Kocher [3] and Naor, Nissim [7]. They make use of authenticated hash trees. For a fixed update time and r as defined above the communication from the CA to the public directories is reduced to $O(\log r)$. In order to check a user's identity, one receives $O(\log r)$ hash values from the directory and computes another $O(\log r)$ hash values. These values are compared with the public directory data in a specified way. Additionally, the root signature of the authenticated hash tree is checked.

The main drawbacks of the offline solutions mentioned so far are:

- The information send by the CA must be authenticated. Therefore, signing the data is necessary. In order to prove the status, the signature must be checked.
- The proof length – the amount of data that has to be checked for validation – is a function of r. Since normally one must prove a key's validity very often, the proof length is the main bottleneck of digital identity revocation.

S. Micali [5, 6] proposed an elegant solution for these two problems based on an idea of offline/online signatures [2], which in turn builds on a work of Lamport [4]. He suggests to add an additional number y – called the user's 0-token – into the certificate. In order to create the 0-token, the CA picks a random number x and a one-way hash function f and computes $y = f^{(l)}(x) := f(f(\ldots f(x)))$, that is, the function f is applied l times to x in order to compute the 0-token y. The parameter l corresponds to the number of update periods, e.g. the days till expiration. On day 1, if user u is not revoked, the CA publishes the 1-token $f^{(l-1)}(x)$ of u. Since f is a one-way function, y can easily be computed from $f^{(l-1)}(x)$ by applying f once, but it is infeasible to find a valid 1-token $\tilde{x}$ with $f(\tilde{x}) = y$. Hence, u can take the 1-token as a proof that his key is valid on day 1. In general, the CA publishes the i-token $f^{(l-i)}$ on day i. This i-token serves as a day-i proof for the validity of u's key. Applying f i times and comparing the result with the 0-token proves the key's validity. In the sequel, we will use the terms token and proof synonymously. Notice that in contrast to the schemes of Kocher [3] and Naor, Nissim [7] this scheme needs only one proof for key validation and no signature of the CA in the daily update period.

Let $U = \{1, 2, \ldots, n\}$ be the set of users and let 2^U denote the power set of U. For a fixed update time let $R \subseteq U$ be the set of all users revoked so far. We set $r = |R|$. The complement $\bar{R} = U - R$ of R is the set of non-revoked users. The problem with Micali's scheme is that each of the $n - r$ non-revoked users in $\bar{R}$ obtains his own proof during an update period. Hence in each update period the CA has to communicate $n - r$ tokens to a public directory. We denote this as the CA-to-directory communication.

ALO (Aiello, Lodha, Ostrovsky) [1] proposed two schemes that reduce the CA-to-directory communication. These schemes are called Hierarchical and Generalized Scheme. The main building block of the ALO schemes is a set $F \subseteq 2^U$. The set F has the property that each set $\bar{R}$ of non-revoked users can be written as the union of the elements in a subset $S(\bar{R})$ of F. Each element $S_j \in F$ has its own 0-token. For each set $S_j \in F$, each user $u \in S_j$ stores the 0-token of S_j in his certificate. That is, the certificate of user u contains $|\{S_j \in F : u \in S_j\}|$ different tokens. We denote the maximal number $\max_{u \in U} |\{S_j \in F : u \in S_j\}|$ of tokens per certificate by $\mathcal{T}$.

In order to issue day-i proofs for the non-revoked users $u \in \bar{R}$, the CA computes a cover $S(\bar{R}) = \{S_{j_1}, S_{j_2}, \ldots, S_{j_m}\}$, $\bigcup_{1 \leq k \leq m} S_{j_k} = \bar{R}$ of the set $\bar{R}$. Next, it publishes the m i-tokens of the sets $S_{j_1}, S_{j_2}, \ldots, S_{j_m}$. Since these sets cover the set $\bar{R}$, each non-revoked user u is contained in at least one set S_j. Recall that u stores the 0-token of S_j in his certificate. Hence, the i-token of the set S_j is a day-i proof for user u.

There may be different ways to cover $\bar{R}$ by elements $S_j \in F$. In ALO's schemes, the CA always takes the minimal number of subsets for the cover in order to minimize the number m of proofs. Let $\max_{\bar{R} \subseteq U : |\bar{R}| = n-r} \{m : \text{CA needs } m \text{ sets to cover } \bar{R}\}$ be the maximal number of proofs that the CA has to publish for a set $\bar{R}$ of size $n - r$. We denote this maximal number of proofs by $\mathcal{P}$.

Note that Micali's revocation scheme fits this description. To obtain Micali's revocation scheme, define F as $F = \{\{1\}, \{2\}, \ldots \{n\}\}$. Hence, the users only have to store one 0-token in their certificate.

Let us define three demands on our key revocation scenario in order of decreasing priority:

Proof of key's validity: In our scenario, a user must prove a key's validity very often. Therefore, we insist on only one proof for key validation as in the revocation schemes of Micali and ALO. The schemes of Kocher and Naor, Nissim do not meet this requirement.

CA-to-directory communication ($\mathcal{P}$): The CA-to-directory communication corresponds to the maximal number of proofs the CA has to send to a public directory. The maximal number of proofs is denoted by $\mathcal{P}$. We have to keep the CA-to-directory communication small to allow frequent update periods. Thus, we want to minimize $\mathcal{P}$.

Tokens per certificate ($\mathcal{T}$): We denote the number of tokens per certificate by $\mathcal{T}$. To make the scheme practical (especially for smart card applications), $\mathcal{T}$ must be kept small, since checking long certificates is inefficient. But assuming that checked keys are stored, certificates normally have to be checked only once.

As mentioned above, Micali's scheme has $\mathcal{T} = 1$ token per certificate. However, the CA-to-directory communication is $\mathcal{P} = n - r$ if r is the number of revoked users. ALO's Hierarchical Scheme improves upon Micali's scheme by reducing the CA-to-directory communication $\mathcal{P}$ to $r \log_2(n/r)$ while increasing the number of 0-tokens $\mathcal{T}$ per certificate to $\log_2 n$. The Generalized Scheme of ALO needs at most $r(\log_c(n/r) + 1)$ proofs per update period and $\mathcal{T} \leq (2^{c-1} - 1) \log_c n$ tokens per certificate. Due to the $(2^{c-1} - 1)$ factor, this scheme is only practical for $c = 2$ or $c = 3$, otherwise the certificates become too large.

Our results build on the work of ALO. We propose a new method for covering the set $\bar{R}$ of non-revoked users by intervals. Therefore, we define a new class for covering problems called interval cover family (ICF). Our ICFs are constructed using interval trees.

The set R of revoked users partitions U in subintervals of non-revoked users, which can be represented by the nodes of an interval tree. Our task is to find a scheme which covers any interval with sets of an ICF. Furthermore, we want that each user $u \in U$ is in a small number of sets. This property is important because as in ALO's schemes, user u must include in his certificate all the 0-tokens of sets that contain u.

Micali's [5, 6] and ALO's Hierarchical scheme [1] also belong to the class of algorithms using interval cover families. The ICF in [5, 6] is the simplest one. It covers intervals by single elements. Thus, the length of the covering intervals is always 1. In ALO's Hierarchical scheme, the set of non-revoked users is covered by intervals with interval lengths that are powers of 2. In this paper, we propose two new methods for covering intervals that might be interesting for other areas of covering problems as well.

In Section 2, we introduce the class ICF of interval cover families. Revocation Scheme 1 (RS1) is presented in Section 3. It is a generalization of ALO's Hierarchical scheme. The length of the covering intervals is a power of $c \geq 2$. For RS1, we obtain the upper bounds $\mathcal{P} \leq (r+1)(2\log_c n - 1)$ and $\mathcal{T} \leq \frac{(c+1)^2}{4}\log_c n$ for some constant parameter c.

Our second Revocation Scheme (RS2) presented in Section 4 leads to a CA-to-directory communication of $\mathcal{P} \leq (r+1)(\log_c n + 1)$, while keeping the number of 0-tokens per certificate upper bounded by $\mathcal{T} \leq \frac{(c+1)^2}{2}\log_c n(1+o(1))$.

Since the new bounds for $\mathcal{T}$ are polynomial in c our systems are practical for larger parameters c than ALO's schemes. Thus, we can reduce the CA-to-directory communication $\mathcal{P}$ by choosing a large c. Since this communication is done during each update period, the system becomes more efficient.

In Section 5, we study the relations of the class ICF to the task of key revocation. Using a more refined analysis, we show that RS1 has a maximal CA-to-directory communication of $\mathcal{P} \leq 2r(\log_c n - \lfloor \log_c r \rfloor)$. Assuming the revoked users to be uniformly distributed, we can further reduce the bound of RS2 to an expected upper bound of $\mathcal{P} \leq (r + 1 - \frac{r(r-1)}{n}))(\log_c n + 1)$.

Scheme	$\mathcal{P}$ proofs from CA to directories	$\mathcal{T}$ tokens per certificate
Micali	$n - r$	1
ALO's Hierarchical	$r\log_2(\frac{n}{r})$	$\log_2 n$
ALO's Generalized	$r(\log_c(\frac{n}{r}) + 1)$	$(2^{c-1} - 1)\log_c n$
RS1	$2r(\log_c n - \lfloor \log_c r \rfloor)$	$\frac{(c+1)^2}{4}\log_c n$
RS2	$(r+1)(\log_c n + 1)$	$\frac{(c+1)^2}{2}\log_c n(1+o(1))$

Table 1. Comparison of our schemes with Micali's and ALO's schemes

If we only want to minimize the CA-to-directory communication, an optimal solution can be obtained from Yao's range query data structures [8]. However, Yao's construction results in prohibitively many tokens per certificate. For the first time in this area, we also prove lower bounds for the number $\mathcal{T}$ of 0-tokens (see Section 6). For example, Corollary 22 provides a lower bound of $\mathcal{T} \geq (\frac{c}{e} - 1) \cdot \log_c n$, where e is the Euler number. This shows that if c is constant, the trade-off in our revocation schemes between CA-to-directory communication $\mathcal{P}$ and the number of 0-tokens $\mathcal{T}$ is optimal up to a constant.

2 Definitions

Consider the universe $U = \{1, 2, \ldots, n\}$ of users with personal identification numbers 1 to n. Let 2^U denote the power set of U. Let $R \subseteq U$ be the subset of revoked users, $\bar{R} = U - R$

the complement of R. In our schemes, the CA has to find a family of sets that covers the subset $\bar{R}$ of all non-revoked users. Then the CA issues the i-tokens for all the sets in the cover. A day-i proof for a non-revoked u is a set that contains u.

Definition 1 (interval set) *The* interval set $V = [a, b]$, $V \subseteq U$ *is defined as* $[a, b] := \{x \in I\!N \mid a \leq x \leq b\}$ *. The interval set* $[a, a]$ *is briefly written as* $[a]$*. The* length *of an interval set* $[a, b]$ *is defined as* $b - a + 1$*.*

Definition 2 (interval cover) *We call a family of subsets* $S \subseteq 2^U$ *an* interval cover (IC) *of the interval set* I *iff* $\bigcup_{V \in S} V = I$ *and all subsets* V *are interval sets. If* $|S| \leq k$*,* S *is called a* k*-IC.*

Definition 3 (interval cover family) $F \subseteq 2^U$ *is an* interval cover family (ICF) of U *iff for every interval set* $I \subseteq U$*, there is a subset* S *of* F *such that* S *is an IC of* I*.* F *is a* k*-ICF of* U *iff there is at least one* k*-IC* $S \subseteq F$ *of* I *for every* $I \subseteq U$*.*

Lemma 4 *Assume we have a* k*-ICF* F *of the universe* $U = \{1, \ldots, n\}$ *and an arbitrary* $R \subseteq U$ *with* $|R| = r$*. Then* F *covers* $\bar{R}$ *with at most* $(r + 1)k$ *interval sets.*

Proof: Notice that a subset R of size $|R| = r$ partitions the interval $[1, n]$ in at most $r + 1$ subintervals $\bar{R}_1 \cup \cdots \cup \bar{R}_{r+1} = \bar{R}$. Thus, it suffices to cover these subintervals for covering $\bar{R}$. Since each $\bar{R}_i$ is coverable by F with at most k interval sets, the claim follows. $\Box$

The number of interval sets needed to cover a set $\bar{R}$ in Lemma 4 corresponds to the maximal number of proofs – denoted $\mathcal{P}_F$ – the CA must send to the public directories during an update period. Hence, for a k-ICF F the size of $\mathcal{P}_F$ is always upper-bounded by $(r + 1)k$.

It is also important for the practicality of a revocation scheme that the size of F is polynomial in n and that for every subset $\bar{R}$ of non-revoked users the corresponding ICs can be computed in time polynomial in $\log(n)$.

For an ICF F and every $u \in U$, we define $h_F(u)$ as the *multiplicity* of u in F, that is the number of sets in F containing the element u. Because every set in F that contains u can be part of an interval cover, user u's certificate must include all $h_F(u)$ 0-tokens that contain u. Thus, the maximal number of 0-tokens, denoted $\mathcal{T}_F := \max_u\{h_F(u)\}$, corresponds to the maximal length of a user's public key certificate. This length should be polynomial in $\log(n)$. There is a trade-off between the number of proofs $\mathcal{P}_F$ the CA must send to a public directory and the number of 0-tokens $\mathcal{T}_F$ in a revocation scheme. For instance in the Micali scheme [5, 6], we have $\mathcal{P} = n - r$ and $\mathcal{T} = 1$. In their Generalized scheme, ALO [1] had $\mathcal{P} \leq r(\log_c(n/r) + 1)$ and $\mathcal{T} \leq (2^{c-1} - 1) \log_c n$.

In the next section, we present a $(2k - 1)$-ICF F with $\mathcal{T}_F = O(kn^{2/k})$ for some system parameter k. Taking $k = \log_c n$ leads to $\mathcal{T} = O(c^2 \log_c n)$.

3 A Revocation Scheme using ICFs

First, we introduce a notation on intervals.

Definition 5 (combinational sum) *The* combinational sum *of an interval* $[a_1, b_1]$ *with an interval* $[a_2, b_2]$ *is defined as the interval* $[\min\{a_1, a_2\}, \max\{b_1, b_2\}]$. *We also say, we combine interval* $[a_1, b_1]$ *with* $[a_2, b_2]$. *Let* $W = \{[a_1, b_1], [a_2, b_2], \ldots, [a_m, b_m]\}$ *be a set of disjoint intervals. We define the* maximal combinational sum *of* W *that is contained in an interval* $[a, b]$ *as the interval* $[\min_{a_i} : a_i \geq a, \max_{b_j} : b_j \leq b]$.

Note that the combinational sum of two intervals $[a_1, b_1]$ and $[a_2, b_2]$ may contain elements which are neither in $[a_1, b_1]$ nor in $[a_2, b_2]$.

Next, we define an interval tree T for the interval $[1, n]$ and a parameter k, that might depend on n. The construction is recursive.

Construction of the interval tree T

- The root is labelled with the interval $[1, n]$.
- Each node labelled with an interval $[a, b]$ of length greater than 1 has $n^{1/k}$ children. The children partition the interval $[a, b]$ into equally long pieces. That is, the children are roots of the interval trees for the intervals $[a + i \cdot \frac{b-a+1}{n^{1/k}}, a + (i+1) \cdot \frac{b-a+1}{n^{1/k}} - 1]$, $0 \leq i < n^{1/k}$ (for simplicity, we assume that $n^{1/k}$ is an integer to avoid rounding).

We store the following contents in each node of the interval tree.

- Each node stores the interval of its label.
- Moreover, each node stores the combinational sums of its label with the labels of its right siblings.

Let combinational sums of nodes be defined as the combinational sums of their labels.

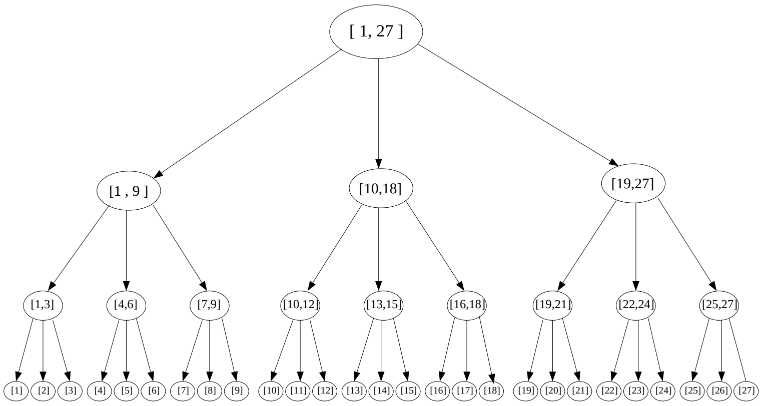

Fig. 1. The interval tree T for $n = 3^3$, $k = 3$

Example: In Figure 1, the node with label $[10, 12]$ stores the interval sets $[10, 12]$, $[10, 15]$ and $[10, 18]$. Its father $[10, 18]$ stores the intervals $[10, 18]$ and $[10, 27]$.

Since in level i the nodes are labelled with intervals of length $n^{(k-i)/k}$, in level k we have interval length 1 and the recursive construction stops. Thus, the interval tree has depth k.

We define the ICF F as the union of all the sets of intervals stored in the nodes of the interval tree T. However, we exclude the root label interval $[1, n]$.

Next, we want to show that F is a $(2k-1)$-ICF, that is, we want to show that we can cover every interval set $I \subseteq [1, n]$ by at most $2k-1$ sets in F. In order to prove this, we present an algorithm that needs a maximum of $2k-1$ combinational sums for covering any interval I.

Algorithm Cover Scheme 1 (CS1)

INPUT: interval $I = [a, b]$

FOR level $i = 1$ TO k in the interval tree T DO
  Take the maximal combinational sums of the label intervals in
  level i of T that is contained in the yet uncovered parts of $[a, b]$.
  IF $[a, b]$ is covered completely, EXIT.

OUTPUT: interval sets $I_1, I_2, \ldots, I_m$ with $\bigcup_{j=1\ldots m} I_j = [a, b]$ and $m \leq 2k-1$.

Example: In Figure 1 on input $I = [2, 21]$, the Algorithm CS1 covers I by taking the intervals $[10, 18]$ (level 1, stored in node $[10, 18]$), $[4, 9]$ and $[19, 21]$ (level 2, stored in nodes $[4, 6]$ and $[19, 21]$) and $[2, 3]$ (level 3, stored in $[2]$).

Lemma 6 *The union F of all intervals stored in the interval tree T is a $(2k-1)$-ICF.*

Proof: We have to show that CS1 needs at most $2k-1$ interval sets to cover $[a, b]$. Notice that CS1 covers the whole interval $[a, b]$ successively from the middle to the borders. In level 1 of the interval tree T, one gets at most one combinational sum $[a_1, b_1]$. The uncovered parts $[a, a_1 - 1]$ and $[b_1 + 1, b]$ both yield at most one additional interval in level 2. This holds because the maximal combinational sums in level 2 are always of the form $[a_2, a_1 - 1]$ respectively $[b_1 + 1, b_2]$. Analogously, we get at most two additional intervals in the subsequent levels. This leads to the upper bound of $2k-1$. $\square$

We define the memory requirement $|F|$ of an ICF F to be the number of interval sets in F. The running time of a k-ICF F on input $I = [a, b]$ is the time to find a k-IC S for I. Further, we define the running time of a k-ICF to be the maximal running time taken over all choices of input intervals I. The following lemma shows that our $(2k-1)$-ICF F can be efficiently implemented.

Lemma 7 *The ICF F has memory requirement $O(n^{1+1/k})$ and running time $O(kn^{1/k})$.*

Proof: In every set of siblings at most $\sum_{i=1}^{n^{1/k}} i = O(n^{2/k})$ intervals are stored. This follows from the fact that each node contains its label and all combinational sums with its right siblings. Hence, F contains at most $O(n^{2/k}) \sum_{i=0}^{k-1} (n^{1/k})^i = O(n^{1+1/k})$ interval sets.

The operations in each level can easily be implemented to run in time $O(n^{1/k})$, that is in the number of children. Thus, the total running time is $O(kn^{1/k})$. $\square$

Definition 8 (RS1) Revocation Scheme 1 (RS1) *uses the* $(2k-1)$*-ICF* F *and Algorithm CS1 in order to cover all interval sets* $\bar{R} = U - R$ *of non-revoked users.*

Theorem 9 *RS1 is a revocation scheme with* $\mathcal{P}_F \leq (r+1)(2k-1)$ *and* $\mathcal{T}_F \leq \frac{1}{4}k(n^{1/k}+1)^2$.

Proof: The number of proofs $\mathcal{P}_F \leq (r+1)(2k-1)$ follows from Lemma 4.

It remains to show the upper bound for $\mathcal{T}_F$. Because the node labels in each level partition the interval $[1, n]$, every element $u \in U$ is stored in exactly one label per level. We want to determine the number of interval sets of a single level in which a user u is contained. Therefore, consider the node with the label interval containing u. Combinational sums are only taken among the $n^{1/k}$ siblings of this node. When enumerating the siblings from left to right, it is easy to see that the i^{th} sibling is in exactly $i \cdot (n^{1/k} - i + 1)$ interval sets. This function in i takes its maximum for $i = (n^{1/k}+1)/2$, leading to $\max_i\{i \cdot (n^{1/k} - i + 1)\} = (\frac{n^{1/k}+1}{2})^2$. Hence, user u can be in at most $(\frac{n^{1/k}+1}{2})^2$ intervals sets per level. Summing over the k levels, we obtain $\mathcal{T}_F \leq \frac{1}{4}k(n^{1/k}+1)^2$. □

Corollary 10 *Choosing* $k = \log_c n$ *for some constant* c*, we obtain a* $(2\log_c n - 1)$*-ICF* F *that needs* $O(n)$ *memory and* $O(\log_c n)$ *running time. RS1 is a key revocation scheme with* $\mathcal{P}_F \leq (r+1)(2\log_c n - 1)$ *and* $\mathcal{T}_F \leq \frac{(c+1)^2}{4}\log_c n$.

Note that our result improves upon the generalized scheme of ALO, who had $\mathcal{T} \leq (2^{c-1} - 1)\log_c n$. A refined analysis of $\mathcal{P}_F$ is given in Section 5.

Even with refined analysis, there remain two problems with the $(2k-1)$-ICF presented above. First, we always assume an upper bound of $2k-1$ for the number of intervals taken by algorithm CS1. Consider a small interval $[a, b]$ with length much shorter than n. Algorithm CS1 will take its first combinational sum in a level i that is close to the leaves in level k. It is easy to see that in this case, CS1 outputs at most $2(k-i)+1$ interval sets. Therefore, Lemma 6 gives a pessimistic bound. Second, after using the first combinational sum in level i, we just need combinational sums of the rightmost or leftmost sibling nodes in the subsequent levels. But we store combinational sums of all sibling nodes. In the next section, we show how to avoid these problems.

4 Another Revocation Scheme based on ICFs

We take an interval tree T' similar to the interval tree T in Section 3. The nodes and their labels remain the same as in T, only their content is changed.

Definition 11 (partial sums) *For each set of sibling nodes in a tree, we call the combinational sums of the leftmost sibling* v *with all other siblings to the right the* right partial sums*. The combinational sums of* v*'s father's leftmost sibling with* v *and all of* v*'s siblings are called the* upper right partial sums *(for an example, see below). The combinational sums of the rightmost sibling* w *with all other siblings to the left – except the leftmost sibling – are called the* left partial sums*. Analogously, the combinational sums of* w*'s father's rightmost sibling with* w *and all of* w*'s siblings are called the* upper left partial sums*.*

Let $W = \{[a_1, b_1], [a_2, b_2], \ldots, [a_m, b_m]\}$ *be a set of partial sums. We define the* maximal partial sum *of* W *that is contained in an interval* $[a, b]$ *as the interval* $[\min_{a_i} : a_i \geq a, \max_{b_j} : b_j \leq b]$.

Example: In Figure 1, the right partial sums of the set $\{[13],[14],[15]\}$ of sibling nodes are $[13]$, $[13,14]$ and $[13,15]$. The left partial sums are $[15]$ and $[14,15]$. The upper right partial sums are the interval sets $[10,13]$, $[10,14]$, $[10,15]$ and the upper left partial sums are $[15,18]$, $[14,18]$ and $[13,18]$.

Notice that we should omit those upper right partial sums where the father of the leftmost sibling v is itself a leftmost sibling, since these combinational sums yield always the label of the father. This holds analogous for the upper left partial sums.

Node contents of the interval tree T′

- Each node stores its label.
- For any sibling nodes in level $k-2j$, $0 \leq j < \frac{k-1}{2}$, the leftmost sibling v stores the right partial sums. Additionally, v stores the upper right partial sums.
- For any sibling nodes in level $k-2j$, $0 \leq j < \frac{k-1}{2}$, the rightmost sibling w stores the left partial sums. In addition, w stores the upper left partial sums.
- Any sibling nodes in level i of the recursion tree are divided in i equally large parts. In any part, each node stores the combinational sums of its label with the labels of its right siblings in this part.

Again, the ICF F' is defined as the union of all intervals stored in the nodes.

Algorithm Cover Scheme 2 (CS2)

INPUT: interval $I = [a,b]$

level $i := 1$
UNTIL a combinational sum is taken DO
  Take the maximal combinational sum of the label intervals in level i of T' that is contained in $[a,b]$. (That combinational sum may consist of up to i intervals.)
  $i := i+1$
FOR level $j = i + (k - i \bmod 2)$ TO k STEP 2 DO
  Take the maximal partial sums of the yet uncovered parts of $[a,b]$.
  IF $[a,b]$ is covered completely, EXIT.

OUTPUT: interval sets $I_1, I_2, \ldots, I_m$ with $\bigcup_{j=1\ldots m} I_j = [a,b]$ and $m \leq k+1$.

Example: We cover the interval $[2,21]$ using Algorithm CS2 and the interval tree T' of Figure 1. CS2 outputs the combinational sum $[10,18]$ in level 1 and the upper partial sums $[2,9]$ and $[19,21]$ in level 3.

Lemma 12 *The union F' of all intervals stored in the nodes of the interval tree T' is a $(k+1)$-ICF.*

Proof: Let Algorithm CS2 take a combinational sum in level i. Since in level i, maximal combinational sums of label intervals can be divided into i parts, we take at most i intervals. In the remaining $k-i$ levels, CS2 can take at most 2 intervals in each of the levels $k-2j$, $0 \leq j < \frac{k-i}{2}$. These are at most $\lceil \frac{k-i}{2} \rceil$ levels. Thus, we obtain the upper bound $i + 2 \cdot \lceil \frac{k-i}{2} \rceil \leq i + 2 \cdot \frac{k-i+1}{2} = k+1$. ◻

Lemma 13 *The $(k+1)$-ICF F' needs $O(n^{1+1/k})$ memory and $O(kn^{1/k})$ running time.*

Proof: The memory requirement of F' is the amount of partial sums and combinational sums. We have at most $2n^{1/k}$ right and left partial sums per set of siblings. The upper partial sums sum up to another $2n^{1/k}$ intervals. Ignoring that these intervals are only taken in each second level we get an upper bound of $4n^{1/k} \cdot \sum_{i=0}^{k-1}(n^{1/k})^i = O(n)$ for the partial sums. Further, each set of siblings stores $O(n^{2/k})$ combinational sums. Summing over the levels gives an upper bound of $O(n^{2/k}) \cdot \sum_{i=0}^{k-1}(n^{1/k})^i = O(n^{1+1/k})$ which is also an upper bound for the total memory requirement.

Since the operations in each level can be implemented to run in time $O(n^{1/k})$, the running time is $O(kn^{1/k})$. ◻

Definition 14 (RS2) *The* Revocation Scheme 2 (RS2) *uses the $(k+1)$-ICF F' and Algorithm CS2 in order to cover all interval sets $\bar{R} = U - R$ of non-revoked users.*

Theorem 15 *The $(k+1)$-ICF F' yields a revocation scheme with $\mathcal{P}_{F'} \leq (r+1)(k+1)$ and $\mathcal{T}_{F'} \leq \frac{k+1}{2} \cdot n^{2/k} + \frac{k}{4} \cdot (2n^{1/k}+1) + \frac{1}{2}(\log k + 1)n^{1/k}$.*

Proof: The upper bound for the number of proofs $\mathcal{P}_{F'}$ follows from Lemma 4.

To complete the proof, we must show that $\mathcal{T}_{F'} \leq \frac{k+1}{2} \cdot n^{2/k} + \frac{k}{4} \cdot (2n^{1/k}+1) + \frac{1}{2}(\log k + 1)n^{1/k}$. Let us start with the partial sums and consider a set of sibling nodes as enumerated from left to right. It is easy to see that the i^{th} sibling is in $n^{1/k} - i + 1$ right partial sums and in $i-1$ left partial sums. This gives a total of $n^{1/k}$ partial sums for each element $u \in U$. Analogously, one can show that each element u is in $n^{1/k}$ upper partial sums. Since each upper partial sum consists of $n^{1/k}$ combinational sums, we get another $n^{2/k}$ intervals. Thus, we have a total of $n^{2/k} + n^{1/k}$ partial sums for every element $u \in U$ in the levels $k - 2j$, $0 \leq j < \frac{k-1}{2}$. Summing over these levels gives us an upper bound of $\lceil \frac{k-1}{2} \rceil \cdot (n^{2/k} + n^{1/k}) \leq \frac{k}{2} \cdot (n^{2/k} + n^{1/k})$.

In addition to the partial sums, we divide all sibling nodes in level i of T' in parts of size $\frac{n^{1/k}}{i}$ and compute the combinational sums with their right siblings in that part. Analogous to the proof of Theorem 9 every element in level i is in no more than $\frac{1}{4}(\frac{n^{1/k}}{i}+1)^2$ of these intervals. Summing over the levels gives

$$
\begin{aligned}
\frac{1}{4}\sum_{i=1}^{k}\left(\frac{n^{1/k}+i}{i}\right)^2 &= \frac{1}{4}\left(\sum_{i=1}^{k}\frac{n^{2/k}}{i^2} + \sum_{i=1}^{k}\frac{2n^{1/k}}{i} + \sum_{i=1}^{k}1\right) \\
&\leq \frac{1}{4}\left(\frac{\pi^2}{6} \cdot n^{2/k} + 2(\log k + 1) \cdot n^{1/k} + k\right)
\end{aligned}
$$

Together with the partial sums computed before we get the desired upper bound for the number of 0-tokens

$$
\mathcal{T}_{F'} \leq \frac{k+1}{2} \cdot n^{2/k} + \frac{k}{4} \cdot (2n^{1/k}+1) + \frac{1}{2}(\log k + 1)n^{1/k}.
$$

◻

Corollary 16 *Taking $k = \log_c n$, RS2 is a revocation scheme with $\mathcal{P}_{F'} \leq (r+1)(log_c n+1)$ and $\mathcal{T}_{F'} \leq \frac{(c+1)^2}{2} \cdot \log_c n(1 + o(1))$.*

If we compare this result with the revocation scheme RS1 of Section 3, we roughly halve the number of proofs $\mathcal{P}$ by doubling $\mathcal{T}$. Since we have update periods frequently, it is preferable to make $\mathcal{P}$ small by slightly enlarging $\mathcal{T}$.

5 ICFs and key revocation

In the previous sections, we studied the covering of *arbitrary intervals* $[a, b]$ by ICFs and connected this to the task of key revocation by Lemma 4. But Lemma 4 yields a pessimistic bound:

- We always expect that r revoked users yield $r + 1$ intervals of non-revoked users. This is no longer true if r becomes large.
- The intervals representing non-revoked users are not arbitrary but disjoint, that is the intervals do not overlap. Further, the average length of the intervals that have to be covered depends on the parameter r.

5.1 The expected number of intervals

In the following, we assume that the revoked users R are uniformly distributed over the interval $[1, n]$ and look for the expected number of intervals of non-revoked users. Let i_1, i_2, ..., i_r be the revoked users in sorted order, that is $i_1 < i_2 < \cdots < i_r$. We call i_j and i_{j+1} a pair iff $i_{j+1} = i_j + 1$. Note that pairs of revoked users do not introduce a new interval that must be covered, since they enclose an interval of non-revoked users of size 0. Let X be the random variable for the number of intervals. Then

$$E_R(X) \leq r + 1 - E_R(\text{number of pairs}).$$

We obtain an upper bound since revoked users at the interval borders 1 and n never yield an additional interval. Thus, the borders 1 and n always pair.
The expected number of pairs is

$$E_R(\text{number of pairs}) = \frac{\binom{n-2}{r-2}}{\binom{n}{r}} \cdot (n-1) = \frac{r(r-1)}{n}.$$

We summarize this in the following lemma.

Lemma 17 *Let the revoked users be distributed uniformly over $[1, n]$ and let F be a k-ICF. Then F yields a key revocation system with an expected upper bound of $\mathcal{P}_F \leq (r + 1 - \frac{r(r-1)}{n})k$ for the CA-to-directory communication.*

5.2 Key revocation with RS1 for growing r

Lemmas 4 and 17 still give pessimistic bounds, since they assume that arbitrary intervals are covered. But CS1 does not always use $2k - 1$ intervals to cover an interval $[a, b]$. If the interval length of $[a, b]$ is small, Algorithm CS1 will not use any intervals in the upper levels of the interval tree T. However, if the number r of revoked users increases then the average interval lengths of intervals that must be covered decreases. Hence, we expect some amortization of costs with growing r.

This fact was studied in ALO [1]. They proved an upper bound of $\mathcal{T} \leq r \log_2(\frac{n}{r})$. Note, that the logarithmic term decreases with increasing r. We show our algorithm to be a generalization of [1] by showing a bound of $2r(\log_c n - \lfloor \log_c r \rfloor) - m$ for arbitrary $c > 2$ and $m = r - c^{\lfloor \log_c r \rfloor}$. Therefore, we adapt the proof techniques of [1]. For $c = 2$, our scheme reduces to the Hierarchical Scheme proposed in ALO[1].

Definition 18 *Let $P(n, R)$ be the number of proofs using the revocation scheme RS1 for covering $U - R$, where $U = \{1, 2, \dots, n\}$. We define $P(n, r) = \max_{R:|R|=r}\{P(n, R)\}$ to be the worst case number of proofs for a revocation set R of r users.*

Assume $n = c^k$.

Theorem 19 *For $r = c^l$, $l \geq 0$, RS1 has $P(n, r) \leq 2r \log_c(\frac{n}{r})$ for $c > 2$, $c \in I\!N$.*

Proof: Similar to the proof in ALO [1]. The proof is given in the full version of the paper. □

In the following theorem, we prove the upper bound for $P(n, r)$ for arbitrary r.

Theorem 20 *For $r = c^l + m$, RS1 yields $P(n, r) \leq 2r(\log_c n - \lfloor \log_c r \rfloor) - m$.*

Proof: The proof is given in the full version of the paper. □

6 Lower bounds for $\mathcal{T}_F$ in a k-ICF F

In this section, we show lower bounds for the number of tokens $\mathcal{T}_F$. Let $U = \{1, 2, \dots, n\}$ be the set of users. We cover arbitrary interval sets $[a, b] \subseteq U$ of non-revoked users by k-ICFs. Comparing the lower bounds to our results in Section 3 and 4 will prove that our revocation schemes are up to constants optimal.

Theorem 21 *Let F be a k-ICF of U, then $\mathcal{T}_F \geq \sqrt[k]{k!} \cdot (n + 1)^{1/k} - k$.*

Proof: We prove a lower bound for covering the interval sets $[1, 1]$, $[1, 2]$, $\dots$, $[1, n]$ with the k-ICF F. This yields a lower bound for covering all interval sets $I \subseteq U$. The bound is proven by induction. For $k = 1$ an optimal family F_1 covering these sets must contain all of the n interval sets. But each of these sets contains the element 1. Thus, $\mathcal{T}_{F_1} = h_{F_1}(1) = n$. The identity

$$\mathcal{T}_{F_k} = h_{F_k}(1), \tag{1}$$

is an invariant of the proof, where F_k denotes an optimal covering scheme with at most k interval sets. The inductive step is done from $k - 1$ to k.

Assume, there's an optimal family F_k covering the sets $[1,1]$, $[1,2]$, ... , $[1,n]$ with at most k interval sets and minimal $\mathcal{T}_{F_k}$. We show in Lemma 24 that we can assume wlog $\mathcal{T}_{F_k} = h_{F_k}(1)$. Hence, invariant (1) holds.

Now, consider the interval sets of F_k containing 1. Let these be the sets $[1,a_1]$, $[1,a_2]$, ..., $[1,a_t]$, where $a_1 = 1$ because F_k must cover the single user 1. By construction, $t = \mathcal{T}_{F_k}$. An auxiliary set is defined by $[1,a_{t+1}]$ with $a_{t+1} = n+1$. The intervals sets $[1, a_{i+1}-1]$, $1 \le i \le t$ are covered by taking the interval set $[1,a_i]$ and an optimal covering in F_k of the remaining interval $[a_i+1, a_{i+1}-1]$ with at most $k-1$ sets.

The element a_i+1 is the first element in the interval $[a_i+1, a_{i+1}-1]$. Hence, it plays the role of the element 1 when covering the sets $[a_i+1, a_i+1], [a_i+1, a_i+2], \ldots, [a_i+1, a_{i+1}-1]$. By equation (1), the element a_i+1 is critical, because it has maximal multiplicity of all the elements in $[a_i+1, a_{i+1}-1]$. Thus, element a_i+1 must be contained in $h_{F_{k-1}}(1)$ interval sets and the induction hypothesis applies with $k-1$ and interval length $a_{i+1}-a_i-1$. Additionally, a_i+1 is contained in the $t-i$ interval sets $[1,a_{i+1}], \ldots, [1,a_t]$. Since $h_{F_k}(1) = t$ and 1 is the element of maximal multiplicity, we obtain for $1 \le i \le t$

$$\sqrt[k-1]{(k-1)!} \cdot (a_{i+1}-a_i)^{\frac{1}{k-1}} - (k-1) + t - i \le t \tag{2}$$

$$\Rightarrow \quad a_{i+1} \le a_i + \frac{(i+k-1)^{k-1}}{(k-1)!}. \tag{3}$$

Solving the recurrence in (3) for $a_1 = 1$ yields

$$\begin{aligned} a_{i+1} &\le a_{i-1} + \frac{(i+k-2)^{k-1}}{(k-1)!} + \frac{(i+k-1)^{k-1}}{(k-1)!} \le \cdots \le \frac{1}{(k-1)!} \sum_{j=1}^{i+k-1} j^{k-1} \\ &< \frac{1}{(k-1)!} \int_{j=0}^{i+k} j^{k-1} dj = \frac{(i+k)^k}{k!} \end{aligned}$$

But we know $a_{t+1} = n+1$, which leads to

$$\begin{aligned} &\frac{(t+k)^k}{k!} \ge n+1 \\ \Rightarrow \qquad & t \ge \sqrt[k]{k!} \cdot (n+1)^{1/k} - k. \end{aligned}$$

□

Using Stirling's formula $k! \sim \sqrt{2\pi k}\left(\frac{k}{e}\right)^k > \left(\frac{k}{e}\right)^k$, we conclude

Corollary 22

$$\mathcal{T}_F > \frac{k}{e} \cdot (n+1)^{1/k} - k > \left(\frac{n^{1/k}}{e} - 1\right) k$$

Taking $k = \log_c n$ yields

Corollary 23

$$\mathcal{T}_F \ge \left(\frac{c}{e} - 1\right) \cdot \log_c n.$$

Lemma 24 *Let F_k be a k-ICF covering $[1,1],[1,2],\ldots,[1,n]$ with minimal $\mathcal{T}_{F_k}$. F_k can be turned into a k-ICF with $h_{F_k}(1) = \mathcal{T}_{F_k}$.*

Proof: The proof is given in the full version of the paper. ◻

Theorem 25 *Let F be a k-ICF of U, then $\mathcal{T}_F > \left(\frac{1}{6}\right)^{\frac{1}{k-1}} n^{\frac{2}{k}}$.*

Proof: Let $F = \{S_1, S_2, \ldots, S_m\}$. Since F is a k-ICF, for every interval set $I \in U$ there exist at most k interval sets $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ such that $S(I) := \{S_{i_1}, S_{i_2}, \ldots, S_{i_k}\}$ is a cover of I. Note, that some S_{i_j} might be empty and there might be several $S(I)$ in F that cover I. For each interval set I we consider an arbitrary but fixed $S(I)$.

Assume $\mathcal{T}_F \leq (\frac{1}{6})^{\frac{1}{k-1}} n^{\frac{2}{k}}$. Fix some interval set $S_i = [a, b]$ of F and consider the number of times this set can be contained in a cover $S(I)$ of some interval set $I = [s, t]$, where $s < a$ or $b < t$. Consider the case $s < a$. By assumption, user $a - 1$ is contained in at most $(\frac{1}{6})^{\frac{1}{k-1}} n^{\frac{2}{k}}$ sets of F. On the other hand, every cover $S(I)$ containing $S_i = [a, b]$ must contain a set S_j which includes the element $a-1$. Next consider the interval $S_j \cup S_i = [c, d]$. Assuming $s < c$ and arguing as above, $S(I)$ must contain one of the $(\frac{1}{6})^{\frac{1}{k-1}} n^{\frac{2}{k}}$ intervals containing $c - 1$. Continuing in this way and using the fact that F is a k-ICF, we conclude that there are at most $\frac{1}{6} n^{\frac{2(k-1)}{k}}$ covers $S(I)$ in which a set S_i can participate. This holds for any S_i:

$$|\{I \subseteq U : S_i \in S(I)\}| \leq \frac{1}{6} n^{\frac{2(k-1)}{k}}. \tag{4}$$

Next, we count the number of elements with multiplicities in all the $\binom{n}{2}$ interval sets I in U. Since there is one interval of length n, two intervals of length $n - 1$, etc., we get $\sum_{I \in U} |I| = \sum_{i=1}^{n} i \cdot (n - i + 1) = \frac{1}{6} n^3 + \frac{1}{2} n^2 + \frac{1}{3} > \frac{1}{6} n^3$. Since the interval sets S_i that cover I can overlap, the following inequality holds

$$\sum_{I \in U} \sum_{S_i \in S(I)} |S_i| \geq \sum_{I \in U} |I| > \frac{1}{6} n^3. \tag{5}$$

Using inequality (4), we also obtain

$$\begin{aligned} \sum_{I \in U} \sum_{S_i \in S(I)} |S_i| &= \sum_{S_i \in F} |S_i| \cdot |\{I \subseteq U : S_i \in S(I)\}| \\ &\leq \frac{1}{6} n^{\frac{2(k-1)}{k}} \sum_{S_i \in F} |S_i| \end{aligned} \tag{6}$$

Combining (5) and (6) leads to

$$\sum_{S_i \in F} |S_i| > n^{3 - \frac{2(k-1)}{k}} = n^{1 + \frac{2}{k}}.$$

Note that $\sum_{S_i \in F} |S_i| = \sum_{u \in U} h_F(u)$. Taking the average number of the multiplicities $h_F(u)$ yields that there must be an element u with $h_F(u) > n^{\frac{2}{k}}$, contradicting the assumption that each element is in at most $(\frac{1}{6})^{\frac{1}{k-1}} n^{\frac{2}{k}}$ sets. ◻

Definition 26 *We call a k-ICF F* δ-optimal, *if for all k-ICFs* $\bar{F}$: $\frac{\mathcal{T}_F}{\mathcal{T}_{\bar{F}}} = O(\delta)$.

Theorem 27 *RS1 uses a* $\min_k\{n^{3/k}, kn^{2/k}\}$*-optimal k-ICF. The* $(k+1)$*-ICF* F' *constructed for RS2 is* $\min_k\{n^{1/k}, k\}$*-optimal.*

Proof: RS1 uses a $(2k-1)$-ICF F with $\mathcal{T}_F = O(kn^{2/k})$. This can be turned into a k-ICF with $\mathcal{T}_F = O(kn^{4/k})$. Dividing by the lower bounds of Corollary 22 and Theorem 25 gives the $min_k\{n^{3/k}, kn^{2/k}\}$-optimality. RS2 uses a $(k+1)$-ICF F' with $\mathcal{T}_{F'} = O(kn^{2/k})$. Applying Corollary 22 and Theorem 25 proves the claim. □

Corollary 28 *For* $k = \log_c n$*, the* $(2k-1)$*-ICF* F *used in RS1 and the* $(k+1)$*-ICF* F' *used in RS2 are* 1*-optimal.*

Note, that we obtain the lower bound in Theorem 21 by covering the intervals $[1,1]$, $[1,2]$, $\ldots$, $[1,n]$. This is only a small subset of all the intervals in $[1,n]$ and the left border is fixed by the element 1. It seems that making both borders variable introduces a factor of $n^{2/k}$, but we can not prove this yet. Thus, we expect a lower bound of $\mathcal{T}_F = \Omega(kn^{2/k})$ for any k-ICF F. This would yield 1-optimality for the ICF F' in RS2 independent of the choice of k.

7 Conclusion

We introduced a new class called ICF for key revocation. Micali's scheme [5, 6] and ALO's Hierarchical scheme [1] belong to this class. We improved upon the former results by reducing the critical update cost for CA-to-directory communication. In practice, the performances of our revocation schemes depend on the expected number r of revoked users. If one expects r to be a small fraction of n, then RS2 is preferable. It avoids a factor of 2 in the communication. RS1 should perform better for large r. We have shown the first lower bounds in this area, proving our schemes to be optimal up to constants.

References

1. W. Aiello, S. Lodha, R. Ostrovsky, "Fast Digital Identity Revocation", Proc. Crypto '98, Lecture Notes in Computer Science Vol. 1462, pages 137-152, 1998
2. S. Even, O. Goldreich, S. Micali, "On-line/Off-line Digital Signing", Proc. Crypto '89, Lecture Notes in Computer Science Vol. 435, pages 263-275, 1989
3. P. Kocher, "A quick introduction to Certificate Revocation Trees", unpublished manuscript, 1998
4. L. Lamport, "Password authentication with insecure communication", Communications of ACM, 24, pages 770-771, 1981
5. S. Micali, "Enhanced Certificate Revocation System", Technical memo MIT/ LCS/TM-542, ftp://ftp-pubs.lcs.mit.edu/pub/lcs-pubs/tm.outbox, 1995
6. S. Micali, "Certificate Revocation System", U.S. Patent number 5666416, 1997
7. M. Naor and K. Nissim, "Certificate Revocation and Certificate Update", Proceedings of 7th USENIX Security Symposium, pages 217-228, 1998
8. Andrew C. Yao, "Space-time trade-off for answering range queries (extended abstract)", Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, pages 128-136, 1982