A Performance Enhanced Privilege Verification Cache for Authorization in Ubiquitous Service Environment

Woomin Hwang, Ki-Woong Park, Chul Lee, Kyu Ho Park
Computer Engineering Research Laboratory, EECS
Korea Advanced Institute of Science Technology
Guseong-Dong, Yuseong-Gu, Daejeon 305-701, Korea
{wmhwang,woongbak,chullee}@core.kaist.ac.kr, kpark@ee.kaist.ac.kr

Abstract

In the current role-based privilege management infrastructure (PMI) scheme based on X.509, a verification server should verify a chain of attribute certificates (called privilege delegation path) for a verification request. It makes the server adopt verification cache to reduce an elapsed time taken for handling repeated certificate verification. In previous cache scheme, one cache entry has a verification result of one certificate. Unfortunately, this approach do not restrain the increase of verification time as a result of the increasing length of privilege delegation path. This paper presents a design and analysis of the delegation-path-based verification cache. In our proposed cache, an entry has an integrated verification result of all certificates in a valid privilege delegation path. Using our proposed cache, a verification server can leave out further lookup of cache when the cached verification result of the delegation path to be verified is found. The performance evaluation convinces that this cache yields a 46% to 62% reduction of the average verification time of a privilege and a 4 to 23 times higher throughput in comparison to conventional certificate cache.

1. Introduction

Authorization systems play a key role in achieving a high security of computer users, and thus various ubiquitous services need to build authorization infrastructure to protect privacy-sensitive resources from unauthorized access. We have implemented a campus-wide ubiquitous environment, U-TOPIA [11], which consists of various service devices such as U-Kiosks, U-Print, Zigbee-enabled appliances. In our environment, users access these devices using our wearable computer platform, UFC[1], under authorization system. In U-TOPIA, most service devices are resource-constrained though the verification procedure is CPU-intensive job. Thus we hand over the verification procedure to several resource-rich servers to enhance verification speed.

[1]UFC : Ubiquitous Fashionable Computer [8]

Since the number of services for campus members increases and the majority members are students with scheduled lectures, service users try to use their privilege in a specific time period. For example, when a lecture having hundreds of students is about to begin, then the verification servers receive substantial service requests in a relatively short period of time. As a result, the converged load of authorization process increases latency of the users. Thus an efficient authorization system is necessary to provide the authorization with proper latency.

As a fundamental way to enable authorization, role-based privilege management infrastructure (PMI) based on ITU-T X.509 attribute certificates (ACs) [1, 2] is used due to its ease of integration with public key infrastructure (PKI) and the convenient management of privileges.

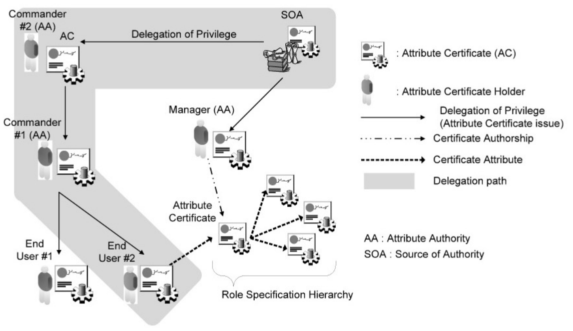

Figure 1. An Example of privilege delegation chain in role-based X.509 PMI

However, the authorization infrastructure is based on certificates that require verification of digital signature, thus the increase of the number of certificates that need to be validated leads to the increase of user's latency. Figure 1 shows an example of privilege delegation path in the role-based X.509 PMI. To verify a privilege of the end user #2, all ACs in role specification hierarchy, ACs of manager, and the ACs of commanders between the end user #2 and the SOA (Source of Authority) have to be verified. If a commander is inserted into the delegation path, highlighted as gray box in Figure 1, ACs of the inserted commander and the ACs of the commander's role specification hierarchy also have to be verified. In addition to this, the centralization of offloaded CPU-intensive job to the server depletes the server's CPU time, which causes long latency for user to use the privilege. Consequently, these overhead degrade throughput of the verification server.

In PERMIS [3] project, the problem is partially solved by integrating certificates in role specification hierarchies into one policy AC written in XML. That approach reduces the verification time of a privilege by reducing the number of signature verifications with policy AC. As a solution about the depletion of CPU time in server, credential cache [9] reduces the expected number of CPU-intensive job by caching verification results. However, the cache does not restrain the increase of verification time as a result of the increasing length of the privilege delegation path.

This paper presents a new privilege verification cache to accelerate the verification procedure. More specifically, this paper focuses on the performance enhancement of cache in verification server with two ideas. First idea is that every privilege delegation chain can share some of the verification result of other delegation path. For example, when the end user #1 and the end user #2 have the same commanders (commander #1, #2), the two end users can share the same delegation subpath composed of the commander #1, the commander #2, and the SOA. Hence the verification server can use verification results of the certificates in the shared delegation subpath whenever the two users' privileges are verified. Second idea is that we can cache the verification result of all certificates of a delegation path into one cache entry because a privilege is valid if and only if all certificates in the delegation path are valid. If the verification server uses the cache proposed in [9], the server still has to lookup entry of next certificate in path. But, in our approach, the server does not have to repeat cache lookup and verification when cache hit occurs because the hit entry contains all verification results about the delegation path. Thus we reduced the average number of signature verifications for verifying a privilege of user.

The use of these ideas yields a 46% to 62% reduction of the average verification time of one privilege, and thus a 4 to 23 times higher throughput in comparison to traditional certificate cache. In addition to this, the number of LDAP (Lightweight Directory Access Protocol) [13] requests for delegation path discovery is also reduced. This paper analyzes the throughput of server and performance impact of this cache in detail.

The rest of this paper is structured as follows. Section 2 describes related works including standards and researches in this area, followed by description of notions and architecture of proposed privilege verification cache in Section 3. Section 4 describes results of performance evaluation. Finally, Section 5 draws conclusion of this paper.

2. Related Works

The verification overhead of the delegation chain in role-based X.509 PMI standard was found by Knight and Grandy [7], and this was also noted by the authors of RFC3281 [4]. But delegation of authority is one important feature required in the secure organizations, federations, and virtual organizations, there are some efforts to make delegation of authority more efficiently. Offloading CPU-intensive operations from resource-constrained devices to the resource-rich servers is an obvious solution to reduce power consumption of service devices and service response time. In RFC3379 [12], a DPV (Delegated Path Validation) server takes certification path validation process from clients and returns the validation result. It is useful at the point of view for its acceleration effect by offloading certificate path discovery and path validation to server. However, it concentrates on offloading principle of validation operation so that there is no consideration of arising problem, centralization of CPU-intensive operation, in offload server. Centralization of offloaded CPU-intensive operations causes longer response time of service devices. Because the digital signature verification of a certificate is a CPU-intensive job, centralization of the certificate verification procedure depletes CPU cycles. Then the delayed time by reason of wait to get CPU time for processing is shifted to users. Finally, the users are forced to wait during the delayed time.

Use of certificate cache for the verification server can solve this problem by caching current status of credentials, that is, validity of X.509 certificates. Online certificate status protocol (OCSP) [10] defined in IETF RFC 2560 describes a network protocol to verify the validity of an X.509 public key certificate. Under OCSP, a verification server, OCSP responder, receives verification requests of certificates and replies the current status of the certificates. To accelerate the verification procedure of the certificates, RFC2560 allows the verification server to deploy a certificate cache to manage status of already verified certificates. Though it is essential for the server to reduce the verification time, there is no detailed explanation about the implementation of cache. The OCSP only represents a communication protocol between verification server and related clients, and it is the administrator's responsibility for the use of certificate cache to enhance server's performance. Furthermore, OCSP burdens the resource-constrained service devices with protocol overhead and the OCSP responder have to sign all responses which depletes server's CPU cycles for the verification requests. Server-based Certificate Validation Protocol (SCVP) [5] which supports certificate validation also has same weak point.

Jiangtao et al. [9] proposed a credential cache to defend the DoS (Denial of Service) attack which depletes CPU time. The contribution of their work is reducing the number of signature verification using the credential cache to reduce latency of the service user. The credential cache is effective to secure CPU time to serve more verification requests, but it can be enhanced. Because it does not consider the fact that the privilege verification is performed by verifying multiple certificates along a delegation path from end entity's certificate to trust anchor. Our work starts from this point.

3. Proposed approach : Path-based Verification Cache (VC)

We present a novel caching mechanism for maximizing effect of caching valid certificates' verification results. The cache entry containing verification result of entire delegation path of a privilege makes it possible for service users to reduce the latency of the verification procedure.

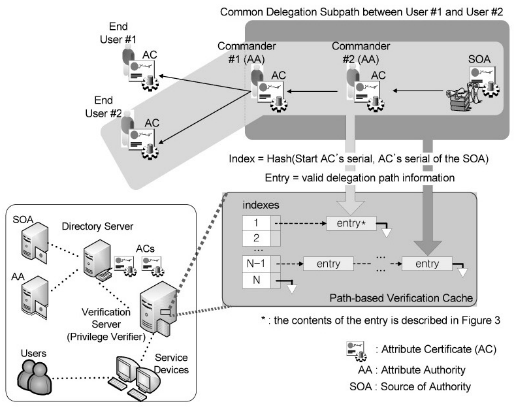
Common Delegation Subpath between User #1 and User #2
End User #1
AC
End User #2
AC
Commander #1 (AA)
AC
Commander #2 (AA)
AC
SOA
Index = Hash(Start AC's serial, AC's serial of the SOA)
Entry = valid delegation path information
indexes
1
2
N−1
N
entry*
entry
entry
Path-based Verification Cache
* : the contents of the entry is described in Figure 3
SOA
Directory Server
ACs
AA
Verification Server (Privilege Verifier)
Users
Service Devices
: Attribute Certificate (AC)
AA : Attribute Authority
SOA : Source of Authority

Figure 2. Overall scheme of proposed path-based verification cache

3.1. Cache Entry based on Privilege Delegation Path

Once a verification procedure along a delegation path is completed, the verification server inserts verification information of the valid delegation path into a cache entry. Figure 2 shows some part of caching procedure of valid delegation path for the end user #2. When a privilege of the end user #2 needs to be verified for the first time, the verification server verifies a certificate of the end user #2, the commander #1, the commander #2, and the SOA sequentially. After the completion of the verification procedure, three entries are inserted into the cache. The first one contains a verification information of the delegation path between the end user #2 and the SOA. The second entry contains information between the commander #1 and the SOA. The third entry holds information about delegation path between the commander #2 and the SOA. Therefore, the verification server can reuse the second entry for the privilege verification procedure of the end user #1. Both cache schemes maintain the same number of entries as many as the number of certificates in the delegation path without the SOA.

```
struct _entry {
        PATH                   delegation_path;
        SERIALS                certs_in_path;
        TIMEPERIOD             validity;
        TIMESTAMP              last_valid_timestamp;
        CRL_LOCATIONS          crldist_point;
};
```

Figure 3. Fields of an entry in path-based verification cache

Figure 3 shows fields of a cache entry. The first field contains a pair of AC's serial numbers, (start AC's serial number, AC's serial number of the SOA), representing a valid delegation path. We use a fingerprint value of the first field as a key for cache lookup because of the serial number's uniqueness. The second field contains all serial numbers of ACs included in the path for deletion of the cache entry. The third field conveys a maximum time period that all ACs in the second field are valid according to the *Validity* field of each AC. Because the cached entry is valid only if all ACs in the entry are valid. The fourth field contains the latest timestamp when the field is certified as valid. After cache hit, the verification server has to look up certificate revocation lists (CRLs) to assure that all ACs in the path are not revoked. But there is no need to search CRLs published before the time when the entry is certified as valid. Thus, the hit entry contains timestamp when the entry is verified last. The last field contains CRL distribution points of whole ACs in the second field of the entry. If all ACs in the second field have *noRevAvail* extension, this field has nothing.

The above cache entries are indexed using hash with a key value of the *delegation_path* field. Insertion of entry occurs when the new subpath is created by the signature verification of certificates in the delegation path. Removal of entry occurs when the entry expires or invalidated certificates are found during CRL lookup. The verification server searches index with the fingerprint value of the key, and traverse the list until target is found or the last entry is identified.

3.2. Privilege verification procedure

Figure 4 shows the verification procedure of the proposed delegation-path-based verification cache. The verification procedure is a combination of a CRL lookup and a recursive work composed of cache lookup and the validation of uncached certificates. The procedure starts with entire path between an end user and the SOA, denoted as *sp(n)*, where *n* indicates the number of certificates in the path. The first thing is calculating fingerprint value of the key representing *sp(n)* and searching the target path in the cache. A pair of two serial numbers, (serial number of the end user, AC's serial number of the SOA), is used as a key. If the cached entry is found, the privilege verifier investigates whether the entry expires or the certificates in the entry are revoked. The privilege delegation path in the hit entry is determined as valid when the entry passes those investigations. However, if there is no such entry in the cache, recursive verification of individual certificates has to be performed. In other words, if the verification result of target path is not cached, the verifier verifies one certificate which the delegation path starts from. If the certificate is valid, then the verifier creates subpath by subtracting already verified certificate from the target delegation path. Then the key for cache search is (serial number of next certificate in the path, AC's serial number of the SOA). The verification result of generated subpath is searched in the next cycle of the verification procedure. This recursive procedure ends with two cases. One thing is when the hit cache entry is determined as valid after CRL lookup. The other thing is when the AC of the SOA is verified.

After cache lookup, the privilege verifier has to confirm that all ACs in the hit entry are still valid at the verification time. To achieve this, the privilege verifier retrieves all CRLs from distribution points in *crldist_point* field of the hit entry. Then the privilege verifier looks up serial numbers in *certs_in_path* field from the CRLs. If no one is found in the CRLs, the path is determined as valid. But if one of the serial numbers is found, the cached verification information is useless, so the entry is deleted and the path is determined as invalid. The final procedure is inserting new entries about the valid subpath, if exists. When the entries of subpaths $sp(n), sp(n-1), \cdots, sp(n-k+1)$ are not in cache but subpath $sp(n-k)$ is in the cache, the entries about subpaths, $sp(n), sp(n-1), \cdots, sp(n-k+1)$ are inserted into the cache.

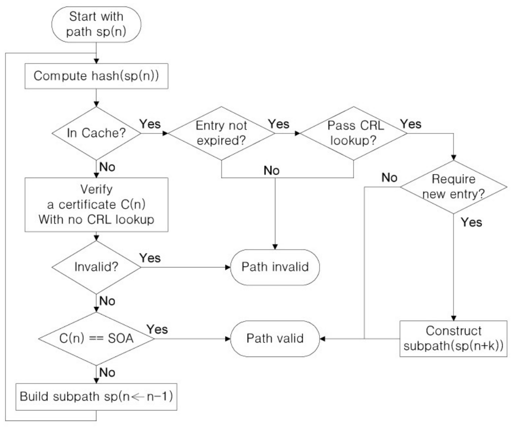

Figure 4. Privilege verification procedure with path-based verification cache. n is the number of certificates in a path and k is the number of cache miss.

3.3. Consistency of cache entry: delayed revocation check

To achieve the consistency of cache entry, we delayed revocation check procedures of certificates in the hit entry to ensure validity of the cached verification result. A cached verification result is a snapshot about the status of the certificates at the cached time. So, the validity of cached verification results cannot be assured at the verification time. In addition to this, a privilege is verified based on the path in the proposed path-based VC, CRL lookup cannot be done with each certificate's verification. As a solution, we delayed CRL lookup procedure of all certificates after cache hit. This procedure, called delayed revocation check, is shown in Figure 4. When a delegation path is hit, then the verification server retrieves CRLs recorded in *crldist_point* and verify whether certificates in path is revoked after the time in *last_valid* field of each entry. After checking all CRLs, the validity of the hit entry is confirmed.

4. Performance Evaluation

In this section we show the performance evaluation result of our delegation path based caching strategy in comparison by simulation. Our simulations were carried out on an Intel 4-way 3.40GHz Xeon processor machine with 4GB of RAM running Linux Fedora Core 5. We assumed that every message transmission is secure and the verification system has already public key verification infrastructure for user and attribute authorities.

The experimental environment consists of four components and communicates with each other. When an end user tries to use his privilege to get a service, a service device sends user's information to the verification server. Then, the verification server investigates whether the delegation path is valid. The procedure includes retrieval of ACs related to the end user's privilege from directory server[2] including revocation information. Then the verification server replies verification result to the service device in a secure way. The service device reacts to the request with the verification result as if the service device verified user's privilege of its own.

Because this proposed method is for the U-TOPIA environment in KAIST campus, the target authorization hierarchy reflects the organization of KAIST. We have assumed that a privilege of a graduate student is delegated along the real path[3] according to the organization chart for KAIST. We have also assumed that 2 to 4 levels of hierarchy are optionally required to manage hierarchy among students. So we decided that the maximum delegation length of a privilege is ten, the number of policy AC for all privilege is one, and the number of CRL is one with one revoked certificate.

We implemented a path-based verification cache based on Figure 4 using C language. The following results are generated from the simulation with the implemented codes. For the measurement of server capacity, we assume that verification requests arrived at the server with Poisson distribution.

In the following subsections, we show how the delegation-path based caching method affects verification delay and server throughput. Then we also show simple calculated results about the effect of the proposed method to the certificate path discovery. In those results, we refer to Jiangtao et al.'s method as CredentialCache, and proposed verification cache as Path-based VC.

4.1. Elapsed time of privilege verification

Table 1 shows the number of cryptographic operations required to verify a privilege on each works. In this table, *n* indicates the number of certificates in the path and *j* indicates the number of policy ACs for the privilege delegation hierarchy. *p* means the hit probability of the verification result for each one certificate. Note that the verification procedure of a delegation path followed description in RFC2459 [6]. But we neglected other operations in verification procedure because those have little portion of operation time in comparison with signing and signature verification operation.

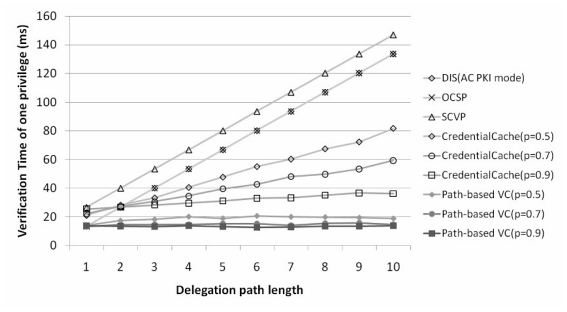

Figure 5. Verification time comparison of one privilege of a user according to the length of the delegation path. The p is hit probability of the cache.

Figure 5 shows that the reduced number of signature verifications results in significant reduction of the verification time of one privilege where the delegation path lengthens. When the hit probability p is 0.9 and the length of the delegation path is one, the proposed cache scheme reduces verification time of a privilege to 13.6ms in comparison to the verification time, 25.5ms, of CredentialCache. Furthermore, even the length increases, proposed path-based VC restrained the increase of verification time despite the result of the CredentialCache increases. When the length of delegation path is ten with same hit ratio, the proposed cache yields a 62.2% reduction of verification time of one privilege in comparison to the CredentialCache. This reduced verification time make the verification system more scalable for the decreased relevance between verification time and the length of the delegation path.

Figure 6 shows the throughput of one verification server represented as the number of requests responded less than 500ms. 100,000 requests were used for the experiment. Because the proposed cache scheme reduces the required verification time of one privilege for one user, more users can be served by one server on time, represented by the throughput of the server. For the path-based VC with hit probability 0.9, server throughput is almost 20 times larger than the CredentialCache, though it has large deviation. The large deviation of the server throughput is the weak point of the proposed path-based VC. It is because entries have same hit probability despite each entry has a verification result about different length of delegation path. We left the reduction of

[2]Directory server stores attribute certificates. We assume that LDAP directory server is used.

[3]President - Vice President of Operations - Head of College of Engineering - Head of Division of Electrical Engineering - Professor - Student

		Signing		Signature Verification	
		Mandatory	Optional	Mandatory	Optional
Verification Server	OCSP_nocache	1	1	$n+j$	1
	SCVP	1		$1+n+j$	
	CredentialCache	1		$(1-p)(n+j)$	
	Path-based VC	1		$(1-p_{policy})j+n\prod_{m=1}^{n}(1-p_m)$ $+\sum_{k=1}^{n}p_k\left(\prod_{l=0}^{k-1}(1-p_l)\right)(k-1)$	

Table 1. Number of cryptographic operations

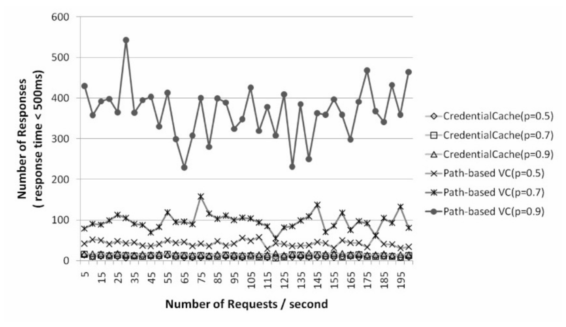

Figure 6. Throughput comparison represented by the number of requests responded under 500ms. The target delegation length is ten and the p is hit probability of the cache.

the large deviation as our further work.

5. Conclusion

In this paper, we propose a privilege verification cache based on delegation path to reduce the privilege verification time and to increase the server processing capacity. The result of this study indicates that caching verification result with proposed path-based verification cache leads to significant performance improvement, even compared with the caching of individual certificate's result. Processing capabilities of the verification server is also enlarged. Furthermore, path-based VC reduces the number of LDAP requests about the delegation path discovery.

As a future work, stabilization of server performance, development of cache replacement algorithms, and evaluation of reduced LDAP requests to the performance of LDAP server is remained.

References

[1] ITU-T Rec. X.509(2005) The Directory: Public-key and attribute certificate frameworks.

[2] ITU-T Rec. X.812(1995) Security Frameworks for open systems: Access control framework.

[3] D. W. Chadwick and A. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. In *7th ACM Symposium On Access Control Models And Technologies (SACMAT 2002), Monterey, USA*, pages 135–140, June 2002.

[4] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. RFC3281, April 2002.

[5] T. Freeman, R. Housley, A. Malpani, D. Cooper, and W. Polk. Server-based Certificate Validation Protocol (SCVP). Internet Draft, September 2007.

[6] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC2459, January 1999.

[7] S. Knight and C. Grandy. Scalability Issues in PMI Delegation. In *Proceedings of NIST 1st Annual PKI Research Workshop*, Gaithersburg, April 2002.

[8] J. Lee, S.-H. Lim, J.-W. Yoo, K.-W. Park, H.-J. Choi, and K. H. Park. A Ubiquitous Fashionable Computer with an i-Throw Device on a Location-Based Service Environment. In *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, pages 59–65, Washington, DC, USA, 2007. IEEE Computer Society.

[9] J. Li, N. Li, X. Wang, and T. Yu. Denial of Service Attacks and Defenses in Decentralized Trust Management. In *Securecomm and Workshops*, pages 1–12, 2006.

[10] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC2560, June 1999.

[11] K. H. Park, J. Lee, J.-W. Yoo, S. H. Lim, K.-W. Park, H.-J. Choi, and K. Wohn. U-TOPIA: Campus-wide Advanced Ubiquitous Computing Environment. In *Conference on Next Generation Computing*, pages 5–16, 2007.

[12] D. Pinkas and R. Housley. Delegated Path Validation and Delegated Path Discovery Protocol Requirement. RFC3379, September 2002.

[13] W. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol. RFC2251, December 1997.