# Public-Key Certificate Verification Method by Tree-List Structure

Sin Chol-Nam, Ri Kuk-Jin, Cha Hung-Dok *

Faculty of Information Science, Kim Il Sung University, Pyongyang,
Democratic People's Republic of Korea

## Abstract

*Certificate verification and revocation are important aspects of public-key infrastructures (PKI). This paper presents the Tree-List Certificate Verification (TLCV) scheme, which uses a novel tree-list structure to provide efficient digital certificate verification. Under this scheme, users in a public-key infrastructure are partitioned into clusters and a separate blacklist of revoked certificates is maintained for each cluster. The verification proof for each cluster's blacklist comes in the form of a hash path and a digital signature. An algorithm to derive an optimal number of clusters that minimizes the TLCV response size was described. The characteristics of TLCV were examined. Simulations were carried out to compare TLCV against a few other schemes and the performance metrics that were examined include computational overhead, network bandwidth, overall user delay and storage overhead. In general, we find that TLCV performs relatively well against the other schemes in most aspects.*

***Keywords:*** *public-key certificate, certificate verification, certificate revocation, tree-list structure, digital signature*

## 1. Introduction

The digital certificate that guarantees a user's identity by using public key cryptography scheme needs to be revoked when the private key has been compromised or due to other reasons which could cause a breach in security [11]. Before using a public-key of another user to verify a digital signature, the digital certificate associated with the key must be validated to ensure that it has not been revoked [19]. The same applies when one wishes to use a public-key of another user to encrypt a document [15]. In order for certificate verification and revocation to achieve its purpose, the mechanism used must provide timely information to users and be scalable for deployment [25]. In this paper, we introduce the Tree-List Certificate Verification (TLCV) scheme, which uses a hybrid tree-list structure that is efficient, practical and straightforward to implement.

One of the very first mechanisms proposed for certificate verification relied on the use of a blacklist that contains serial numbers of revoked certificates [8]. Such a blacklist is referred to as the certificate revocation list (CRL) [12]. Most, if not all, internet browsers support the downloading of a CRL for certificate verification [23]. One major drawback of using CRLs is that when the number of revocations is large, a large amount of response data would have to be downloaded [1]. In addition, the network bandwidth that the distribution server needs to support is also affected [7]. To reduce the network bandwidth required for certificate verification, the Internet Engineering Taskforce (IETF) came up with the Online Certificate Status Protocol (OCSP) [21], in which a server known as an OCSP responder issues a signed response for each certificate verification request. While OCSP provides small responses and has low network bandwidth requirements, the need to compute a digital signature for every response makes it non-scalable when the number of requests is large. Variants of CRL have also been introduced for better performance. Examples include the segmented CRL, sliding window delta CRL [4], and more recently, the augmented CRL [14].

In [12], Kocher took a different approach by proposing a tree-based method known as the certificate revocation tree (CRT) that provides reasonably small responses for certificate verification. However, when changes to certificate status occur, the entire tree needs to be reconstructed, incurring a substantial amount of hash computations [20]. In [22], Naor introduced the authenticated dictionary (AD) that uses a 2-3 tree in place of a binary tree to provide greater efficiency when handling revocation updates. However, one shortcoming of AD is that it is complicated and not straightforward to implement.

Based on insights gained from these works, we propose the TLCV scheme, which uses a novel tree-list structure to provide better performance for certificate verification without the shortcomings faced by CRT or AD.

* Corresponding author.
E-mail address: Hd.Cha@start-co.net.kp

The paper is organized as follows. We explain the proposed tree-list structure and describe an algorithm to derive an optimal tree height that minimizes the response size for TLCV. We also discuss the benefits and possible drawbacks when implementing TLCV, and compare its performance against other previously proposed schemes with the use of simulations. The simulations were carried out based on a constant-rate certificate arrival model, as well as a real-world model. Various aspects of performance, including computational overhead, network bandwidth, overall user delay and storage overhead, were examined. In general, we find that TLCV performs better than most of the other schemes in most aspects.

## 2. Background

The certificate revocation tree (CRT) is a certificate revocation and verification method that is based on a binary hash tree [9]. The CRT arranges the serial numbers of revoked certificates in an ordered manner and uses a path along the hash tree and a digitally signed root to verify the integrity and authenticity of the certificate verification response [18]. This allows a certificate to be validated with a significantly smaller response as compared to CRL. However, one problem of this solution is that the entire CRT needs to be reconstructed when changes to certificate status occur, i.e. when there are new revocations that have to be inserted or when expired revocations need to be removed [16].

In [22], Naor introduced an Authenticated Dictionary (AD) as an improvement over the CRT. The proposed AD is based on a 2-3 tree instead of a binary tree. Using a 2-3 tree allows updates to be performed without the need to reconstruct the entire tree. The insertion or deletion of a leaf node due to a change in certificate status will only involve a re-computation of the hash path for the associated leaf node. However, balancing a 2-3 tree is a complicated affair and putting the AD into implementation is not straightforward [13].

In this paper, we introduce a practical and efficient tree-list structure that only requires recomputing hash paths for changed leaf nodes and does not involve complicated rebalancing of the tree. The number of hash paths that need to be recomputed is at most that required for an AD and the size of a hash path is guaranteed to be lesser than that of a CRT.

Other related works that have been carried out include the hash-based NOVOMODO [17], the QuasiModo Tree [5] and the Huffman Merkle Hash Tree [24]. From the previous works, we find that since the introduction of the CRT, there has been no significant improvement to tree-based certificate verification approaches without much implementation complications, such as that exhibited by AD [2, 3].

In this paper, we introduce the TLCV scheme, which uses a hybrid tree-list structure that is efficient, practical and straightforward to implement. We conduct experiments to compare the performance of TLCV against CRT, which TLCV is based on, and AD, which is an efficient improvement over CRT. In addition, we also compare TLCV against those schemes with known practical implementations – namely OCSP, ACRL (the most efficient CRL variant known to date), and NOVOMODO. Here, we note that OCSP differs from the other schemes in that it is essentially an online mechanism that provides real-time certificate verification, while the rest are based on periodic updates to certificate status or revocation information. Nonetheless, we contend that it would be necessary to investigate how TLCV fares against OCSP, especially since OCSP is an openly adopted certificate verification mechanism for the public internet.

## 3. The Proposed Scheme

### 3.1. Description of Scheme

The proposed Tree-List Certificate Verification (TLCV) scheme is based on partitioning users in a PKI into clusters. Under TLCV, the public-key certificate issued for each keypair would include a cluster number in addition to the serial number. The certificate authority (or revocation authority) maintains a separate blacklist of revoked certificates for each cluster. Similar to CRT and AD, a hash path, together with the cryptographically signed root of a hash tree, is used to provide integrity protection and origin authentication for each cluster's black list [6]. The blacklist for each cluster is first hashed using a secure one-way hash function (e.g. SHA-1 or SHA-256) to produce the leaf node of a hash tree, as shown in Fig. 1. The rest of the tree is produced the same way as in a CRT, i.e. the value of each non-leaf node is computed by hashing a concatenation of the values of its children [10]. The root of the hash tree is then signed (e.g. using RSA) together with a timestamp and other relevant information that is to be included in the verification response. The resulting digital signature and the hash path provided can then be used to verify whether the contents of the response have been received correctly and whether the response originated from the trusted certification authority.

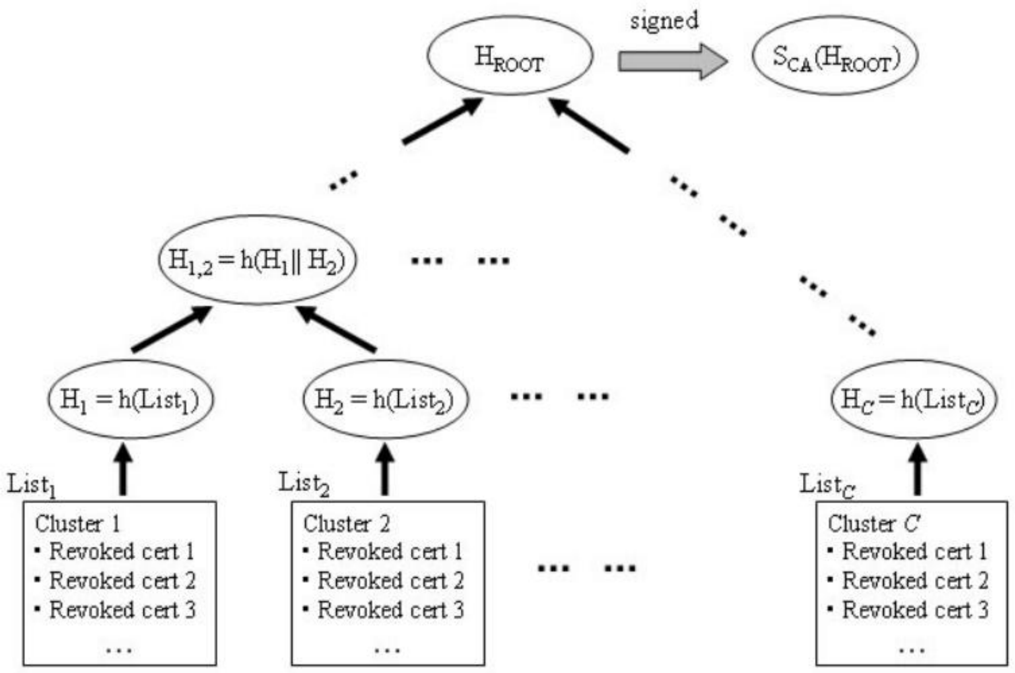

Fig. 1. The proposed tree-list structure under the TLCV scheme

| |
|---|
| • **Header**: Issuer Name (32bytes)<br>Cluster Number (3)<br>Last Update (6)<br>Next Update (6) |
| • **Data**: Revoked Entries (10bytes/entry)<br>- Serial Number (3)<br>- Revocation Date (6)<br>- Reason Code (1) |
| • **Proof**: Siblings of Hash Path (20bytes/node)<br>- SHA-1 Hash (20)<br>Signed Root (RSA) (128) |

Fig. 2. Format of a verification response under the TLCV scheme

With this scheme, each public-key certificate would now contain an additional field for the cluster number. During certificate verification, a user sends a request containing the cluster number of the certificate in question. The response contains the blacklist for the cluster, the siblings of the hash path associated with the blacklist, and the signed root (see Fig. 2).

Under TLCV, the size of a response depends on the number of revoked certificates in a cluster and the size of the hash path. Let $N_r$ denote the total number of revoked certificates in the PKI population and $C$ denote the number of clusters (note that this is equal to the number of leaf nodes in the tree – in the case of a binary tree, $C = 2^h$ for some positive integer $h$). Assuming that the revoked certificates are uniformly distributed across the clusters, then the average number of revoked certificates in each cluster would be

$$N_c = \frac{N_r}{C} \tag{1}$$

The size of each hash path in the tree is given by $h = \log_2 C$. Hence, the size of a response under the TLCV scheme effectively depends on $C$(or $h$) and $N_r$. In a study performed in [12], the authors showed that $N_r$ reaches a constant value after a certain duration of time, as new revocations are balanced by revocations that expire. Furthermore, we note that in most PKI deployments, the PKI population is known and is more or less fixed. If the number of revocations at steady state is known, or if this can be predicted or estimated given the PKI population size, then we can use the information to determine a suitable value for $C$(or $h$) such that the size of the response is minimized. Henceforth, we shall assume that $N_r$ refers to the number of revoked certificates at steady state.

## 3.2. Finding the Optimal Number of Clusters

Based on the response format shown in Fig. 2 and assuming that the revoked certificates are uniformly distributed across clusters, the size of a verification response under TLCV is

$$S_{resp} = S_{header} + S_{entry}\left(\frac{N_r}{C}\right) + S_{hash}(\log_2 C) + S_{sign} \tag{2}$$

When expressed in terms of the height of the tree, we have

$$S_{resp} = S_{header} + S_{entry}\left(\frac{N_r}{2^h}\right) + S_{hash}(h) + S_{sign} \tag{3}$$

Our aim is to obtain the value of $C$ or $h$ that minimizes $S_{resp}$. Using (2), we solve for the value of $C$ that satisfies

$$min\left\{S_{header} + S_{entry}\left(\frac{N_r}{C}\right) + S_{hash}(\log_2 C) + S_{sign}\right\} \tag{4}$$

and obtain the optimal number of clusters and tree height as

$$C^* = \frac{S_{entry}N_r \log_e 2}{S_{hash}} \tag{5}$$

$$h^* = \log_2\left(\frac{S_{entry}N_r \log_e 2}{S_{hash}}\right) \tag{6}$$

However, the $h^*$ derived from (6) may not necessarily be a positive integer. To ensure that this condition is met, we replace (5) and (6) with the following:

$$C^* = 2^{\left\lceil \log_2\left(\frac{S_{entry}N_r \log_e 2}{S_{hash}}\right)\right\rfloor} \tag{7}$$

$$h^* = \left\lceil \log_2\left(\frac{S_{entry}N_r \log_e 2}{S_{hash}}\right)\right\rfloor \tag{8}$$

In this case, (7) and (8) give approximations to the optimal number of clusters and tree height. We now describe a simple algorithm that allows us to obtain the precise optimal number of clusters that minimizes the response size:

1. Based on the known (or estimated) number of revoked certificates at steady state, compute $h^*$ using (8).
2. Compute $S_{resp}$ by substituting $h = h^*$ into (3) to obtain $s$. Repeat with $h = h^* - 1$ and $h = h^* + 1$ to obtain $s^-$ and $s^+$ respectively.
3. (a) If $s^- \le s \le s^+$, then $h^*$ is the optimal tree height. Output $C = 2^{h^*}$.
   (b) Otherwise, obtain $s' = min(s^-, s^+)$. The corresponding value of $h$, say $h'$, gives the optimal tree height. (This can be verified by checking that the values of $S_{resp}$ obtained from $h' - 1$ and $h' + 1$ are greater than or equal to that obtained from $h'$.) Output $C = 2^{h'}$.

Using this algorithm, we can construct a tree-list structure that guarantees a low response size. Fig. 3 shows the graph of response size against $N_r$ for TLCV compared with CRT and AD (in the average case and for a full ternary tree).

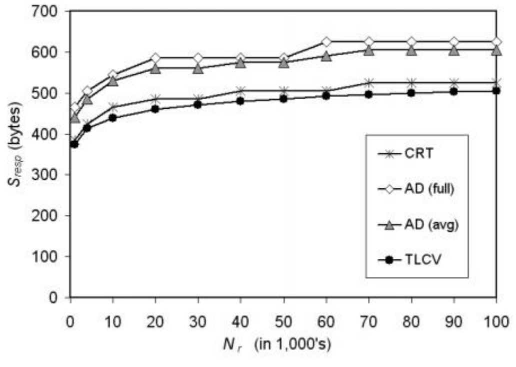

Fig. 3. Graph of response size $S_{resp}$ against number of revoked certificates $N_r$ for the various tree-based certificate verification schemes

The responses for CRT and AD are assumed to be of the format shown in Fig. 4. From the graph, we find that TLCV generally provides smaller responses than the other tree-based schemes.

| | |
|---|---|
| • Header: | Issuer Name (32bytess) |
| | Last Update (6) |
| | Next Update (6) |
| • Data: | 2-tuple $(S_1,\ S_2)$ (6) |
| | Revocation Date for $S_2$ (6) |
| | Reason Code (1) |
| • Proof: | Siblings of Hash Path (20bytes/node) |
| | - SHA-1 Hash (20) |
| | Signed Root (RSA) (128) |

Fig. 4. Format of a verification response under CRT and AD

## 3.3. The Characteristics of TLCV

In this section, we discuss the benefits and possible drawbacks of TLCV. The benefits of TLCV can be listed as follows:

– Just like CRT and AD, TLCV only requires the computation of a single digital signature during each update period. Meanwhile additional hash operations need to be performed, the time taken for them is only a small fraction of that required for a signature. This is an advantage over on-demand schemes like OCSP, whereby a signature needs to be computed for every response, causing scalability problems when the number of requests get large.

– Changes to revocation entries only affect the path from the affected leaf node (corresponding to the cluster the revoked certificate is in) to the root node. In cases that multiple revoked certificates belong to the same cluster, only a single path needs to be updated for those revocations. Moreover, at the optimal number of clusters, the height of a TLCV is lower than that for a CRT or AD, i.e. the hash path for a TLCV is shorter than that for CRT and AD. Hence, during each update, the number of hash computations required for a TLCV is lesser than that required for a CRT or an AD.

– Unlike AD, there is no need to rebalance the tree for TLCV. The structure of the tree remains the same as only the contents of the leaf nodes change. The updating of a TLCV is straightforward and easier to implement than an AD based on a 2-3 tree.

– As witnessed in the previous section, the response size of TLCV is smaller than that of CRT or AD. Consequently, the network bandwidth required to support the download of verification responses under TLCV is expected to be smaller than that under CRT or AD.

On the other hand, some possible drawbacks to TLCV are as follows:

– There is a need to have an a priori knowledge of the PKI population size and revocation rate in order to determine the optimal number of clusters. While the PKI population size can be pre-determined in many cases, information on the revocation rate may not be readily available. Under such circumstances, an estimate for the revocation rate would have to be used.

– If the size of the PKI population changes drastically or the actual number of revoked certificates at steady state differs greatly from what was expected, the number of clusters may not be optimal anymore. This could possibly affect the performance of TLCV.

– When deriving the optimal TLCV, it is assumed that the number of revoked certificates is uniform across different clusters. However, in a real-life scenario, clusters may have unequal numbers of revoked certificates. Thus, a client could end up downloading a larger response if the certificate to be validated belongs to a cluster that has a larger number of revoked certificates.

In general, the first two drawbacks would be more significant in a highly dynamic environment, where users join or leave the PKI in an unpredictable fashion and the number of users in the PKI varies drastically with time. In a fixed organization whereby the number of users is more or less stable, those drawbacks would have a lesser effect TLCV. With regards to the third drawback, we note that if the certificate to be validated is equally likely to be from any cluster, then the size of data downloaded will be averaged out due to clusters that have smaller number of revoked certificates (i.e. smaller responses). However, if the certificate to be validated is more likely to belong to a cluster containing a larger number of revoked certificates, the performance of TLCV would suffer.

## 3.4. Effects of Deviations to Expected $N_r$

In this section, we investigate how the TLCV scheme would be affected if the number of revoked certificates happens to differ from its expected value. We first investigate how changes in the total number of revoked certificates could affect the performance of TLCV by examining the optimal TLCV tree height $h^*$ for various ranges of $N_r$. From Table 1, we see that if $h^* = h$ is the optimal tree height for $N_r = n$, then the optimal tree height for $N_r = 2n$ would be $h^* = h + 1$. In other words, if the actual value of $N_r$ is double of what was expected, then the height of the TLCV tree would need to be increased by one in order to make it be optimal. (This is characteristic of a binary tree, where the number of leaf nodes in the tree doubles each time the height is increased by one). This suggests that on the average, TLCV would only deviate from optimality if the number of revoked certificates at steady state is double or more than double of what was initially expected. Table 2 shows the response sizes for the various tree-based schemes, including the cases of TLCV where the tree height is not optimal (represented by TLCV#2, TLCV#5 and TLCV#10). TLCV#2 represents the case where the TLCV was designed to provide optimal response size for $\frac{1}{2}N_r$, i.e. the tree height is optimal for $\frac{1}{2}N_r$(but not $N_r$). TLCV#5 and TLCV#10 represent the cases where the TLCV was designed to provide optimal response size for $\frac{1}{5}N_r$ and $\frac{1}{10}N_r$ respectively.

Table 1. Ranges of $N_r$ and the corresponding $h^*$ under optimal TLCV

| $N_r$ | $h^*$ |
|---|---|
| 1 024 - 2 047 | 9 |
| 2 048 - 4 095 | 10 |
| 4 096 - 8 191 | 11 |
| 8 192 - 16 383 | 12 |
| 16 384 - 32 767 | 13 |
| 32 768 - 65 535 | 14 |
| 65 536 - 131 071 | 15 |

Table 2. Comparison of response sizes (in bytes) for non-optimal TLCV

| $N_r$ | $S_{resp}$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | TLCV | Non-optimal TLCV | | | CRT | AD | |
| | | #2 | #5 | #10 | | (avg) | (full) |
| 1, 000 | 374 | 393 | 451 | 588 | 385 | 440 | 465 |
| 5, 000 | 419 | 424 | 530 | 706 | 445 | 500 | 505 |
| 10, 000 | 439 | 444 | 550 | 726 | 465 | 530 | 545 |
| 50, 000 | 486 | 496 | 537 | 639 | 505 | 575 | 585 |
| 100, 000 | 506 | 516 | 557 | 659 | 525 | 605 | 625 |

The results under these columns correspond to the situation that the actual number of revoked certificates at steady-state is twice, five times and ten times (respectively) bigger than what was initially expected for the PKI. From the table, we find that TLCV#2 performs reasonably well, with slightly larger response sizes than CRT but smaller than those achieved under AD. TLCV#5 has slightly larger response sizes than AD when $N_r$ is below 50, 000 but smaller response sizes for $N_r = 50,000$ and $N_r = 100,000$. TLCV#10 gives responses that are around 25-50% larger than the responses obtained under CRT and 5-30% larger than the responses obtained under AD. This shows that the performance of TLCV would only deteriorate to levels below that of AD if the actual number of revoked certificates at steady state is drastically larger (over five times more) than what was initially expected. In summary, we find that TLCV can still perform at an optimal level if the number of revoked certificates deviate slightly from its expected value. Even when the deviation is large enough to cause the TLCV to become non-optimal, performance gains over the other schemes can still be achieved unless the deviation is drastically larger (over five times more) than the expected value.

## 3.5. Adapting the TLCV Structure when the Expected Value is Deviated

To improve the performance of TLCV in a highly dynamic environment that causes $N_r$ to deviate drastically from its expected value, adaptive changes to the tree-list structure can be made during the deployment of TLCV. The tree-list structure can be easily adapted to regain optimality in situations when the actual value for $N_r$ is drastically different from its expected value. When $N_r$ is overestimated, the resulting number of clusters would be greater than optimal. In order to compensate for this, we can combine the cluster lists to obtain a reduced tree, as shown in Fig. 5.

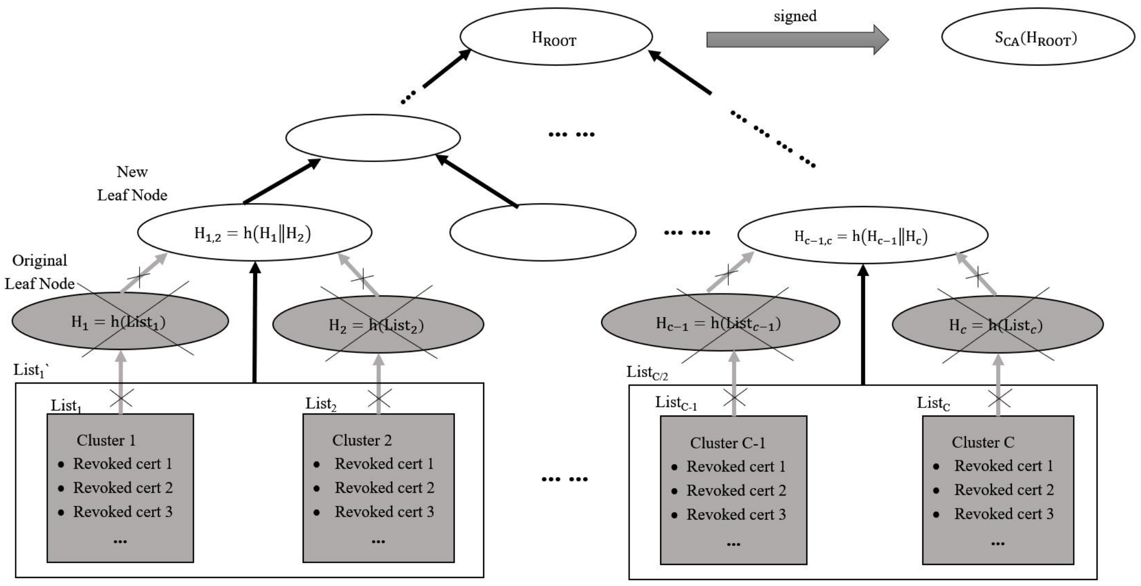

Fig. 5. Adapting the TLCV structure when $N_r$ is overestimated

The lists for two adjacent clusters are combined into a single new list and the leaf node (corresponding to the new list) of the resulting hash tree would be the parent of the original leaf nodes corresponding to the original lists. The new hash tree would simply be the original tree with its leaf nodes removed and the tree height reduced by one. Hence, the hash nodes are reused and there is no need to make any new hash computations. However, the new verification response would need to include an additional cluster number since both cluster numbers are required to identify the separate clusters within the new list. Based on Fig. 2, this would only incur an additional 3 bytes in the response.

On the other hand, if $N_r$ is underestimated, there will be a need to increase the number of clusters and expand the tree, as shown in Fig. 6. In this case, the number of clusters is doubled and new leaf nodes are added for the revocation lists corresponding to the new clusters. New users joining the PKI would then be assigned to those new clusters when they apply for their digital certificates. The existing nodes from the original hash tree will be reused and need not be recomputed. The original root node now becomes the child of the new root and the height of the hash tree is increased by one. Hash values would have to be computed for the intermediate nodes that lie along the new paths between the new leaf nodes and the new root. We note that this adaptation is useful for PKIs where new membership can be expected and possibly result in an increase in the size of the PKI population. In PKIs whereby the users are fixed, then such an adaptation would not be applicable.

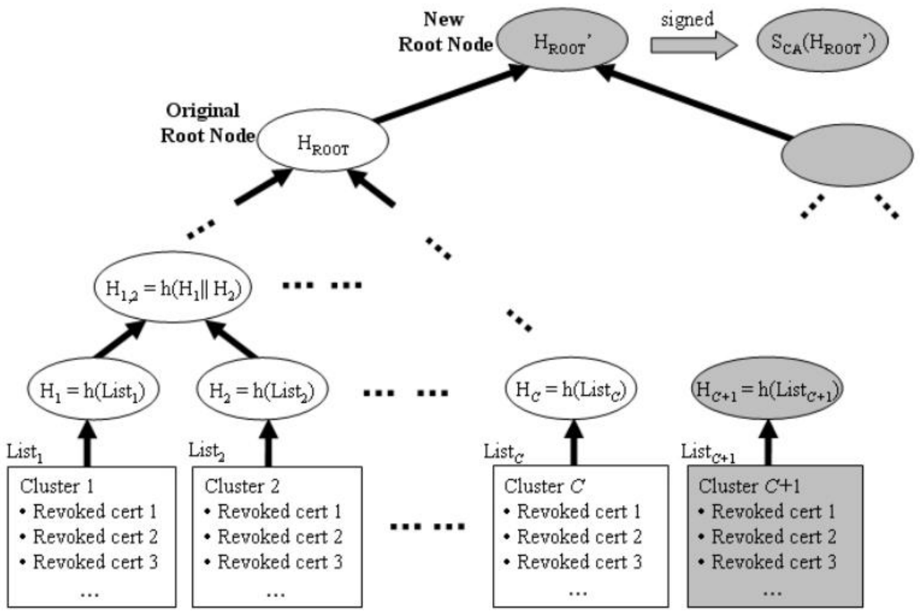

Fig. 6. Adapting the TLCV structure when $N_r$ is underestimate

## 4. Performance Analysis

In this section, we conduct simulations in MATLAB to analyze the performance of TLCV against a few other schemes, namely CRT, AD, OCSP, NOVOMODO and ACRL. We note that OCSP differs from the other schemes in that it is an online mechanism that provides real-time certificate verification, while the rest are based on periodic updates to certificate status or revocation information [26]. While this is the case, we contend that it would be interesting to find out how TLCV fares against OCSP.

### 4.1. PKI Assumptions

We assume a PKI population of $N$ users, whereby each user owns a single asymmetric key pair and a public-key certificate that binds the user to the key pair. Each certificate is valid for a duration $L$, after which it expires and is renewed (replaced by a new certificate) if it has not been revoked. Clients validate certificates (receive certificates that have not expired) at an average rate of $v$ per client. Certificates are revoked at a constant rate $r_r$ and revoked certificates expire at a rate $r_x$. In addition, the following assumptions were made:

- $L$ =365 days
- $v$ =10 certs/day
- $r_r = 0.05\% \times N = 0.0005N$ certs/day= $r_x$

As in [4], we assume further that the total number of certificates that have been revoked and have not yet expired at any one time is given by $N_r = \frac{1}{2} r_r$ , $L = 0.0913N$. If valid certificates expire at a rate $r_v$ and the proportion of valid certificates that expire each day is the same as the proportion of revoked certificates that expire each day, then

$$\frac{r_v}{N} = \frac{r_r}{N_r} \Leftrightarrow r_v = \frac{r_r}{N_r} N = 0.00548N \text{ certs/day}$$

The total number of unexpired certificates in circulation, is $N_{total} = N + N_r = 1.0913N$. Hence, when a client receives a certificate that has not expired, the probability that it is revoked is $P_r = \frac{N_r}{N_{total}} = 0.08$ and the probability that it is valid is $P_v = 1 - P_r = 0.92$.

### 4.2. Certificate Verification System Model

For the underlying certificate verification network, we assume a basic model shown in Fig. 7.

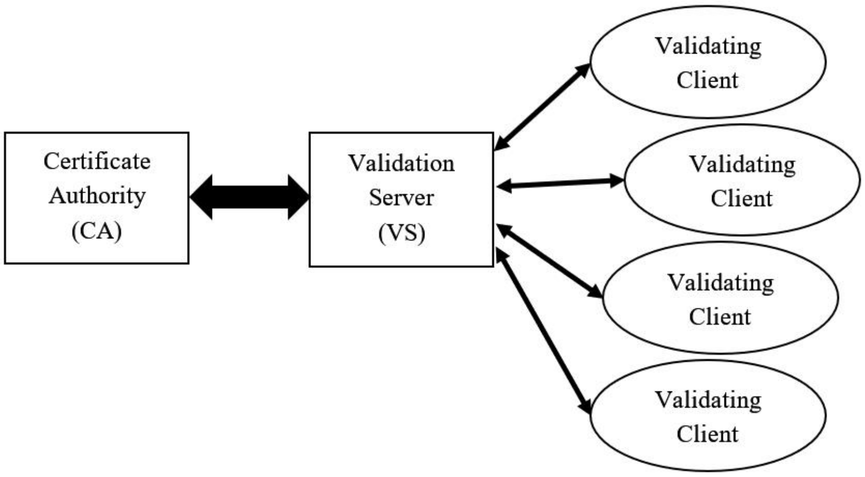

Fig. 7. The network architecture for certificate verification

| OCSP |
|---|
| • Header: Issuer Name (32bytes)<br>Effective Time (6) |
| • Data: Certificate Status (1)<br>Revocation Date (6)<br>Reason Code (1) |
| • Proof: RSA Signature (128) |

| NOVOMODO |
|---|
| • Header: Issuer Name (32bytes)<br>Last Update (6)<br>Next Update (6) |
| • Data: Certificate Status (1)<br>Revocation Date (6)<br>Reason Code (1) |
| • Proof: SHA-1 Hash (20) |

| ACRL |
|---|
| • Head: Same as NOVOMODO (44bytes) |
| • Data: CRL Entries (10byte/entry)<br>- Serial Number (3)<br>- Revocation Date (6)<br>- Reason Code (1)<br>Expired Entries (3bytes/entry)<br>- Serial Number (3) |
| • Proof: RSA Signature (128) |

Fig. 8. Formats of verification responses for OCSP, NOVOMODO and ACRL

For the arrival of digital certificates to clients, we consider two different models. The first model is the constant-rate model adopted by Cooper in [4]. In this model, each client receives certificates for verification at a constant rate $v$ and the inter-arrival time between two successive certificates can be modelled using an exponential distribution with mean $\frac{1}{v}$. The second model is a real-world model that more accurately reflects the actual deployment and operation of an information technology (IT) infrastructure. It is based on the daily activity within an organization, i.e. the load during working hours is expected to be heavy while the load outside working hours would be light or close to zero. To model such a load pattern, we gather statistics on access logs of clients to the server. Statistics on access logs were collected from a web server in a network of approximately 500 computers. The access log for all work days in a single month were consolidated, from which the average access log pattern for a single work day was derived. Fig. 9 shows the resulting graph. This was extrapolated to model a PKI of N users, each validating certificate at an average rate $v$ for each day.

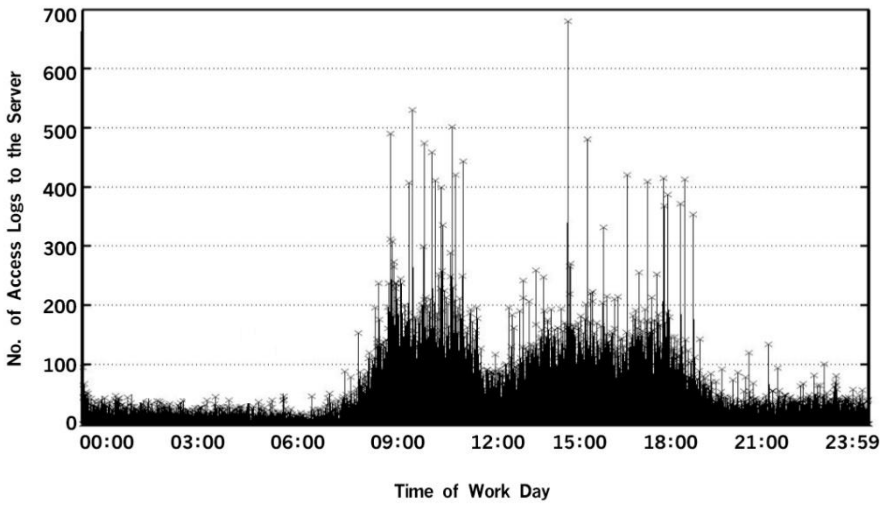

Fig. 9. The access logs pattern to the server (obtained by summing the logs over 1 min intervals) for a network of 500 computers on a single workday

## 4.3. Performance Metrics

In our performance analysis, we make use of some typical metrics that can be used to determine the scalability of each scheme for implementation and deployment in a PKI. The metrics are:

– **Peak computational overhead.** We assume that the main overhead comes from cryptographic computations and measure computational costs in terms of the total time taken for cryptographic operations. The following costs are assumed based on benchmarks obtained in [13]:
  • time to compute an RSA signature, $T_{sign} = 5\text{ms}$
  • time to verify an RSA signature, $T_{verify} = 0.2\text{ms}$
  • time to compute a SHA-1 hash, $T_{hash} = 0.3\mu s$

– **Peak network bandwidth required at VS.** Network bandwidth is obtained by computing the number of bytes transmitted from the VS to clients per unit time.

– **Peak delay experienced by a client.** This is measured by summing the peak computational time, the time taken to transfer the relevant data structures between the CA and the VS, the peak network delay between the VS and a client, and the computational time required at the client.

– **Storage overhead at VS.** This gives the amount of storage memory required at the VS.

## 4.4. Simulation Setup

Simulations were carried out on the optimized TLCV, as well as the non-optimal TLCV (labelled TLCV#), using MATLAB. For TLCV#, we assume that the number of revoked certificates is under-estimated by one-fifth, i.e. the actual Nr is five times bigger than the estimated $N_r$. Other schemes that were also simulated include CRT, AD, ACRL, NOVOMODO and OCSP. For the periodic schemes (schemes other than OCSP), the interval for updating the revocation information at the VS was set to one minute in order to make certificate verification as close to real time as possible without incurring an unreasonable overhead. Separate sets of simulations were conducted under Cooper's constant-rate model, as well as the real-world model, with the average rate of certificate arrival at each client set to $v = 10$ certs/day. The simulations were carried out for various values of $N$ between 100,000 to 1 million.

## 4.5. Results and Discussion

**Peak Computational Overhead.** Fig. 10 shows results for peak computational overhead under Cooper's constant-rate model and the real-world model.

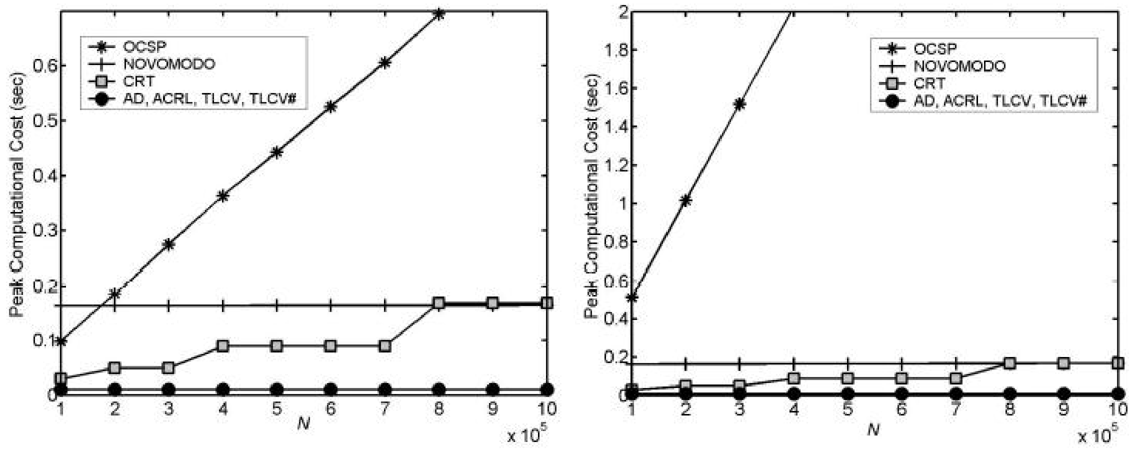

Fig. 10. Graph of peak computational overhead against $N$ under the constant-rate model (left), and the real-world model (right)

TLCV performs well since only one digital signature needs to be computed at every revocation update and the additional hash computations make up an almost negligible fraction of the time taken for computing a digital signature. The same goes for AD. ACRL has similar computational overheads since only one digital signature needs to be computed per update. CRT has higher computational costs than TLCV, TLCV# and AD due to the large number of hash operations required to reconstruct the entire tree. OCSP experiences the highest amount of computations due to the large number of digital signatures that have to be computed for the responses. NOVOMODO also experiences a relatively large computational overhead due to the computation of hash chains for new validity targets when expired certificates are renewed and revoked certificates are replaced.

Table 3 shows the number of hash operations required at the CA (similarly, at the VS) under the tree-based schemes during each update.

Table 3. Number of hash operations per update for tree-based schemes

| $N$ | CRT | AD | TLCV | TLCV# |
|---|---|---|---|---|
| 100, 000 | 32, 767 | 30 | 26 | 20 |
| 300, 000 | 65, 535 | 48 | 42 | 36 |
| 500, 000 | 131, 071 | 51 | 45 | 39 |
| 1, 000, 000 | 262, 143 | 90 | 80 | 70 |

From the table, we see that TLCV and TLCV# require lesser hash operations than both AD and CRT. This is because the height of a TLCV tree is generally smaller than that for CRT or AD, which results in a shorter hash path that needs to be re-computed when a change occurs. We note that the non-optimal TLCV# performs better than optimized TLCV because the optimization was carried out with respect to response size rather than computational overhead. In the following sections, we will see that response size has a greater effect on the performance of both schemes and TLCV has the edge over TLCV# in terms of overall performance.

**Peak Network Bandwidth at VS.** From Fig. 11, we find that TLCV requires lower network bandwidth than CRT and AD. This is due to its smaller response size, as witnessed in Table 2. While TLCV# requires slightly larger network bandwidth than CRT, it is still lesser than AD. Comparing TLCV against ACRL, we find that TLCV performs better for larger $N$ (when $N \geq 700,000$ under Cooper's constant-rate model and $N \geq 600,000$ under the real-world model). This is because for larger user populations, the size of the revocation list grows tremendously, causing the performance of ACRL to deteriorate. OCSP and NOVOMODO require smaller network bandwidths than TLCV due to their smaller response sizes, making them perform better than TLCV in this aspect.

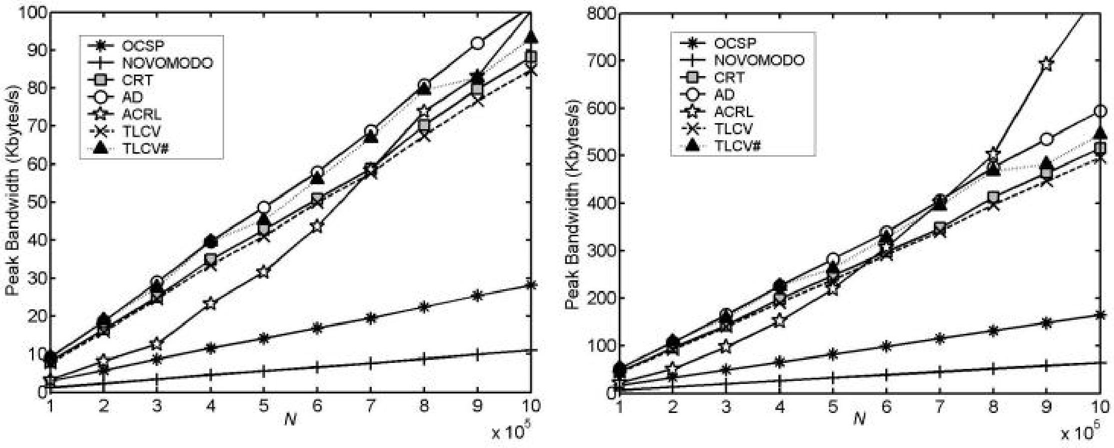

Fig. 11. Graph of peak network bandwidth required at the VS against $N$ under the constant-rate model (left), and the real-world model (right)

**Peak Delay at Client.** Fig. 12 shows the peak delay experienced by a client.

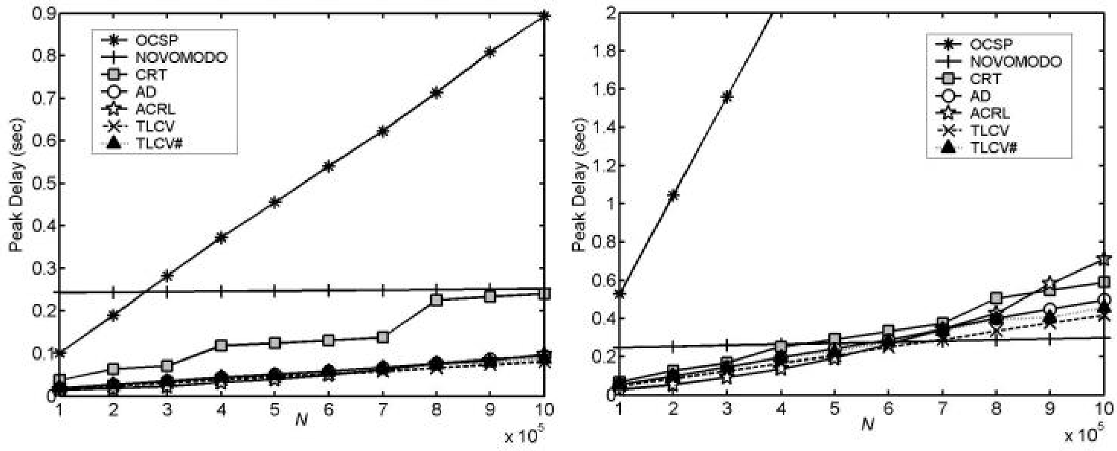

Fig. 12. Graph of peak delay against N under the constant-rate model (left), and the real-world model (right)

Under Cooper's constant-rate model, we find that TLCV performs better than most of the schemes, losing out only to ACRL in the range of $100\,000 \leq N \leq 600\,000$ For the range $N \geq 700,000$, TLCV performs the best. Under the real-world model, TLCV performs better than most of the schemes as well. In the range $100,000 \leq N \leq 600,000$, only ACRL performs better and in the range $700\,000 \leq N \leq 1$ million, only NOVOMODO performs better. These results show that in terms of overall performance (considering the sum of computational delay and network delay), TLCV performs better than most of the schemes. In fact, TLCV performs the best among the tree-based schemes.

**Storage Overhead at VS.** Fig. 13 shows results for storage overhead at the VS. OCSP has negligible storage overhead since it does not store any data at the VS. However, it requires a secure connection with the CA to retrieve revocation information from the CA and secure storage for the secret key that is used to sign the responses. This requires the VS to be trusted and secured against tampering, which implies extra overhead. On the other hand, TLCV, CRT, AD and ACRL can be deployed with distrusted VS since the data structures are integrity-protected by the signature pre-computed by the CA. In general, we find that TLCV incurs lower storage overhead than CRT or AD and slightly larger storage overhead than ACRL. Non-optimal TLCV# incurs the lowest storage overhead among the various schemes but this comes at the expense of larger response size and overall delay.

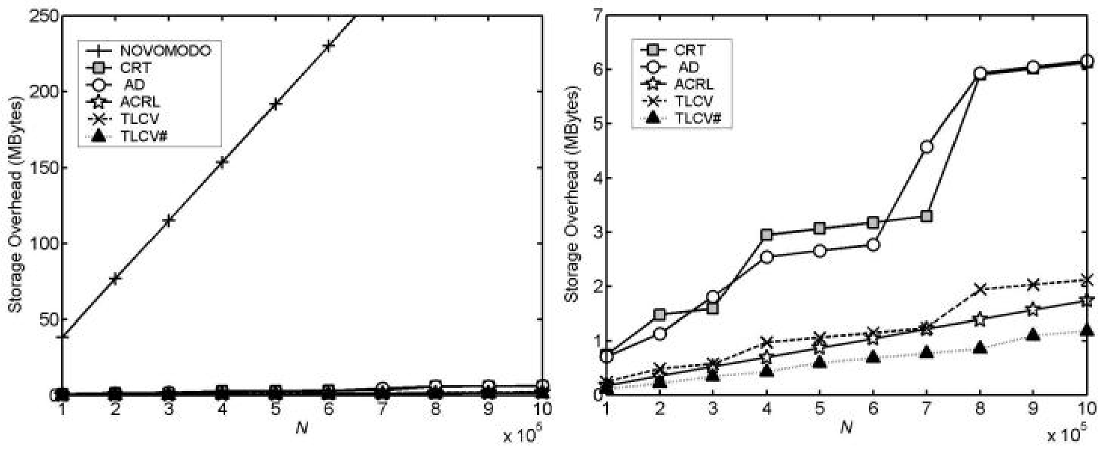

Fig. 13. Graph of storage overhead at VS against N. (The graph on the right is a scaled version of the graph on the left, with results for NOVOMODO excluded.).

**Summary of Results.** In general, we find that TLCV performs well compared against the other schemes. Among the tree-based schemes, TLCV requires lesser computations, network bandwidth, client delays and storage than CRT or AD. Comparing TLCV against OCSP and NOVOMODO, we find that while OCSP and NOVOMODO require smaller network bandwidths to support, the large computational overheads incurred cause them to perform poorly against TLCV. Moreover, both schemes require the VS to be trusted and secured against tampering since cryptographic secrets are kept at the VS. This incurs additional costs to these two schemes. Moreover, NOVOMODO requires an extremely large storage overhead. Comparing TLCV with ACRL, we find that both schemes perform closely in all four areas. In terms of network bandwidth and overall client delay, TLCV performs better for larger N and ACRL performs better for smaller $N$. Considering results for non-optimal TLCV#, we find that while the network bandwidth and overall client delay deteriorates when TLCV deviates from optimality, this deterioration is not very significant even when the actual $N_r$ is five times bigger than what was initially expected. TLCV# is still able to perform better than CRT, and performs close to AD. Moreover, TLCV# has a slight edge over the rest of the schemes in certain aspects, for example computational overhead and storage overhead.

## 5. Conclusion

We presented a novel tree-list structure for efficient certificate verification that partitions users into clusters and maintains a blacklist of revoked certificate for each cluster. We described an algorithm to obtain the optimal number of clusters that gives the minimal response size. We discussed the characteristics of TLCV and performed simulations to compare its performance with a few other certificate verification schemes. From the results, we find that TLCV performs better than most of the other schemes in most aspects. TLCV scheme can be effectively used to verify users' identity rapidly, correctly and safely in a network environment in which the issue, revocation, and update of digital certificates are performed periodically.

## References

[1] J. Blomer, A. May, "Key Revocation with Interval Cover Families", *SAC 2001*, vol. 2259, pp. 325-341, 2001.

[2] HY.Chien, "Dynamic Public Key Certificates for IoT and WSN Scenarios", *Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference*, pp. 646–651, 2018.

[3] HY.Chien, "Dynamic Public Key Certificates with Forward Secrecy", *Electronics*, vol. 2009, no. 10, pp. 1-14, 2021.

[4] D. Cooper, "A More Efficient Use of Delta CRLs", *IEEE Symposium on Security and Privacy 2000*, pp. 190–202, 2000.

[5] F. Elwailly, C. Gentry, Z. Ramzan, "QuasiModo: Efficient Certificate Validation and Revocation", *PKC 2004*, vol. 2947, pp. 375–388, 2004.

[6] M. Goodrich, R. Tamassia, "Efficient Authenticated Dictionaries with Skip Lists and Commutative Hashing", *Technical Report, Johns Hopkins Information Security Institute*, 2000.

[7] T. Hewa et al., "Blockchain-based Automated Certificate Revocation for 5G IoT", *Proceedings of the ICC 2020 IEEE International Conference on Communications*, pp. 1–7, 2020.

[8] G. Hua et al., "Efficient and dynamic key management for multiple identities in identity-based systems", *Information Sciences*, vol. 221, pp. 579–590, 2013.

[9] J. Li et al., "Certificate-based signencryption with enhanced security features", *Computers and Mathematics with Applications*, vol. 64, pp. 1587–1601, 2012.

[10] J. Zhang et al., "Digital certificate management: Optimal pricing and CRL releasing strategies", *Decision Systems*, vol. 58, pp. 74-78, 2014.

[11] J. Li et al., "Provably secure certificate-based signature scheme without pairings", *Information Sciences*, vol. 233, pp. 313-320, 2013.

[12] P. Kocher, "On Certificate Revocation and Validation", *FC 1998*, vol. 1465, pp. 172–177, 1998.

[13] Y. Kortesniemi, "Survey of certificate usage in distributed access control", *Computers & Security*, vol. 13, pp. 1-17, 2014.

[14] A. Lakshminarayanan, T. Lim, "Augmented Certificate Revocation Lists", *ACISP 2006*, vol. 4058, pp. 87–98, 2006.

[15] L. Wang et al., "Certificate-based proxy decryption systems with revocability in the standard model", *Information Sciences*, vol. 247, pp. 188-201, 2013.

[16] M. Cheminod et al., "A comprehensive approach to the automatic refinement and verification of access control policies", *Computers & Security*, vol. 80, pp. 186-199, 2019.

[17] S. Micali, "NOVODOMO - Scalable Certificate Validation and Simplified PKI Management", *Proceedings of the 1st Annual PKI Research Workshop*, 2002.

[18] Y. Mingxu et al., "Efficient integrity verification of replicated data in cloud computing system", *Computers & Security*, vol. 65, pp. 202-212, 2017.

[19] M. Mohammad et al., "Towards efficient certificate status validations with E-ADOPT in mobile ad hoc networks", *Computers & Security*, vol. 49, pp. 17-27, 2015.

[20] Y. Murat, S. Mehmet, A. Hac, "CertLedger: A New PKI Model with Certificate Transparency Based on Blockchain", *Computers & Security*, vol. 5, pp. 1-42, 2018.

[21] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public-Key Infrastructure Online Certificate Status Protocol - OCSP", *RFC 2560*, 1999.

[22] M. Naor, K. Nissim, "Certificate Revocation and Certificate Update", *Proceedings of the 7th USENIX Security Symposium*, 1998.

[23] Q. Li et al., "A Mobile-Certificate Security Method of Satellite-Earth Integration Networks", *ICICA 2012*, pp. 88-97, 2012.

[24] C. Santosh et al., "A novel access control protocol using proxy signatures for cloud-based health information exchange", *Computers & Security*, vol. 67, pp. 73 -88, 2017.

[25] M. Wang, C. Qian, X. Li, S. Shi, "Collaborative Validation of Public-Key Certificates for IoT by Distributed Caching", *IEEE/ACM Trans. Netw*, vol. 29, pp. 847–855, 2019.

[26] Y. Yao, Z. Li, "A novel fuzzy identity based signature scheme based on the short integer solution problem", *Computers and Electrical Engineering*, vol. 9, 2013.