Chaos in the Chain: Evaluate Deployment and Construction Compliance of Web PKI Certificate Chain

Jia Yao
Tsinghua University
Beijing, China
yaojia3000@126.com

Yiming Zhang
Tsinghua University
Beijing, China
zhangyiming@tsinghua.edu.cn

Baojun Liu
Tsinghua University
Beijing, China
lbj@tsinghua.edu.cn

Zhan Liu
Tsinghua University
Beijing, China
liuz24@mails.tsinghua.edu.cn

Mingming Zhang
Zhongguancun Laboratory
Beijing, China
zhangmm@mail.zgclab.edu.cn

Haixin Duan
Tsinghua University
Beijing, China
duanhx@tsinghua.edu.cn

Abstract

Transport Layer Security (TLS) is a cornerstone to secure Internet communications. It requires proper deployment and validation of certificate chains. During validation, clients must first construct the chain from server-provided certificates. However, existing research often integrates chain construction into the broader validation process, lacking independent analysis of this crucial step.

This paper presents the first systematic assessment of certificate chain construction, covering server-side deployment compliance and client-side capabilities. On the server side, we summarized structural requirements from RFC standards and evaluated real-world website compliance. We found that approximately 3% of Tranco Top 1M domains have deployed non-compliant chains, with common issues including reversed sequences and incomplete chains. The compliance would be influenced by HTTP server and Certificate Authority checks and guidance during the configuration process. On the client side, we evaluated 9 types of chain-building capabilities across 8 mainstream TLS implementations, uncovering prevalent deficiencies like inadequate backtracking and difficulties with long chains. These deficiencies could compromise TLS security, causing a fallback to insecure HTTP or making the service unavailable. Our findings highlight critical gaps in current certificate chain practices. Based on our findings, we also propose recommendations for improving the deployment and construction of certificate chains.

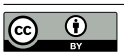
CCS Concepts

• **Security and privacy → Web protocol security**; • **Networks → Network measurement**.

Keywords

PKI; Certificate Chain; Web; Security

ACM Reference Format:
Jia Yao, Yiming Zhang, Baojun Liu, Zhan Liu, Mingming Zhang, and Haixin Duan. 2025. Chaos in the Chain: Evaluate Deployment and Construction Compliance of Web PKI Certificate Chain. In *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25), October 28–31, 2025, Madison, WI, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3730567.3732921



ACM ISBN 979-8-4007-1860-1/2025/10
https://doi.org/10.1145/3730567.3732921

1 Introduction

Transport Layer Security (TLS) and Public Key Infrastructure (PKI) are essential for secure and authenticated communication over the Internet. Within TLS interactions, authenticating the peer entity during the handshake is of great importance. Specifically, in the "Server Hello" message, the server provides X.509 certificates to the client, which the client then validates according to RFC standards [11]. The server is trusted only if the certificate chain is valid.

Due to the critical nature of certificate chain verification, extensive studies have examined the (HTTPS) deployment of websites [8, 29, 42, 43] and validation issues on TLS clients [29, 38]. However, an important detail has been overlooked: the construction of the certificate chain. A structurally compliant chain should include a leaf certificate, one or more intermediate certificates, and a root certificate in the order of issuance. In practice, servers only provide a list of certificates, leaving the client responsible for constructing the certificate path before validation. Improper chain construction can cause security issues. For example, in 2020, the expiration of the `AddTrust External CA Root` certificate caused many clients to fail to identify a valid certificate path, leading to the unavailability of numerous websites[39]. Besides, the CVE-2024-0567 vulnerability identified that servers using GnuTLS, like Cockpit, could experience disruptions when verifying client certificates that included cyclic cross-signed certificates, potentially resulting in a DoS attack [6].

Research gap. The RFC standards have acknowledged concerns about certificate chain deployment and construction. TLS 1.2 [14] imposes "structural" requirements for server-deployed certificate chains, e.g., each certificate in the chain MUST directly authenticate the preceding certificate. So far, no scalable measurements have been conducted to evaluate real-world server-side compliance with these requirements. Furthermore, TLS 1.3 [36] states that non-compliant server deployments are prevalent, and requires the clients to handle disorganized certificate chains. However, path construction is inherently complex for clients, especially without clear or standardized normative guidance. While RFC 4158 [12] (informational document, not standard) offers suggestions for chain construction, these suggestions are not mandatory. To date, there

has been no systematic study of how TLS clients construct chains in practice.

Research questions. In this paper, we conduct the first study on the security implications of certificate chain construction issues, focusing on server deployments and client-side certificate chain construction. The research questions we aim to answer include: (RQ1) How structurally compliant are certificate chains provided by web servers in the wild? and (RQ2) Do mainstream TLS clients have the capability of handling malformed (non-compliant) certificate chains, and what are the security impacts? We also expect to provide suggestions on the practices for providing and constructing certificate chains.

Our work. For server-side deployment evaluation, we first summarized the "structural" requirements of certificate chains from RFC standards and developed rules for compliance analysis. We collected certificate chains from Tranco Top 1M domains [34, 41]. Our analysis revealed widespread non-compliance, affecting 2.9% of Tranco Top domains. Primary issues included improper issuance orders and incomplete chains (see Section 4). While these issues may stem from configuration errors by network administrators, we found that the checks and guidance provided by HTTP servers and Certificate Authorities (CAs) during the configuration process also matter. Automated checks could mitigate non-compliance (e.g., HTTP servers like Azure check duplicate leaf certificates), yet messy guidelines further complicate configuration (e.g., CAs or resellers like GoGetSSL provide users with certificates in reverse order).

For client-side evaluation, we first empirically analyzed the chain-building logic of open-source TLS implementations (e.g., Chromium), and summarized 9 common chain-building capabilities (see Table 2). We then designed targeted sample certificate chains to test each capability and evaluated 8 mainstream TLS clients (4 browsers and 4 libraries). Our evaluation revealed that, except for CryptoAPI, libraries typically underperform browsers (see Table 9). Key deficiencies include missing basic capabilities such as AIA support, and the lack of prioritization features. Furthermore, through differential testing, we demonstrated that real-world non-compliant chains would fail validation due to deficiencies in client-side chain construction, impacting service availability (see Section 5). Based on these findings, we propose recommendations to improve the security and reliability of certificate chain deployment and construction (see Section 6).

Contributions. Our main contributions include:

- We conducted the first large-scale analysis from the perspective of structural compliance in server-side certificate chain deployments.
- We evaluated the certificate chain construction capabilities in mainstream TLS implementations and found significant discrepancies and deficiencies.
- We provide recommendations to improve the compliance and security of certificate chains.

2 Background

Public Key Infrastructure is a framework used to secure communications between entities through digital certificates. In this paper, we focus on Web PKI, where clients validate X.509 certificates to authenticate website entities during HTTPS connections. This section introduces the fundamental concepts of X.509 certificates and certificate chains, followed by an overview of existing research in this domain.

2.1 Certificate Chain Validation

Web PKI mechanism. In the trust model of Web PKI, website owners need to apply for certificates from publicly trusted Certificate Authorities (CAs), which would issue certificates in X.509 formats [11] after verifying the applicant's identity (i.e., domain ownership). For security and flexibility, CAs typically first use their root certificates to issue intermediate certificates and then use the intermediate ones to issue the entity (leaf) certificate. The leaf certificate, along with (one or more) intermediate certificates and the root certificate, forms the certificate chain. During the TLS handshake, the server would typically send the certificate chain to the client via `Certificate` Message [14]. Then, the client performs a series of checks on the chain's validity, e.g., the presence of the root cert in trusted lists, expiration or revocation status, and hostname matching, to ensure the trustworthiness of the server and the security of the TLS connection.

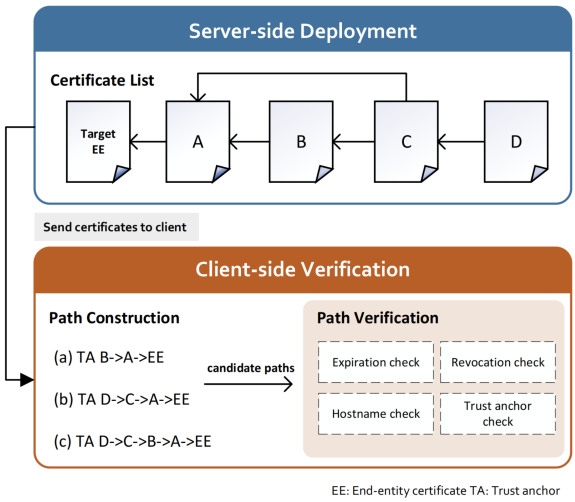


Figure 1: Certification path processing

X.509 certificate. Standardized in RFC 5280 [11], an X.509 version 3 certificate includes basic fields such as the issuer, subject, validity and a set of extensions. Key extensions that are relevant to chain construction include: Subject Alternative Name (SAN), which specifies additional identities such as domain names and IP addresses; the Subject Key Identifier (SKID), which uniquely identifies the public key in the certificate and facilitates path construction; the Authority Key Identifier (AKID), which identifies the public key used to sign the certificate, typically by referencing the SKID of the issuing CA or the issuer's name and serial number; the Authority Information Access (AIA), which provides access information for obtaining the issuer's certificate or checking certificate status; and the Basic Constraints, which indicate whether the certificate is issued to a Certificate Authority (CA) and define the maximum certification path length.

Certificate Chain. Notably, in practice, the `Certificate` message just provides a *list* of certificates, which has not been guaranteed to be a well-structured *chain* (i.e., from leaf to root in the exact order of issuance). Although TLS 1.2 [14] requires the server to place the entity certificate first, followed by certificates in their issuance order (with the root certificate optionally omitted), real-world deployments may still provide "messy" chains due to misconfiguration [8, 42, 43]. Therefore, chain building, i.e., constructing a chain of certificates to be verified based on the certificates provided by the server, has become an inescapable task for TLS clients. TLS 1.3 [36] explicitly specifies that all TLS implementations SHOULD be prepared to handle potentially extraneous certificates and arbitrary orderings. In this paper, we use *certificate list* to refer to the list of certificates provided by the server via `Certificate` message, and *certificate path* to denote the chain constructed from this list by the client during the validation process.

Certificate validation. As discussed above, TLS clients cannot directly validate the certificates provided by the server. They must first construct a certificate path—a list of certificates starting with the leaf and ending with the root, and all certificates follow the issuance order. Therefore, the client-side certificate validation can be considered to include two steps (as shown in Figure 1):

(1) **Path construction** involves assembling candidate certification paths, where "candidate" highlights that while the certificates may be correctly sequenced, the path may still not meet other criteria, such as path length, name constraints, or certificate policies.
(2) **Path validation** ensures that each certificate in the path meets certain criteria: it must be within its validity period, not revoked, and comply with any applicable constraints.

Currently, there is no unified standard for how TLS clients should construct a certificate chain or for the relationship between chain construction and validation. RFC 4158 [12], as an informational document rather than a formal standard, provides recommendations for chain construction, but the specific implementation is left to the client. For instance, to expedite finding a valid certificate path, a client might validate certain attributes (e.g., expiration date) during the construction process.

2.2 Related Work

Security of server certificate chain deployment. In recent years, numerous studies have been conducted to measure the Web PKI ecosystem. Holz et al. [23] were among the first to conduct a large-scale measurement and analysis of the HTTPS certificate ecosystem, utilizing OpenSSL's verify module to validate collected certificate chains and analyze various errors. Subsequent research focused on building large-scale certificate datasets [16, 25], detecting HTTPS hijacking [17, 24], and other explorations. Measures to enhance the security of Web PKI certificates, such as certificate revocation mechanisms [27, 30] and Certificate Transparency [21, 37, 40], have also been extensively measured and analyzed. Kumar et al. [28] assessed the compliance of individual certificate contents, identifying non-compliant practices during issuance by certification authorities. Despite extensive research on the security of Web PKI certificates, these studies have either focused on individual certificate content or supplementary mechanisms, or they have "assumed" that the server-provided chains are compliant and validated them accordingly. There has never been a systematic assessment of whether the server-provided chains meet the structural compliance requirements of TLS protocols.

Certificate verification issues. Security concerns during the certificate chain verification process have always been a hot topic in PKI. Brubaker et al. [9] generated "frankencerts" through random mutations to automate differential testing of certificate verification issues in browsers and libraries. Sivakorn et al. [38] in their HVLearn project focused on the "hostname" field of individual certificates, using a black-box testing framework to identify non-compliant behaviors in TLS implementations that could be exploited for man-in-the-middle attacks. Non-browser software also has a history of studies on TLS certificate verification issues. In 2012, Georgiev et al. [19] manually identified prevalent certificate validation errors in critical software that relied on various SSL/TLS implementations. Pourali et al. [35] conducted a fine-grained attribution of TLS certificate verification issues in Android systems, uncovering certificate validation hijacking phenomena. While the aforementioned studies focus on testing potential problems clients might encounter during final certificate verification, and they do not explore potential issues that may arise during the certificate chain construction process.

Certificate chain construction. Only a few research papers directly address the process of certificate chain construction. In 2013, Durumeric et al. [16] conducted a large-scale measurement of the HTTPS ecosystem, mentioning the need to reorder and supplement missing certificates in the validation section, but did not provide specific technical details. Hiller et al. [22] developed a certificate chain construction tool to study the relationships of cross-signed certificates, aiming to traverse all viable certificate chains within passive datasets (e.g., Censys [15] or CT logs [21]), though details of the tool's implementation and considerations for individual link constructions were not provided. Zhang et al. [43] noted that server-deployed certificate chains could be disordered or incomplete and designed rules to locate root certificates within the chain, but did not analyze methods for certificate chain construction. Debnath et al. [13] attempted to create a certificate verification implementation fully compliant with RFC standards; their Chain-Builder Module identifies certificate issuance relationships using AKI.keyid and SKI.keyid and could construct multiple candidate chains. However, their approach lacks a detailed design for handling more complex scenarios, such as missing intermediate certificates, ordering multiple candidates, and whether to set attempt limits. Larisch et al. [29] also used KeyID matching and subject/issuer DN to construct certificate chains, incorporating AIA to locate new certificates, but did not consider complex scenarios either. To date, the only project that examines the chain construction capabilities of TLS implementations is BetterTLS[1] (2020). It evaluated whether clients reject invalid certificates (e.g., expired or violating name constraints) and select alternative valid chains. In other words, BetterTLS primarily targets "validation correctness" rather than "decision-making among multiple certificate paths". Our study investigates a more essential question: given multiple certificates, which one does the client prefer for chain construction (i.e., priority preferences). We

[1] https://bettertls.com/

also evaluate client behavior in more complex scenarios, such as reversed certificate order or missing intermediate certificates. A detailed comparison with BetterTLS is shown in Table 1. Overall, our work provides a more systematic investigation into the chain construction capabilities of TLS clients.

Table 1: Comparison of client certificate chain building capabilities between BetterTLS and this work

Type		BetterTLS	This Work
Basic Capabilities	ORDER_REORGANIZATION	×	✓
	REDUNDANCY_ELIMINATION	×	✓
	AIA_COMPLETION	×	✓
Priority Preferences	EXPIRED	✓	✓
	NAME_CONSTRAINTS	✓	×
	BAD_EKU	✓	×
	MISS_BASIC_CONSTRAINTS	✓	×
	NOT_A_CA	✓	×
	DEPRECATED_CRYPTO	✓	×
	BAD_PATH_LENGTH	×	✓
	BAD_KID	×	✓
	BAD_KU	×	✓
Restriction Settings	PATH_LENGTH_CONSTRAINT	×	✓
	SELF_SIGNED_LEAF_CERT	×	✓

3 Methodology

This section outlines our research methodology. On the server side, e conducted large-scale measurements to access the compliance of deployed certificate chains from three aspects: leaf certificate location, chain order, and chain integrity (based on RFC specifications [14, 36]). On the client side, we assessed the chain construction capabilities and their potential security implications of mainstream TLS implementations through heuristic construction of malformed certificate chains and semi-automated differential testing.

Terminology. The TLS 1.2 standard (RFC 5246, Section 7.4.2) [14] specifies three basic requirements for certificate chain deployment: (1) The sender's certificate MUST come first in the list; (2) Each following certificate MUST directly certify the one preceding it; (3) The root certificate MAY be omitted. TLS 1.3 (RFC 8446, Section 4.4.2) slightly relaxes the second rule [36], requiring only that each certificate "shall" directly authenticate its predecessor. In this paper, we define a "compliant" chain as one that fully conforms the above three rules, i.e., (1) the end-entity certificate appear first in the certificate list; (2) certificates are ordered by issuance relationship; and (3) the certificate list includes all certificates required to construct a complete chain except for the optional omission of the root certificate.

Besides, as noted in TLS 1.3, servers may provide non-compliant certificate chains, requiring clients to reconstruct the chain. For the same non-compliant input, different clients may construct different certificate paths, potentially leading to divergent validation results. For a particular client, assuming that there are one or more certificate paths in a non-compliant certificate chain that can be verified as valid by the client, and the client constructs any one of them, we say that the client chooses a "correct" path. Conversely, if the client only constructs other paths that lead to invalid results, we consider the paths selected by the client to be "incorrect".

3.1 Server-Side Chain Deployment

This paper analyzes the certificate chain structure deployed by popular web servers to evaluate their compliance with TLS standards. The data collection and analysis approaches are described below.

Data collection. We selected the certificates deployed by Tranco Top 1M (ID is 833KV) [34] domains as our primary dataset. Using ZGrab2 [3], we collected the certificate chains returned by their web servers during the TLS handshake from two VPSs (in the United States and Australia) in March 2024. During the scanning process, we limited the scanning rate to below 500KB/s to prevent any adverse effects on the servers and network. We further compared the data collected under TLS 1.2 and TLS 1.3 and found that 98.8% of the same domains received identical certificate chains. The remaining discrepancies were primarily due to different servers handling TLS 1.2 and TLS 1.3 requests, each deploying different certificates. Manual inspection of these chains reveals minimal structural differences, such as having identical intermediate and root certificates but differing only in the leaf certificate. Consequently, we chose to use the certificates collected with TLS 1.2 as our research dataset. With TLS 1.2, we obtained certificate data for 870,113 domains on the US VPS and 867,374 domains on the Australian VPS separately. The dataset we analyzed is a union of the two, containing a total of 906,336 unique certificate chains and 861,747 unique certificates. For domains with different certificate chains on two VPSs, we considered the server-side configuration to be non-compliant if one of the VPSs was non-compliant.

Leaf certificate analysis. Although RFC 5246 [14] and RFC 8446 [36] both require the leaf certificate (i.e., the server certificate in our context) is correctly placed at the beginning of the certificate list, they do not provide clear criteria for determining whether a given certificate qualifies as a leaf. This omission is not addressed in RFC 5280 [11] either.

In this work, we utilize the Common Name (CN) and Subject Alternative Name (SAN) fields of the certificate for determination. Firstly, if the CN or SAN of the first certificate in the chain matches the domain name, the chain would be classified as *Correctly Placed and Matched*. If not, we further check if these fields are formatted as domain names or IP addresses. If so, the chain is *Correctly Placed but Mismatched*. Otherwise, we check the remaining certificates in the chain and categorize them as *Incorrectly Placed but Matched* or *Incorrectly Placed and Mismatched*, depending on whether any certificate beyond the first matches the domain or follows the domain/IP format in its CN or SAN fields. All other cases are classified as *Other* and will be reviewed manually.

Order of certificates. The key to checking this requirement is to determine whether an issuance relationship exists between two adjacent certificates in the chain. RFC 5280 [11] does not provide a clear definition; however, previous studies [29, 43] have effectively distilled three primary criteria if Certificate A issuing Certificate B: (1) The public key of Certificate A must be able to verify the signature of Certificate B; (2) The subject of Certificate A needs to match the issuer field of Certificate B; (3) The Subject Key Identifier (SKID) of Certificate A needs to match the Authority Key Identifier (AKID) of Certificate B. In cases where a certificate may lack one of these fields, compliance with the validation criteria is considered fulfilled if either the second or third condition is met. This approach

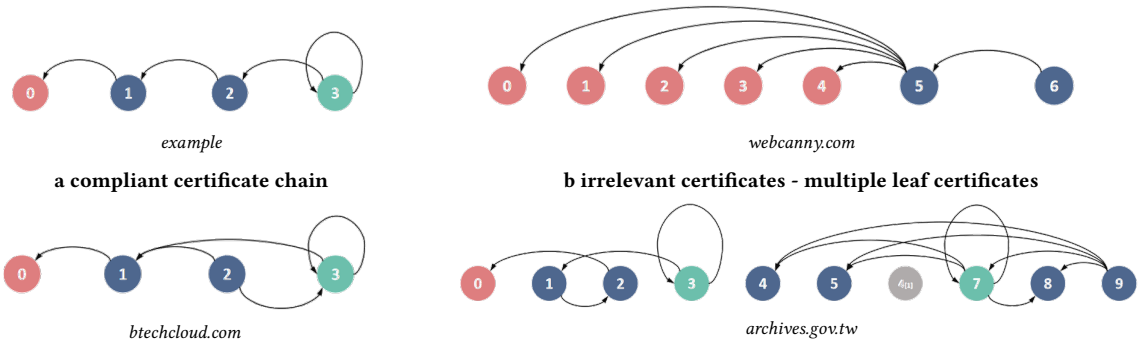


Figure 2: Server-Side certificate chain topology (red: leaf certificate; blue: intermediate certificate; green: root certificate; gray: duplicate certificate)

allows for some flexibility in handling certificates that might not fully adhere to the typical structure but are still functionally secure within the chain. Therefore, we can verify the issuance relationships of certificates in a chain from front to back, determining if they are orderly placed.

Since certificates can be out of order in various ways, such as duplicates, reversed paths, or missing necessary certificates, we use a topological graph to formally illustrate the issuance structure of the certificate chains, which allows us to further classify and understand differences between out-of-order types. Specifically, we place the chain horizontally from left to right, as per the order provided by the server. In a compliant deployment, the leftmost certificate is the server (leaf) certificate, and the rightmost is either the root certificate or an intermediate cert whose parent is the root. We treat each certificate as one node. The node is denoted by C_p ($p \geq 0$), p indicating its position in the chain. For a certificate chain containing four certificates (Figure 2a), it can be represented as: C_0, C_1, C_2, C_3. Then we first check *Duplicated Certificates*, i.e., two or more bit-for-bit identical certificates in one chain. Certificate chains with duplicated certificates are definitely in violation of the order requirement.

Subsequently, we analyze more complex scenarios using topological graphs. Considering that duplicate certificates can increase the complexity of the issuance order, we keep only the first (i.e., the leftmost one) of duplicate certificates when constructing the topological graph. We replace the original label of duplicated certificates with $Cp[i]$, where p represents the position where the certificate first appears, and i indicates its i^{th} occurrence. As shown in Figure 2d, the node originally designated as '6', due to its certificate duplicating that of node '4', is relabeled as '4[1]'. Based on the topological graph, we can check *Irrelevant Certificates* (Figure 2b), i.e., one or more certificates that cannot be connected to the leaf certificates (C_0). Besides, we can find all paths in the graph that terminate at leaf certificates. When the number of such paths exceeds one, we classify it as *Multiple Paths* (Figure 2c). Finally, we check all paths within the certificate chain to see if any issuer certificates appear before their corresponding subject certificates. If such cases are found, they are classified as *Reversed Sequences* (Figure 2c).

Certificate chain completeness analysis. Standards suggest that root certificates may be omitted since clients are assumed capable of supplementing them to complete the chain building. However, it is not stated that other certificates, particularly intermediate certificates, can be omitted. Therefore, we check whether the certificate chain includes all necessary intermediate certificates required for chain building. Utilizing the topological graph, we identify all potential chain paths. For the last certificate in each path, we make the following assessments:

- If it is a self-signed certificate, we consider that there is at least one complete certificate chain without an omitted root certificate.
- If it is not a self-signed certificate, we search for its issuer using its AKID/AIA fields. Specifically, we first check if the certificate's AKID matches the SKID of any certificates in the root store from Mozilla [5], Microsoft [7], Chrome [1] and Apple [26]. Next, we download the certificate issuer through the AIA and then check if it is a self-signed certificate. The AIA mechanism allows clients to retrieve missing issuer certificates via HTTP-accessible URIs specified in the "caIssuers" field of the AIA extension. If the issuer cannot be found or the direct issuer is not a self-signed certificate, we determine that the certificate chain lacks the necessary intermediate certificates[2].

3.2 Client-Side Chain Construction

The relaxation of server-side certificate chain deployment requirements in TLS 1.3 increases the demands on client-side chain construction capabilities, expecting clients to be prepared to handle non-compliant chains. Although prior studies have examined various problems in the chain verification of major browsers and TLS libraries [9, 17–19], they have treated "chain construction" and "chain verification" as a unified process. The specific issues that occur during the "chain construction" phase have not yet been systematically

[2]Here we use the concatenation of common root stores, so the incomplete chains identified likely affect nearly all clients. Clients may experience more issues, depending on the completeness of their specific root store.

Table 2: Certificate chain construction capability test

#	Capability Type	Description	Test Case (formal description)[1]
1	**Order Reorganization**	Provide a chain with disordered certificates to test the client's construction capabilities.	$\{E, I_2, I_1, R\}$
2	**Redundancy Elimination**	Provide a chain containing irrelevant certificates to test the client's ability to eliminate redundancies.	$\{E, X, I, R\}$
3	**AIA Completion**	Provide a chain missing intermediate certificates and test if the client can use AIA to construct the chain correctly.	$\{E, I_1\}$ URI in I_1's AIA caIssuer point to I_2
4	**Validity Priority**	Priority decision among issuer certificates with differing Validity periods.	$\{E, I, I_1, I_2, I_3, R\}$ I, I_1, I_2, and I_3 have the same subject; I's validity period is 1 year and is valid, I_1 is invalid, I_2's validity period is 1 year but is more recent, and I_3 has the same start date as I but a validity period of 10 years.
5	**KID Matching Priority**	Priority decision among issuer certificates with varying KID statuses.	$\{E, I, I_1, I_2, R\}$ I, I_1, and I_2 have the same subject; I's KID matches, I_1's KID mismatches, and I_2 lacks the KID field.
6	**KeyUsage Correctness Priority**	Priority decision among issuer certificates with differing KeyUsage settings.	$\{E, I, I_1, I_2, R\}$ I, I_1, and I_2 have the same subject; I's KeyUsage is correct, I_1's KeyUsage is incorrect, and I_2 lacks the KeyUsage field.
7	**Basic Constraints Priority**	Priority decision based on correct or incorrect Path Length constraints.	$\{E, I_1, I_2, I_3, R\}$ I_2 and I_3 have the same subject and is I_1's issuer; I_2's path_length is correct and I_3's path_length is incorrect.
8	**Path Length Constraint**	Maximum chain length the client can construct.	$\{E, I_1, I_2, \ldots, I_n, R\}$ I_n is the issuer of I_{n-1}
9	**Self-signed Leaf Certificate**	Whether the client allows a self-signed certificate as a leaf in chain construction.	$\{ES, E, I, R\}$ E and ES have the same subject but ES is self-signed.

[1] E: End-entity certificate. I: Intermediate certificate. R: Root certificate. ES: Self-signed server certificate. X: Irrelevant certificate.

studied. In this work, we evaluate the chain construction capabilities of 8 mainstream TLS clients, including 4 browsers (Chrome (v128.0.6613.114), Firefox (v126.0), Microsoft Edge (v128.0.2739.54), Safari (v17.4)) and 4 TLS libraries (OpenSSL (v3.0.2), MbedTLS (v3.5.2), GnuTLS (v3.7.3), CryptoAPI (v10.0.19041.5072)).

Empirical analysis. Due to the lack of public information on the chain construction process of TLS clients, we first conducted an empirical study. We selected all open-source projects from the TLS clients under study, including Chromium [10] (the open-source project of Chrome), Mozilla NSS [32], OpenSSL [33], GnuTLS [20] and MbedTLS [31], and analyze their source code to understand the basic principles behind the process. Our analysis revealed that all TLS clients use a forward construction approach, building the path from the leaf certificate to a trust anchor. Notably, each implementation employs a unique method for selecting the issuer of the given certificate. For instance, Chromium prioritizes candidate certificates by first checking for the Key Identifier (KID) match, then evaluating whether the certificate is self-signed, and finally comparing their validity. In contrast, OpenSSL, in addition to considering the subject name and KID, also checks if the signature algorithms match and compares the validities of certificates. We also found that TLS clients could vary in how they integrate chain construction and validation processes. For example, OpenSSL completes the entire chain construction before conducting validation checks. While MbedTLS performs "partial" validation during chain construction, i.e., immediately verifying certificate signatures and revocation status upon selecting the candidate certificate. Only certificates that pass this validation will be added to the chain.

Construction capability test. Based on the empirical study, we summarized a series of common principles for certificate chain construction, as listed in Table 2. They can be categorized into three types: *basic capabilities* (Type 1-3), which include order reorganization, redundancy elimination, and AIA completion; *priority preferences* (Type 4-7), which assess if the client prioritizes certificates based on specific attributes in scenarios where multiple candidate certificates could serve as the potential issuer; and *restriction settings* (Type 8-9), which cover limitations on chain length and whether self-signed leaf certificates are permitted in chain construction. For *basic capabilities*, we consider a client to possess the capability if it successfully passes the validation. For each *priority preferences* test, we generate multiple intermediate certificates with the same subject field, but differing in certain other fields. By altering their arrangement and observing the certificate chain constructed by the client, we can infer its method of priority selection.

Real-world impact evaluation. After assessing client-side chain-building capabilities and server-side deployment compliance, we want to further evaluate whether current TLS clients can handle real-world (potentially non-compliant) certificate chains and identify related security implications. We use the dataset described in Section 3.1 for evaluation. As the chains collected from real servers may have various configuration errors or structural non-compliance, there is no standard answer for the validation results.

Therefore, we use differential testing, i.e., comparing the differences in the validation results across different TLS clients. The disparate results are then manually analyzed to infer root causes and security implications.

4 Server-Side Deployment

This section presents the results of structural compliance of server-deployed certificate chains using the dataset mentioned in Section 3.1.

4.1 Leaf Certificate Deployment

Our evaluation shows that leaf certificate deployment is widely compliant with standards, i.e., locating as the first cert in the chain. Of the 906,336 Tranco Top 1M domains, 838,354 (92.5%) are *Correctly Placed and Matched*, 62,536 (6.9%) are *Correctly Placed but Mismatched*, 5,445 (0.6%) fall into the *Other* category, and only 1 was *Incorrectly Placed and Mismatched* (see Table 3). It is `mot.gov.ps`, which first certificate's `CN=SophosApplianceCertificate_xxx` (not in domain format), but the issuer is a self-signed certificate with `CN=www.mot.gov.ps` (in domain format). We manually reviewed the *Other* category and found that these sites configured certificates with empty CN fields or CNs indicating test use (e.g., Plesk[3], localhost, testexp).

Overall, the compliance regarding the leaf cert location is notably high. Our investigation into the checking mechanisms provided by mainstream HTTP servers (see Table 4) during the configuration process reveals that, since users must configure the private key (of the server certificate), servers typically verify that the private key corresponds to the first certificate in the list. Failure to match results in an "SSL_CTX_use_PrivateKey failed" error. This verification may contribute to the observed high compliance level.

Table 3: Leaf certificate deployment

Place	Match	#Tranco Top 1M domains
✓	✓	838,354 (92.5%)
✓	×	62,536 (6.9%)
×	✓	0 (~0%)
×	×	1 (~0%)
Other		5,445 (0.6%)

4.2 Issuance Order

We next analyzed whether certificates in the chain were placed in their compliant issuance order. Using the topological method introduced in Section 3.1, we found that even for Tranco Top 1M domains, 16,952 (1.9%) domains had non-compliant certificate chain ordering. Table 5 summarizes the four types of non-compliant deployment.

Duplicate certificates. There are 5,974 (35.2%) chains containing duplicate certificates, including 4,730 chains with duplicated leaf certificates, 1,354 with duplicated intermediate certificates, and 401 with duplicated root certificates. The maximum number of duplicate certificates in a single chain is 26. Among the chains with duplicate leaf certificates, 4,231 chains have two leaf certificates placed at the front of the chain. No common pattern was observed for duplicate intermediate or root certificates. However, we identified 4 servers (`ns3.link`, `ns3.com`, `ns3.cx`, `n0.eu`) with a similar topology: a leaf certificate followed by two intermediate certificates (`Let's Encrypt`, `CN=R3`, and `ISRG Root X1`), forming a certificate path. These two intermediate certificates are then duplicated repeatedly, ultimately resulting in chains containing up to 29 certificates.

We speculate that these duplications stem from the specific configuration features of HTTP servers. We use Nmap [2] to fingerprint the server applications by referring to the `Server` field in the HTTP headers. The results (Table 10 in Appendix B) revealed that duplicated certificate rates were notably higher for Apache servers. We further investigate the mainstream HTTP servers, focusing on their certificate deployment mechanisms, such as automatic certificate management support, required certificate fields, and potential issues during deployment (e.g., mismatches between the leaf certificate and private key, or the presence of duplicate certificates). We manually deployed problematic certificate chains to observe how different HTTP servers respond (e.g., whether error messages are returned), and cross-validated the deployment results by accessing the websites directly. Detailed results are presented in Table 4. We found that Apache (prior to version 2.4.8) and AWS ELB require two separate files for certificates: `SSLCertificateFile` (for the end-entity certificate only) and `SSLCertificateChainFile` (for intermediate and root certificates). Users may misunderstand the usages of these two files and include leaf certificates in `SSLCertificateChainFile`, causing duplicates. Apache updated to use the same approach as Nginx after version 2.4.8, placing the complete chain directly in one file, which may mitigate the misconfiguration. In contrast, Microsoft-Azure-Application-Gateway checks for duplicate leaf certificates during certificate upload, providing a more robust configuration.

Duplicate certificates may not directly cause client chain construction errors but can lead to excessively long chains, exceeding the maximum length limit supported by TLS clients (see Section 5.1, e.g., GnuTLS). Besides, clients like MbedTLS do not eliminate duplicate certificates, which may increase resource consumption during the build process.

Irrelevant certificates. 3,032 (17.9%) certificate chains deployed by the Top 1M domains contain irrelevant certificates, i.e., certificates with no direct or indirect issuing relationship to the leaf cert. Duplicate certificates are not counted.

We analyzed different scenarios of irrelevant certificates. First, 225 chains contain unrelated self-signed certificates, i.e., root certificates that have no issuing relationship with leaf certificates. Of the 159 certificate chains identified, each has a leaf certificate that is self-signed and originates from non-authoritative CAs, which should ideally terminate the chain. However, the servers have also deployed additional, irrelevant root certificates from public CAs. Secondly, 444 chains contain multiple distinct leaf certificates, and only one of them would be used for chain building. Among these, 338 chains have multiple leaf certificates differing only in validity periods, likely because the outdated certificates are not removed during updates. For example, the servers for `webcanny.com` (illustrated in Figure 2b) deploy 5 leaf certificates from the same CA

[3]Plesk is a web hosting control panel, and a certificate with `Plesk` as the CN usually signifies a self-signed certificate generated by default during testing or development.

Table 4: Characteristics of SSL certificates deployed across different HTTP servers

Deployment Characteristics	Apache	Nginx	Microsoft-Azure-Application-Gateway	IIS	AWS ELB
Automatic Certificate Management	✓	✓	✓	✗	✓
Supported Certificate Fields	<2.4.8 SF_1 >=2.4.8 SF_2	SF_2	SF_3	SF_3	SF_1
Private Key and Leaf Certificate Matching Check	✓	✓	✓	✓	✓
Duplicate Leaf Certificate Check	✗	✗	✓	✓	✗
Duplicate Intermediate/Root Certificate Check	✗	✗	✗	✗	✗

SF_1: CertificateFile.pem, Ca-bundle.pem, Privkey
SF_2: FullChain.pem, Privkey
SF_3: CertificateFile.pfx

CertificateFile.pem - Contains only the leaf certificate.
Ca-bundle.pem - Contains only intermediate/root certificates.
Fullchain.pem - Contains the complete certificate chain.
Privkey - The private key.
CertificateFile.pfx - A PFX-formatted certificate chain.

Table 5: Chains with non-compliant issuance order

Type	#Tranco Top 1M domains
Duplicate Certificates	5,974 (35.2%)
Irrelevant Certificates	3,032 (17.9%)
Multiple Paths	246 (1.5%)
Reversed Sequences	8,566 (50.5%)
Total	16,952

(`Sectigo RSA Domain Validation Secure Server CA`). The validity of these certificates ranges from 3 months to 1 year, spanning 2019 to 2024. They are arranged with the most recent on the left, progressively moving to older certificates towards the right.

Finally, we identified 840 certificate chains with irrelevant certificates that had issuance relationships with each other, i.e., they appeared to be (part of) another chain. For instance, the Taiwan government site `archives.gov.tw` (Figure 2d) includes a primary chain of certificates from nodes 0–3, with `ePKI Root Certificate Authority` as the final CA. However, the server also provides certificates 4–9. Cert 7 is a CA certificate (`TWCA Global Root CA`) commonly used in Taiwanese government sites, and others are intermediate certificates issued by it. One possible reason for this is that multiple domains are managed by the same administrator (who holds multiple CAs and intermediate CAs).

While irrelevant certificates do not impede chain construction, they can result in excessively long chains. This may exceed client limits (See Section 5.1) or consume additional resources through unnecessary chain construction attempts.

Multiple paths. A total of 246 (1.5%) chains have more than one path, with up to three paths observed. Of these, 241 certificate chains were caused by cross-signed certificates, as illustrated in Figure 2c, where node 2 and node 3 have the same subject (`USERTrust RSA Certification Authority`) and SKID but different issuers. Additionally, another 5 certificate chains were caused by the presence of several intermediate certificates with identical `subject` and `issuer` but differing validity periods. Cross-signed certificates, while increasing availability by providing multiple paths when the server cannot determine the client's trusted root certificate, can also introduce complexity. Notably, we identified 29 certificate chains containing expired cross-signed certificates, indicating current management flaws for timely renewing cross-signed certificates by website administrators.

We acknowledge that, in practical deployments, multiple paths caused by cross-signed certificates may be intentionally provided to enhance reliability by offering clients multiple chain construction options. It is important to note that the existence of multiple paths due to cross-signing does not necessarily imply a violation of the chain compliance requirements. In Figure 2c, a compliant chain order can be achieved simply by reordering the certificates (e.g., swapping node 2 and node 3 to follow the issuance order). Despite this possibility, since the server does not provide the certificates in the correct issuance order, we still classify this case as non-compliant. In contrast, there are more complex cross-signing topologies, such as those involving three distinct issuers or multiple layers of cross-signing, it may be impossible to satisfy the issuance-order requirement through any reordering. Such cases should be considered as exceptions and excluded from the non-compliance classification. Upon manual inspection, we did not find such cases in our dataset, so our measurement results are not affected.

Overall, the presence of multiple paths increases the complexity of certificate chain construction. Without robust backtracking capabilities or comprehensive prioritization settings, clients risk constructing incorrect certificate chains. This issue will be further evaluated in Section 5.

Reversed sequences. The reversed order is the most prevalent issue of non-compliance. We found 8,566 (50.5%) chains of Top 1M domains contained at least one path in reversed order, and 8,370 had all paths reversed. Among them, the reversed order of chains containing multiple paths is mostly due to the inappropriate insertion positions of cross-signed certificates. For example, in Figure 2c, node 2 is a cross-signed certificate. Placing it after node 3 maintains the compliant issuance order and avoids a reversed path. However, it was discovered that the server placed it before node 3.

Besides, there are 8,365 certificate chains that have only one path but still have the order reversed. We identified the two most common chain path structures: 1->2->0 (5,248 chains) and 1->2->3->0 (1,769 chains). We suspect that the issue is related to the certificate files provided by different CAs or their resellers during the issuance process. In Table 11 (see Appendix C), We documented the number of non-compliant certificate chains issued by various CAs or their resellers and found that `GoGetSSL`, `cyber_Folks S.A.`,

Table 6: Characteristics of SSL certificates issued by different CAs or resellers

Issuance Characteristics	Let's Encrypt	ZeroSSL	GoGetSSL	Trustico	cyber_Folks S.A.
Automatic Certificate Management	✓	✓	✗	✗	✗
Provide Fullchain File	✓	✗	✗	✗	✗
Provide Ca-bundle File	✓	✓	✓	✓	✓
Provide Root Certificate	✗	✗	✗	✓	✓
Compliant Issuance Order in Ca-bundle File	✓	✓	✗	✗	✗
Provide Certificate Installation Guide	✓	✓	✗	✗	only Apache/IIS

and `Trustico` have a high incidence of reversed sequences. To further investigate the root cause, we applied for an SSL certificate from these CAs or resellers and discovered that it provides two files: one containing only the leaf certificate and another containing the required intermediate and root certificates in reverse order (see Table 6). When administrators receive these two files, they may simply merge them without altering the order of the (intermediate) root certificates, leading to a reversed path.

For clients (e.g., MbedTLS) without the ability to reorder certificates, a certificate chain with reversed sequences will directly result in verification failure (See Section 5.1).

Table 7: Completeness of certificate chain

Type	#Tranco Top 1M domains
Complete Chain w/ Root	79,144 (8.7%)
Complete Chain w/o Root	815,105 (89.9%)
Incomplete Chain	12,087 (1.3%)

4.3 Completeness of Certificate Chain

We also evaluated whether the certificate lists (chains) provided by Tranco Top 1M domains contain all the necessary certificates to form a complete path. Our definition of completeness requires that the certificate path either include the root certificate or that the immediate issuer of the final intermediate certificate is a root certificate. For the latter case, in practical scenarios, clients need to complete the chain by retrieving the root certificate themselves. To avoid overstating related issues, we assume that clients support AIA fetching and use a unified root store that combines Mozilla, Chrome, Microsoft, and Apple. Under this assumption, the collected chains can be classified into three categories: (complete) chains that contain both the intermediate and the root certificate (79,144 (8.7%)), chains that contain intermediate but omit the root certificate (815,105 (89.9%)), and chains that lack intermediate certificates and therefore cannot directly form a complete path (12,087 (1.3%)). It can be seen that omitting the root certificate is a common practice. The last type is considered non-compliant. Of them, we found 8,729 (72.2%) chains were missing a single intermediate certificate, which could be fixed by adding the missing cert. While the remaining chains were missing more than one intermediate certificate.

The AIA mechanism is designed to resolve the issue of missing certificates. Ideally, the issuer certificate for a particular certificate can be dynamically downloaded via the URI in the AIA field of its content. Among the 12,087 certificate chains that lack necessary intermediate certificates, we found that 11,419 (94.5%) chains can be completed by recursively downloading certificates from AIA. However, there are also cases where AIA fields are missing (579 chains), URI access fails (88 chains), or the certificates retrieved through AIA are not the correct issuers (1 chain). For example, the certificate located at the AIA URI (http://www.CAcert.org/class3.crt) for `CAcert Class 3 Root` is the certificate itself, not its issuing authority `CA Cert Signing Authority`. Additionally, dynamically downloading certificates via AIA can have security risks, as the primary downloading method is HTTP, which may be susceptible to man-in-the-middle attacks and would also raise privacy concerns [4].

Table 8: Additional incomplete chains in Tranco Top 1M domains due to differences in root stores and AIA support across various TLS clients

Root Store	Mozilla	Chrome	Microsoft	Apple
AIA Supported	66	66	5	4
AIA Not Supported	225,608	225,608	225,538	225,360

As mentioned above, the incomplete chains identified here represent a lower bound. In practice, clients often rely on a single root store and may lack AIA support, resulting in more chains being deemed incomplete. Table 8 quantifies the additional incomplete chains in Tranco Top 1M, for clients using individual root stores with or without AIA. Each column corresponds to a specific root store. The results indicate that root store differences have a limited impact for clients when AIA is supported. AIA capability plays a more critical role for achieving chain completeness.

Summary of server-side evaluation: We found that there is a general structural non-compliance in the current server-side certificate chain deployment, affecting 26,361 (2.9%) of Top 1M domains. Among them, the violation of the placement of issuance order (64.3%) and missing necessary intermediate certificates (45.9%) are the main non-compliant behaviors. Although the primary cause is misconfiguration by domain administrators, our survey showed that the configuration features/guidance provided by HTTP servers and Certificate Authorities also matter. No clear configuration guidance from CAs (or even providing certificates in reverse order) can exacerbate this issue. While automated checks by HTTP servers during configuration can effectively mitigate this problem, such checks are currently far from fully implemented.

Table 9: Differences in the capabilities of TLS implementations

Type	OpenSSL	GnuTLS	MbedTLS	CryptoAPI	Chrome	Microsoft Edge	Safari	Firefox
Order Reorganization	✓	✓	×	✓	✓	✓	✓	✓
Redundancy Elimination	✓	✓	✓	✓	✓	✓	✓	✓
AIA Completion	×	×	×	✓	✓	✓	✓	×
Validity Priority	VP_1	\	VP_1	VP_2	VP_2	VP_2	VP_2	VP_1
KID Matching Priority	KP_1	KP_1	\	KP_2	KP_2	KP_2	KP_1	\
KeyUsage Correctness Priority	\	\	KUP	KUP	KUP	KUP	KUP	KUP
Basic Constraints Priority	\	\	BP	BP	BP	BP	BP	BP
Path Length Constraint	>52	**=16**	=10	=13	>52	=21	>52	=8
Self-signed Leaf Certificate	×	×	✓	×	×	×	✓	×

✓ - Supported × - Not Supported \ - No priority ordering

VP_1 - Selects the first valid cert.
VP_2 - Prioritizes most recent, then longest validity among valid certs.
KP_1 - KID match/absence prioritized over mismatch.
KP_2 - KID match prioritized over absence and mismatch.
KUP - Correct/missing KeyUsage prioritized over incorrect.
BP - Correct basic constraints prioritized.

5 Client-Side Capability

Section 4 demonstrated that real-world servers do not yet strictly adhere to certificate structural requirements in RFC [14, 36], imposing additional demands on client-side abilities to construct certificate paths. This section presents testing results on client-side chain construction and uses differential testing to reveal the impact of server-side non-compliance and client-side deficiencies.

5.1 Capabilities of TLS Implementations

Using the methodology outlined in Section 3, we summarized 9 types of common certificate chain-building capabilities and designed targeted test samples for evaluation.

Overall evaluation. The results for mainstream browsers and TLS libraries are listed in Table 9. It is apparent that libraries (aside from CryptoAPI) generally perform worse than browsers, mainly due to the lack of basic chain construction capabilities like AIA completion. Although Firefox also lacks AIA completion, it compensates by caching intermediate certificates to complete the chain.

Differences in priority choices. We observed that all TLS implementations prioritize specific fields in certificates when selecting issuer certificates to construct a chain. Still, there are differences in the priority choices and processing logic among implementations. For example, in the *KID matching priority test*, MbedTLS and Firefox do not differentiate by priority and simply select the first certificate. In contrast, clients including OpenSSL, GnuTLS, and Safari, treat missing and mismatched `KID` fields with equal priority, while other clients prioritize missing `KID` over mismatches. These differences in priority settings can lead to variations in the certificate chains constructed by different clients.

5.2 Real-World Impact

We conduct differential testing on the libraries and browsers mentioned in Section 3.2 using certificate chains deployed by servers. This evaluation aims to assess the real-world impacts caused by non-compliant server deployments and deficiencies in client capabilities.

Result overview. The test dataset is our collection of 906,336 certificate chains. We specifically focused on the results of 26,361 non-compliant chains. Among them, 61.1% passed the validation in all 3 browsers (we excluded Safari for it cannot retrieve specific certificate chain validation error messages in the same way as the other browsers), and 47.4% passed the validation in all 4 libraries. This gap highlights the impact of chain-building capabilities on validation. After excluding network-related issues, 3,295 chains showed discrepancies across browsers, while 10,804 chains exhibited inconsistencies across libraries. We manually reviewed the cases to investigate the underlying causes of discrepancies. The validation issues due to structural problems of certificate chains are summarized below.

I-1: the lack of order reorganization capability. We identified 51 certificate chains where MbedTLS could not correctly find the issuer certificates due to their not being deployed in the issuance order, resulting in MbedTLS reporting an error (`X509_BADCERT_NOT_TRUSTED`) unlike the other three clients which successfully constructed the certificate path, impacting the validation of 22 websites of Taiwan governments.

I-2: inability to process overly long chains. Experimental results revealed that GnuTLS fails to build and validate 10 certificate chains due to the number of certificates deployed exceeding its maximum limit of 16. Unlike other clients, GnuTLS imposes a limit on the number of certificates in the original list rather than the maximum length of the constructed chain. Consequently, if a server deploys multiple irrelevant or duplicate certificates, validation may fail. As illustrated in Figure 3, the correct chain should be 8->1->16->0. However, the construction and validation fail as '16' exceeds the length limit of GnuTLS.

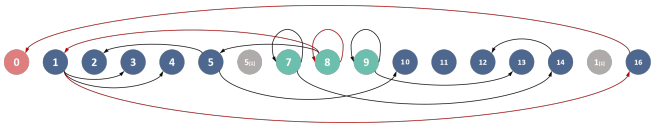


Figure 3: Certificate list of `assiste6.serpro.gov.br`. The red arrow indicates the certificate path.

I-3: the lack of backtracking ability. We identified a case where OpenSSL, GnuTLS, and MbedTLS, lacking backtracking capabilities, constructed incorrect certificate chains. The certificate chain

topology deployed by `moex.gov.tw` (Figure 4) presents three candidate paths; nodes 1 and 4 are root certificates, but node 1 is not included in any of the four trust stores discussed in Section 3.1. Both OpenSSL and GnuTLS incorrectly included node 1 (paths 1 or 2), while CryptoAPI correctly selected path 3 by backtracking after detecting that node 1 is not trusted. MbedTLS also produced path 3, but only because it lacks reordering capability. When we swapped the order of nodes 1 and 2, MbedTLS also included node 1 in the construction path.

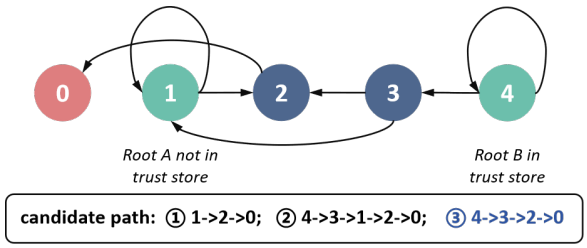


Figure 4: Certificate list of `moex.gov.tw`

I-4: the lack of AIA completion capability. In our differential tests across libraries, we found that 8,553 certificate chains could be constructed successfully by CryptoAPI but failed in the other three libraries. This is likely because CryptoAPI supports AIA fetching. To confirm this, we disabled AIA in the CryptoAPI validation code, which led to 8,373 (97.9%) of the chains failing to construct (the remaining chains could be fetched from the Windows system's Intermediate Certificate Store). In browser differential tests, we discovered 1,074 certificate chains for which Edge and Chrome showed consistent results (`date_invalid/OK/domain mismatch`), while Firefox marked them as `SEC_ERROR_UNKNOWN_ISSUER`. This discrepancy is likely due to Firefox relying on intermediate certificate caching rather than AIA fetching.

Besides, Chrome and Edge showed more consistent results than Firefox in both capability and differential testing, probably due to their shared Chromium engine, unlike the Firefox Gecko engine, which relies on NSS for handling secure communication.

Summary of client-side evaluation: We identified discrepancies and deficiencies in the chain construction capabilities of mainstream TLS implementations, where libraries (excluding CryptoAPI) generally underperform compared to browsers in both basic capabilities and priority selection. Given the widespread non-compliance of certificate chains on real-world servers, the deficiency in chain-building capabilities can lead to validation failures. For TLS libraries, this typically means an inability to establish a TLS connection, while for browsers, it results in warning pages that impact service availability. For Tranco 1M websites, 40.9% certificate chains encounter building issues in TLS libraries, and for browsers, the percentage is 12.5%, highlighting a notable impact on the availability of network services.

6 Discussion

In this section, we propose improvements for both the server side and the client side, as well as provide guidance for other involved parties. We also discuss the limitations of this study.

6.1 Server-Side Recommendation

Our analysis showed that the compliance of certificate chain deployment could be improved by clear instructions from CAs and thorough checks and reminders from HTTP servers. Therefore, we propose the following recommendations.

For Certificate Authorities: Provide users with complete and compliant certificate chains accompanied by detailed deployment instructions, such as specifying the placement of files on common server configurations.

For HTTP servers: Implement automated checks during certificate deployment to identify and resolve common errors, such as detecting and removing duplicate certificates.

For web administrators: Beyond carefully following CA and server guidelines, adopting automated certificate management solutions (e.g., already offered by major CAs like `Let's Encrypt`) can simplify the process. Automation not only ensures compliant deployment but also facilitates automatic certificate renewal, reducing the risk of non-compliant configurations.

6.2 Client-Side Recommendation

This section proposes recommendations for client-side chain construction based on our experimental findings.

Construction capability. As shown in Section 5.2, the most critical factor affecting a client's ability to construct valid certificate chains is to complete missing certificates (effecting 8,373 Top 1M domains). This can be achieved through AIA fetching or alternative mechanisms such as those adopted by Firefox. In addition, backtracking, which allows the client to attempt an alternative path when the initially constructed chain is invalid, is also important. Order reorganization, which enables the client to correctly process certificates presented in a disordered sequence, further contributes to construction. Our empirical results show that clients equipped with all three capabilities exhibit a significantly higher success rate in validating server certificate chains.

Prioritization. We also provide practical recommendations on prioritization strategies when multiple candidate issuer certificates are available. A primary consideration is KID matching. Clients typically begin by identifying issuers whose `subject_DN` matches the `issuer_DN` of the current certificate. To further ensure the cryptographic correctness, we recommend that candidates be prioritized based on KID matching in the following order: match >empty >mismatch. When both `subject_DN` and KID match, clients may still encounter multiple candidates. We identified 785 such chains for Tranco 1M domains. The most common scenario, observed in 744 chains, involves an intermediate certificate and a self-signed root certificate that share the same `subject_DN` and KID. Notably, all of the root certificates in these cases are present in the trusted root store. To reduce unnecessary chain construction attempts and improve efficiency, we recommend prioritizing the trusted self-signed root certificate when available. The remaining 42 chains involve multiple intermediate certificates that differ only in validity periods. Examples of these certificates are provided in Figure 5. In such cases, the most recently issued certificate should be preferred, as it is more likely to reflect the current configuration of the issuing authority.

Figure 5: Priority selection cases

6.3 Limitations

First, our primary analysis and research in this paper focus on the more popular domains within the Tranco Top 1M list. It is possible that less popular domains may exhibit a higher prevalence of non-compliant deployments, potentially leading to more profound certificate chain construction security issues. Nevertheless, we discovered that even among these popular domains, about 3% exhibit non-compliant deployments (Section 4), leading to discrepancies in 1,440 (5.6%) certificate chains during differential testing in various libraries (Section 5). Besides, although we employed a heuristic approach to construct test cases for malformed certificate chains, it may not have covered all possible scenarios, such as the impacts of caching, certificate policies, and certificate revocation. These factors do indeed influence the certificate chain construction process; however, their variability and unpredictability make them difficult to capture comprehensively. Therefore, our study only provides insights into the worst-case scenarios. Moreover, the differential testing method used to assess real-world impacts might suffer from false positives, where all TLS implementations exhibit the same behavior, which could inaccurately reflect an erroneous condition.

7 Conclusion

This paper presents the first comprehensive assessment of certificate chain construction, covering both server-side deployments and client-side capabilities. We found that 2.9% of popular domains exhibit non-compliant deployments. Significant deficiencies were also found in client-side chain construction capabilities, including missing basic capabilities and inappropriate prioritization, which could impact network service availability. Based on our findings, we recommend improvements to both deployment practices and construction logic.

Acknowledgments

We thank our shepherd and the anonymous reviewers for their valuable feedback. This work is supported by the National Natural Science Foundation of China (62302258, 62102218). Baojun Liu and Yiming Zhang are both the corresponding authors.

References

[1] [n. d.]. Chrome Root Store. https://chromium.googlesource.com/chromium/src/+/main/net/data/ssl/chrome_root_store/root_store.md
[2] [n. d.]. Nmap. https://nmap.org/
[3] [n. d.]. ZGrab 2.0. https://github.com/zmap/zgrab2
[4] 2020. "all trusted Web PKI Certificate Authority certificates known to Mozilla will be cached locally". https://lwn.net/Articles/817182/
[5] 2023. Mozilla Project. https://wiki.mozilla.org/CA/Included_Certificates
[6] 2024. CVE-2024-0567. https://nvd.nist.gov/vuln/detail/CVE-2024-0567
[7] 2024. List of Participants - Microsoft Trusted Root Program. https://learn.microsoft.com/en-us/security/trusted-root/participants-list
[8] Mustafa Emre Acer, Emily Stark, Adrienne Porter Felt, Sascha Fahl, Radhika Bhargava, Bhanu Dev, Matt Braithwaite, Ryan Sleevi, and Parisa Tabriz. 2017. Where the Wild Warnings Are: Root Causes of Chrome HTTPS Certificate Errors. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1407–1420. doi:10.1145/3133956.3134007
[9] Chad Brubaker, Suman Jana, Baishakhi Ray, Sarfraz Khurshid, and Vitaly Shmatikov. 2014. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 114–129. doi:10.1109/SP.2014.15
[10] Chromium. [n. d.]. The Chromium Projects. https://source.chromium.org/chromium/
[11] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and W. Timothy Polk. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *RFC* 5280 (2008), 1–151. doi:10.17487/RFC5280
[12] Matt Cooper, Yuriy Dzambasow, Peter Hesse, Susan Joseph, and Richard Nicholas. 2005. Internet X.509 Public Key Infrastructure: Certification Path Building. *RFC* 4158 (2005), 1–81. doi:10.17487/RFC4158
[13] Joyanta Debnath, Sze Yiu Chau, and Omar Chowdhury. 2021. On Re-engineering the X.509 PKI with Executable Specification for Better Implementation Guarantees. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, New York, NY, USA, 1388–1404. doi:10.1145/3460120.3484793
[14] Tim Dierks and Eric Rescorla. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. *RFC* 5246 (2008), 1–104. doi:10.17487/RFC5246
[15] Zakir Durumeric, David Adrian, Ariana Mirian, Michael D. Bailey, and J. Alex Halderman. 2015. A Search Engine Backed by Internet-Wide Scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM, New York, NY, USA, 542–553. doi:10.1145/2810103.2813703
[16] Zakir Durumeric, James Kasten, Michael D. Bailey, and J. Alex Halderman. 2013. Analysis of the HTTPS certificate ecosystem. In *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013*, Konstantina Papagiannaki, P. Krishna Gummadi, and Craig Partridge (Eds.). ACM, New York, NY, USA, 291–304. doi:10.1145/2504730.2504755
[17] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael D. Bailey, J. Alex Halderman, and Vern Paxson. 2017. The Security Impact of HTTPS Interception. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society. https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/security-impact-https-interception/
[18] Sascha Fahl, Marian Harbach, Thomas Muders, Matthew Smith, Lars Baumgärtner, and Bernd Freisleben. 2012. Why eve and mallory love android: an analysis of android SSL (in)security. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM, 50–61. doi:10.1145/2382196.2382205
[19] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. 2012. The most dangerous code in the world: validating SSL certificates in non-browser software. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM, New York, NY, USA, 38–49. doi:10.1145/2382196.2382204
[20] GnuTLS. [n. d.]. The GnuTLS Projects. https://gitlab.com/gnutls/gnutls
[21] Google. [n. d.]. Certificate Transparency. https://certificate.transparency.dev
[22] Jens Hiller, Johanna Amann, and Oliver Hohlfeld. 2020. The Boon and Bane of Cross-Signing: Shedding Light on a Common Practice in Public Key Infrastructures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1289–1306. doi:10.1145/3372297.3423345
[23] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. 2011. The SSL landscape: a thorough analysis of the x.509 PKI using active and passive measurements. In *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC '11, Berlin, Germany, November 2-, 2011*, Patrick Thiran and Walter Willinger (Eds.). ACM, New York, NY, USA, 427–444. doi:10.1145/2068816.2068856
[24] Lin-Shung Huang, Alex Rice, Erling Ellingsen, and Collin Jackson. 2014. Analyzing Forged SSL Certificates in the Wild. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 83–97. doi:10.1109/SP.2014.13
[25] ICSI. [n. d.]. ICSI Certificate Notary. http://notary.icsi.berkeley.edu/
[26] Apple Inc. 2023. Lists of available trusted root certificates in macOS. https://support.apple.com/en-us/103723
[27] Nikita Korzhitskii and Niklas Carlsson. 2021. Revocation Statuses on the Internet. In *Passive and Active Measurement - 22nd International Conference, PAM 2021, Virtual Event, March 29 - April 1, 2021, Proceedings (Lecture Notes in Computer*

Science, Vol. 12671), Oliver Hohlfeld, Andra Lutu, and Dave Levin (Eds.). Springer, 175–191. doi:10.1007/978-3-030-72582-2_11

[28] Deepak Kumar, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, J. Alex Halderman, and Michael D. Bailey. 2018. Tracking Certificate Misissuance in the Wild. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 785–798. doi:10.1109/SP.2018.00015

[29] James Larisch, Waqar Aqeel, Michael Lum, Yaelle Goldschlag, Leah Kannan, Kasra Torshizi, Yujie Wang, Taejoong Chung, Dave Levin, Bruce M. Maggs, Alan Mislove, Bryan Parno, and Christo Wilson. 2022. Hammurabi: A Framework for Pluggable, Logic-Based X.509 Certificate Validation Policies. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM, New York, NY, USA, 1857–1870. doi:10.1145/3548606.3560594

[30] Yabing Liu, Will Tome, Liang Zhang, David R. Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, Aaron Schulman, and Christo Wilson. 2015. An End-to-End Measurement of Certificate Revocation in the Web's PKI. In *Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*, Kenjiro Cho, Kensuke Fukuda, Vivek S. Pai, and Neil Spring (Eds.). ACM, New York, NY, USA, 183–196. doi:10.1145/2815675.2815685

[31] MbedTLS. [n. d.]. The MbedTLS Projects. https://github.com/Mbed-TLS/mbedtls

[32] Mozilla NSS. [n. d.]. The Mozilla NSS Projects. https://github.com/nss-dev/nss

[33] OpenSSL. [n. d.]. The OpenSSL Projects. https://github.com/openssl/openssl

[34] Victor Le Pochat, Tom van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/tranco-a-research-oriented-top-sites-ranking-hardened-against-manipulation/

[35] Sajjad Pourali, Xiufen Yu, Lianying Zhao, Mohammad Mannan, and Amr Youssef. 2024. Racing for TLS Certificate Validation: A Hijacker's Guide to the Android TLS Galaxy. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, 683–700. https://www.usenix.org/conference/usenixsecurity24/presentation/pourali

[36] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. *RFC* 8446 (2018), 1–160. doi:10.17487/RFC8446

[37] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, and Matthias Wählisch. 2018. The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. In *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*. ACM, New York, NY, USA, 343–349. https://dl.acm.org/citation.cfm?id=3278562

[38] Suphannee Sivakorn, George Argyros, Kexin Pei, Angelos D. Keromytis, and Suman Jana. 2017. HVLearn: Automated Black-Box Analysis of Hostname Verification in SSL/TLS Implementations. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 521–538. doi:10.1109/SP.2017.46

[39] Ryan Sleevi. 2020. Path Building vs Path Verifying: The Chain of Pain. Medium. https://medium.com/@sleevi_/path-building-vs-path-verifying-the-chain-of-pain-9fbab861d7d6

[40] Emily Stark, Ryan Sleevi, Rijad Muminovic, Devon O'Brien, Eran Messeri, Adrienne Porter Felt, Brendan McMillion, and Parisa Tabriz. 2019. Does Certificate Transparency Break the Web? Measuring Adoption and Error Rate. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 211–226. doi:10.1109/SP.2019.00027

[41] Tranco. 2024. *Tranco Top Domain Name List.* https://tranco-list.eu/

[42] Wenya Wang, Yakang Li, Chao Wang, Yuan Yan, Juanru Li, and Dawu Gu. 2021. Re-Check Your Certificates! Experiences and Lessons Learnt from Real-World HTTPS Certificate Deployments. In *Network and System Security - 15th International Conference, NSS 2021, Tianjin, China, October 23, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13041)*, Min Yang, Chao Chen, and Yang Liu (Eds.). Springer, 17–37. doi:10.1007/978-3-030-92708-0_2

[43] Yiming Zhang, Baojun Liu, Chaoyi Lu, Zhou Li, Haixin Duan, Jiachen Li, and Zaifeng Zhang. 2021. Rusted Anchors: A National Client-Side View of Hidden Root CAs in the Web PKI Ecosystem. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, New York, NY, USA, 1373–1387. doi:10.1145/3460120.3484768

A Ethics

When assessing server-side deployment compliance, we utilize domain data from publicly available datasets. During our scans of domain servers, we implement rate control, ensuring that the data transfer does not exceed 500KB/s to prevent any negative impact on the availability of the domain servers or the network. Additionally, we take care to avoid conducting multiple consecutive scans on a single server, further minimizing any potential disruption to server operations.

Vulnerability disclosure. The issue of inconsistent certificate chain length limits in GnuTLS has been reported to the GnuTLS team. We have also contacted the CAs or resellers (`GoGetSSL`,`cyber_FolksS.A.`, and `Trustico`) that provided the ca-bundle file with certificates in the non-compliant issuance order. Trustico does not consider the certificate order to be problematic. It states that users can manually rearrange the certificates if necessary. However, our measurements show that expecting users to understand certificate chain configuration and make manual adjustments is often ineffective in practice.

B HTTP Servers for Non-Compliant Chains

As briefly noted in Section 4.2, we used Nmap to gather statistics on HTTP servers associated with non-compliant certificate chains. Table 10 lists the top 6 HTTP servers and the percentage of their usage across each type of non-compliance. We found that most websites currently use Apache and Nginx for deployment. Additionally, it is evident that chains deployed on Azure rarely encounter issues with *duplicate certificates*, particularly there are no instances of *duplicate leaf certificates*. From our practical experience, Azure checks certificate chain files upon upload to ensure that exactly one leaf certificate matches the private key. For cloudflare, it offers a fully automated certificate chain deployment service, which typically prevents non-compliant deployments. Therefore, we hypothesize that the non-compliant deployments indicated in the table result from using *Advanced Certificate Manager*, which allows for uploading custom user certificates.

C Certificate Authorities for Non-Compliant Chains

In addition to analyzing the usage of HTTP servers, we also examined the relationship between CAs and non-compliant deployments. We selected 8 CAs or resellers based on market share, non-compliance rate, and certificate issuance costs (see Table 11). We found that `Let's Encrypt` has the highest issuance rate while maintaining the lowest rate of non-compliant deployments, largely due to its automated certificate issuance and deployment implementation. For `GoGetSSL`, `cyber_Folks S.A.`, and `Trustico`, which exhibit a higher proportion of non-compliant deployments, we have confirmed that the disorderly arrangement of intermediate and root certificates in the provided ca-bundle file leads to a high proportion of *reversed sequences*. On the other hand, `TAIWAN-CA`'s non-compliance is primarily due to *incomplete chain*. The main issue appears to be the omission of an intermediate certificate (subject = `TWCA Global Root CA`, issuer = `TWCA Root Certification Authority`) in their deployments.

Table 10: HTTP servers used by domains with non-compliant certificate chains

Non-compliant Type	Apache	Nginx	Azure [1]	cloudflare	IIS	AWS ELB	Other	Total
Overview	6,482 (39.7%)	5,826 (35.7%)	901 (5.5%)	537 (3.3%)	488 (3.0%)	375 (2.3%)	1,714 (10.5%)	**16,323**
Duplicate Certificates	2,179 (56.1%)	876 (22.6%)	9 (0.2%)	131 (3.4%)	74 (1.9%)	217 (5.6%)	396 (10.2%)	**3,882**
Duplicate Leaf	2,086 (63.3%)	548 (16.6%)	0 (0.0%)	106 (3.2%)	57 (1.7%)	201 (6.1%)	300 (9.1%)	**3,298**
Duplicate Intermediate	104 (16.6%)	328 (52.4%)	9 (1.4%)	26 (4.2%)	34 (5.4%)	9 (1.4%)	116 (18.5%)	**626**
Duplicate Root	42 (16.4%)	121 (47.3%)	5 (2.0%)	5 (2.0/%)	33 (12.9%)	12 (4.7%)	38 (14.8%)	**256**
Irrelevant Certificates	1,023 (53.0%)	633 (32.8%)	18 (0.9%)	65 (3.4%)	29 (1.5%)	27 (1.4%)	135 (7.0%)	**1,930**
Irrelevant Leaf	220 (72.6%)	49 (16.2%)	0 (0.0%)	4 (1.3%)	11 (3.6%)	5 (1.7%)	14 (4.6%)	**303**
Multiple Paths	38 (32.5%)	59 (50.4%)	0 (0.0%)	3 (2.6%)	3 (2.6%)	1 (0.9%)	13 (11.1%)	**117**
Reversed Sequences	1,219 (23.1%)	2,015 (38.2%)	750 (14.2%)	171 (3.2%)	210 (4.0%)	139 (2.6%)	764 (14.5%)	**5,268**
Incomplete Chain	2,633 (39.6%)	2,689 (40.4%)	145 (2.2%)	202 (3.0%)	199 (3.0%)	117 (1.8%)	669 (10.1%)	**6,654**

[1] Microsoft-Azure-Application-Gateway

Table 11: CAs or resellers for non-compliant certificate chains

Type	Let's encrypt	Digicert	Sectigo Limited	ZeroSSL	GoGetSSL	TAIWAN-CA	cyber_Folks S.A.	Trustico [1]
Non-compliant	4,620 (1.2%)	4,784 (7.9%)	5,118 (10.7%)	209 (2.5%)	270 (16.7%)	248 (50.4%)	94 (66.2%)	71 (65.7%)
Duplicate Certificates	3,259 (0.8%)	771 (1.3%)	639 (1.3%)	86 (1.0%)	41 (2.5%)	7 (1.4%)	3 (2.1%)	1 (0.9%)
Irrelevant Certificates	400 (0.1%)	726 (1.2%)	496 (1.0%)	35 (0.4%)	34 (2.1%)	8 (1.6%)	8 (5.6%)	1 (0.9%)
Multiple Paths	51 (∼0.0%)	6 (∼0.0%)	134 (0.3%)	0 (0.0%)	7 (0.4%)	0 (0.0/%)	0 (0.0%)	0 (0.0%)
Reversed Sequences	81 (∼0.0%)	1,736 (2.9%)	2,537 (5.3%)	2 (∼0.0%)	125 (7.7%)	47 (9.6%)	86 (60.6%)	67 (62.0%)
Incomplete Chain	1,155 (0.3%)	2,245 (3.7%)	1,998 (4.2%)	120 (1.5%)	112 (6.9%)	206 (41.9%)	8 (5.6%)	4 (3.7%)
Total	400,737	60,894	48,042	8,219	1,617	492	142	108

[1] The Trustico Group Ltd