International Journal of Mechanical Engineering

OCSP modification for fast processing of stapled message

Eun-Gi Kim[*1]

[*1]Professor, Dept. of Information and Communication Engineering, Hanbat National Univ.,
125 Dongseo-daero Yuseong-gu Daejeon City, 34158, Republic of Korea

Abstract.
BACKGROUND/OBJECTIVES: The public certificate used in PKI provides a function to bind the user's identity and public key, and is actively used as a means of identification and digital signature.
Methods/Statistical analysis: When a certificate is created, it is valid until the expiration date. However, when the private key of the certificate is exposed or the user's affiliation is changed, the certificate with the validity period may become invalid. In this case, the user verifying the certificate must be able to know information about the revoked certificate.
FINDINGS: Certificate authority (CA) stores information about revoked certificates in the Certificate Revocation List (CRL), which is updated periodically. Methods for providing information on the revoked certificate to the user include the CRL distribution method and the use of the Online Certificate Status Protocol (OCSP) protocol. In the CRL distribution method, all users periodically download the updated CRL, which incurs a lot of overhead for users. Another way to provide information about the revoked certificate to the user is to use the OCSP protocol. In this method, the CA sends a CRL that is periodically updated to the OCSP responder. The user sends an OCSP request including the serial number of the certificate to be verified to the responder. Receiving the user's request, the responder checks the validity of the certificate by referring to the contents of the CRL and transmits the result as an OCSP response.
IMPROVEMENTS/APPLICATIONS: In this study, we propose a fast verification method that can be applied in the case of validating certificates using the OCSP protocol and confirm the results through simulation.

Keywords: CRL, certificate, OCSP, stapled, fast processing

1. INTRODUCTION

PKI (public key infrastructure) was initiated to facilitate the safe use of information in various sectors such as finance, e-commerce, and the public sector on the Internet. To support this, PKI is defining the underlying technology necessary to create, manage, distribute, use, store, and revoke digital certificates and manage public key encryption[1,2]. And the X.509 standard defines the structure of public key certificates, and these certificates are used in many Internet protocols including SSL/TLS. An X.509 certificate includes the identity of a user (a host, an organization, or an individual name), the public key used by the identity, and the sign of a certificate authority (CA) that can verify these two facts[3].

The X.509 certificate configured in this way provides a function to bind the user's identity and public key, and is actively used as a means of identification and digital signature throughout the Internet. Users can verify whether the certificate information is forged or altered with the CA sign on the certificate. In addition, whether or not the contents of a document signed by a user have been forged can be verified using the public key in the user's certificate[4].

Representative asymmetric cryptography algorithms for certificate verification include RSA (Rivest–Shamir–Adleman) and ECDSA (Elliptic Curve Digital Signature Algorithm)[5,6]. RSA is a public key cryptographic algorithm published in 1977 and is based on the integer factorization problem. Currently, NIST recommends that the RSA key length should be 2048 bits or more, and this key length has a disadvantage that requires a lot of time for signature generation and verification. ECDSA is an elliptic curve series of digital signature algorithm using the difficulty of the discrete logarithm problem. The use of ECC (elliptical curve cryptography) algorithms is increasing actively as they use smaller key lengths compared to conventional public key algorithms that include RSA. The NIST currently recommends using 224 bits or more for ECDSA keys[7].

The generated certificate is valid until the expiration date. However, if the private key of the certificate is exposed or the user's affiliation is changed, the certificate with the validity period may become invalid. CA stores the serial number of the revoked certificate in the CRL (Certificate Revocation List)[8]. Therefore, a user who wants to verify a certificate must first check whether the certificate is valid.

Methods for validating certificates include using CRL distribution methods and the Online Certificate Status Protocol (OCSP) protocol[9]. In the CRL distribution method, all terminals copy the CRL and use it when verifying the certificate. Using the OCSP protocol, the CA sends the CRL to the OCSP responder (server), and the user sends the serial number of the certificate to the OCSP responder to verify the validity of the certificate.

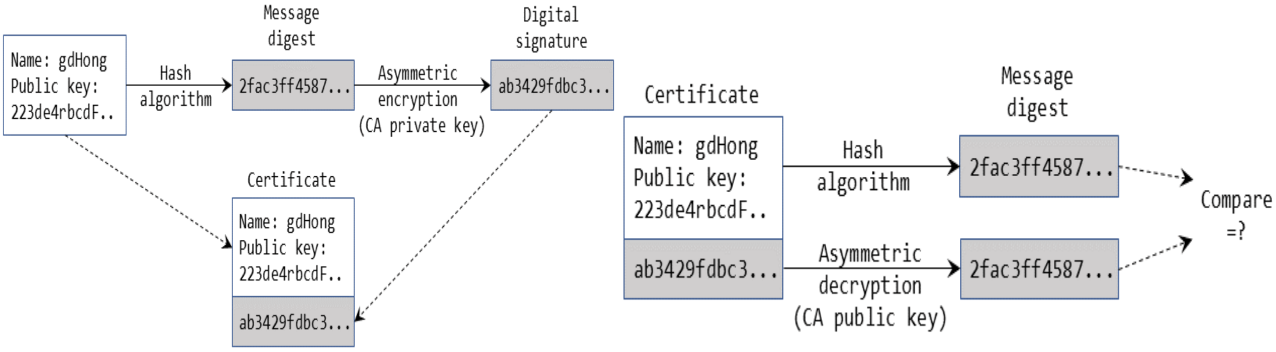
In this study, we proposed ways to reduce the overhead of revoked certificate verification process using OCSP and analyzed the performance through experiments. It is thought that the method proposed in this study can be used for vehicle communication systems, where real-time processing is essential[10]. In Chapter 2, related studies are described, and in Chapter 3, the proposed method and its performance are analyzed. Finally, the conclusion and future research plans are described.

2. RELATED RESEARCHES

In this chapter, we describe the contents related to certificate verification.

2.1. CERTIFICATE GENERATION AND VERIFICATION

Figure 1 shows the certificate creation and verification process. As you can see in figure 1(a), the CA that wants to generate a certificate calculates the hash value of the message including the name of the certificate issuer and the public key. And the calculated hash value is encrypted with the CA's private key and used as a digital signature of the certificate.



(a) Certificate generation **(b) Certificate validation**

Figure 1. Certificate generation/validation procedures

Figure 1(b) shows the verification process of the certificate. For certificate verification, as you can see in figure, the hash value for the contents of the certificate is calculated. The digital signature of the certificate is decrypted with the CA's public key and the hash value of the message is compared. If the two values are the same, the certificate verification succeeds.

2.2. CERTIFICATE REVOCATION STATUS CHECK

If the membership of the user who owns the certificate is changed or the secret key associated with the certificate's public key is exposed, the certificate must be revoked. These certificates are treated as revoked even if they remain valid. During certificate verification, the user should examine the revocation status of the certificate, along with comparing the hash value of the certificate contents. Methods to check the certificate revocation status include CRL distribution method and OCSP protocol.

The CRL distribution method distributes the CRL generated by CA to all users. Users can check whether the certificate to be verified has been revoked by using their own CRL. However, since the CRL can be updated continuously, all users have the disadvantage of having to update their CRL periodically.

Figure 2 Shows the operation of the OCSP protocol. [10] As shown in the figure, the client can be able to verify the revocation status of a certificate by sending the serial number of the certificate they want to investigate to the OCSP responder (server). This approach has the advantage that all users do not need to download CRLs, but it has the disadvantage of causing problems such as delay in transmission and DOS attacks during query/response over the network.

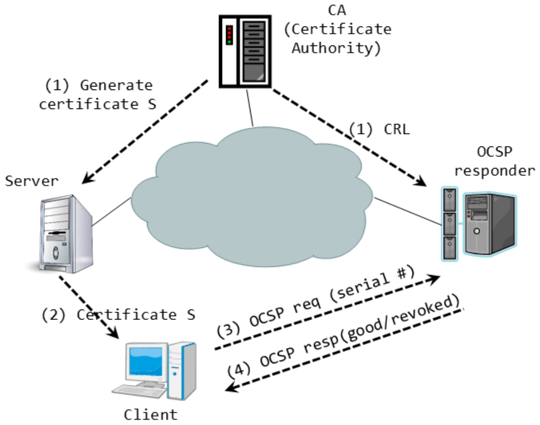
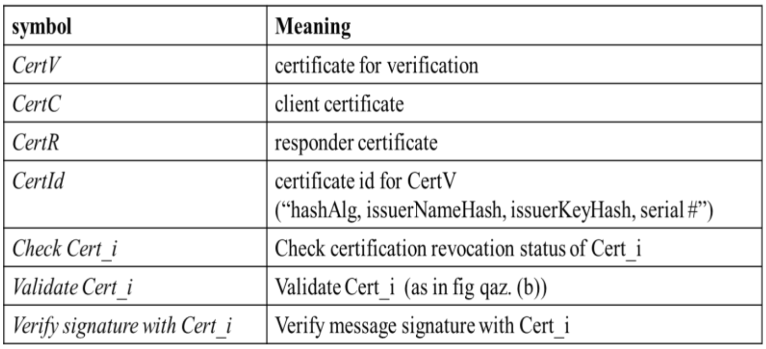


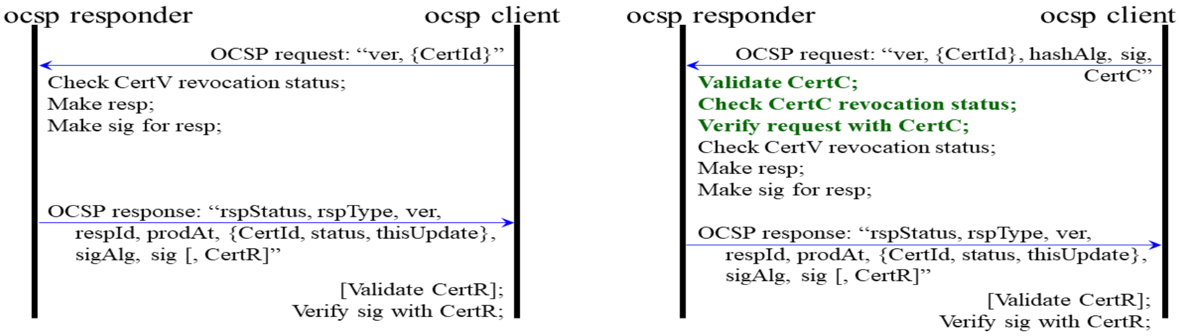
Figure 2. The operations of OCSP protocol

The verification latency that occurs when using the OCSP method may be a significant issue depending on the application. For example, in the case of a vehicular network, a vehicle in operation periodically broadcasts messages related to safety. For these safety messages, they are sent with certificates, and signatures of messages to support the non-repudiation property for sending messages. The user who receives the safety message must go through the process of checking the revocation status of the certificate, verifying the certificate, and verifying the message. Since this process must be carried out quickly in real time, the time delay that occurs during the certificate revocation process makes it difficult to apply OCSP in the vehicular network. Table 1 Shows the notations to be used in this study.

Table 1. List of notations



symbol	Meaning
CertV	certificate for verification
CertC	client certificate
CertR	responder certificate
CertId	certificate id for CertV ("hashAlg, issuerNameHash, issuerKeyHash, serial #")
Check Cert_i	Check certification revocation status of Cert_i
Validate Cert_i	Validate Cert_i (as in fig qaz. (b))
Verify signature with Cert_i	Verify message signature with Cert_i


ocsp responder
ocsp client
OCSP request: "ver, {CertId}"
Check CertV revocation status;
Make resp;
Make sig for resp;
OCSP response: "rspStatus, rspType, ver,
respId, prodAt, {CertId, status, thisUpdate},
sigAlg, sig [, CertR]"
[Validate CertR];
Verify sig with CertR;
ocsp responder
ocsp client
OCSP request: "ver, {CertId}, hashAlg, sig,
CertC"
Validate CertC;
Check CertC revocation status;
Verify request with CertC;
Check CertV revocation status;
Make resp;
Make sig for resp;
OCSP response: "rspStatus, rspType, ver,
respId, prodAt, {CertId, status, thisUpdate},
sigAlg, sig [, CertR]"
[Validate CertR];
Verify sig with CertR;

(a) OCSP request without signature **(b) OCSP request with signature**

Figure 3. OCSP message processing

Figure 3 Shows the processing of messages sent and received between the client and responder during OCSP operation. As shown in the figure 3, OCSP requests may or may not include signatures of OCSP clients. If the OCSP request does not include signature, responder simply checks the revocation status of the CertV to be verified and sends the OCSP response. When the signature is included in the OCSP request, the OCSP responder performs the validation and check process of CertC included in the request, and verifies the signature of the OCSP request. The OCSP responder checks the CertV when this process is completed successfully.

2.3. OCSP STAPLING

The OCSP protocol for certificate revocation check has a several problems. For example, if multiple clients access a server at the same time, there may be problems with the rapid increase in requests to the responder because all clients perform OCSP verification process for the server certificate. In addition, delays in the OCSP process performed by clients to verify the server's certificate may slow the overall processing speed of the browser.

To solve these problems, the OCSP stapling method was proposed. [11] Figure 4 Shows the operation of the OCSP stapling method. As you can see in the figure, the server sends an OCSP request and receives an OCSP response confirming that its certificate is normal. The server then sends its certificate to the clients along with an OCSP response that indicates that the certificate is healthy. The client can use the received OCSP response to verify that the received certificate has not been revoked.

The OCSP stapling method has the advantage of reducing the burden on OCSP responder because the server sends OCSP response to a large number of clients.

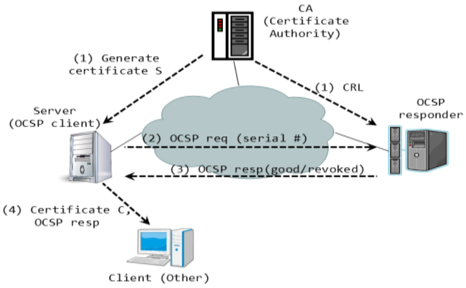
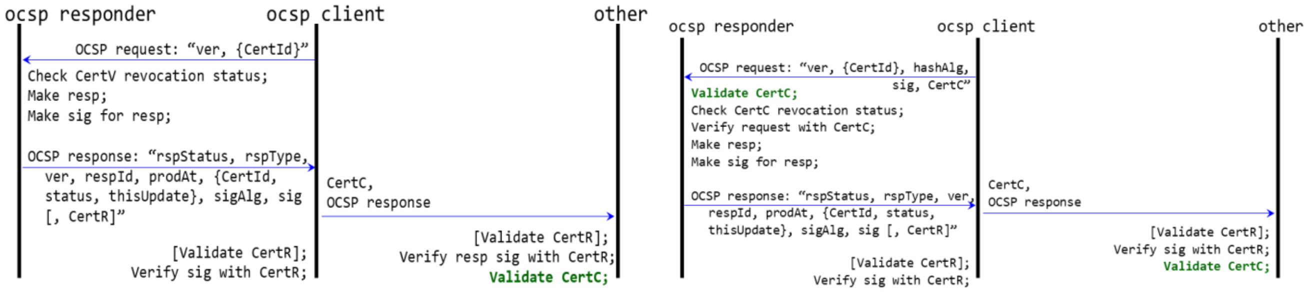


Figure 4. OCSP Stapling operations

Figure 5 shows the message processing process between OCSP responder, server (OCSP client), and client (other) in OCSP stapling method. Figure 5(a) is the case that the OCSP request does not have the server's signature, and figure 5(b) is the case where there is a signature.


ocsp responder
ocsp client
other
OCSP request: "ver, {CertId}"
Check CertV revocation status;
Make resp;
Make sig for resp;
OCSP response: "rspStatus, rspType, ver, respId, prodAt, {CertId, status, thisUpdate}, sigAlg, sig [, CertR]"
CertC,
OCSP response
[Validate CertR];
Verify sig with CertR;
[Validate CertR];
Verify resp sig with CertR;
Validate CertC;
ocsp responder
ocsp client
other
OCSP request: "ver, {CertId}, hashAlg, sig, CertC"
Validate CertC;
Check CertC revocation status;
Verify request with CertC;
Make resp;
Make sig for resp;
OCSP response: "rspStatus, rspType, ver, respId, prodAt, {CertId, status, thisUpdate}, sigAlg, sig [, CertR]"
CertC,
OCSP response
[Validate CertR];
Verify sig with CertR;
[Validate CertR];
Verify sig with CertR;
Validate CertC;

(a) OCSP request without signature (CertC=CertV) **(b) OCSP request with signature (CertC=CertV)**

Figure 5. Processing of OCSP stapling message

As the figure 5 shows, the other who received the CertC and OCSP response verifies the OCSP response message with the certR and validates the CertC. If the other does not validate the CertC, the attacker will be able to perform the following actions:

- The attacker copies the transmitted OCSP response and CertC.
- The attacker arbitrarily changes the contents of the part excluded from the CertId field of CertC and transmits the forged CertC to other.
- Other judges that the forged certificate is a normal certificate because the value of the certId field sent by the attacker is the same as the certId field of the OCSP response.

3. DESIGN AND IMPLEMENTATION OF OCSP FOR HIGH-SPEED PROCESSING

3.1. STAPLED OCSP TO SUPPORT FAST PROCESSING

As can be seen in Figure 5, in the process of the existing OCSP stapling method including the signature in the OCSP request, it can be seen that the OCSP responder and the other perform the process of validating CertC redundantly. This is because CertId field in OCSP request and response is composed of "hashAlg, issuerNameHash, issuerKeyHash, serial number", etc[11].

If the CertId field in RFC 6960 is changed to "hashAlg, hash(CertC), serial number", it is possible to prevent duplicate validation of CertC. The OCSP responder performs the CertC validation process, and if validation is valid, this information is stored in the status of the OCSP response. Therefore, users who have received CertC and OCSP responses can skip the validation process of CertC. Figure 6 shows the entire process of the new OCSP stapling method proposed in this study.

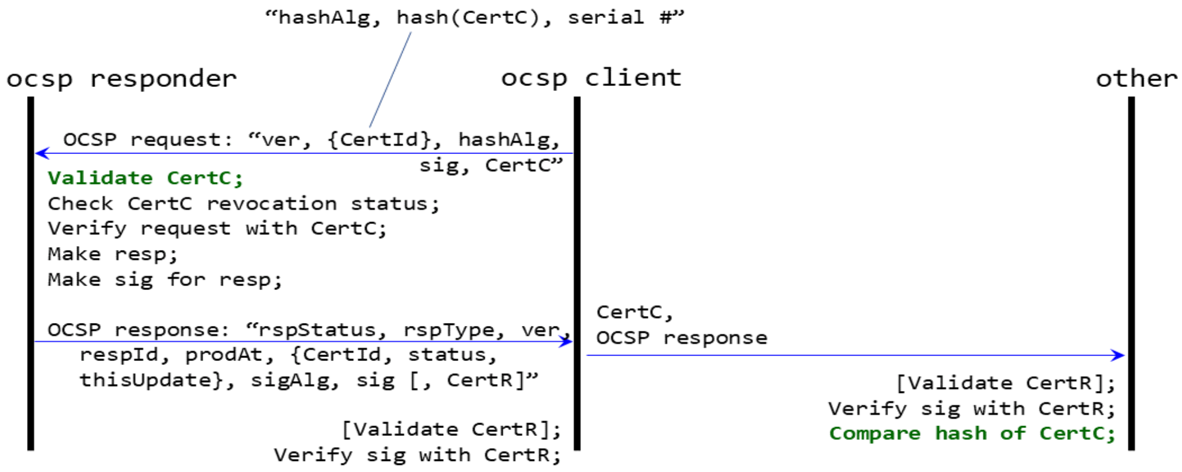

"hashAlg, hash(CertC), serial #"
ocsp responder
ocsp client
other
OCSP request: "ver, {CertId}, hashAlg, sig, CertC"
Validate CertC;
Check CertC revocation status;
Verify request with CertC;
Make resp;
Make sig for resp;
OCSP response: "rspStatus, rspType, ver, respId, prodAt, {CertId, status, thisUpdate}, sigAlg, sig [, CertR]"
[Validate CertR];
Verify sig with CertR;
CertC,
OCSP response
[Validate CertR];
Verify sig with CertR;
Compare hash of CertC;

Figure 6. Modified OCSP stapling operation

As the figure shows, the validation process of CertC is performed only once in the OCSP responder, and the validation results are recorded in the OCSP response and sent to the OCSP client. OCSP client (application server) transmits OCSP response including CertC and CertC validation results to other party. Therefore, the other party will be able to verify the revocation status of the certificate by comparing only the hash value of CertC without validating the certificate.

3.2 OCSP PERFORMANCE FOR HIGH-SPEED PROCESSING

The performance was analyzed by comparing the processing speed of the existing OCSP stapling method and the method proposed in this study. The parameters used in the performance analysis are as follows.

- Hash algorithm: SHA-256
- Cryptography algorithm: ECDSA
- EC domain parameters: prime256v1 (NIST P-256, secp256r1)
- Simulation Board: BeagleBone Black Wireless (OSD3358 1GHz ARM® Cortex-A8, 512MB DDR3 RAM)
- Kernel: Linux BeagleBone 4.14.108-ti-r113 (Debian 9.13)
- Simulation code: C with openssl library

As shown in the figure 7, the method proposed in this study shows faster processing speed because it simply compares hash values without performing the ECDSA verification process that traditional stapling methods should perform.

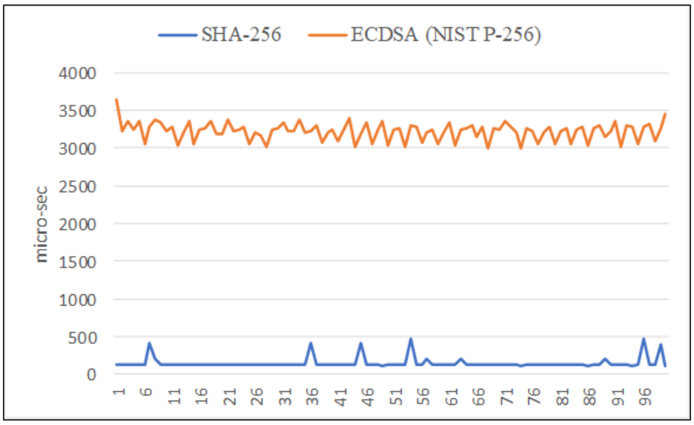


Figure 7. Performance evaluation of our designed methods

4. CONCLUSION

The use of public key certificates is increasing as a means of identification and electronic signature of Internet users. Certificates are usually used as valid until extension date. However, if the affiliation of the certificate user is changed or the private key of the certificate is exposed, the certificate can no longer be used normally. Methods to enable end users to check information about these revoked certificates include CRL distribution, OCSP, and OCSP stapling.

In this study, the new OCSP stapling method for fast verification of certificate revocation status was presented and its performance was analyzed. The method proposed in this study supports the end user to quickly process the OCSP stapling operation through a simple change of the OCSP message structure in RFC. The results of this study are considered to be applicable to various environments requiring real-time verification of certificate revocation status. Later, a study will be carried out to apply this study to the VANET (Vehicular ad hoc network) environment for automotive communication.

5. ACKNOWLEDGMENT

This research was supported by the research fund of Hanbat National University in 2020.

6. REFERENCES

1. Adams, Carlisle, Lloyd, Steve. Understanding PKI: concepts, standards, and deployment considerations. Addison-Wesley Professional; 2002. p.11–5.
2. P Gutmann. PKI: it's not dead, just resting. IEEE Computer. 2002 Nov; 35(8):41~9.
3. Cooper D, Santesson S, Farrell S. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Internet Standard, RFC 5280. 2008 May. Available from: https://tools.ietf.org/html/rfc5280.html
4. Jeff Woods. Understanding Public Key Infrastructure and X.509 Certificates. linux journal. 2019 Jun. Available from: https://www.linuxjournal.com/content/understanding-public-key-infrastructure-and-x509-certificates
5. Johnson J, Kaliski B. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. Internet Standard, RFC 3447. 2003 Feb. Available from: https://tools.ietf.org/html/rfc3447
6. Information Technology Laboratory. Digital Signature Standard (DSS). NIST FIPS PUB 186-4. 2013 Jul. Available from: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
7. Elaine B Barker, Allen L Roginsky. Transitioning the Use of Cryptographic Algorithms and Key Lengths. NIST Special Publication 800-131A Revision 2. 2019 Mar.
8. Michael Cobb. Certificate Revocation List (CRL). TechTarget. 2016 May. Available from: https://searchsecurity.techtarget.com/definition/Certificate-Revocation-List
9. Stefan Santesson, Michael Myers, Rich Ankney. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP. Internet Standard, RFC 6960. 2013 Jun. Available from: https://tools.ietf.org/html/rfc6960
10. Qianpeng Wang, Deyun Gao, Du Chen. Certificate Revocation Schemes in Vehicular Networks: A Survey. IEEE Access 2020 Jan;8:26223–34. DOI: 10.1109/ACCESS.2020.2970460.
11. D. Eastlake 3rd. Transport Layer Security (TLS) Extensions: Extension Definitions. Internet Standard, RFC 6066. 2011 Jan. Available from: https://tools.ietf.org/html/rfc6066