Fast Digital Identity Revocation

(Extended Abstract)

William Aiello[1] Sachin Lodha[2] Rafail Ostrovsky[3]

[1] Bell Communications Research, *email:* aiello@bellcore.com
[2] Rutgers University Computer Science Department,
e-mail: lodha@paul.rutgers.edu. Part of this work was done while this author visited Bellcore, also partially supported by DIMACS.
[3] Bell Communications Research, *email:* rafail@bellcore.com

Abstract. The availability of fast and reliable Digital Identities is an essential ingredient for the successful implementation of the public-key infrastructure of the Internet. All digital identity schemes must include a method for *revoking* someone's digital identity in the case that this identity is stolen (or canceled) before its expiration date (similar to the cancelation of a credit-cards in the case that they are stolen). In 1995, S. Micali proposed an elegant method of identity revocation which requires very little communication between users and verifiers in the system. In this paper, we extend his scheme by reducing the overall CA to Directory communication, while still maintaining the same tiny user to vendor communication. We contrast our scheme to other proposals as well.

KEYWORDS: Certificate authority, certificate revocation, signatures, public-key infrastructure, digital identities.

1 Introduction

MOTIVATION: Digital identities (and author-identities of messages) are essential for business, private, and government use of the Internet. For example, they are needed for on-line shopping, business-to-business transactions, on-line banking, code-authentication, company-internal identities, etc. (see, for example, [16, 17]). The U.S. Federal government, NIST, the U.S. Post-office, Visa and Master Card, some major banks, and private companies (like VeriSign, SIAC, IBM, GTE, and Microsoft) are all building digital identity infrastructures. While the general design of all these schemes is similar, and relies on public-key cryptography and Certificate Authority services, the details (and hence the efficiency) of what it means for a digital identity to be *valid*, and *how* it can be revoked differs from scheme to scheme.

THE GENERAL SETTING: The general setting consists of two ingredients blended together, namely, public-key cryptography [1] and a Certificate Authority Infrastructure (PKI). We briefly review both notions.

Public-key cryptography encompasses *digital signatures* and *public-key encryption*. A digital signature scheme is a triple of algorithms, a *key generation* algorithm which generates a *public key* and a *secret key*, the *signing* algorithm and the *verification algorithm*. The signing algorithm uses the public and private key to generate a signature for an arbitrary document. The verification algorithm uses just the public key to verify the signature of a document. The specifications require that all signatures generated with the valid secret key are verified by the verification algorithm; and, without the private key it is computationally infeasible to generate a signature for a (previously unsigned) message which the verification algorithm would accept. Public-key encryption is also a triple of algorithms, a *key-generation* algorithm which produces private and secret keys an *encryption* algorithm and a *decryption* algorithm. Anyone, given the public key can encrypt a message using the encryption algorithm, but only those who have the secret key can efficiently decrypt. For formal definitions and further details see [13, 3, 4].

When using public-key cryptography over the Internet, the main issue is the ability to associate the right public-keys to the right individuals/organizations. The overall strategy for doing so is as follows. In its simplest form, we assume one or more trusted **Certificate Authority (CA)** centers which initially run the signature key generation algorithm to compute its own public and secret keys. Each CA holds its secret key under great security but widely distributes (say, publishes in NY-Times) its public key (or uses some other method to "bind" the public key to the CA). Now, any entity, who wishes to establish its digital identity with this CA, first generates its own public-key/private-key pair. Then, it submits its public key along with its name and other required identifying information to this CA. The CA checks the name and identifying information and if satisfied signs the public key and the name of the individual along with some subset of the identifying information, including expiration date a serial number, and possibly other information. (The amount of identifying information and the scrutiny of the information checking establish the level of trust with which a public key is "bound" to the identified entity.)

Now, any user can verify that a message came from this individual by verifying both the signature of the message against a given public key and also verifying that this public key together with the right name and the valid expiration date are properly signed by CA. The above simple scheme extends to various *hierarchies* of CA structures (such as Federal Public Key Infrastructure (PKI) hierarchy, banks, company-internal CA, etc.) where a user will belong to many such hierarchies [14]. (See also [15] and references therein for more general structures.) For all these schemes, no matter what the structure is, one of the main bottlenecks is the issue of *revocation*.

THE MAIN BOTTLENECK: When a public-key, together with an expiration date, is signed by a CA, there remains a risk. That is, a user's private key and certificate may be stolen/compromised, or a company's private key and certificate may be retained by a former employee. Without a method for certificate revocation, these keys may be used by unauthorized parties. *Just as with credit-cards, occasional revocation is a fact of life.* According to some estimates, about 10% of public keys will usually be revoked before they expire [9]. Thus, an important element of any CA scheme (or hierarchy) is its revocation procedure.

Notice, that with revocation, the above simple setting becomes more involved, since now to verify that some public key is valid, one must not only check that it has been signed by the proper CA, but also that is has *not* been revoked. (Moreover, in case of CA hierarchy, one must certify that the CA public key has not been revoked either). Thus, the definition of one's identity depends on the revocation rules in an essential way. In this paper, we extend an elegant method of S. Micali [9, 10] of doing revocations. Before we explain our approach, let us review the design objectives and previous work done on this issue.

THE SETTING: Our setting is the one identical to [9]. We review it here. We consider some finite universe of users. The setting consists of one or several trusted certificate authorities, which distribute, at regular intervals (for example, each day), information regarding revoked certificates to several untrusted directories D. (The reason for having many directories is to allow *replication* of data) For each day, any directory should be able to provide a proof whether any user's u public key is valid or revoked. Of course, we insist that the directory (which is untrusted) can not cheat and change revocation status for any user. The proof of the revocation status the Directory can provide either to user himself (who can then pass it along to any vendor or any other user who wishes to check that the user's identity is still valid). The critical costs of the scheme are of course the size of the proof of the validity of user's identity (i.e., this proof should be as small as possible, since this is the most frequently used operation,) as well as the communication from Certificate Authority to the Directories for each period.

PREVIOUS WORK: We briefly discuss some of the earlier work done on this problem. Previous to our work, three possible revocation strategies were proposed: the "centralized on-line solution"; Certificate Revocation Lists (see, for example, [16]) and Micali's proposal and patent [9, 10]. Independent of our work, two other schemes by Kocher [6] and Naor and Nissim [11] have been proposed. We review all these schemes below and point out the differences.

In the *centralized on-line solution,* similar to credit-cards, there is a CA trusted central database, which maintains the status of all public-keys that have not expired. The user, in order to verify that a public-key has not been revoked makes a query to this central database, and receives an answer. The answer must be authenticated, either implicitly by use of a private network, with a CA signature, or with CA private-key authentication, provided that the CA shares with this user a private authentication key. This centralized solution has several drawbacks: the CA must be accessed for the verification processes on-line (or risk using revoked public-key) and its answer must be authenticated (to prevent man-in-the-middle attacks of somebody impersonating to the CA).

The centralized on-line solution is not acceptable in many cases, and thus a distributed solution is sought. One such solution is the so-called *Certificate Revocation List* (CRL) (see, for example, VeriSign [16]). In this approach the CA composes a list, say each day, of all keys thus far revoked and signs this list. This list along with the CA's signature of the list is the CRL for that day. Each day the CA sends the CRL to all the untrusted revocation *directories* all over the network. To verify a public key certificate, the verifier first checks the expiration date. If the certificate has not expired it contacts any directory for that day's

CRL. If the public key *does not* appear in the CRL it is still valid. This solution is better than the centralized solution, since now many sites maintain the CRL list, and every such site, instead of maintaining the status of all public-keys must only maintain a list of revoked public-keys, where typically R, the number of revoked keys is much less than the number of keys N. The drawback is that the proof that some key is still valid is the entire CRL list, signed by the CA.

In 1995 Micali proposed a scheme in which a verifier needs only a small number of additional bits beyond a user's signature and the user's public key certificate and only a small amount of extra computation to insure that the user has not been revoked [9, 10]. In this scheme, Micali uses the idea of off-line/on-line signatures [2], which in turn builds on the work of Lamport [7]. The basic idea is as follows. Let f be a one-way permutation which is easy to compute but hard to invert (in practice, MD5 or other hash-functions are used). Now, for each user, u, the CA picks a random x_u and computes $y_u = f(f(f \cdots f(x_u)))$, where f is applied, for example, for 365 times to compute y_u. This y_u is called user u's "0-token." For each user the CA includes the user's 0-token along with the users public key, and expiration date and other certificate information in its signed certificate. Now, after the first day, for each user u whose public key has not been revoked that day, the CA releases user u's "1-token", $f^{-1}(y_u)$, to the directories in the network (which is can compute from x_u or from cached intermediate values of $f^i(x_u)$). Note that, given a 0-token for a user u it is easy to check whether a candidate token is a 1-token for user u, but computationally infeasible to compute a 1-token for user u. To verify on the second day that the public key of a user u was not revoked on the first day, a verifier gets a candidate token from a directory or a candidate token from the user (who retrieved it already from a directory). The verifier then checks whether the candidate is a 1-token for the user. In general, if u's certificate is not revoked up through day i, the CA provides u's i-token, $f^{-i}(y_u)$, to the directories. Verifier on the $i+1$st day check whether candidate tokens are i-tokens for users. The advantage of the scheme is that communication between a directory and a user (or verifier) is small and need not be authenticated. If a given protocol calls for the user to retrieve the day i-token from the directory then the token remains a short certificate for day $i+1$ that user has not been revoked. In addition, verifying a day i-token is less expensive then verifying the signature of a CRL. The disadvantage is that the scheme requires (for each day) the CA to send $N - R$ tokens to each directory where N is the number of public keys and R is the number of revoked keys. Micali also considers (as in our setting) the case where a directory may act maliciously by not forwarding the day i-token of a user it received from the CA to the user or verifier. A non-revoked user could thus be considered revoked by a verifier. To prevent this Micali modifies his scheme as follows. For each user the CA also chooses a random x'_u, the "revocation token," and includes $f(x'_u)$ in u's certificate, along with the 0-token and other certificate information. When u's public key is revoked the CA sends the revocation token to the directories. When a directory is queried about a user on day $i+1$ it must return either the day i-token or the revocation token for that user.

Recently, Naor-Nissim [11] have improved upon a scheme of Kocher [6]. We will describe the requirements of their scheme. For the details of the scheme see [11]. For R revoked keys, in the Naor-Nissim scheme the CA maintains an authenticated 2-3 tree of hash values of depth $O(\log R)$. For each public key

which is revoked in an update period, the CA must update this special tree by computing $O(\log R)$ new hash values. As long as one key is added to this revocation tree the root must be recomputed and signed. The CA sends to the directories the lists of users to insert and delete, and a signature of the root. The directories insert and delete the users from the tree, recompute the hash values and check the signature of the root. When a user queries a directory concerning the revocation status of a key, the directory sends $O(\log R)$ hash values and the signature of the root. To verify the revocation status of a key, a user must compute $O(\log R)$ hash values, compare them in a specified way with the hash values from the directory and verify the signature of the root hash value. In summary, the CA to Directory communication of Naor-Nissim scheme is superior to our scheme (as well as Micali's scheme) but the verification of the revocation status of the user (or vendor) is smaller for Micali scheme and our scheme.

OUR RESULTS: We will build on Micali's scheme [9, 10] to maintain its advantage of efficient verification. Recall that in Micali's scheme every user's certificate contains one 0-token. In order for the user to remain unrevoked on day $i+1$ the CA must publish the i-token associated with the user's 0-token. In our scheme the CA will incorporate k 0-tokens into the certificate for each user rather than just one, where k will be a parameter of our scheme. The sets of 0-tokens for the users will be distinct, however, the sets may intersect substantially. The rule for remaining unrevoked will now be slightly different. In order for a user to remain unrevoked on day $i+1$ the CA need only publish an i-token associated with *one* of the user's 0-tokens. Since the sets of 0-tokens may intersect, one i-token may act as a certificate of non-revocation for many users. Thus the CA may not need to publish $N - R$ tokens when R keys have been revoked as in Micali's scheme. For example, we show that if the number of 0-tokens in each certificate is $\log N$, and the sets of 0-tokens in each user's certificate are selected properly, then the number of tokens a CA must publish during an update period is $R \log N/R$.

In this paper we insist that the proof from user to vendor of the validity of user's ID remains very small, as in Micali's scheme. For the scheme of Naor-Nissim, this is not the case (i.e., while [11] substantially save on the CA to Directories communication over [9], their proof of validity is bigger). In this paper, we study the tradeoff between the number of 0-tokens needed in a certificate and the number of tokens a CA must publish per update period. We summarize our results (and compare it to Micali's scheme) below. Note that N stands for the number of not-yet-expired certificates, R stands for number of revoked certificates, and c is a parameter in the *general scheme*, typically a constant.

Scheme Name	# Tokens from CA to directories	# Tokens per Certificate
Micali's Scheme	$N - R$	1
Our Hierarchical Scheme	$R \log (N/R)$	$\log N$
Our Generalized Scheme	$R \log_c (N/R) + R$	$(2^{c-1} - 1) \log_c N$

Table 1. Comparison of Our Results and Micali's scheme

Notice that if the number of revocations is small our scheme is faster then Micali's approach. However, even if the number of revocations is large, and in fact for any value of R, $0 \leq R \leq N$ we show that the information update cost is better then the above table indicates, and in fact is always smaller then $N - R$ of Micali's scheme. In practical settings, both our hierarchical scheme and our general scheme with $c = 3$ provide savings in communication compared to the Micali's approach. (For example, for the hierarchical scheme, with 10% revocation rate we gain 2.7 multiplicative factor improvement, and for 1% revocation rate we gain 14.9-fold improvement over Micali's communication from CA to Directories, while still required a single hash-value (i.e. 128 bits) proof of the validity of one's identity. For the generalized scheme with $c = 3$ we get 2.9-fold improvement for the 10% revocation rate and 19-fold improvement for 1% revocation rate) If the percentage of revoked users is smaller then the savings are more substantial. We should remark here that Micali's method to avoid malicious directories by including one extra token in each certificate applies to our scheme as well and we will not explicitly include this in the description of our scheme.

In the Naor-Nissim scheme [11], the communication from CA to Directories is smaller then our scheme and Micali's scheme. However, both Micali's scheme and our scheme, the directories need only send one token (of 128 bits) in response to a query about any key.

PAPER OUTLINE: In the next section we describe our scheme and prove a bound of $R \log R/N$ on the number of tokens the CA must send the directories. In Section 3 we describe a generalization of our scheme which requires the CA to send asymptotically fewer tokens to the directories using a novel complement cover construction. (We believe that this construction is of independent interest, and might be useful for other purposes as well.) In Section 4 we describe a variant of our scheme which includes an *incremental feature*. That is, if some public key is revoked on day i, we do not have to pay for this on the following days (i.e. we pay for each revocation only once.) We now proceed with the details of our scheme.

2 Hierarchical Scheme

2.1 Description of the Scheme

For concreteness, we will assume that certificates are valid for D days and that certificates are revoked once per day. Let f be a one-way permutation from n bits to n bits. Consider the following sequence: $(r, f(r), f^2(r), \ldots, f^D(r))$ where r is a uniformly chosen n bit string. We will call this the *chain of* r and denote it by $\mathbf{c}_r$. We will index this sequence in reverse order and starting from 0. That is, the 0th element of the sequence,$\mathbf{c}_r(0)$, is $f^D(r)$. We will call this the 0-token of the chain of r. Similarly, $\mathbf{c}_r(1) = f^{D-1}(r)$ which we will call the 1-token of the chain of r. Note that the 1-token is the inverse of the 0-token, i.e., $\mathbf{c}_r(1) = f^{-1}(\mathbf{c}_r(0)) = f^{D-1}(r)$. More generally, define $\mathbf{c}_r(j) = f^{-j}(\mathbf{c}_r(0)) = f^{D-j}(r)$, $j \geq 1$. We will call $\mathbf{c}_r(j)$ the j-token of the chain of r.

Note it is easy to verify whether a token τ is an i-token of a chain once the 0-token (or a j-token, $0 \geq j < i$) of a chain is known. Simply compute

$f^i(\tau)$ ($f^{i-j}(\tau)$) and check for equality with the 0-token (j-token). However, by the properties of a one-way permutation, given any number of the j-tokens of a chain, $0 \geq j < i$, it is computationally infeasible to compute the i-token of the chain.

Before we describe our revocation scheme it will be useful to review Micali's revocation scheme. In the Micali scheme, for each user v there is a random n bit string r_v and a chain $\mathbf{c}_{r_v}$. For notational simplicity we will hereafter refer to this as the chain of v and denote it by $\mathbf{c}_v$. In this scheme the 0-token of v, $\mathbf{c}_v(0)$, is incorporated into the certificate of v which is signed by the CA. If user v has not been revoked up to and including day i then the CA will issue the i-token of v in response to queries about the status of user v's certificate. Otherwise, the CA will not issue an i-token.

To verify that signer v has not been revoked up to and including day i, a verifier must have v's certificate and a candidate i-token τ for v which he obtains from either the CA, a directory, or v (who had previously obtained it from the CA or a directory). He accepts that v has not been revoked up to and including day i if τ is verified to be the i-token of v.

We will now describe our first and simplest scheme for revocation. For simplicity we will assume that the number of users $N = 2^l$ is a power of 2 and that all users are issued certificates at the beginning of day 1. (We remark that these assumptions can be relaxed. We introduce them here for the clarity of exposition.) Each user is given an l bit id $v \in \{0, 1\}^l$. We associate these ids with the leaves of a labeled complete binary tree of height l in the natural way. That is, label the root by the null string ϕ, and label its two children by 0 and 1. Given any node in the tree whose label is a binary string s of length less than l, label one child with $s0$ and the other with $s1$. For each node u in this tree the CA computes a chain $\mathbf{c}_u$.

For an l bit id, v, let $(\phi = v_0, v_1, \ldots, v_{l-1}, v_l = v)$ denote the nodes on the path from the root to v in this tree. Call the set of chains for this path the *path of chains of* v which we denote by $\mathbf{pc}_v$, i.e., $\mathbf{pc}_v = (\mathbf{c}_{v_0}, \ldots, \mathbf{c}_{v_l})$. Furthermore, call the set of i-tokens of this path of chains the *path of i-tokens of v* and denote this symbolically as $\mathbf{pc}_v(i)$, i.e., $\mathbf{pc}_v(i) = (\mathbf{c}_{v_0}(i), \ldots, \mathbf{c}_{v_l}(i))$. Of course, given the path of 0-tokens of v it is easy to verify whether a token τ is on the path of i-tokens of v (and even easier to verify whether τ is the i-token of, say, the jth node on the path).

The CA's Certificates: When creating a certificate for user v, the CA includes $\mathbf{pc}_v(0)$, the path of 0-tokens of v, in the certificate (rather than just including the 0-token of chain $\mathbf{c}_v$ in the certificate as in the Micali scheme). Note that this path consists of $\log N + 1$ 0-tokens.

The CA's Update Operation: Let R_i be the set users which have been revoked at some point up to and including day i. At the end of day i, the CA will calculate a subset of nodes of the labelled tree, called the *day-i verification nodes* with the following two properties.

1. For every $v \notin R_i$, there is at least one node of the path of v which is a day-i verification node.
2. For every user $r \in R_i$, none of the nodes on the path of r is a day-i verification node.

For each node which is a day-i verification node, the CA can compute (or has cached) the i-token of that node. Call the collection of all such i-tokens the *day-i verification tokens*.

The CA sends the day-i verification tokens to the directories.

A Directory's Query Response: In response to queries on day $i+1$ concerning user u, if u has not been revoked through the end of day i, then at least one node on the path of u is a day-i verification node. The directory issues the i-token of one such node. This i-token is, by definition, a member of the day-i verification tokens. If u has been revoked, no node on the path of u is a member of the day-i verification nodes and, hence, the directory cannot issue an i-token.

The Verifier's Operation: A verifier v must verify both the signature of a signer u and u's certificate. This certificate contains the path of 0-tokens of u. We assume that the verifier has retrieved u's certificate and a candidate i-token, τ, from either the signer, the CA, or a directory. He accepts that u has not been revoked up to and including day i if τ is verified to be an i token of one of the nodes on the path of u.

By property 1 of the day-i verification nodes, for a non-revoked signer u, an i token of one of the nodes on the path of v will be available for the verifier and the verifier will thus accept v.

By property 2 of the day-i verification nodes, and the definition of one-way permutation, it is computationally infeasible for a revoked user, u, or any accomplice to deliver a token to the verifier which the verifier will accept as being an i-token of a node on the path of u.

Calculating the Day-i Verification Nodes Given a set of revoked users R_i, the CA can calculate the day-i verification nodes by operating on the labelled tree as follows. First, mark as "revoked" every node in the path of every revoked user. Now, the day-i verification nodes consists of all non-revoked nodes which are children of a revoked node. It is easy to verify that this set satisfies the two properties above and moreover, that it is the minimal set which satisfies these two properties.

As an example, suppose that none of the users have been revoked on day one, then the set of day-1 verification tokens consists of a single token: the 1-token of the root. Since this 1-token is on every users' path of 1-tokens, a verifier which is given this token will accept every user.

As another example, suppose that the user denoted by the string of all 1s is revoked the 1st day. The day-1 verification tokens consists of the 1-tokens of the l nodes $0, 10, 110, \ldots, 11\cdots110$. It is easy to verify that this set can be used to accept all but the revoked user.

In the next section we analyze the size of the set of verification nodes as a function of the number of revoked users.

2.2 Size of the Set of Verification Nodes

In this section we derive an upper bound on the number of day-i verification nodes (and hence the number of day-i verification tokens) as a function of the number of nodes which have been revoked up to and including day i. Let N denote the number of users and r be the number of revoked users.

Definition 1. *Let $\bar{T}(N,R)$ be the number of verification nodes in the labelled tree for N users when the set of revoked users (leaves) is R. Let $T(N,r) = \max_{|R|=r} \bar{T}(N,R)$.*

We will assume that N is a power of 2. We now prove the following.

Theorem 1. *For $r = 0$, $T(N,r) = 1$. For $1 < r \leq N/2$, $T(N,r) \leq r \log(N/r)$. And, for $N/2 \leq N$ $T(N,r) \leq N - r$.*

Proof Sketch: Let $N = 2^l$ so that the labelled tree is a complete binary tree of height l. When $|R| = r = 0$, the set of verification nodes just consists of the root of the labelled tree and thus $T(N,0) = 1$ for all N. When $r = 1$, our algorithm will mark as revoked all the nodes on the path of this revoked user. The algorithm puts all nodes which are children of revoked nodes into the set of verification nodes. There are $\log N$ such children. So, $T(N,1) = \log N$ $\forall N$.

Now consider the case when $r > 1$. First consider the case when $r = 2^k$, $k \geq 1$, is a power of 2. Consider the r subtrees in the labelled tree whose roots are the nodes at depth k in the labelled tree. These trees have height $l - k$. We define the family of revoked sets $\tilde{\mathcal{R}}$ as follows. $\tilde{R}$, for $|\tilde{R}| = r$, is in $\tilde{\mathcal{R}}$ if there is exactly one revoked user of $\tilde{R}$ in each of the subtrees of height $l - k$.

By our previous analysis for one revoked user in a tree, for any $\tilde{R} \in \tilde{\mathcal{R}}$ each subtree of height $l - k$ requires $l - k$ verification nodes. Note that no other verification nodes are required. Hence, for any $\tilde{R} \in \tilde{\mathcal{R}}$, the total number of verification nodes, $T(N,\tilde{R})$, is just $r(l-k) = r \log N/r$. We will now show that for any set $R \notin \tilde{\mathcal{R}}$, that $\bar{T}(N,R) < r \log N/r$. Actually we will prove the stronger statement in the lemma below. We will use the following definition. For any set of revoked users R let s_R be the number of subtrees of height $l - k$ with no elements of R.

Lemma 1. *For r a power of 2 and for any set of revoked users R of size r there is a $\tilde{R}$ in $\tilde{\mathcal{R}}$ such that $\bar{T}(N,R) \leq \bar{T}(N,\tilde{R}) - s_R$.*

Proof: We begin with two simple lemmas.

Lemma 2. *Let R be the set of revoked leaves and consider a maximal subtree with no revoked leaves of height h. Let u be a leaf of this subtree. Then, $\bar{T}(N, R \cup \{u\}) = \bar{T}(N,R) + h - 1$.*

To see this note that for the set R the subtree was covered by one verification node at the root of the subtree. However, the number of verification nodes in the subtree after u is added to the revoked set is h. □

Lemma 3. *Let R, for $|R| \geq 2$ be the set of revoked leaves and consider a minimal subtree containing exactly two elements of R of height h'. Let u be one of the two elements. Then $\bar{T}(N, R - \{v\}) = \bar{T}(N,R) - (h' - 2)$.*

To see this note that in R the subtree has $2h - 2$ verification nodes whereas $R - \{v\}$ it has h verification nodes.

We will prove the Lemma 1 by induction. That is, we will show that for any R with $|R| = r$ and $s_R \geq 1$, there is an R' with $|R'| = r$ such that $s_{R'} = s_R - 1$ and $T(N,R) \leq T(N,R') - 1$. Clearly, the lemma will then follow. □

So, consider an R with $|R| = r$ and $s_R \geq 1$ subtrees of height $l-k$ with no elements of R. Let u be one of the leaves of one of these subtrees. Let v be a leaf in R of a subtree of height $l-k$ with two or more leaves in R. Such a subtree must exists. Let $R' = (R/\{v\}) \cup \{u\}$. Note that the number of subtrees of height $l-k$ with no elements of R' is $s_R - 1$. The maximal subtree containing u but containing no elements of R has height at least $l-k$. The minimal subtree containing v and exactly one other element of R has height at most $l-k$. Thus using the two claims, we conclude that $\bar{T}(N,R) \leq \bar{T}(N,R') - 1$. This concludes the proof of the lemma. □

We now consider the case when r is not a power of 2. We write r as $2^k + m$ where $k = \lfloor \log r \rfloor$ and $0 < m < 2^k$. We define the family of revocation sets $\tilde{\mathcal{R}}$ in a similar fashion to the case when r is a power of two. The revocation set $\tilde{R}$ is in $\tilde{\mathcal{R}}$ if $\tilde{R}$ has the following properties. Call a subtree of height $l-k-1$ a i-subtree if i of its leaves are in $\tilde{R}$. Now, of the 2^{k+1} subtrees of height $l-k-1$, $r = 2^k + m$ of them are 1-subtrees and $2^k - m$ of them are 0-subtrees. Furthermore, each of the 0-subtrees is "paired" with a 1-subtree. That is, the sibling of the root of each 0-subtree is a root of a 1-subtree. (Note that such a pairing is possible since the number of 0-subtrees is strictly less than the number of 1-subtrees.)

Now let us consider the number of verification tokens required for a $\tilde{R}$ in $\tilde{\mathcal{R}}$. Because each 0-subtree is paired with a 1-subtree, the root of each 0-subtree must be a verification node. The number of verification nodes required for each 1-subtree is just $l-k-1$. Thus, for each $\tilde{R}$ in $\tilde{\mathcal{R}}$, $T(N,\tilde{R}) = r(l-k-1) + 2^k - m$. If $r > N/2$ so that $k = l-1$, then $T(N,\tilde{R}) = 2^k - m = N - r$. Otherwise $T(N,\tilde{R}) = r(l-k) - 2m$. Since

$$r \log (r/2^k) = r \log (1 + \frac{m}{2^k}) \leq m \frac{r}{2^k} < 2m$$

it follows that $r \log (N/2^k) - 2m \leq r \log (N/r)$.

As before, for an arbitrary revocation set R we define s_R to be the number of 0-subtrees for R. Note that $s_R \geq 2^k - m$ for every R of size $r = 2^k + m$. As in the case when r is a power of two, the revocations sets in $\tilde{\mathcal{R}}$ have the maximum number of verification nodes. This follows from the following lemma.

Lemma 4. *Consider a set of revoked users R whose size, $2^k + m$, is not a power of 2. For each such R there is a $\tilde{R}$ in $\tilde{\mathcal{R}}$ such that $\bar{T}(N,R) \leq \bar{T}(N,\tilde{R}) - (s_R - (2^k - m))$.*

Proof: The proof is in two stages. In the first stage we start with an arbitrary R and find an R' with $2^k - m$ 0-subtrees such that $\bar{T}(N,R) \leq \bar{T}(N,R') - (s_R - (2^k - m))$. The first stage is identical to the proof of Lemma 1. It proceeds by induction. Each step reduces the number of 0-subtrees by one and increases the number of verification nodes by at least one. The second stage begins with R' and ends with an $\tilde{R} \in \tilde{\mathcal{R}}$. We will illustrate the first step of this stage. If $R' \notin \tilde{\mathcal{R}}$ then there are two 0-subtrees whose roots are siblings and two 1-subtrees whose roots are subtrees. Let v be one of the elements of R' in one of the two 1-subtrees and let u be a leaf of one of the 0-subtrees. Let $R'' = (R'/\{v\}) \cup \{u\}$. By our previous claims $T(N,R') \leq T(N,R'') - 1$. This process can proceed by induction on the number of 0-subtrees paired with 1-subtrees until we have a verification set $\tilde{R} \in \tilde{\mathcal{R}}$. □

We remark that $T(N,r) \leq N - r$. This can be shown algebraically or just by observing that by construction, the number of verification nodes is never more than the number of non-revoked users. The number of tokens the CA needs to issue after r users have become revoked is at most $T(N,r)$ in our scheme, and hence, this is at most the number of tokens the CA needs to issue in the Micali scheme which is $N - r$. For small r there is a large gap between the number of tokens which need to be issued in both schemes. For example, if $r = N/64$ (i.e., a revocation rate of approximately 1.5%), then $N - r = (63/64)N$ is larger than $T(N,r) \leq (6/64)N$ by almost a factor of ten.

3 Generalized Hierarchical Scheme

In this section we will first describe a set-theoretic object we call a *complement cover family*. We will then show that it properly captures the idea of assigning tokens to the non-revoked users. Finally, we will describe several complement cover families which thus yield constructions for the revocations problem.

3.1 Complement Cover Families

Consider a universe U of size N. Let R be any subset of U and let $\mathcal{S}$ and $\mathcal{F}$ denote families of subsets of U. Let $\bar{R} = U - R$ denote the complement of R.

Definition 2. $\mathcal{S}$ is a *complement cover* of R iff $\bigcup_{W \in \mathcal{S}} W = \bar{R}$.

Definition 3. $\mathcal{F}$ is a *complement cover family* of U iff for every subset R of U, $\mathcal{F}$ contains a complement cover of R. That is, for every subset R of U, there is a subset $\mathcal{S}$ of $\mathcal{F}$ such that $\mathcal{S}$ is a complement cover of R.

We will see below that for all of our constructions of a complement cover family $\mathcal{F}$, for every subset R of U, there is a unique subset $\mathcal{S}$ of $\mathcal{F}$ which is a complement cover of V. We will call such families *unique* complement cover families.

For any unique complement cover family $\mathcal{F}$, and every subset R of U, let $CC_{\mathcal{F}}(R)$ denote the unique complement cover of R which is contained in $\mathcal{F}$. Furthermore, denote the function $\delta_{\mathcal{F}}$ as the size of the complement cover of $\mathcal{F}$ which is defined as follows: $\delta_{\mathcal{F}}(r) = \max_{R \in U, |R|=r} |CC_{\mathcal{F}}(R)|$.

For every complement cover $\mathcal{F}$ and every element of $u \in U$, define $H_{\mathcal{F}}(u)$ as the sets in $\mathcal{F}$ which contain u. Let $h_{\mathcal{F}}(u) = |H_{\mathcal{F}}(u)|$ denote the *height* of u in $\mathcal{F}$. By abuse of notation, define the *height of* $\mathcal{F}$, $h_{\mathcal{F}}$ as $\max_{u \in U} h_{\mathcal{F}}(u)$. Note that if $u \notin R$ then the intersection of $CC_{\mathcal{F}}(R)$ and $H_{\mathcal{F}}(u)$ is not empty, otherwise it is empty. This follows from the definition of a complement cover family.

We will now show how a unique complement cover family can be used as a data structure for certificate revocation. Let U be the set of users and $\mathcal{F}$ a unique complement cover family of U. The CA creates a chain for each set S of $\mathcal{F}$. Call this chain $\mathbf{c}_S$.

The CA's Certificates: Recall that for every user u, $H_{\mathcal{F}}(u)$ is the collection of subsets which contain u. For each such subset there is a chain. Call this set of chains the chains of $H_{\mathcal{F}}(u)$. Denote the set of i-tokens of the chains of $H_{\mathcal{F}}(u)$ as the i-tokens of $H_{\mathcal{F}}(u)$. The certificate signed by the CA of each user u contains the 0-tokens of $H_{\mathcal{F}}(u)$.

The CA's Update Operation: As before, let R_i denote the set of users which have been revoked at some point up to and including day i. At the end of day i, The CA will set the day-i verification sets to be $CC_{\mathcal{F}}(R_i)$, the complement cover of R_i. Define the set of day-i verification tokens to be the i-tokens of each set in the complement cover of R_i. The CA sends the day-i verification tokens to the directories.

A Directory's Query Response: In response to queries on day $i+1$ concerning user v, if v has not been revoked through the end of day i, the directory computes the intersection of $H_{\mathcal{F}}(v)$ and $CC_{\mathcal{F}}(R_i)$. The directory issues an i-token of one of the sets in the intersection. Note that this i-token is in the set of day-i verification tokens. If v has been revoked, the intersection is empty and, hence, the directory cannot issue a token.

The Verifier's Operation: The day-i verification tokens are used by verifiers on day $i+1$ to test whether users have or have not been revoked at some point up through the end of day i. This is done as follows. For a signer, v, as before a verifier is given v's certificate. This certificate contains the 0-tokens of $H_{\mathcal{F}}(v)$. In addition the verifier retrieves a candidate i-token, τ, from either the signer, the CA, or some other source. He accepts that v has not been revoked up to and including day i if τ is verified to be one of the i-tokens of $H_{\mathcal{F}}(v)$.

By the definition of the complement cover of R_i, for a non-revoked user v one of the i-tokens of $H_{\mathcal{F}}(v)$ will be available for the verifier and the verifier will thus accept v. In addition, by the definition of one-way permutation, it is computationally infeasible for a revoked user or any accomplice to deliver a token to the verifier which the verifier will accept as being an i-token of $H_{\mathcal{F}}(v)$.

Since unique complement cover families are useful as revocation data structures we are interested in finding a family $\mathcal{F}$ of U such that

1. $|\mathcal{F}|$ is $O(N^c)$ for some small constant c, i.e., number of chains required in the system is practical.
2. $h_{\mathcal{F}}$ is small, i.e., the number of 0-tokens in the certificate of every user is small.
3. The complement cover of every set, $CC_{\mathcal{F}}$, can be efficiently computed.
4. The size of the complement covers, $\delta_{\mathcal{F}}(\cdot)$ as a function of the size of the revoked users is small.

3.2 Example Complement Cover Families

We consider two extreme, but simple, solutions for this problem :

- $\mathcal{F} = \{\{1\},\{2\},\ldots,\{N\}\}$. Given any R, $U - R = \bigcup_{x \notin R}\{x\}$. Thus, $\mathcal{F}$ is a complement cover of U. Clearly, $|\mathcal{F}| = N$. Note that $\delta_{\mathcal{F}}(r) = N - r$ and $h_{\mathcal{F}} = 1$. Observe that this corresponds to Micali's scheme.
  Note that since $\delta_{\mathcal{F}}(r) = N - r$, the number of tokens the CA must issue when r is small is quite large.
- $\mathcal{F} = 2^U$. Given any R, $U - R \in \mathcal{F}$. So, $\delta_{\mathcal{F}}(r) = 1$. Thus, the number of day-i verification tokens is 1 regardless of the number of revoked users. However, the certificate size is $h_{\mathcal{F}} = 2^{N-1}$ and $|\mathcal{F}| = 2^N$. Thus, the solution is obviously not practical because the size of the certificates and the CA's data structure is exponential.

These two families stand at two opposite poles - the first family has small size and small $h_{\mathcal{F}}$ but a large $\delta_{\mathcal{F}}$, and the second family has a small $\delta_{\mathcal{F}}$ but large size and large $h_{\mathcal{F}}$. Below we will describe a scheme which has the advantages of both of the above.

3.3 Description of General Scheme

We will now describe a construction for a unique complement cover family which is parameterized by integer $c \geq 2$. For simplicity, suppose that $N = c^l$ for some positive l. The first subset we put into the family is the entire set of c^l elements. We next divide the elements into c subsets of size c^{l-1} each and put these subsets into the family. These are level 1 subsets. Further divide each level 1 subset into c subsets of size c^{l-2}. These are level 2 subsets. Continue dividing the subsets and adding them to the family until all the level l subsets of size one have been added to the family. The collection of level i subsets which are subsets of the same level $i-1$ subset are called siblings and the level $i-1$ subset is called their parent. For example, in Fig. 1, $\{4\}, \{5\}$ and $\{6\}$ are siblings in level 2 and their parent is $\{4,5,6\}$. Note that thus far we have $(c^{l+1}-1)/(c-1)$ subsets in the family and every element of U is in $l+1$ sets.

If $c = 2$ we are done. Otherwise, we will continue to add to this family as follows. For each collection of sibling subsets do the following. First, form subsets which are the union of any two siblings. There are $\binom{c}{2}$ such subsets. Add these to the family. Then, form the subsets which are the union of any three siblings. There are $\binom{c}{3}$ such subsets. Add these to the family. In general, form the $\binom{c}{i}$ subsets which are the union of i siblings, for $1 < i < c$, and add these to the family. This completes the subsets of the family.

Using edges to represent the (maximal, nontrivial) subset relation, this complement cover family can be represented by a graph. See Fig. 1 for an example when $c = 3$ and $N = 3^2$.

The size of this family is

$$\frac{c^l - 1}{c-1}(2^c - 2) + 1 \;\leq\; N\frac{2^c}{c-1}.$$

Also, it is easy to see that every element of U is in $l(2^c-1)+1 = h_{\mathcal{F}} \leq 2^c \log N / \log c$ subsets of $\mathcal{F}$.

The complement cover of a revoked set R, $CC_{\mathcal{F}}(R)$, can be computed as follows. As in the hierarchical scheme, first mark as revoked each singleton set whose sole member is a member of R. In Fig. 1 these sets are denoted by an asterisk. Then mark as revoked each subset of $\mathcal{F}$ which is a superset of any of these marked sets. In the graph representation, start by marking the revoked leaves. Then mark the immediate supersets of the marked leaves. Continue working up the graph until every superset of a marked set has been marked. In Fig. 1, the marked sets are marked with an asterisk. The elements of $CC_{\mathcal{F}}(R)$ are the maximal subsets of $\mathcal{F}$ which have not been marked. In Fig. 1, they are circled. These elements are easy to find. For each set of siblings at a given level in which at least one of the sets, but not all, is marked as revoked do the following. If exactly one of the siblings is not marked, it is a member of $CC_{\mathcal{F}}(R)$. For example, node $\{3\}$ in Fig. 1. Otherwise, there is a unique subset of the parent of the siblings

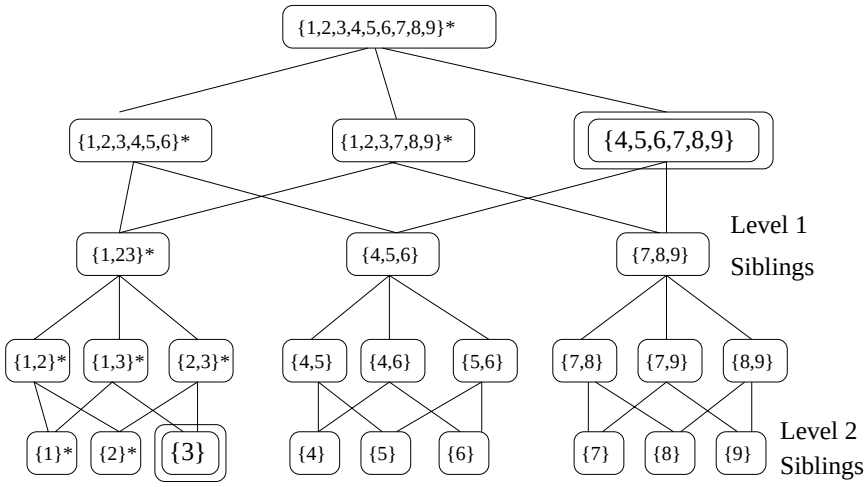

Fig. 1. An example of the complement cover of $\{1, 2\}$ for $c = 3$ and $N = 9$. The revoked nodes are marked with an asterisk. The sets which comprise the complement cover are circled.

which is the union of the unmarked siblings. For example, node $\{4, 5, 6, 7, 8, 9\}$ in Fig. 1. This set is a member of $CC_{\mathcal{F}}(R)$.

The size of the complement covers is given by $\delta_{\mathcal{F}}(r) \leq r \log_c N/c^k - m \leq r \log_c N/r - c^k$. This can be shown using methods similar to those for the hierarchical scheme.

It is instructive to instantiate the scheme for several values of c.

1. $c = 2$: This is the Hierarchical scheme. The size of the complement cover is $|\mathcal{F}| = 2N - 1$, $\delta_{\mathcal{F}}(r) \leq r \log(N/r)$ and $h_{\mathcal{F}} = \log N + 1$.
2. $c = 3$: This is an interesting case. It gives us a complement cover family of size $3N-2$ with $\delta_{\mathcal{F}}(v) = \frac{v}{\log 3} \log \frac{N}{v}$). Note that there is a 50% increase in size of $\mathcal{F}$ compared to the Hierarchical scheme, but more than a 50% decrease in δ (since $\log 3 = 1.585$). So, it could be an alternative to the Hierarchical scheme.
3. $c = \log N$: $|\mathcal{F}| = (O(N^2)$, $\delta_{\mathcal{F}}(v) \leq v \frac{\log N/v}{\log \log N})$ and $h_{\mathcal{F}} \leq N \frac{\log N}{\log \log N}$. Although this setting achieves the lowest value for the size of the complement cover, $|\mathcal{F}|$ and $h_{\mathcal{F}}$ may not be practical for moderately large N.

4 Incremental Version

Under the current scheme, if r_1 users are revoked on day 1 and r_2 users are revoked on day 2, the number of day 2 verification nodes is a function of r_1 whereas the number of day 3 verification nodes is based on $r_1 + r_2$ although the number of users revoked the previous day was only r_2. We propose the following modification to our scheme such that the number of verification nodes needed for a given day depends only on the number of users revoked the previous day and not on all of the users revoked thus far.

For the first modification we replace each chain of tokens of length 365 with a hash tree of depth $2 \cdot 365$, analogous to [8, 5]. The root value of the hash tree becomes the 0-token. There is one such hash tree and 0-token associated with every node in the labelled tree of N elements. On day i, for each node v in the day-i verification set, the CA gives the $\log(2 \cdot 365) + 1$ hash values needed to compute the path from leaf $2i$ in the the hash tree to the 0-token associated with the node v. Call this the day-i verification path for node v.

Note that given the day-j, for $1 \leq j < i$, verification paths for node v, it is computationally infeasible to compute the day-i verification path for v. This motivates our second modification to the scheme. Recall that each user's certificate contains $\log N$ 0-tokens, one for each node in the labelled tree on the path from the user's id to the root. Previously, for a user to be verified on day i, the verifier would a need a day-i verification path for one of the user's 0-tokens. We change the protocol as follows. On day i, the CA computes the day-i verification set of nodes in the labelled tree using just the users revoked the previous day. The CA sends to the directory a day-i verification path for each node in the day-i verification set. Now, for a user to be verified on day i, the verifier will need a day-1 through day-i verification path with the additional property that the root of each of the verification paths is contained in the set of 0-tokens in the user's certificate.

This scheme reduces the CA to directory communication costs substantially. It can be shown that the average daily cost is proportional to at most $(R/365)\log(365N/R)$ where R is the total number of revocations throughout the year. Of course, this reduction is gained at the expense of a larger communication requirement for the verifier which is proportional to i for day i. This latter communication requirement can be reduced to $\log 365$ by requiring the CA to produce incremental verification paths for the $\log 365$ times scales of 2 days, 4 days, 8 days, etc. This only increases the average daily communication cost of the CA by a factor of 2.

References

1. W. Diffie, M. Hellman, "New directions in cryptography", *IEEE Trans. on Inf. Theory*, IT-22, pp. 644–654, 1976.
2. S. Even, O. Goldreich and S. Micali "On-line/Off-line Digital Signatures" *CRYPTO* 1989.
3. S. Goldwasser, S. Micali "Probabilistic Encryption" JCSS Vol. 28 No. 22. April 1984.
4. S. Goldwasser, S. Micali, R. Rivest "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks" SIAM Journal of Computing, Vol. 17, No 2, April 1988, pp. 281-308.
5. C. Jutla and M. Yung "Paytree: amortized signature for flexible micropayments" In Second USENIX workshop on Electronic Commerce, November 1996.
6. P. Kocker, "A quick introduction to Certificate Revocation Trees (CRTs)," http://www.valicert.com/company/crt.html.
7. L. Lamport "Password authentication with insecure communication" Communications of ACM, 24(11):770-771, November 1981.
8. R. C. Merkle, "A Certified Digital Signature," *Proceedings of Crypto'89*, pp. 234–246, 1989.

9. S. Micali "Enhanced Certificate Revocation System" *Technical memo MIT/LCS/TM-542*, November 1995. *available online* URL ftp://ftp-pubs.lcs.mit.edu/pub/lcs-pubs/tm.outbox/
10. S. Micali "Certificate Revocation System" U.S. Patent number 5666416, issued Sep. 9, 1997.
11. M. Naor and K. Nissim, "Certificate Revocation and Certificate Update," *Proceedings of USENIX '98.*
12. R. Rivest "The MD5 message-digest algorithm" Internet Request for Comments, April 1992. RFC 1321 *available online* URL http://theory.lcs.mit.edu/rivest/publications.html
13. R. Rivest, A. Shamir, L. Adleman "A Method for Obtaining Digital Signature and Public Key Cryptosystems" Comm. ACM, Vol 21, No. 2, 1978.
14. R. Rivest, B. Lampson "SDSI-A Simple Distributed Security Infrastructure" *available online* URL http://theory.lcs.mit.edu/ rivest/sdsi10.html
15. M. Reiter, S. Stubblebine "Towards Acceptable Metrics of Authentication" In Proc. of 1997 IEEE Symposium on Security and Privacy.
16. VeriSign Corporation, *available online* URL http://www.verisign.com/
17. Microsoft "Proposal for Authenticating Code Via the Internet" April 1996, *available online* URL http://www.eu.microsoft.com/security/tech/authcode/authcode.htm