Vol. 47 No. 8 IPSJ Journal Aug. 2006

*Regular Paper*

# Online Certification Status Verification with a Red-Black Hash Tree

Hiroaki Kikuchi,[†] Kensuke Abe[††] and Shohachiro Nakanishi[†††]

Certificate revocation is a critical issue for a practical, public-key infrastructure. A new efficient revocation protocol using a one-way hash tree structure (instead of the classical list structure, which is known as a standard for revocation) was proposed and examined to reduce communication and computation costs. A tree approach, however, may run in $\mathcal{O}(n)$ time, in the worst case, i.e., when all the entries are sorted in descending order. A red-black tree is a binary sorted tree, with one extra bit per node, which is used for balancing the tree and guarantees that search and insertion operations take $\mathcal{O}(\log n)$, even in the worst case. In this paper, we propose a binary red-black hash tree for certificate revocation and prove that it reduces costs more than any other tree balancing approache. We also compare the estimated communication and computation cost reductions of the red-black hash tree to those of the conventional binary search tree.[☆]

## 1. Introduction

Timely certificate revocation information is required for practical applications such as high-value funds transfer and large stock trades. The periodic Certificate Revocation Lists (CRLs) commonly used in the current public-key infrastructure (PKI), are not effective when a private key is compromised. In addition, CRLs have very expensive data structures, because the size of the CRL is proportional to the number of certificates, $n$, issued by a certification authority (CA), and for the most part, CRLs may not be used to verify a target certificate. Many other reasons were pointed out by Rivest[12)].

The Online Certificate Status Protocol (OCSP), was proposed by the IETF PKIX Working Group[3),4)]. OCSP responses consist of the identification of the responder, a target certificate identifier, and a certificate status, either "good", "revoked", or "unknown". This OCSP must be digitally signed by a trusted responder whose public key is certified by the CA who issued the target certificate. One drawback of OCSPs, is in their level of security: the responder has to be online, but at the same time, has to protect its private key against intruders from the Internet. Thus, an off-line server is much safer.

The first step in addressing this difficulty was taken by Kocher, who proposed a Certificate Revocation Tree (CRT), in which the leaves are statements concerning revoked certificates, and the root hash value is signed by the CA[5),11)]. The responder can prove the status of any given certificate by showing the path from the root to the target leaf without digitally signing the response, because the digital signatures for any leaf (or revoked certificate) are identical. Thus, no trust in the responder is necessary. The communication costs between responder, CA, and end users are as low as the resulting OCSP costs.

Naor and Nissim[13)] proposed an Authenticated Dictionary, which further reduces communication by balancing the revocation tree. In a binary search tree, basic operations such as: search, insert, and delete, run in $O(\log n)$ on average, where $n$, is the number of revoked certificates. However, in the worst case scenario, if revocation happens in order of the serial numbers, the cost would be $O(n)$. "Bulk revocation" is likely to occur when a group of certificates issued in a certain interval are all revoked for some reason, e.g., a faulty operator, a misconfiguration of CA private key, a smart cryptanalysis of signature algorithm, or a bankruptcy, and this effect should be taken into account.

There are many tree balancing algorithms. AVL trees are balanced by applying rotation,

† Department of Information Media Technology, School of Information Technology and Electronics, Tokai University
†† Panasonic Mobile Communication Co. Ltd.
††† Department of Information Science, School of Information Technology and Electronics, Tokai University

☆ The original version of this paper was presented at the First International Workshop for Asian PKI (IWAP2001).

and B-trees are balanced by manipulating the degrees of the nodes [1]. Naor and Nissim discussed the use of 2-3 trees, in which any node of a tree may contain at most three children, as balancing trees. However, in the particular application of public-key revocation, the degree of the B-tree does not necessarily minimize the overall communication between directory and end-users.

We propose the red-black tree as an appropriate alternative for certificate revocation. The red-black tree is a tree-balancing algorithm that has one extra bit of information (red or black) per node that is used to balance the whole tree. In this paper, we propose a new scheme for certificate revocation using a red-black hash tree. Our scheme consists of (1) a red-black hash tree in which all revoked certificates are specified as leaves and corresponding hash values are nodes and (2) a subtree, defined by the path of a given target certificate (leaf) to the root, that proves the integrity of response to certificate status queries.

The advantage of the red-black hash tree is in communication cost. We prove that the binary balanced tree has the optimal degree, minimizing the number of bits in a path from the root to the target node in a hash tree. The size of the path results in an optimized communication cost between the OCSP responder and the end user. In addition, any path in a binary tree is individually represented by a sequence of hash values, which is easily contained in the conventional CRL format.

The goal of this paper is to clarify:

- the optimal degree of a balanced revocation tree in terms of communication cost between the directory and users,
- how much the red-black certificate revocation tree can be expected to reduce communication and computation costs for verification processes, and
- how much overhead is required to balance the directory that computes a path from the CRT root to the requested nodes.

In order to answer these questions, we implemented an online certificate status server using a red-black certificate revocation tree. This paper describes the performance of the server and estimates cost reduction based on actual revocation data.

The rest of this paper is organized as follows. In Section 2, the fundamental definitions and assumptions are laid out and related work is described. In Section 3, we propose a scheme using a red-black tree structure and detail the communication cost properties of the scheme. In Section 4, we describe how to deploy our scheme in the existing PKI environment. The revocation syntax and encoding rules are presented in ASN.1 format. With a trial implementation, we demonstrate the performance of the scheme in terms of communication and computation overhead. In Section 5, we conclude this paper.

## 2. PKI Model and Certificate Revocation

### 2.1 PKI Model

In a PKI, there are three entities, as shown in **Fig. 1**:

1. **Certification Authority (CA):** A trusted party responsible for certification of public keys. A CA issues public-key certificates that specify user identification, a corresponding public key, an expiration date, a serial number, and related certificate information, all of which is digitally signed with the CA's private key. For security reasons, CAs are isolated from the Internet.
2. **Directory:** A non-trustworthy set of distributed servers maintains certificates and a CRL (or CRT) database. A directory provides online services to the users, including certificate and CRL distribution, search facilities, and certificate status checking. A directory guarantees the consistency of the database, but is not responsible for its contents. (The term "directory" includes a certificate repository, an OCSP server, and a responder.)
3. **Users:** A certificate user with a corresponding private key. Users do not trust

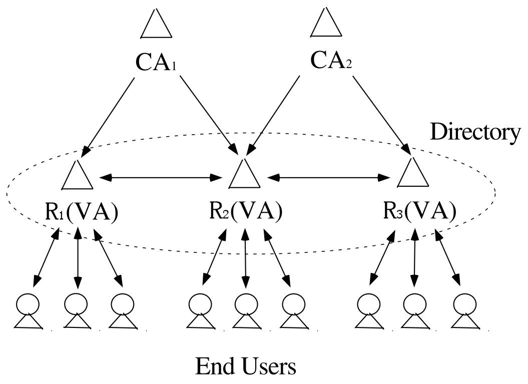

**Fig. 1** PKI model.

each other, trusting only the CA(s). A user sends a signed message to another user, who then verifies the digital signature of the certificate. When a user needs more reliable verification about a certificate, e.g., in a high-value funds transfer, the verifier sends a query to the directory as to whether the certificate is revoked or not.

### 2.2 Related Work

This section gives a brief overview of certificate revocation schemes that have been presented so far. In the following section, we will compare the proposed scheme and the existing schemes.

#### 2.2.1 Online Certificate Status Protocol (OCSP)

Myerrs, et al. proposed an Online Certificate Status Protocol (OCSP)[4], where a trusted party called an OCSP responder provides timely certificate status, which is digitally signed by the OCSP responder. Although OCSP is an IETF standard and widely used for commercial services such as VeriSign[20], it has the drawback of needing the responder to be online and thus vulnerable to online attacks including DoS. In addition, the large overhead for digital signatures on request is an issue.

#### 2.2.2 Certificate Revocation Status (CRS)

Micali presented a lightweight certificate validity checking scheme using one-way hash functions[6]. The scheme, called Certificate Revocation Status (CRS), allows the responder to answer revocation queries without any signing operation. The scheme was revised in 2002[7] so that the responder can be distributed.

#### 2.2.3 Efficient and Fresh Certification (EFECT)

Gassko, et al. presented a new certification scheme using c-statement. The scheme, called EFECT, which stands for Easy Fast Efficient Certification Technique, certifies individuals with a hash tree of c-statements, in which the root is signed by the CA. EFECT uses a B-tree for certificate retrieval and thus requires $O(\log n)$ cost for retrieval.

#### 2.2.4 One-Way Accumulator

Faldella et al. proposed the use of a one-way accumulator, which is cryptographically primitive, verifying many statements with a constant size certificate in Ref. 9).

#### 2.2.5 Security Mediator (SEM)

Bonel, et al. proposed SEcurity Mediator (SEM)[10] in 2001. In their scheme, using the threshold RSA algorithm, a public key is split into two parts, one for the user and the other for the SEM. Every time a user wishes to decrypt or sign a message he needs to interact with the SEM, thereby giving the SEM control over revocation.

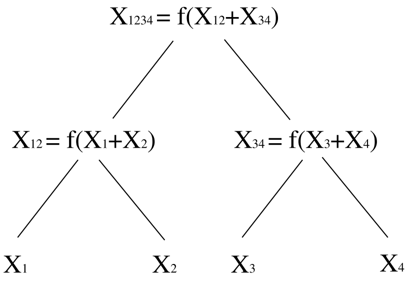

**Fig. 2** Certificate Revocation Tree. Four certificates identified by serial numbers, $X_1, \ldots, X_4$ are jointly fed into a one-way hash function $f$.

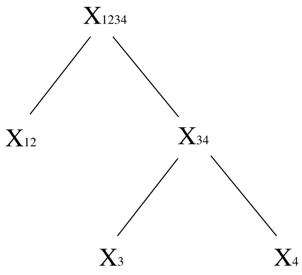

**Fig. 3** Subtree to revoke $x_4$. Hash value $X_{12}$ is used instead of $X_1$ and $X_2$.

#### 2.2.6 Certificate Revocation Tree (CRT)

A CRT is a digitally signed hash tree in which leaves represent revoked certificates identified by serial numbers, $X_1, \ldots, X_n$, where $X_1 < \cdots < X_n$. In a CRT, nodes are hash values computed for a concatenation of child nodes. Let us suppose that a CRT has four leaves, as in **Fig. 2**. A node, $X_{i,i+1}$, is given by

$$X_{i,i+1} = f(X_i + X_{i+1}),$$

where $f()$ is a collision intractable hash function and + denotes concatenation. In practice, any one-way hash function such as MD5[21] or SHA 1[22] can be used as a collision intractable hash function. In this example, the root hash value, $X_{root} = f(f(X_1+X_2)+f(X_3+X_4))$, is dependent on all leaves and cannot be forged by a new tree such that $X_{root} = X'_{root}$ and $X'_{root}$ consist of serial numbers other than $X_1, \ldots, X_4$, assuming collision intractable hash function $f()$.

On a verification request from a user, a directory responds with a subtree, which contains a path from a root to an appropriate leaf. When the requested certificate has already been revoked, a directory extracts the subtree consisting of the revoked certificate $x_4$, as shown in **Fig. 3**.

### 2.2.7 Other Schemes

Zheng categorizes and analyzes many other attempts, summarizing the important properties of various certificate revocation schemes [14].

## 3. Revocation with a Red-Black Tree

### 3.1 Red-Black Tree

A red-black tree [1] is a binary search tree that satisfies the following red-black properties:

(1) Every node is either red or black.
(2) Every leaf (NIL[☆]) is black.
(3) If a node is red, then both its children are black.
(4) Every simple path from a node to a descendant leaf contains the same number of black nodes.

Each of the $n$ nodes in a red-black tree has the fields $p$ (parent), *key*, *left*, and *right*. Fields $p$, *left*, and *right* designate the nodes corresponding to parent, left child, and right child, respectively. The field *key* stores key value so as to satisfy $key[y] \leq key[x] \leq key[z]$ for any node $y$ in the left subtree of $x$ and any node $z$ in the right subtree.

The number of black nodes on node $x$ of a leaf is called the black height, and denoted by $hb(x)$.

The basic operations include search, insert, and delete, which are used to verify whether a target certificate has been revoked and to update the database about any revoked certificates. These operations run in $O(h)$ time, where $h$ is the height of the tree. The scheme guarantees that the red-black tree does not exceed twice the optimal height.

**Theorem 1** [1] A red-black tree with $n$ internal nodes has maximum height $= 2\log(n+1)$.

The insertion of a node into red-black tree, $T$, is accomplished in time $= O(\log n)$. The following gives an outline of the procedure described in Cormen et al. [1].

**Algorithm** RB-Insert($T$, $x$)

```
1 Insert($T$, $x$)
2 $color[x]$ = Red
3 if $color[y]$ = Red
4    then Rotate1($T$, $x$)
5    else if $x$ is right child
6            then Rotate2($T$, $x$)
7            else Rotate3($T$, $x$)
8 $color[root[T]]$ = Black,
```

[☆] NIL is a special symbol to denote an empty node. We regard these NILs as leaves of the tree [1].

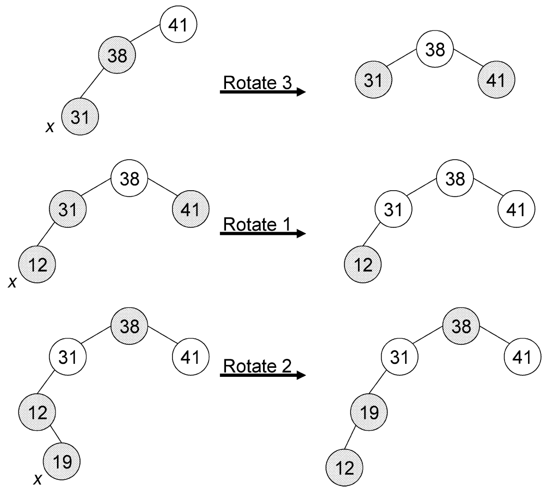

**Fig. 4** Rotations in red-black tree. Recently inserted nodes are labeled with $x$.

where $y$ is the "uncle" of $x$, defined by $y = right[p[p[x]]]$ (or $y = left[p[p[x]]]$). A new node $x$, is inserted according to the ordinal procedure for a binary search tree and then colored red. In order to satisfy the red-black properties, the three procedures for rotation, Rotate 1, 2 and 3, are applied depending on the color of $y$. Rotate 1 requires a recursive step after setting the new node $x = p[p[x]]$, until the target node comes to the root node. **Figure 4** illustrates how each rotation works when nodes 31, 12, and 19 are inserted in the tree in turn. From this we see that the insertion time increases as the height of the tree increases. Note that other rotations do not vary this way. Accordingly, the basic operations require time $= \mathcal{O}(h)$ even in the worst case. The details of this procedure and of other operations are provided by Cormen et al. [1].

**Figure 5** illustrates the red-black tree, and the ordinal binary search tree, successively inserting nodes 41, 38, 31, 12, 19, 8 into an initially empty tree. All the paths from root node 38 to the leaves have exactly the same black-height, $hb = 3$. The circles around nodes 8 and 19 indicate the color red. Note that the balance of the red-black tree is not optimal, and there exists a shorter optimal tree.

### 3.2 Certificate Revocation with a Red-Black Hash Tree

In this section, we define red-black certificate revocation trees, in which nodes correspond to revoked certificates and the path to the root is

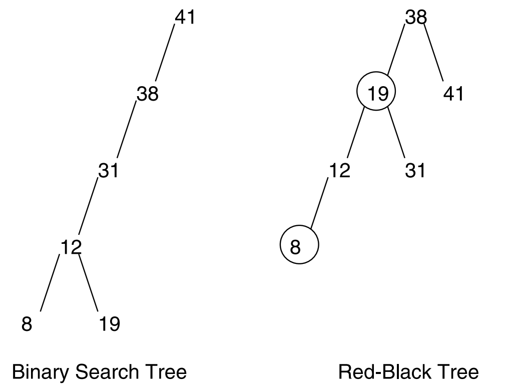

**Fig. 5** Example red-black tree algorithm balancing a binary search tree (left). When keys 41, 38, 31, 12, 19, and 8 are successively inserted, the red-black tree is rotated so as to satisfy red-black properties.

used to verify revocation status.

A key of node $x_i$ is defined by

$$k_i = key[x_i] = h(s_i + v_i),$$

where $h()$ is the secure collision intractable hash function, and symbol $+$ denotes concatenation. Here, $s_i$ and $v_i$, are the serial number of a revoked certificate and the revocation date, respectively, with which a hash value $k_i$ is computed.

The commutative hash value of node $x_i$, in tree $T$, with $n$ internal nodes is defined by:

$$H_i = h(x_i) = h(h(left[x_i]) + key[x_i] + h(right[x_i])).$$

At the leaves of the tree, the hash value computation terminates with $h(NIL) =$ null. The root hash value of tree $T$, $h(T) = h(Root[T])$, is used to digitally sign the whole tree. The security of the hash tree is based on the assumption that no collision with the same root hash value can be found.

Note that the hash values are independent of the colors of nodes. The attribute of color is used only to balance the updated tree with a newly revoked certificate. Verification processes at the directory and end users, are identical to those of the conventional binary search tree. This simplifies the implementation and the communication protocol.

The unique order of insertion of the entries determines a tree. The tree with $n$ internal nodes is of the form:

$$Z = z_1, z_2, \ldots, z_n$$

where, for $i = 1, \ldots, n$,

$$z_i = k_i + H_i + c_i + s_i + v_i.$$

The value, $c_i$, is a Boolean value indicating the color of the node, i.e., true = black, and false = red.

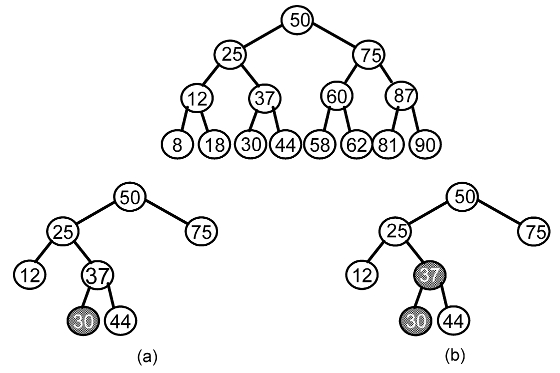

**Fig. 6** Subtrees — (a) verifying that node 30 has been revoked and (b) verifying two neighbor nodes of node 33, i.e., node 33 has NOT been revoked so far.

On receiving a verification request from the end user, the directory responds to the subtree with the digital signature of $T$. If the target certificate has been revoked, the subtree consists of a simple path of nodes, $P$, from the root to the target node; otherwise the subtree consists of a path to two neighbor nodes. For instance, the red-black tree in **Fig. 6** shows two types of subtrees — (a) proves that node 30 has been revoked and (b) proves that node 33 has not yet been revoked by showing the two nearest neighbor nodes, 30 and 37. The directory need not send the whole tree shown above.

A subtree with $m$ nodes is of the form:

$$Y = y_1, y_2, \ldots, y_m$$

where, for $j = 1, \ldots, m$,

$$y_j = \left\{ \begin{array}{cl} k_i & \text{if } x_i \in P \\ k_i : H_i & \text{if } p[x_i] \in P \end{array} \right.$$

where $P$ is a subset of nodes in a simple path from the root. The siblings of a node in $P$ are necessary to compute the hash value of the subtree.

### 3.3 Optimal Degree for Communication Cost

In this section, we explain the cost of communication in revoking certificates using a $k$-ary hash tree. **Figure 8** illustrates sample $k$-ary trees for $k = 3$ and $k = 4$.

**Definition 1** Let $k$ be a degree of hash tree. The cost of communication in revoking certificates using a $k$-ary hash tree is the bit size of data structure to be sent as evidence of revocation (and of non-revocation).

In a standard CRL, communication cost is

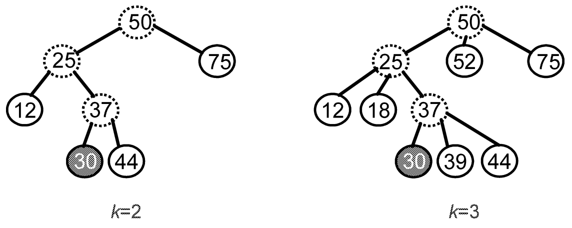

**Fig. 7** Siblings in terms of degree $k$.

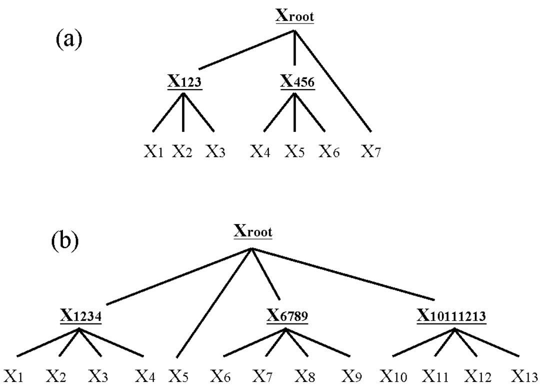

**Fig. 8** Examples of 3-ary and 4-ary hash tree.

$O(n)$. In a certificate revocation tree, the cost of communication is the average size of a subtree containing a path from root to leaves.

Clearly, as $k$ increases, the height of tree decreases, with communication cost decreasing correspondingly. See for instance **Fig. 7**, where sibling nodes along the path to the root are indicated for $k = 2$ and 3. Generally, the average height of a subtree is $d = \lceil \log_k(n) \rceil$, which is easily minimized by $k = n$. This is trivial because the tree in which the root directly follows all leaves is the same as the ordinal CRL. It should be noted that any path from the root to a leaf requires the remaining $n-1$ adjacent leaves to be sent as well. The trivial tree, therefore, does not reduce communication at all.

Let us consider the overhead cost of adjacent hash values in a non-trivial $k$-ary hash tree. The number of hash values adjacent to the path increases with $k$. See for instance (Fig. 8), which proves that $X_7$ being in tree (a) requires the directory to send hash values not only on the path from $X_{root}$ but also hash values on the paths from the adjacent nodes, $X_{123}$ and $X_{456}$ to $X_7$. In the example of tree (b) ($k = 4$), hree adjacent hash values, $X_{1234}$, $X_{6789}$, and $X_{10111213}$, are required to prove $X_5$.

**Theorem 2** Cost of communication in revocation of certificate using a $k$-ary hash tree is $\Theta(k \log_k(n))$.

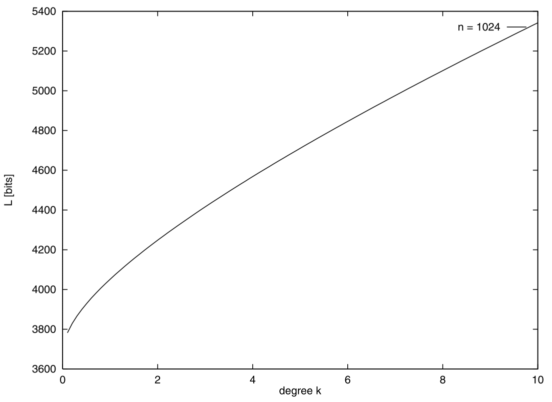

**Fig. 9** Size of a partial tree with $k$-ary hash with respect to $k$ when $n = 1{,}024$. Behavior of $L(k)$ represents expected communication cost.

**Proof:** A number of adjacent hash values in a balanced $k$-ary subtree is given by $c_k = d(k-1)$, where, $d$ is the average height. Hence, we have the size of subtree as

$$L_k = \alpha c_k + \beta = \alpha(k-1)\lceil \log_k(n) \rceil + \beta, \quad (1)$$

where $\alpha$ is the bit length of one hash, and $\beta$ is a constant that doesn't depend on $n$. Function $\log_k(n)$ is monotonic in degree $k$ for all values of $k > 1$. So, we have

$$L_k = \alpha(k \log_k(n) - \log_k(n)) < O(k \log_k(n)).$$

Since we can naturally assume $k < n$, letting $\gamma = \log_n(n)$, we have

$$\alpha(O(k \log_k(n)) + \gamma) + \beta < L_k.$$

Thus, we have shown that the size of a subtree is bounded above and below by $\mathcal{O}(k \log_k(n))$. The communication cost, which is defined as proportional to $L_k$, is therefore asymptotically bounded in $\Theta(k \log_k(n))$. □

**Theorem 3** The optimal degree of $k$-ary hash tree in terms of communication cost is 2.

**Proof:** In order to identify the optimal degree in terms of communication, by taking partial derivatives of Eq. (1) with $k$, we have

$$\frac{dL_k}{dk} = \alpha \log_k n - \alpha \frac{\log n}{(\log k)^2} + \alpha \frac{\log n}{k(\log k)^2},$$

where the second term cannot exceed the sum of the first and third terms when $k > 1$ as

$$\alpha \log n + \alpha \frac{\log n}{(\log k)^2}\left(\frac{1}{k} - 1\right) > 0,$$

which shows that the function $L_k$ is increasing for all $k > 1$. $k = 1$ cannot happen because it would yield hash chaining with no effect on reduction of communication cost. Therefore, the minimum meaningful value of $k$is 2. □

Based on the sample CRL we assign the size of CRL as $\alpha$=64[bit] and $\beta$=3,608[bit], and demonstrate the performance in **Fig. 9**, which shows that size $L$ increases as $n$ increases and as $k$ increases. Therefore, we actually see that $k = 2$ minimizes communication cost $L$.

## 4. Implementation and Estimation

### 4.1 Hash Values of Red-Black Tree

The X.509 version 2 CRL syntax is defined by the ASN.1 shown in **Fig. 10**. Revoked certificates are uniquely identified by the issuer and the serial number `CertificateSerialNumber`, which are locally assigned by the issuer. Optionally, the revocation date and the entry extensions can be provided for each entry.

The sample red-black tree in Fig. 5 is specified by a sequence in **Table 1**. Note that we omit hash computation of $k_i$, which should be as long as the commutative hashes because of the compatibility with the example in Fig. 5. In this particular example, we use the MD5 algorithm as a collision intractable hash function. For the sake of reducing the balancing overhead, the nodes were sorted with the higher nodes coming first.

### 4.2 Encoding Rules of Subtrees

**Table 2** shows the sequence used to verify that the certificate with serial number 19 has already been revoked in the red-black tree of Fig. 5. The target entry is indicated by $[k_i]$. The sequence is necessary for and sufficient to recover the hash value of the root in Table 1.

We define the syntax for subtrees in **Fig. 11**. Note that most elements are the same as the CRL syntax except for the `hash` to identify hash algorithms used in hash trees and `hashValue` to provide hash values of the nodes and `target` to show if the node is the target certificate to be verified.

### 4.3 System Specifications

We have developed an online certificate status server using the red-black tree in order to estimate the cost reduction of the proposed protocol. **Table 3** shows our system's specifications. The server and client are implemented as a Java application.

### 4.4 Communication Cost

The communication between the directory and end users is proportional to the size of the directory response, which contains the subtree that verifies the authenticity of the target certificate. The size of the subtree depends on the height of revocation trees. **Figure 12** illustrates the behaviors of the communication costs of the binary search tree and the red-black

```
CertificateList ::= SEQUENCE {
tbsCertList TBSCertList,
signatureAlgorithm AlgorithmIdentifier,
signatureValue BIT STRING }

TBSCertList ::= SEQUENCE {
version Version OPTIONAL,
-- if present, shall be v2
signature AlgorithmIdentifier,
issuer Name,
thisUpdate Time,
nextUpdate Time OPTIONAL,
revokedCertificates SEQUENCE OF SEQUENCE {
userCertificate CertificateSerialNumber,
revocationDate Time,
crlEntryExtensions Extensions OPTIONAL
-- if present, shall be v2
} OPTIONAL,
crlExtensions [0] EXPLICIT Extensions OPTIONAL
-- if present, shall be v2
}
```

**Fig. 10** X.509 version 2 CRL syntax. Data to be signed is ASN.1 DER encoded.

**Table 1** Sequence of a revoked certificate with hash values.

| $k_i$ | $H_i$ | $c_i$ |
|---|---|---|
| 38 | 6668E1611A4AB23E914331331289A436 | true, |
| 19 | ADCF72C11A19927D45E3186802694145 | false, |
| 12 | 1AB899864B6F979EB680D52C779AD659 | true, |
| 8 | C4D9042407EE87310AB5EB633D81EF35 | false, |
| 31 | 7BFF3DE72AC837CF4DD82476688AF75B | true, |
| 41 | 8D2F06B58513A943BD634C595E85FFBB | true |

**Table 2** Sequence representing subtree that verifies node 19 has already been revoked.

| $k_i$ | $H_i$ |
|---|---|
| 38 | N/A, |
| [19] | N/A, |
| 12 | 1AB899864B6F979EB680D52C779AD659, |
| 31 | 7BFF3DE72AC837CF4DD82476688AF75B, |
| 41 | 8D2F06B58513A943BD634C595E85FFBB |

```
CertificateTree ::= SEQUENCE {
tbsCertTree TBSCertTree,
signatureAlgorithm AlgorithmIdentifier,
signatureValue OCTET STRING }

TBSCertTree ::= SEQUENCE {
version Version OPTIONAL,
signature AlgorithmIdentifier,
hash AlgorithmIdentifier,
issuer Name,
thisUpdate Time,
nextUpdate Time OPTIONAL,
revokedCertificates SEQUENCE OF SEQUENCE {
userCertificate CertificateSerialNumber,
revocationDate Time,
hashValue OCTET STRING,
target Boolean OPTIONAL
} OPTIONAL
}
```

**Fig. 11** Red-black tree syntax verifying a target certificate has been revoked.

**Table 3** Specifications of Implementation

| | |
|---|---|
| Platform | Sun Ultra S-7/300U 167 MHz Solaris 2.5 |
| Programming Language | JAVA Development Kit Ver.1.1 |
| Hash Function | MD5 (coded in JAVA) |
| Communication Protocol | proprietary protocol |

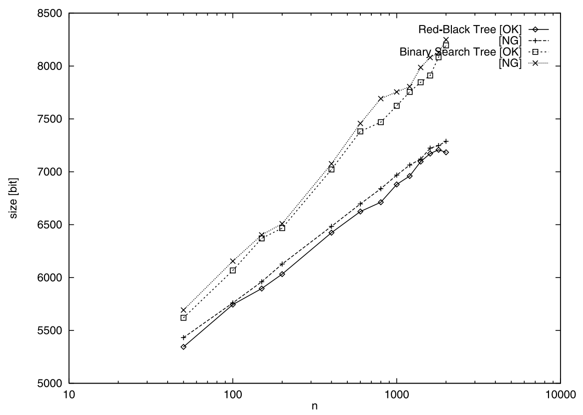

**Fig. 12** Communication cost reduction in red-black tree compared with binary search tree (unbalanced). Labels ‘OK’ and ‘NG (No Good)’ indicate cases where a certificate to be examined is revoked (exists in tree), or not revoked, respectively.

tree, with respect to the number of revoked certificates, $n$. The cases where the target certificate has been revoked, and not revoked, are labeled as “OK”, and “NG (No Good)”, respectively. Since communication costs depend on the height of a target certificate, we took the average size of a subtree of 100 samples chosen in a uniform probability distribution in a certain interval of serial numbers. The revocation tree was randomly generated.

The experimental results demonstrate that the red-black tree reduced the communication costs for online certificate status response services by up to $\mathcal{O}(\log n)$ time, which shows linear behavior on a log scale. The log cost of the red-black tree is scalable to the size of PKI, but is slightly higher than that of the OCSP response, whose size is constant.

Although the performance results are from a trial implementation, which omits additional components such as the OCSP header and the object identifiers, the overall performance with a full implementation is expected to be the same.

From observation of the results, the difference in whether the target certificate is revoked or not is negligible and is smaller than that between the red-black tree and the unbalanced binary search tree, which will increase as a fraction of communication time as $n$ increases.

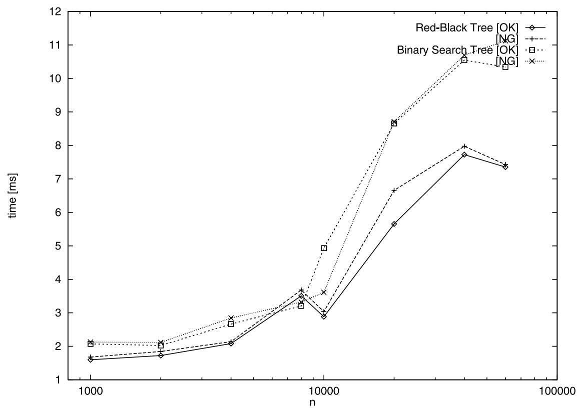

**Fig. 13** Cost reduction for searching in red-black tree. Execution time for looking up a certificate is supposed to be $\mathcal{O}(\log n)$. Side effect of garbage collection is included in figure.

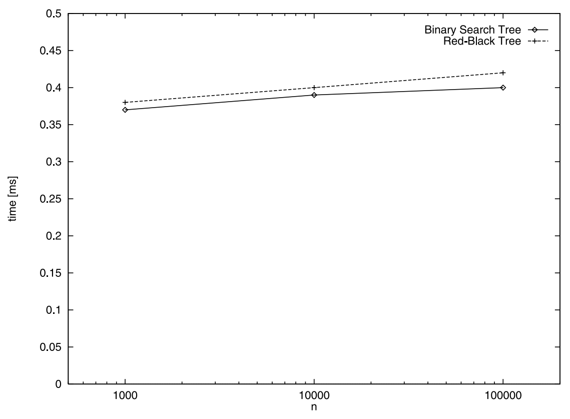

**Fig. 14** Overhead for balancing during insertion of a newly revoked certificate into red-black tree. No significant difference can be observed.

### 4.5 Computation Cost

The balancing in the red-black tree reduces the computation overhead in subtree verification. On the other hand, it increases the number of computations, such as insertion and deletion, needed to update the tree. We show the average execution time for search in **Fig. 13**, and insertion in **Fig. 14**, in order to estimate the behaviors of the computation overheads of the two systems. Measurement was based on the OS clock, and average time over 1,000 trials.

The reduction of computation cost in a red-black tree was at most 3.7 ms at $n = 60{,}000$, which is not nearly as large a reduction as in communication cost. We presume that the garbage collection invoked by the Java virtual machine is one of the reasons why computation does not run in a standard time. Likewise, the overhead involved in balancing is not as significant to the overall performance as shown in Fig. 14.

## 4.6 Estimation Based on Actual Revocation Data

In the actual PKI environment, certificate revocation processes are not likely to be even and may even be skew for certain reasons, such as bulk revocation or independence of revocation. Using the actual CRL data in **Table 4**, we estimate the effect on balancing the tree. Since revoked certificates are sorted in the order of the serial numbers in the CRLs, we placed them in order of revocation date, and used the sorted revocation data as the input for the developed revocation system.

**Figure 15** shows the average heights of trees given the revocation data extracted from the CRL in Table 4. The heights are computed as the average height for all nodes. Both the heights of the binary search tree and the red-black tree increase as $\mathcal{O}(\log n)$ with time. The red-black tree is always shorter than the binary search tree, and approaches the optimum height of a completely balanced tree. The maximum reduction at $n = 20{,}000$ was 0.75, which implies a reduction in computation cost for insert, delete, and search.

**Table 4** CRL Data sheet

| | |
|---|---|
| issuer | VeriSign, Inc. |
| CRL Name | RSASecureServer.crl |
| duration | 1997.02.14–1999.09.27 |
| # of revoked certificates | 20336 |
| size of serial numbers | 128 [bit] |

## 4.7 Comparison with Other Schemes

In **Table 5**, we summarize our proposed scheme along with other revocation schemes proposed so far in terms of computational cost at and between directory and end users. In the table, the columns labeled “hash” and “sign” show the number of performances of hash function and signing operations, respectively, required by the directory to respond to queries regarding certificate status. The column labeled “bandwidth” indicates communication costs between the directory and users, where $n$ is the number of certificates.

Note that the given cost of the CRT is worst case and would be $\log n$ on average. According to the statistical estimation in Section 4.6, the proposed scheme makes the reduction of 75% in bandwidth against the CRT (binary search tree). The constant bandwidth $\mathcal{O}(1)$ in the OCSP and the SEM is provided under the as-

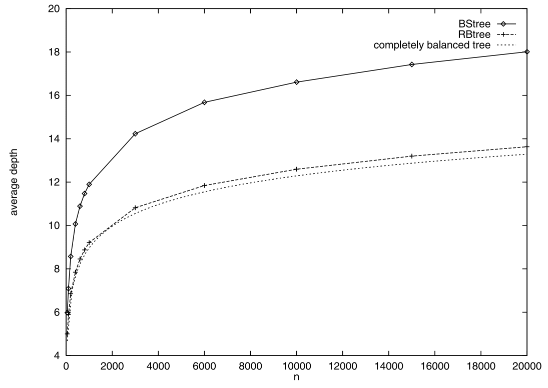

**Fig. 15** Average height of a red-black tree generated by actual revocation data. Average height of a red-black tree approaches optimal height of a completely balanced tree.

**Table 5** Costs in Certification Revocation Schemes.

| schemes | hash | sign | bandwidth | technlogies |
|---|---|---|---|---|
| CRL [3] | N/A | N/A | $\mathcal{O}(n)$ | |
| OCSP [4] | 1 | 1 | $\mathcal{O}(1)$ | signed single message |
| CRS [6] | 1 | N/A | $\mathcal{O}(1)$ | hash chain |
| CRT [5] | $\mathcal{O}(n)$ | N/A | $\mathcal{O}(n)$ | binary search tree |
| 2-3 Tree [13] | $\mathcal{O}(\log n)$ | N/A | $\mathcal{O}(\log n)$ | 2-3 Tree |
| EFECT [8] | $\mathcal{O}(\log n)$ | N/A | $\mathcal{O}(\log n)$ | B-Tree |
| SEM [10] | 1 | 1 | $\mathcal{O}(1)$ | Threshold RSA |
| Accumulator [9] | N/A | $\mathcal{O}(n)$ | $\mathcal{O}(1)$ | RSA Accumulator |
| proposed | $\mathcal{O}(\log n)$ | N/A | $\mathcal{O}(\log n)$ | red-black tree |

sumption that the directory server is trustworthy. Trust of the directory is not necessary for any other scheme. Among tree based approaches, such as the 2-3 tree or the B-tree, the red-black tree is the most efficient in terms of the optimal degree mentioned in Section 3.3.

## 5. Conclusion

We have proposed a scalable secure certificate status server using a red-black tree as an internal revoked certificate database. Our implementation, based on Java, ensures that the balancing based on the red-black tree properties reduces both communication and computation costs of the directory, i.e., the certificate status server in comparison with the ordinary binary search tree. If our proposed method were used in an existing public key infrastructure with $n = 20{,}000$, the communication between the directory and the end users would be 0.75 times that of the binary search tree, according to actual revocation data.

## References

1) Cormen, T., Leiserson, C. and Rivest, R.: *Introduction to algorithms*, MIT Press (1990).
2) ITU-T Recommendation X.509—ISO/IEC 9594-8 (1995).
3) Housley, R., Ford, W., Polk, W. and Solo, D.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, Internet RFC 2459 (1999).
4) Myers, M., Ankney, R., Malpani, A., Galperin, S. and Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol — *OCSP*, Internet RFC 2560 (1999).
5) Kocher, P.: A Quick Introduction to Certificate Revocation Trees (CRTs). http://www.valicert.com/company/crt.html
6) Micali, S.: Efficient Certificate Revocation, Technical Report, MIT-LCS-TM-542 (1995).
7) Micali, S.: NOVOMODO: Scalable Certificate Validation and Simplified PKI Management, *1st Annual PKI Research Workshop — Proceedings* (2002).
8) Gassko, I., Gemmell, P.S. and MacKenzie, P.: Efficient and fresh certification, *Proc. Public Key Cryptography*, pp.342–353 (2000).
9) Faldella, E. and Prandini, M.: A Novel Approach to On-Line Status Authentication of Public-Key Certificates, *16th Annual Computer Security Applications Conference*, IEEE (2000).
10) Boneh, D., Ding, X., Tsudik, G. and Wong, C.M.: A Method for ast Revocation of Public Key Certificates and Security Capabilities, *The 10th USENIX Security Symposium* (2001).
11) Kocher, P.: On Certificate Revocation and Validation, *Proc. Financial Cryptography'98, Springer LNCS 1465*, pp.172–177 (1998).
12) Rivest, R.: Can We Eliminate Certificate Revocation Lists?, *Proc. Financial Cryptography'98, Springer LNCS 1465*, pp.178–183 (1998).
13) Naor, M. and Nissim, K.: Certificate Revocation and Certificate Update, *Proc. Seventh USENIX Security Symposium*, pp.217–228 (1998).
14) Zheng, P.: Tradeoffs in Certificate Revocation Schemes, *ACM SIGCOMM Computer Communication Review*, Vol.33, Issue 2, pp.103–112 (2003).
15) Adams, C. and Farrell, S.: Internet X.509 Public Key Infrastructure Certificate Management Protocols, Internet RFC 2510 (1999).
16) Boeyen, S., Howes, T. and Richard, P.: Internet X.509 Public Key Infrastructure Operational Protocols — LDAPv2, Internet RFC 2559 (1999).
17) Housley, R. and Hoffman, P.: Internet X.509 Public Key Infrastructure Operational Protocols, Internet RFC 2585 (1999).
18) Boeyen, S., Howes, T. and Richard, P.: Internet X.509 Public Key Infrastructure LDAPv2 Schema, Internet RFC 2587 (1999).
19) Kikuchi, H., Abe, K. and Nakanishi, S.: Performance Evaluation of Certificate Revocation Using k-ary Hash Tree, *International Information Security Workshop* (*ISW'99*), *Springer Lecture Notes in Computer Science 1729*, pp.103–117 (1999).
20) VeriSign. http://www.verisign.com
21) Rivest, R.: "The MD5 message-digest algorithm", Internet RFC 1321 (1992).
22) U.S. National Institute of Standards and Technology: Secure Hash Standard, *Federal Information Processing Standards Publication* 180 (1993).
23) Lafore, R.: *Data Structure & Algorithms in Java*, The Waite Group (1998).
24) Wirth, N.: *Algorithms and Data Structures* (1986).

(Received November 30, 2005)
(Accepted June 1, 2006)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.513–523.)

**Hiroaki Kikuchi** was born in Japan. He received B.E., M.E., and Ph.D. degrees from Meiji University in 1988, 1990, and 1994. After working for Fujitsu Laboratories Ltd. from 1990 through 1993, he joined Tokai University in 1994. He is currently a professor in the Department of Information Media Technology, School of Information Science and Technology, Tokai University. He was a visiting researcher at the School of Computer Science, Carnegie Mellon University in 1997. His main research interests are fuzzy logic, cryptographical protocol, and network security. He is a member of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE), the Information Processing Society of Japan (IPSJ), the Japan Society for Fuzzy Theory and Systems (SOFT), IEEE and ACM.

**Kensuke Abe** received B.E and M.E degrees from Tokai University in 1998 and 2000. He jointed Matsushita Communication Industrial Co., Ltd. in 2000.

**Shohachiro Nakanishi** received B.E., M.E., and Dr.E. degrees from Tokai University. He joined Tokai University in 1994. Currently, he is a professor in the Department of Electrical Engineering, Faculty of Engineering, Tokai University. His main research interests are fuzzy systems, neural networks, genetic algorithms, and expert systems. Prof. Nakanishi is a member of the Japan Society of Fuzzy Theory and Systems, SICE, IEEE, IFSA, and NAFIPS.