LAMPS
Internet-Draft
Intended status: Informational
Expires: 6 December 2025

P. Liu, Ed.
Pengcheng Laboratory
R. Fu, Ed.
China Unicom
R. Chen, Ed.
China Mobile
X. Liu, Ed.
Y. Zhang, Ed.
Pengcheng Laboratory

4 June 2025

# Technical guidelines of Web server certification path validation for Interent browser
draft-liu-lamps-certification-path-validation-11

## Abstract

In Web PKI, certificate path construction and validation are crucial security review processes. At present, there are many Internet browser manufacturers around the world. With the change of security practices and the development of technology, the certificate validation process in Internet browser industry is relatively messy, including various private implementations of manufacturers. It is a typical patching improvement method, and the implementation of process functions is relatively arbitrary. This document provides a technical guide for certification path validation of web server certificates during Internet SSL/TLS protocol communication for Web PKI, including the basic path verification process, as well as reference procedures, guidance and suggestions for input, initialization, and basic certificate processing in the verification process. This document is applicable to the development and application of Internet browsers, as well as other similar digital certificate authentication systems requiring SSL/TLS security protocol secure communication.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 6 December 2025.



Table of Contents

## 1. Introduction

For the relying parties of Web PKI, certificate path construction and certificate validation are necessary security review processes. With regard to the implementation of certificating validation process for Internet browsers, the mainstream Internet browser implementation generally follows "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" [RFC5280] standard formulated in 2008. This version of the standard has a long history, is not in line with existing practice of Interent browser and not entirely suitable for Web PKI. With the development of technology and new features, such as the invention and standardization of certificate transparency system CT [RFC6962] and online certificate status protocol OCSP [RFC6960], browser manufacturers have not fully followed or enabled them. These new inspection features are very helpful for building practical certificate security and are crucial for secure SSL/TLS authentication. In addition, considering the needs of manufacturers, the implementation of Internet browsers inevitably includes various private code implementations, and the certificate validation process in the Internet browser industry is relatively messy and arbitrary. In view of this situation, this document proposes a unified reference procedure for the path verification process of SSL/TLS digital certificates in Internet browsers in combination with RFC5280 and the current latest practice, so as to provide reference for Internet browser manufacturers.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Web server certification path validation for Interent browser

This section provides technical guidelines for certification path validation of Web server certificates during SSL/TLS protocol communication, including basic path verification process, as well as reference procedures, guidance and suggestions for input, initialization, basic certificate processing, etc. in the verification process.

Certification path validation procedures for the Internet PKI are based on the algorithm supplied in [X.509]. Certification path processing verifies the binding between the subject distinguished name and/or subject alternative name and subject public key. The binding is limited by constraints that are specified in the certificates that comprise the path and inputs that are specified by the relying party. The basic constraints and policy constraints extensions allow the certification path processing logic to automate the decision making process.

This section describes an algorithm for validating certification paths. Conforming implementations of this specification are not required to implement this algorithm, but MUST provide functionality equivalent to the external behavior resulting from this procedure. Any algorithm may be used by a particular implementation so long as it derives the correct result.

In Section 3.1, the text describes basic path validation. Valid paths begin with certificates issued by a trust anchor. The algorithm requires the public key of the CA, the CA’ s name, and any constraints upon the set of paths that may be validated using this key.

The selection of a trust anchor is a matter of policy: it could be the top CA in a hierarchical PKI, the CA that issued the verifier’s own certificate(s), or any other CA in a network PKI. The path validation procedure is the same regardless of the choice of trust anchor. In addition, different applications may rely on different trust anchors, or may accept paths that begin with any of a set of trust anchors.

Section 3.2 describes methods for using the path validation algorithm in specific implementations.

Section 3.3 describes the steps necessary to determine if a certificate is revoked when CRLs are the revocation mechanism used by the certificate issuer.

Section 3.4 describes the steps required by the relying party to determine whether a certificate has been revoked when OCSP is the revocation mechanism used by the certificate issuer.

Section 3.5 describes the steps required for the relying party to determine the legitimacy and authority of a certificate through the CT mechanism.

### 3.1. Basic Path Validation

This section describes an algorithm for X.509 path processing. A conforming implementation MUST include an X.509 path processing procedure that is functionally equivalent to the external behavior of this algorithm. However, support for some of the certificate extensions processed in this algorithm are OPTIONAL for compliant implementations. Clients that do not support these extensions MAY omit the corresponding steps in the path validation algorithm.

For example, clients are not required to support the policy mappings extension. Clients that do not support this extension MAY omit the path validation steps where policy mappings are processed. Note that clients MUST reject the certificate if it contains an unsupported critical extension.

The algorithm presented in this section validates the certificate with respect to the current date and time. A conforming implementation MAY also support validation with respect to some point in the past. Note that mechanisms are not available for validating a certificate with respect to a time outside the certificate validity period.

The trust anchor is an input to the algorithm. There is no requirement that the same trust anchor be used to validate all certification paths. Different trust anchors MAY be used to validate different paths, as discussed further in Section 3.2.

The primary goal of path validation is to verify the binding between a subject distinguished name or a subject alternative name and subject public key, as represented in the target certificate, based on the public key of the trust anchor. In most cases, the target certificate will be an end entity certificate, but the target certificate may be a CA certificate as long as the subject public key is to be used for a purpose other than verifying the signature on a public key certificate. Verifying the binding between the name and subject public key requires obtaining a sequence of certificates that support that binding. The procedure performed to obtain this sequence of certificates is outside the scope of this specification.

To meet this goal, the path validation process verifies, among other things, that a prospective certification path (a sequence of n certificates) satisfies the following conditions:

1. for all x in {1, ..., n-1}, the subject of certificate x is of certificate x+1;

2. certificate 1 is issued by the trust anchor;

3. certificate n is the certificate to be validated (i.e., the target certificate); and

4. for all x in {1, ..., n}, the certificate was valid at the time in question.

A certificate MUST NOT appear more than once in a prospective certification path.

When the trust anchor is provided in the form of a self-signed certificate, this self-signed certificate is not included as part of the prospective certification path. Information about trust anchors is provided as inputs to the certification path validation algorithm.

A particular certification path may not, however, be appropriate for all applications. Therefore, an application MAY augment this algorithm to further limit the set of valid paths. The path validation process also determines the set of certificate policies that are valid for this path, based on the certificate policies extension, policy mappings extension, policy constraints extension, and inhibit anyPolicy extension. To achieve this, the path validation algorithm constructs a valid policy tree. If the set of certificate policies that are valid for this path is not empty, then the result will be a valid policy tree of depth n, otherwise the result will be a null valid policy tree.

A certificate is self-issued if the same DN appears in the subject and issuer fields (the two DNs are the same if they match according to the rules specified in RFC5280). In general, the issuer and subject of the certificates that make up a path are different for each certificate. However, a CA may issue a certificate to itself to support key rollover or changes in certificate policies. These self-issued certificates are not counted when evaluating path length or name constraints.

This section presents the algorithm in four basic steps: (1) initialization, (2) basic certificate processing, (3) preparation for the next certificate, and (4) wrap-up. Steps (1) and (4) are performed exactly once. Step (2) is performed for all certificates in the path. Step (3) is performed for all certificates in the path except the final certificate. Figure 1 provides a high-level flowchart of this algorithm.

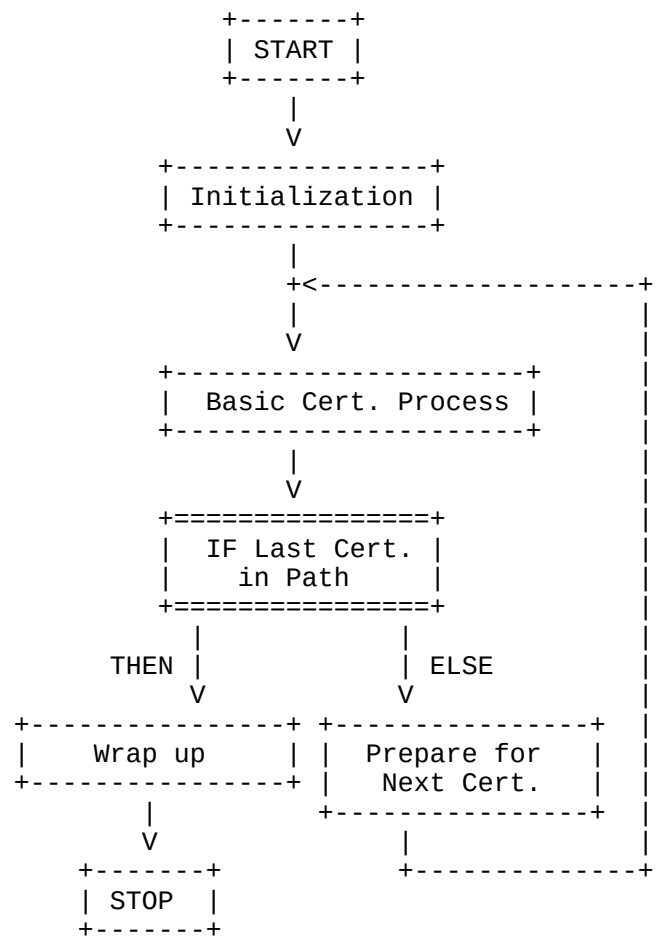


Figure 1: Certification Path Processing Flowchart

### 3.1.1. Inputs

This section references the Section "6.1.1. Inputs" of RFC5280.

### 3.1.2. Initialization

This section references the Section "6.1.2. Initialization" of RFC5280.

### 3.1.3. Basic Certificate Processing

This section references the Section "6.1.3. Basic Certificate Processing" of RFC5280, but with the following conditions added or modified when verifying the basic certificate information:

(1) The signature on the certificate can be verified using working_public_key_algorithm, the working_public_key, and the working_public_key_parameters.

(2) The Certificate MUST be of type X.509 v3, serialNumber MUST be a non-sequential number greater containing at least 64 bits of output from a CSPRNG.

(3) The certificate validity period includes the current time. The validity period in the certificate is within a reasonable range. Expired certificates may be maliciously exploited, and website certificates may not be effective yet. It is necessary to verify the validity field information of the certificate to see if the current date is within the validity period of the certificate. In addition, according to the Baseline Requirements for the Issuance and Management of Publicly-Trusted TLS Server Certificates after version V2.0.8, SSL/TLS Certificates issued before 15 March 2026 SHOULD NOT have a Validity Period greater than 397 days and MUST NOT have a Validity Period greater than 398 days; SSL/TLS Certificates issued on or after 15 March 2026 and before 15 March 2027 SHOULD NOT have a Validity Period greater than 199 days and MUST NOT have a Validity Period greater than 200 days; SSL/TLS Certificates issued on or after 15 March 2027 and before 15 March 2029 SHOULD NOT have a Validity Period greater than 99 days and MUST NOT have a Validity Period greater than 100 days; SSL/TLS Certificates issued on or after 15 March 2029 SHOULD NOT have a Validity Period greater than 46 days and MUST NOT have a Validity Period greater than 47 days. For the purpose of calculations, a day is measured as 86,400 seconds. Any amount of time greater than this, including fractional seconds and/or leap seconds, shall represent an additional day. For this reason, SSL/TLS Certificates SHOULD NOT be issued for the maximum permissible time by default, in order to account for such adjustments. Certificates that do not meet this requirement are considered to have a risk of validity period setting, and the validity field information of the certificate needs to be checked to verify whether it meets the validity period setting requirements.

(4) At the current time, the certificate is not revoked and is valid. The certificate status can be determined by obtaining appropriate protocol status information such as CRL (Section 3.3), OCSP (Section 3.4), and checked by using out of band mechanisms, such as CT (Section 3.5). If the revocation certificate status is queried

through both CRL and OCSP, but the two query results are inconsistent, the worst result status shall prevail. That is, if one of the two results is revoked, it is considered that the certificate has been revoked.

(5) The certificate issuer name is the working_issuer_name. For each Certificate in the Certification Path, the encoded content of the Issuer Distinguished Name field of a Certificate SHALL be byte-for-byte identical with the encoded form of the Subject Distinguished Name field of the Issuing CA certificate. For each CA Certificate in the Certification Path, the encoded content of the Subject Distinguished Name field of a Certificate SHALL be byte-for-byte identical among all Certificates whose Subject Distinguished Names can be compared as equal, and including expired and revoked Certificates.

(6) The signature algorithm of the certificate is valid, that is, the signatureAlgorithm field of the certificate must be consistent with the signature field in TBSCertificate.

(7) The certificate does not use insecure weak keys, such as RSA, ECDSA, and SM2 public key cryptography algorithms. This document recommends that the RSA algorithm has a key length of at least 2048 bits, the modulus size, in bits, is evenly divisible by 8, the value of the public exponent is an odd number equal to 3 or more. Additionally, the public exponent SHOULD be in the range between 2^16 + 1 and 2^256 - 1. The modulus SHOULD also have the following characteristics: an odd number, not the power of a prime, and have no factors smaller than 752. The ECDSA algorithm recommends elliptic curves such as NIST P-256, NIST P-384, and NIST P-521, and the ISO/IEC 14888-3/AMD1 SM2 algorithm has a key length of at least 256 bits.

(8) The certificate format should comply with the IETF RFC5280 standard format for X.509 certificates and meet the relevant requirements for server certificates in the CA/Browser Forum certificate issuance and management baseline requirements.

(9) The website domain name included in the certificate is consistent with the website domain name.

(10) The strength of the hash function used in the signature meets the requirements: it is recommended to use SHA256, SHA384, SHA512, or SM3 hash algorithms, and check whether the hash algorithm of the signatureAlgorithm field is the recommended hash algorithm.

(11) The domain name in the certificate is configured with CAA records: The domain owner can only issue the certificate by configuring CAA record values and specifying allowed trusted CAs.

During the certificate issuance process, the CA will complete the certificate issuance process only after successful CAA verification. The CAA record information should be checked to see if there are valid CAA records.

(12) Check if the website hostname (host_name or ech_name, if ECH is enabled) is included in the Subject AltName list of the certificate. If not included, the certificate name is considered invalid.

(13) The deployment configuration environment of certificates is secure: check whether insecure versions of protocols and cipher suites are used, and whether the server-side supports and enables HSTS. Using insecure protocols and cipher suites can threaten private key security (for example, using certificates that support DH public key parameter reuse can pose a risk of Heartbleed site vulnerabilities). TLS protocols include SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2, and TLS 1.3; The security of certificate deployment configuration should be determined by checking the TLS version and cipher suite. Regarding the judgment of TLS protocol, it is recommended to consider SSL 2.0 and SSL 3.0 as insecure and using algorithms with known security vulnerabilities, both of which have been abandoned; TLS 1.0 and TLS 1.1 are Weak and risky, officially abandoned in 2021; TLS 1.2 and TLS 1.3 are generally considered secure. The cipher suite used is secure: If any of the following situations occur in the cipher suite, it is considered a risk: RSA, DH, ECDH do not have forward encryption and have key recovery attack vulnerabilities; DSA is a weak asymmetric encryption algorithm; RC4 stream cipher is abandoned; CBC mode encryption is vulnerable to BEAST and Lucky 13 vulnerabilities; There is a risk of cracking the SHA1 algorithm; The cipher suites using anonymous authentication and null encryption algorithm is also considered insecure; According to the latest practice, the cipher suites listed in Appendix A are considered secure and it is recommended to use these cipher suites.

(14) The result of the HSTS parameter security check is secure: HSTS is an Internet security policy mechanism published by IETF. Websites that adopt HSTS policy will ensure that the browser is always connected to the HTTPS access of the website. The way for the server to enable HSTS is to include the Strict-Transport-Security field in the HTTP response header returned by the server when the client sends a request through HTTPS. The content format is: Strict-Transport-Security: max-age=expireTime [; includeSubDomains] [; preload]; If max-age is less than 10368000 seconds (120 days), the configuration is considered risky; If max-age is greater than 10368000 seconds but less than 31536000 seconds(365 days), the configuration is considered secure. If max-age is greater than 31536000 seconds, the configuration is considered very secure; and includeSubDomains is an optional parameter, if this parameter is specified, all subdomains of

this website must be accessed through the HTTPS protocol; Preload is an optional parameter, indicating that HTTPS protocol will also be enforced for the first visit.

#### 3.1.4. Preparation for Certificate i+1

This section references the Section "6.1.4. Preparation for Certificate i+1" of RFC5280.

#### 3.1.5. Wrap-Up Procedure

This section references the Section "6.1.5. Wrap-Up Procedure" of RFC5280.

#### 3.1.6. Outputs

This section references the Section "6.1.6. Outputs" of RFC5280.

### 3.2. Using the Path Validation Algorithm

This section references the Section "6.2. Using the Path Validation Algorithm" of RFC5280.

### 3.3. CRL Validation

This section references the Section "6.3. CRL Validation" of RFC5280.

### 3.4. OCSP Validation

In this section, this document describes the steps required to determine whether a certificate has been revoked when OCSP is the certificate status query mechanism used by certificate issuers. The OCSP protocol is an online query service used to verify the validity of certificates, typically provided by the CA to which the certificate belongs. Some clients will query the OCSP interface in real-time during further negotiation during the SSL/TLS handshake phase and block subsequent processes until the results are obtained. OCSP query is essentially a complete HTTP request/response operation. OCSP Stapling refers to the server actively obtaining the OCSP query results and sending them to the client along with the certificate, allowing the client to skip the verification process and improve the efficiency of SSL/TLS handshake. For details, please refer to RFC6066 and RFC6961 standards. OCSP protocol is mainly used to replace CRL in public key infrastructure (PKI) to query the status of digital certificates. When a user attempts to access a Web server through an Internet browser, the Internet browser sends a request for certificate status information through the online certificate status

protocol. The OCSP server responds with a "valid," "revoked," or "unknown. OCSP is a relatively simple request/response protocol implemented in C/S mode, without specifying the transmission mechanism used by the protocol or the structure of the OCSP system. The specific mechanism can be found in standard RFC6960, which is not limited in this document. This document assumes that the Internet browser client has been configured with OCSP server, such as signature key pair, server address, etc. The implementation of the consistency function supporting OCSP for specific scenarios does not require a one-to-one implementation of this algorithm, but when processing OCSP parameters, it must be functionally equivalent to the external behavior generated by the procedures described in this document. Any algorithm can be used by a specific implementation as long as it produces the correct results. The algorithm described in this section defines a set of inputs and processing steps for each certificate in the authentication path. The algorithm output is the revocation status of the certificate.

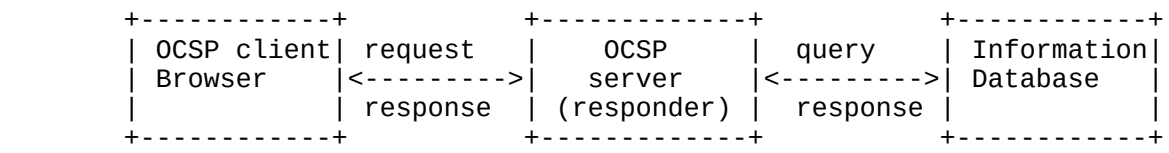


Figure 2: OCSP Request/Response Service Mechanism

### 3.4.1. OCSP Inputs

To support OCSP revocation status query operations, the algorithm requires three inputs:

1. certificate: This algorithm requires the Hash algorithm identifier, the Hash value of the certificate issuer (CA) name (DN), the Hash value of the certificate issuer (CA) public key, and the certificate serial number to determine whether the certificate is valid.

2. use-signature: This input indicates whether to add a signature to the OCSP request, in order for the OCSP responder (server) to authenticate the requester’s identity.

3. use-specified-server: This input indicates whether to include the specified response server URI in the OCSP request.

### 3.4.2. Initialization and Revocation State Variables

To support OCSP processing, the algorithm requires the following state variable:

1. cert_status: This variable contains the status of the certificate. This variable can be assigned one of the following values: REVOKED, UNREVOKED, or UNDETERMINED. This variable is initialized to the special value UNEVOKED.

### 3.4.3. OCSP Processing

Before processing the revocation status query, the algorithm assumes that the target certificate has not been revoked, that is, the initial value of the variable "cert_status" is UNREVOKED. The algorithm performs the following operations:

1. Analyze and extract the statis_request message extension field of the SSL/TLS protocol to obtain the ocsp response sent by the Web server in a Staking manner.

2. If the Web server does not send OCSP related message extensions, that is, does not support OCSP Stapling function, then construct an OCSP request OCSPRequest and send it to the OCSP server. The request contains the target certificate CertID (certificate identifier) for querying the certificate, which includes the hash algorithm identifier, the hash value of the certificate issuer (CA) name (DN), the hash value of the certificate issuer (CA) public key, and the certificate serial number; Determine whether to use the specified OCSP server address based on input parameter use-specified-server and attach it to the ServiceLocator field in the singleRequestExtensions extension of the request; And based on the input parameter use-signature, decide whether to use the request signature to sign the message.

3. After receiving the signed response OCSPResponse from the OCSP server, whether through online queries or stapling, the OCSP response may be a definite response or an error message indicating an abnormal situation. For each definite response, the responder must sign, and there is no need to sign for error messages. The confirmed response includes the protocol version number, responder name, reply to each certificate to be queried, optional extensions, signature algorithm object identifier, and signature value. The following verifications need to be performed. If all verifications pass, the algorithm accepts this response; Otherwise, this response is considered invalid.

(1) Is the certificate identified by the CertID in the received response consistent with the request; (2) Is the Signature signature in BasicOCSPResponse valid; (3) Whether the response signer is trusted by the client (whether the response certificate is in the client’s trust list); (4) Is the response generation time in the response the current time.

Only when the value of responsiveStatus is "successful", the value of responsiveBytes in the OCSPResponse response will be set. There are six values for responsiveStatus response status: "successful" indicates that the response is valid and the query is successful; "malformedRequest" indicates that the received query request format is inconsistent with the OCSP standard syntax; "internalError" indicates that the OCSP responder is currently in an uncoordinated internal state, requiring the client to attempt the request again or try other responders; "trylater" is used to indicate that the OCSP service still exists but is busy and temporarily unable to respond to client requests; "SigRequired" means that the OCSP server requires the current client to attach signature data when sending requests; "Unauthorized" means that the current client does not have permission to query the OCSP responder. If the response status is a certain error situation, the response byte will not be set, and the client needs to determine the revocation status of the certificate based on the returned status:

If the response field CertStatus is good, it indicates a positive response from the responder to the status query. At a minimum, this positive response means that the certificate with the requested certificate serial number has not been revoked within the current validity period. Set the variable "cert_status" to UNREVOKED; If the response field CertStatus is revoked, it indicates that the certificate has been revoked. Set the variable "cert_status" to REVOKED and resolve the reason for revocation based on the CRLReason field in the RevokedInfo field (if this extension field exists) (the definition of revocation reason values can be found in Section 5.5); If the response field variable CertStatus is unknown, it means that the responder cannot determine the certificate status of the request. Set the variable "cert_status" to UNDERTMINED.

### 3.5. CT Validation

In this section, this document describes the steps required for the relying party to determine the legitimacy and authority of a certificate through the certificate transparency CT mechanism. Certificate transparency is a mechanism used to monitor and audit the behavior of SSL/TLS certificate authorities (CAs), by recording and publicly issuing all SSL/TLS certificates, allowing third parties to verify the legitimacy and authority of certificates, preventing certificate abuse and fraud. For the signature certificate timestamp SCT in the CT mechanism, the Internet browser client can obtain it in three ways: one is directly extended through the X509v3 certificate; After submitting the pre signed certificate to the log server, the certificate authority (CA) will return the SCT, and the CA will attach the SCT as an X.509v3 extension to the signed Web server certificate; Another extension of the OCSP Stapling protocol during SSL/TLS negotiation; The third way is to directly extend the SSL/TLS protocol; The specific mechanism can be found in standard RFC6962, which is not limited in this document. This document assumes that the Internet browser client has the relevant configuration of the log server, such as the signature key pair, server address, and so on. The implementation of consistency functionality for CT support in specific scenarios does not require a one-to-one implementation of this algorithm, but when processing CT parameters, the implementation of various algorithms must be functionally equivalent to the external behavior generated by the procedures described in this document. Any algorithm can be used by a specific implementation as long as it produces the correct results. The algorithm described in this section defines a set of inputs and processing steps for each certificate in the authentication path. The algorithm output is the validity status of the certificate.


Figure 3: CT Request/Response Service Mechanism

#### 3.5.1. CT Inputs

To support CT status query operations, the algorithm requires three inputs:

1. certificate: This algorithm requires the SCT list in the certificate to determine if the certificate is valid.

2. OCSP_response: This algorithm requires the SCT list in OCSP Stapling to determine if the certificate is valid.

3. ServerHello: The ServerHello message on the server-side in SSL/TLS protocol, this algorithm requires the SCT list in the SSL/TLS protocol extension to determine whether the certificate is valid.

### 3.5.2. Initialization and Revocation State Variables

To support CT processing, the algorithm requires the following state variable:

1. cert_status: This variable contains the status of the certificate. This variable can be assigned one of the following values: SCT_STATUS_OK, SCT_STATUS_INVALID, and SCT_STATUS_UNNOWN. This variable is initialized to the special value SCT_STATUS_OK.

### 3.5.3. CT Processing

Before querying the validity status of the certificate, the algorithm assumes that the target certificate is valid, that is, the initial value of the variable "cert_status" is SCT_STATUS_OK. The algorithm performs the following operations:

1. Analyze and extract the X509v3 extension field SignedCertificateTimestampList from the certificate to obtain the SCT list related to the certificate; Analyze and extract the OCSP_response extension in SSL/TLS protocol, and further extract the extension field SignedCertificateTimestampList contained in it to obtain the certificate related SCT list; Analyze and extract the SSL/TLS protocol extension field signed_certificate_timestamp from the ServerHello message sent by the server during handshake negotiation to obtain the certificate related SCT list.

2. As long as any of the above SCT list fields exist, parse and validate the SCT list; For each SCT record item, verify its record signature. If the signature is incorrect, the certificate is considered invalid, and set the variable "cert_status" to’ SCT_STATUS_INVALID ’; Verify its timestamp,Verify the CT signature information contained in the certificate, requiring that the certificate must contain 2 SCT data for validity periods less than 180 days and 3 SCT data for validity periods greater than 180 days, to ensure sufficient transparency in certificate issuance; Determine the validity period of the certificate

through validity, and then verify whether the number of SCTs meets the requirements through SCT field information. If the timestamp is not reasonable and has not yet taken effect, it is considered that the certificate is invalid. Set the variable "cert_status" to’ SCT_STATUS_INVALID ’; Otherwise, it is considered that the certificate is valid, and the variable "cert_status" is set to’ SCT_STATUS_OK ’; If the relevant log server cannot be found locally, set the variable "cert_status" to’ SCT_STATUS_UNNOWN ’.

## 4. Appendix A

According to the latest practice, the cipher suites listed in below table are considered secure. The following Abbreviations in table are shown as below.

1. CSN: Cipher Suite Name;
2. MTLS: Minimum TLS protocol version supported;
3. KEA: Key exchange algorithm;
4. IAA: Identity authentication algorithm;
5. DEA: Data encryption algorithm;
6. MDA: Message digest algorithm;

| CSN | MTLS | KEA | IAA | DEA | MDA |
|---|---|---|---|---|---|
| TLS_AES_256_GCM_SHA384 | TLSv1.3 | Kx=any | Au=any | Enc= AESGCM(256) | Mac=AEAD |
| TLS_CHACHA20_POLY1305_SHA256 | TLSv1.3 | Kx=any | Au=any | Enc=CHACHA20 /POLY1305(256) | Mac=AEAD |
| TLS_AES_128_GCM_SHA256 | TLSv1.3 | Kx=any | Au=any | Enc= AESGCM(128) | Mac=AEAD |
| TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 | TLSv1.2 | Kx= ECDH | Au= ECDSA | Enc= AESGCM(256) | Mac=AEAD |
| TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 | TLSv1.2 | Kx= ECDH | Au= RSA | Enc= AESGCM(256) | Mac=AEAD |
| TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 | TLSv1.2 | Kx=DH | Au= RSA | Enc= AESGCM(256) | Mac=AEAD |

| | | | | | |
|---|---|---|---|---|---|
| TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 | TLSv1.2 | Kx=ECDH | Au=DSA | Enc=CHACHA20/POLY1305(256) | Mac=AEAD |
| TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 | TLSv1.2 | Kx=ECDH | Au=RSA | Enc=CHACHA20/POLY1305(256) | Mac=AEAD |
| TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 | TLSv1.2 | Kx=DH | Au=RSA | Enc=CHACHA20/POLY1305(256) | Mac=AEAD |
| TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | TLSv1.2 | Kx=ECDH | Au=ECDSA | Enc=AESGCM(128) | Mac=AEAD |
| TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | TLSv1.2 | Kx=ECDH | Au=RSA | Enc=AESGCM(128) | Mac=AEAD |
| TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 | TLSv1.2 | Kx=DH | Au=RSA | Enc=AESGCM(128) | Mac=AEAD |
| TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 | TLSv1.2 | Kx=ECDH | Au=ECDSA | Enc=AES(256) | Mac=SHA384 |
| TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 | TLSv1.2 | Kx=ECDH | Au=RSA | Enc=AES(256) | Mac=SHA384 |
| TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 | TLSv1.2 | Kx=DH | Au=RSA | Enc=AES(256) | Mac=SHA256 |
| TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | TLSv1.2 | Kx=ECDH | Au=ECDSA | Enc=AES(128) | Mac=SHA256 |
| TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | TLSv1.2 | Kx=ECDH | Au=RSA | Enc=AES(128) | Mac=SHA256 |
| TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 | TLSv1.2 | Kx=DH | Au=RSA | Enc=AES(128) | Mac=SHA256 |
| TLS_RSA_PSK_WITH_AES_256_GCM_SHA384 | TLSv1.2 | Kx=RSAPSK | Au=RSA | Enc=AESGCM(256) | Mac=AEAD |
| TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 | TLSv1.2 | Kx=DHEPSK | Au=PSK | Enc=AESGCM(256) | Mac=AEAD |
| TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256 | TLSv1.2 | Kx=RSAPSK | Au=RSA | Enc=CHACHA20/POLY1305(256) | Mac=AEAD |
| TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256 | TLSv1.2 | Kx=DHEPSK | Au=PSK | Enc=CHACHA20/POLY1305(256) | Mac=AEAD |

| | | | | | |
|---|---|---|---|---|---|
| TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256 | TLSv1.2 | Kx=ECDHEPSK | Au=PSK | Enc=CHACHA20/POLY1305(256) | Mac=AEAD |
| TLS_RSA_WITH_AES_256_GCM_SHA384 | TLSv1.2 | Kx=RSA | Au=RSA | Enc=AESGCM(256) | Mac=AEAD |
| TLS_PSK_WITH_AES_256_GCM_SHA384 | TLSv1.2 | Kx=PSK | Au=PSK | Enc=AESGCM(256) | Mac=AEAD |
| TLS_PSK_WITH_CHACHA20_POLY1305_SHA256 | TLSv1.2 | Kx=PSK | Au=PSK | Enc=CHACHA20/POLY1305(256) | Mac=AEAD |
| TLS_RSA_PSK_WITH_AES_128_GCM_SHA256 | TLSv1.2 | Kx=RSAPSK | Au=RSA | Enc=AESGCM(128) | Mac=AEAD |
| TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 | TLSv1.2 | Kx=DHEPSK | Au=PSK | Enc=AESGCM(128) | Mac=AEAD |
| TLS_RSA_WITH_AES_128_GCM_SHA256 | TLSv1.2 | Kx=RSA | Au=RSA | Enc=AESGCM(128) | Mac=AEAD |
| TLS_PSK_WITH_AES_128_GCM_SHA256 | TLSv1.2 | Kx=PSK | Au=PSK | Enc=AESGCM(128) | Mac=AEAD |
| TLS_RSA_WITH_AES_256_CBC_SHA256 | TLSv1.2 | Kx=RSA | Au=RSA | Enc=AES(256) | Mac=SHA256 |
| TLS_RSA_WITH_AES_128_CBC_SHA256 | TLSv1.2 | Kx=RSA | Au=RSA | Enc=AES(128) | Mac=SHA256 |

## 5. IANA Considerations

This memo includes no request to IANA.

## 6. Security Considerations

For the relying parties of Web PKI, certificate path construction and certificate validation are necessary security review processes.The quality of implementations that process certificates also affects the degree of assurance provided. The path validation algorithm described in Section 6 relies upon the integrity of the trusted CA information, and especially the integrity of the status information associated with the certificate. By providing multiple checking mechanisms, an attacker cannot trick the user into accepting false certificates.this document proposes some updates of the latest

reference of certificate status information mechanism description for the RFC5280 in line with existing practice, to provide security reference for Internet browser manufacturers.

## 7. References

### 7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

### 7.2. Informative References

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/rfc/rfc6960>.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <https://www.rfc-editor.org/rfc/rfc6962>.

## Authors' Addresses

Penghui Liu (editor)
Pengcheng Laboratory
No.2 Xingke 1 Street
Shenzhen
518055
China
Email: liuph@pcl.ac.cn

Yu Fu (editor)
China Unicom
No.1 Zhonghe Street
Beijing
100000
China
Email: fuy186@chinaunicom.cn

Meiling Chen (editor)
China Mobile
No.32 Xuanwumen West Street
Beijing
100000
China
Email: chenmeiling@chinamobile.com

Xiang Liu (editor)
Pengcheng Laboratory
No.2 Xingke 1 Street
Shenzhen
518055
China
Email: liux15@pcl.ac.cn

Yu Zhang (editor)
Pengcheng Laboratory
No.2 Xingke 1 Street
Shenzhen
518055
China
Email: zhangy08@pcl.ac.cn