Inside Certificate Chains Beyond Public Issuers: Structure and Usage Analysis from a Campus Network

Hongying Dong
University of Virginia
Charlottesville, VA, USA
hd7gr@virginia.edu

Yizhe Zhang
University of Virginia
Charlottesville, VA, USA
yz6me@virginia.edu

Hyeonmin Lee
University of Virginia
Charlottesville, VA, USA
frv9vh@virginia.edu

Yixin Sun
University of Virginia
Charlottesville, VA, USA
yixins@virginia.edu

Abstract

Digital certificates are crucial for securing Internet communications. Certificates issued by trusted Certificate Authorities (CAs) can be validated by following the chain of trust, consisting of leaf, intermediate, and root certificates. However, such certificate chain structure may not be followed by issuers who are not subject to public monitoring and auditing. This paper takes a first look at certificate chains involving certificates issued by issuers that do *not* appear in public databases (e.g., major browsers' root stores and CCADB). Utilizing a year's worth of TLS traffic collected from a campus network, we dissect the certificate chain structures and analyze their usage in TLS connections. While we observe positive acts such as the logging of certificates that are issued by issuers outside public databases and anchored to trust roots into Certificate Transparency (CT) logs, we also identify potential misconfigurations by servers where unnecessary certificates are included in the certificate chains, which may lead to validation and connection failures.

CCS Concepts

• **Networks** → **Network measurement**; • **Security and privacy** → **Security protocols**.

Keywords

Transport Layer Security; Public Key Infrastructure; Certificate chain; Certificate

ACM Reference Format:
Hongying Dong, Yizhe Zhang, Hyeonmin Lee, and Yixin Sun. 2025. Inside Certificate Chains Beyond Public Issuers: Structure and Usage Analysis from a Campus Network. In *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25), October 28–31, 2025, Madison, WI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3730567.3764503

1 Introduction

The Public Key Infrastructure (PKI) is the cornerstone of securing communications over the Internet. Digital certificates [10] are issued by Certificate Authorities (CAs) in a hierarchical structure, chaining from the root certificates, through the intermediate certificates, to the leaf certificates that bind to end entities (e.g., domain names). Consequently, certificate validation also follows this hierarchy by verifying through the *certificate chain.*

Given the importance of digital certificates, many studies have analyzed their characteristics and usage [11–13, 18, 21, 24, 26]. However, most studies focused on individual certificates with limited attention to the certificate chain structure, which can reveal CA practices in managing their chains and server practices in delivering the chains. A recent study [15] analyzed the characteristics of certificate chains using publicly available certificate data from Certificate Transparency (CT) [25] logs (i.e., `crt.sh` [14]). However, since CT logs only include certificates chained to trusted public issuers, a significant portion of certificates—over 60% [13, 18]—that are self-signed or chained to issuers outside public databases (e.g., major browsers' root stores [9, 27, 28] or CCADB [19]) are excluded, leaving their chain structures unstudied.

In this paper, we take a first step towards dissecting the structure of chains involving certificates issued by issuers that are not included in public databases [9, 19, 27, 28]. For clarity, we refer to issuers listed in such databases as *public-DB issuers*, and those absent as *non-public-DB issuers*; detailed definitions are provided in Section 3.2.1. We use SSL logs (with TLS connection data) and X.509 logs (with certificate data) collected from a large campus network from 2020-09-01 to 2021-08-31 (12 months). Our dataset consists of 259.30 million TLS connections involving certificate chains associated with non-public-DB issuers and 731,175 unique certificate chains, encompassing a total of 743,993 distinct certificates. We further conduct a retrospective analysis to evaluate the current state of servers utilizing chains from non-public-DB issuers by scanning the servers and comparing the obtained chains with our logs. We examine how certificate chains associated with non-public-DB issuers are structured and used, which is largely unexplored. Our key findings include:

1) *Positive indicators*: Most leaf certificates that are issued by non-public-DB issuers and anchored to a public trust root are correctly logged in CT logs; over 98% of chains with more than one certificate and solely from non-public-DB issuers provide a well-formed certificate chain.
2) *Unnecessary certificates*: Many certificate chains delivered by servers, even when anchored to a public trust root, include unnecessary certificates—certificates that do *not* contribute to the construction of the chain of trust. This potentially leads to inconsistent validation results across applications: browsers like Chrome often succeed using maintained trust stores, while mechanisms relying solely on presented chains may fail.
3) *Shift towards Let's Encrypt*: Many domains have transitioned from chaining leaf certificates issued by non-public-DB issuers to public trust roots, to using public issuers like Let's Encrypt, reflecting a preference for automated, user-friendly solutions that minimize misconfigurations and validation failures.

ACM ISBN 979-8-4007-1860-1/2025/10
https://doi.org/10.1145/3730567.3764503

Artifacts. Due to Infosec and IRB regulations, we are unable to share the original campus network data. Instead, we make other non-sensitive data and tools available [1].

2 Background and Related Work

Certificate chain verification and non-public-DB issuers. A certificate chain is a sequence starting from a leaf (end-entity) certificate, followed by one or more intermediates, and ending at a root certificate. Chain validation involves checking issuer–subject name matches, verifying digital signatures using issuer public keys, and ensuring revocation status and validity periods. Trusted root certificates, often pre-installed in systems or maintained in root stores, anchor this validation process. A chain is valid if it terminates at a trusted root after successfully following the chain. Some organizations operate non-public-DB issuers. While such chains can be cryptographically validated if complete, they are generally not trusted unless anchored to a recognized root [9, 27, 28]. Prior studies [13, 18] show many certificates fail validation against public trust stores. These chains remain understudied, despite potentially exhibiting unique structures outside the constraints of public issuer policies.

Related work. Research on digital certificates for TLS has been extensive, but most studies have focused on individual certificates issued by (trusted) public issuers [11–13, 18, 21, 24, 26], with limited attention to certificate chains, especially those associated with non-public-DB issuers. One notable study explored cross-signing practices across the PKI ecosystem [22], encompassing both public-DB and non-public-DB issuers. For certificates issued by non-public-DB issuers, studies such as [13, 18] employed active scanning and CT logs to examine their prevalence. A recent study [16] highlighted problematic practices associated with these certificates, leveraging datasets from passive measurements. However, none of these studies examined the associated certificate chains. Only a few studies have explored certificate chains [15, 17, 23]. Early work [17, 23] assessed chain lengths and identified unnecessary certificates, but without deeper analysis such as chain structures or related issuers. The most recent study [15] conducted a more detailed analysis of chain structures, but focused exclusively on CT-logged chains issued by public issuers for high-profile domains, leaving chains involving non-public-DB issuers unexamined. This work addresses this research gap with the first analysis of certificate chains associated with issuers outside public databases, offering new insights into their structure and real-world usage.

3 Dataset

3.1 Data Collection

We collaborated with the university's Information Security Department to conduct passive data collection at the university's border gateway over a continuous 12-month period, from September 1, 2020, to August 31, 2021. Throughout this time, raw traffic at the border gateway was processed using the Zeek network monitoring software [33], and two key log files, `SSL.log` and `X509.log`, were streamed to a secure cluster, where we conduct further analysis:

- Zeek's `SSL.log` file captures a broad spectrum of network traffic that utilizes the TLS protocol. `SSL.log` uses dynamic protocol detection (DPD) [8] to identify TLS traffic, as well as provide comprehensive connection details such as IP address, port number, Server Name Indication (SNI), certificate chain information, and connection status.
- The `X509.log` file generated by Zeek provides detailed information on certificates exchanged during TLS handshakes. This log contains attributes such as the certificate's issuer, subject, serial number, validity period, and encryption details. Each certificate entry in `X509.log` is cross-referenced with corresponding `SSL.log` entries using unique identifiers, offering a complete picture of certificate use within TLS connections.

Utilizing data from `SSL.log` and `X509.log`, our study examines both the structural characteristics of certificate chains and their actual usage in TLS connections. In compliance with IRB and university policies, raw certificates were not collected; our analysis is based solely on structured log data.

Ethics. Our data collection and usage protocols were approved by the University's Infosec Department and the IRB. Only authorized fields from Zeek logs were used, with no access to raw traffic. All data is securely stored and processed within a protected university cluster, accessible only via a restricted network by trained, authorized personnel. Details are in Appendix A.

3.2 Data Enrichment

3.2.1 *Certificate classification.*

We begin by identifying certificates based on their issuers and subsequently categorizing the associated certificate chains.

Identifying non-public issuer-issued certificates. Zeek [33] leverages Mozilla Network Security Services (NSS) [28] for validating certificate chains. We expand this validation by including additional trust stores including Apple [9], Microsoft [27], and the Common CA Database (CCADB) [19]. The CCADB is a repository of root and intermediate certificate data contributed by participating public root store operators. To be included, an intermediate certificate must chain to a trusted root from a participating root program (e.g., Mozilla, Microsoft, Apple, Google, or Oracle) and either be technically constrained or subject to public, standards-compliant audits. We classify certificates as issued by public-DB or non-public-DB issuers as follows, similar to [16]:

- Issued by ***public-DB issuers***: If the issuer, either the intermediate or root certificate, is listed in at least one of those major Web PKI root stores [9, 27, 28] or CCADB [19].
- Issued by ***non-public-DB issuers***: If the issuer's certificate is not included in any of the Web PKI root stores or CCADB. This also includes self-signed certificates that are not present in the Web PKI root stores or CCADB.

Identifying TLS interception cases. A distinct subset of certificates issued by non-public-DB issuer is associated with TLS interception, where encrypted traffic is decrypted and re-encrypted by an intermediary using its own private key, such as a firewall performing deep packet inspection [4]. This process alters the issuer information and may bias our analysis. To identify TLS interception, we first filter connections whose leaf certificate issuers do not appear in major trust stores, suggesting possible non-public-DB issuers. We then cross-reference CT logs [14] to check whether the observed issuer matches any recorded issuer for the same domain and certificate validity period. A mismatch implies possible interception, for which we conduct manual investigation through web search. As a result, we identify 80 TLS interception issuers and categorize them in Table 1. The largest group consists of security and network software vendors, including providers of security monitoring tools and network filtering services such as *Zscaler*, *McAfee*, *FireEye*, and *Fortinet*. In addition to 26 corporate entities, such as *Freddie Mac*, we also observe cases where TLS interception is conducted by organizations across various sectors, including public schools, educational software (e.g., *Securly*), finance company (e.g., *Nationwide*), and multiple U.S. government departments.

Category	#. Issuers	% Connections	#. Client IPs
Security & Network	31	94.74	17,915
Business & Corporate	27	4.99	4,787
Health & Education	10	0.02	35
Government & Public Service	6	0.24	25
Bank & Finance	3	0.00	14
Other	3	0.00	73

Table 1: Categories of issuers conducting TLS interception.

These interception activities are likely the result of institutional security policies and the deployment of security software on client-side machines. See further discussion in Appendix B. While this method cannot detect cases where the original certificate was issued by a non-public-DB issuer and thus absent from CT logs, our goal is not to identify all interception instances, but rather to provide a best-effort approach for filtering out a cleaner set of certificate chains involving non-public-DB issuers for structural analysis. We also discuss considerations for additional scenarios in Appendix B.

3.2.2 ***Certificate chain categorization.*** We next categorize certificate chains based on the certificate classification above into the following categories:

- ***Public-DB-only***: Certificate chains that exclusively comprise certificates issued by public-DB issuers.
- ***Non-public-DB-only***: Certificate chains that exclusively comprise certificates issued by non-public-DB issuers, excluding those associated with TLS interception cases.
- ***Hybrid***: Some certificate chains were found to include certificates issued by both public-DB and non-public-DB issuers. Thus, we separately analyze these chains.
- ***TLS interception***: Certificate chains containing certificates issued by entities identified as performing TLS interception.

	Non-public-DB-only	Hybrid	TLS int.
#. Cert chains	429 K	321	301 K
#. TLS connections	216.47 M	78.26 K	42.75 M
#. Client IPs	231,228	11,933	19,149

Table 2: Statistics of certificate chains. 'TLS-int.' stands for 'TLS-interception'.

We show the statistics of certificate chains in Table 2. We collect 731,175 certificate chains, of which 28% are associated with non-public-DB issuers (i.e., *non-public-DB-only*: 16.24%, *hybrid*: 0.02%, *TLS interception*: 11.19%). These chains are delivered in 259.30 million TLS connections, involving over 231,000 unique client IPs (see Appendix C for port distribution). Note that a single client IP may represent multiple clients, as our network traffic is subject to Network Address Translation (NAT).

4 Certificate Chain Analysis

In this section, we take a close look into certificate chains associated with non-public-DB issuers delivered by servers.

4.1 Certificate Chain Length

We first examine the length of certificate chains across the categories, as shown in Figure 1. Three chains are excluded from Figure 1 due to their unusually long lengths of 3,822, 921, and 41—possibly resulting from misconfigurations, as each was observed only once. These are all *non-public-DB-only* chains resulting in *unestablished* TLS connections.

First, we observe that over 60% of *public-DB-only* chains are advertised with a length of 2, which is consistent with prior work [15], as root certificates are often omitted from the delivered chain [31]. In contrast, the majority (80%) of *non-public-DB-only* chains consist of a single certificate, while 20% have a length greater than 1, suggesting that there may exist some structure for these chains. For *TLS interception* chains, more than 80% consistently include 3 certificates. However, *hybrid* chains stand out by exhibiting the widest range of chain lengths, with no dominant length.

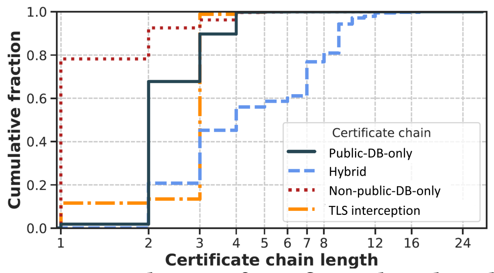

Figure 1: Distribution of certificate chain length.

Our findings highlight that certificate chains containing certificates issued by non-public-DB issuers (i.e., *non-public-DB-only*, *hybrid*, and *TLS interception*) exhibit distinct characteristics compared to *public-DB-only* chains. In particular, *hybrid* chains lack a dominant chain length, implying potentially diverse behaviors in chain construction. Next, we explore how these chains are structured, starting with *hybrid* chains.

4.2 Hybrid Certificate Chains

Methodology. To investigate the validity of certificate chains, we use a combination of methods, as illustrated in Figure 2. Unfortunately, the X509 logs did not capture public keys and signatures of certificates. Instead, we focus on two fields in each certificate: the certificate *issuer*, representing the entity that issued the certificate, and the certificate *subject*, representing the entity the certificate was issued to. For each certificate chain, we verify whether the issuer of one certificate matches the subject of the next certificate. In *hybrid* chains, cross-signing by public-DB issuer-issued certificates may occur, potentially leading to inaccurate validation results.

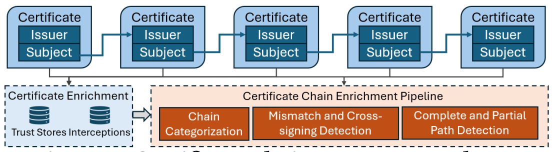
Certificate
Issuer
Subject
Certificate Enrichment
Trust Stores
Interceptions
Certificate Chain Enrichment Pipeline
Chain Categorization
Mismatch and Cross-signing Detection
Complete and Partial Path Detection

Figure 2: Certificate chain structure analyzer.

To address this, we identify cross-signed certificates by comparing our matching results with Zeek's validation results and CA announcement [32]. Our objective is to investigate mismatches between issuer and subject certificates that may undermine the establishment of trust. We provide further details and a comparison with the public key–signature validation in Appendix D.

We define the following terminologies for our further analysis, where the term *path* refers to a sequence of certificates:

- ***Complete matched path***: A certificate path where all issuer-subject pairs match throughout the chain and include a valid leaf certificate.
- ***Mismatch ratio***: The proportion of mismatched issuer-subject pairs to the total number of issuer-subject pairs within the chain.

Figure 3 illustrates two certificate chains: the top chain represents a *complete matched path* with no additional certificates in its chain, while the bottom chain includes a partially matched path (without a valid leaf certificate), a *complete matched path*, and an additional leaf certificate, resulting in a mismatch ratio of 0.4.

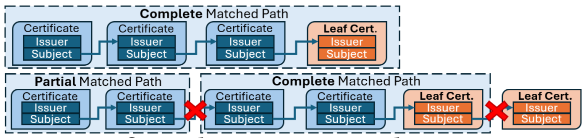

Figure 3: Certificate chain structure. Red crosses suggest mismatched issuer-subject pairs.

Structures of *hybrid* chains. We identify 321 *hybrid* chains, in which many public-DB issuer-issued intermediate certificates are used across various *hybrid* chains. We show details of these chains in Table 3 and in Appendix E. We also identify a pattern where an intermediate certificate is linked to a root, which is subsequently followed by another intermediate certificate in the same chain. Furthermore, many public-DB issuer-issued intermediate certificates are subsequent to non-public-DB issuer-issued roots, indicating potential misconfigurations within the delivered certificate chains. Notably, 19 servers present multiple distinct *hybrid* chains over the observed period. This behavior can be attributed to two primary factors: (1) replacement of leaf certificates due to expiration, and (2) inclusion of different unnecessary certificates. We then proceed to examine the constitution of these certificate chains in more detail.

Hybrid chain category			#. Chains
(1) Chain **is**	*Complete matched path*	Non-pub. chained to Pub.	26
		Pub. chained to Prv.	10
(2) Chain **contains**		-	70
(3) **No**		-	215
Total			321

Table 3: Statistics of *hybrid* certificate chains.

Certificate chain *is* a complete matched path. As shown in Table 3, 36 *hybrid* chains consist solely of a *complete matched path*, without additional certificates (e.g., the upper chain in Figure 3). Of these, 97.56% of associated connections are successfully established [1]. 26 chains feature non-public issuer-issued leaf certificates anchored to public trust roots and are served from various regions. [2] These signing issuers are typically affiliated with their respective non-public-DB issuers and appear to support domain-specific or localized services (details in Appendix F.1). Standards [20, 25] specify that non-public-DB issuer-issued leaf certificates chained to public-trust roots and used for public-facing domains are required to be logged in CT logs. We query CT logs and confirm that all of these leaf certificates are properly logged. However, 3 certificate chains containing expired leaf certificates, with the longest expiration period exceeding 5 years. Additionally, 10 certificate chains exhibit a pattern where a *complete matched path*, consisting of public-DB issuer-issued leaf and two intermediates, are followed by a non-public-DB issuer-issued certificate whose subject matches the issuer of the preceding certificate but has a different issuer. These chains are associated with backend service domains, likely reflecting internal infrastructure deployments (see Appendix F.1).

[1]We refer to the "established" field in Zeek SSL.log files.

[2]The USA, Brazil, and South Korea.

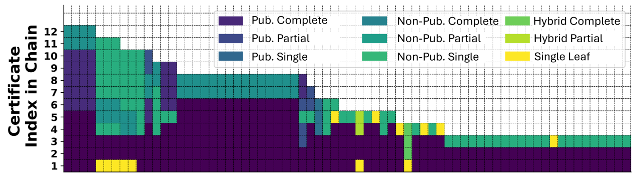

Figure 4: Chain structures of hybrid certificate chains that *contain* a complete matched path. Index 1 indicates the bottom of the trust hierarchy.

Certificate chain *contains* a complete matched path. 70 (21.81%) *hybrid* chains contain a *complete matched path* linked to public anchors but also include additional certificates issued by non-public-DB and/or public-DB issuers that are not part of the matched path, as shown in Table 3. 92.04% of the associated connections are successfully established. Figure 4 illustrates structures of these 70 chains. Each colored cell represents a certificate, and a vertical column of cells represents a certificate chain. The y-axis denotes the position of each certificate within the chain. While some instances appear to result from poor certificate management practices in corporate environments—where enterprise-issued self-signed certificates are mistakenly appended to otherwise valid chains—many others stem from misconfigured certificate management software that generates and inserts these unnecessary certificates into the chain (see Appendix F.2). Additionally, while the majority of chains include unnecessary certificates appended after the *complete matched path*, several chains begin with a leaf certificate followed by the *complete matched path*. In these cases, the issuer of the leaf certificate does *not* match the subject of the subsequent certificate. This chaining practice can lead to certificate chain validation failures, as discussed in Section 5.

No complete matched path. 215 (55.56%) chains do *not* contain any *complete matched path*. These were observed in 38,085 connections from 4,987 client IPs, of which 57.42%—corresponding to 2,937 IPs—were successfully established. The mismatch ratios for these chains range from 0.1 to 1.0, with 56.74% exhibiting a ratio of 0.5 or higher (see Appendix G), indicating a broad spectrum of misconfiguration types (see Appendix F.3). Notably, 56 chains include a public-DB issuer-issued leaf certificate but lack any intermediate certificate that issues the leaf. These 56 chains appeared in 19,366 TLS connections from 4,444 distinct client IPs. 56.08% of these connections were successfully established, involving 2,772 client IPs.

Established connections. TLS connections delivering certificate chains that are *complete matched paths* exhibit a significantly higher establishment rate (97.69%) compared to those that contain a *complete matched path* (92.04%) and those lacking any matched issuer-subject pairs (55.56%). This suggests that unnecessary certificates may contribute to certificate chain validation failures, resulting in unsuccessful connections. Further details are discussed in Section 5.

(*Takeaway 4.2*) Hybrid certificate chains exhibit diverse structures, with only a subset forming complete trust paths without unnecessary certificates. Chains containing unnecessary certificates often result from mismanagement or misconfiguration of certificate-related software and are associated with lower TLS connection success rates, suggesting that structural irregularities may hinder trust establishment.

4.3 Non-public-DB-only and Interception Certificate Chains

Methodology. For chains with multiple certificate, we apply the same issuer-subject comparison approach described in Section 4.2. However, unlike the *hybrid* chains, we do *not* evaluate the presence of a leaf certificate; certificates from and chained to non-public-DB issuers are not bound by the standards applied to public issuers and often lack the `basicConstraints` extension [10], making it challenging to identify leaf certificates. For example, we observe that 55.31% of non-public-DB issuer-issued certificates that are first presented in a chain (as delivered to the client) omit this extension, and 78.32% of those presented subsequent to the first also omit it, rather than explicitly setting it to a boolean value (`TRUE` or `FALSE`) as required by the specification [10]. Instead, we focus on determining whether a matched path exists within the chain and whether unnecessary certificates are included.

Single-certificate chains. 78.10% of *non-public-DB-only* chains consist of a single certificate, 94.19% of which are self-signed (i.e., issuer and subject are identical). These chains are delivered by various servers and appear in 140 million TLS connections from 221,924 client IPs, with 86.70% observed in sessions lacking an SNI. We identify 80 entities as TLS interception issuers (see Appendix B). 13.24% of *TLS interception* chains are single-certificate, 93.43% of which are self-signed, indicating that *TLS interception* chains tend to exhibit more complex structures than *non-public-DB-only* chains.

Single-certificate chains - Special case (DGA). Among single-certificate chains with *distinct* issuer and subject fields, we identify a cluster associated with domains generated by a Domain Generation Algorithm (DGA). Certificates in this cluster follow a consistent format, with both the issuer and subject fields containing randomly generated domain names (distinct from each other) that adhere to the same pattern.[3] The validity periods of these certificates range randomly from 4 to 365 days. These certificate chains are presented to 761 client IP addresses in 21,880 connections.

Chains with more than one certificate. We summarize statistics for *non-public-DB-only* and *TLS interception* chains in Appendix H. We observe a notably high proportion of matched paths in these chains: 99.76% of *non-public-DB-only* chains and 98.94% of *TLS interception* chains with more than one certificate. This high ratio suggests that chains associated with non-public-DB-only or *TLS interception* generally maintain a complete matched path without unnecessary certificates.

While most non-public-DB-only certificate chains employ a straightforward PKI structure—where intermediate certificates are linked to at most two other intermediate certificates across all chains—we identify certificate chains that adopt more intricate PKI structures. In these cases, intermediate certificates are linked to at least three distinct intermediate certificates across different chains; see Appendix I for details.

(*Takeaway 4.3*) Certificate chains associated with non-public-DB-only and *TLS interception* often consist of self-signed, single-certificate chains. Meanwhile, multi-certificate chains typically are complete matched paths, though instances of anomalous usage patterns highlight the diverse and sometimes opaque nature of non-public-DB issuer deployments.

5 Revisit: Hybrid and Non-Public-DB-Only Certificate Chains

To assess the current status of *hybrid* and *non-public-DB-only* certificate chains, we conducted an experiment in November 2024. We used an AWS instance to connect to servers previously identified as delivering those chains using OpenSSL [30] `s_client` command
`openssl s_client -connect $domain:443 -showcerts`
to retrieve their certificate chains.

Hybrid chains. We successfully accessed 270 out of 321 servers that previously delivered *hybrid* chains. Of these, 231 now deliver chains entirely issued by public-DB issuers with the majority being Let's Encrypt, which differs from their previous public-DB issuers. On the other hand, 4 present chains entirely issued by non-public-DB issuers. Among the remaining 35 servers that still have *hybrid* chains, 9 deliver chains with a *complete matched path* containing no unnecessary certificates, and 3 provide chains with a *complete matched path* with unnecessary certificates. The rest present chains without any matched paths. We further validated 3 certificate chains with a *complete matched path* anchored to a public trust root but containing unnecessary certificates, using OpenSSL and Chrome. Interestingly, the two tools produced different validation results. Chrome successfully validates these chains as long as a public-DB issuer-issued leaf certificate is present and its corresponding trust anchor is included in Chrome's trust store. In contrast, OpenSSL yields different results due to variations in validation mechanisms and the trust anchors maintained by the host (e.g., the OS) running the application. This suggests that unnecessary certificates may play a role in chain validation failures, potentially explaining the lower establishment rate observed in Section 4.2.

Non-public-DB-only chains. Among all connections previously identified as delivering *non-public-DB-only* chains, 341,356 (79.49%) were presented without an SNI. Thus, we were only able to extract and access 12,404 servers. All servers still use *non-public-DB-only* chains. However, 9,849 servers (79.40%) now deliver *non-public-DB-only* chains with more than one certificate. Of these, only 3,841 (39.00%) previously used chains longer than one, while 53.44% formerly served a single self-signed certificate (i.e., with identical issuer and subject) and the remaining 7.56% delivered a single certificate with distinct issuer and subject. In other words,

[3] `www[dot]randomstring[dot]com`.

over 60% of servers transitioned from delivering a single certificate to a longer certificate chains, highlighting a trend toward adopting a more hierarchical or complete chain structure. 97.61% of these chains are *complete matched paths* without unnecessary certificates, consistent with our previous observation.

(*Takeaway 5*) Most servers that used *hybrid* chains have transitioned to public-DB issuers, especially Let's Encrypt. At the same time, non-public-DB-only certificate chains are adopting hierarchical chains. These trends reflect a move toward automated and standards-aligned certificate management. Meanwhile, client-side validation inconsistencies underscore the risks posed by unnecessary certificates to trust establishment.

6 Discussion

6.1 Practical impact

Including unnecessary certificates in TLS chains introduces several operational challenges that affect connectivity, performance, and interoperability.

Inconsistent chain validation outcomes. Unnecessary certificates can lead to inconsistent validation outcomes across different applications (see Section 5). Major browsers such as Chrome often succeed in validating these chains because they rely on local trust stores to complete the chain. However, applications with alternative validation logic, particularly those that validate chains strictly based on the presented certificates, such as custom PKI implementations or OpenSSL with specific verification options—may reject the same chains due to the inability to construct a valid trust path. Although our study does not directly evaluate the impact of inconsistent chain validation, applications that fail to validate such chains could, in practice, encounter both availability and security issues: from higher connection retry rates or simple connection failures to downgrades to insecure protocols (e.g., HTTP), which in turn might open avenues for attackers to exploit. Furthermore, such inconsistencies can also result in fragmented reliability, where browsers and other applications experience different availability of the same server. This highlights the importance of adhering to chain construction best practices to ensure broad interoperability and maintain security.

Bandwidth and latency costs. Unnecessary certificates increases the TLS handshake latency and consumes additional network bandwidth due to additional data transmission.

6.2 Observed trends and recommendations.

In response to these challenges, we identify emerging deployment trends and recommend directions to improve certificate chain management.

Shift towards Let's Encrypt. From our campus data collection to our retrospective study in 2024, we have observed a shift towards using public-DB issuers—mainly Let's Encrypt—for previously *hybrid* chains that did not involve Let's Encrypt. This shift aligns with the rise in certificates issued by Let's Encrypt [6], indicating a growing preference for automated, user-friendly certificate management, especially for servers that previously struggled with properly delivering *hybrid* chains. This shift reduces the risk of misconfigurations and potential validation failures associated with *hybrid* chains.

Need for improved support. The observed shift highlights the benefits of automated and user-friendly certificate management. Our analysis reinforces the importance of this trend: We find that many unnecessary certificates in chains originate from poor certificate management and misconfigured certificate management software. These underscore the need for improved tooling and automation to reduce human error and ensure consistent, interoperable TLS deployments.

6.3 Generalization, limitations, and future work

Our study leverages campus network traffic to analyze real-world usage patterns (e.g., number of established connections and clients) associated with different certificate chain structures. While our campus, serving over 35,000 registered users (with additional guest users), offers generalizability to similar environments—such as other educational institutions and open-access networks—results may vary across network types. For instance, enterprise or government networks may enforce stricter security policies, centralized device management, and more uniformed configurations, in contrast to the heterogeneous and ad hoc nature of campus environments. Also, campus traffic tends to be more education-focused [16], differing from residential usage patterns. Another limitation is that passive monitoring cannot capture certificates for TLS 1.3 due to encryption, which comprises about a quarter of TLS connections. Additionally, our issuer–subject validation method effectively detects subject–issuer mismatches but cannot identify malformed certificates or public key integrity issues (see Appendix D). Future studies may generalize and broaden the certificate chain analysis by performing active scanning of the entire IP address space, combined with network traffic logs from operators to obtain connection statistics to pinpoint the actual usage of the chains for TLS clients.

7 Conclusion

In this paper, we analyzed certificate chains associated with issuers outside public databases using data collected from a campus over 12 months and provided new insights into previously unexplored practices and structures. While we observe a positive trend toward public issuers like Let's Encrypt—highlighting a shift toward automation and reduced misconfiguration—we also identify cases where unnecessary certificates in chains can lead to overlooked validation failures, underscoring the need for improved tooling and automation to reduce human error and enhance the interoperability and robustness of TLS deployments.

Acknowledgments

We thank anonymous reviewers for their insightful and constructive suggestions and feedback. This work is supported by National Science Foundation CNS-2154962 and CNS-2319421, and the Commonwealth Cyber Initiative. Results from the retrospective study presented in this paper were obtained using CloudBank [29], which is supported by the National Science Foundation under award CNS-1925001.

References

[1] 2025. Artifact for "Inside Certificate Chains Beyond Public Issuers:Structure and Usage Analysis from a Campus Network". https://github.com/yzzhn/certchain validator (accessed Sep 30, 2025).

[2] 2025. Athenz, an open source platform for X.509 certificate based service authentication and fine-grained access control in dynamic infrastructures. https://github.com/AthenZ/athenz (accessed Jun 16, 2025).
[3] 2025. Fortinet, FortiGate open ports. https://docs.fortinet.com/document/fortigate/6.4.0/ports-and-protocols/303168/fortigate-open-ports(accessed Jun 16, 2025).
[4] 2025. Fortinet, ssl-tls-deep-inspection. https://docs.fortinet.com/document/fortigate/7.6.0/best-practices/598577/ssl-tls-deep-inspection)accessed Jun 16, 2025).
[5] 2025. Let's Encrypt Forum, CN=Fake LE Intermediate X1. https://community.letsencrypt.org/t/cn-fake-le-intermediate-x1/13437 (accessed Jun 16, 2025).
[6] 2025. Let's Encrypt Stats. https://letsencrypt.org/stats/)(accessed Jun 16, 2025).
[7] 2025. Welcome to pyca/cryptography. https://cryptography.io/en/latest/ (accessed Jun 16, 2025).
[8] 2025. Zeek, Dynamic protocol detection (DPD). https://docs.zeek.org/en/master/logs/dpd.html (accessed Jun 16, 2025).
[9] Apple. 2024. Available trusted root certificates for Apple operating systems. https://support.apple.com/en-us/103272 (accessed Oct 29, 2024).
[10] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. https://doi.org/10.17487/RFC5280
[11] Carl Magnus Bruhner, Oscar Linnarsson, Matus Nemec, Martin Arlitt, and Niklas Carlsson. 2022. Changing of the guards: Certificate and public key management on the internet. In *International Conference on Passive and Active Network Measurement*. Springer, 50–80.
[12] David Cerenius, Martin Kaller, Carl Magnus Bruhner, Martin Arlitt, and Niklas Carlsson. 2024. Trust Issue (r) s: Certificate Revocation and Replacement Practices in the Wild. In *International Conference on Passive and Active Network Measurement*. Springer, 293–321.
[13] Taejoong Chung, Yabing Liu, David Choffnes, Dave Levin, Bruce MacDowell Maggs, Alan Mislove, and Christo Wilson. 2016. Measuring and applying invalid SSL certificates: The silent majority. In *Proceedings of the 2016 Internet Measurement Conference*. 527–541.
[14] crt.sh. 2015. Certificate Transparency Logs. https://crt.sh/ (accessed Oct 29, 2024).
[15] Marcus Döberl, York Freiherr von Wangenheim, Carl Magnus Bruhner, David Hasselquist, Martin Arlitt, and Niklas Carlsson. 2024. Chain-Sawing: A Longitudinal Analysis of Certificate Chains. In *Proceedings of IFIP Networking*.
[16] Hongying Dong, Yizhe Zhang, Hyeonmin Lee, Kevin Du, Guancheng Tu, and Yixin Sun. 2024. Mutual TLS in Practice: A Deep Dive into Certificate Configurations and Privacy Issues. In *Proceedings of the 2024 ACM on Internet Measurement Conference*. 214–229.
[17] Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman. 2013. Analysis of the HTTPS certificate ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference*. 291–304.
[18] Syed Muhammad Farhan and Taejoong Chung. 2023. Exploring the Evolution of TLS Certificates. In *International Conference on Passive and Active Network Measurement*. Springer, 71–84.
[19] The Linux Foundation. 2024. Common CA Database. https://www.ccadb.org/ (accessed Oct 29, 2024).
[20] Google. [n. d.]. Certificate Transparency in Chrome. https://googlechrome.github.io/CertificateTransparency/ (accessed Nov 20, 2024).
[21] David Hasselquist, Ludvig Bolin, Emil Carlsson, Adam Hylander, Martin Larsson, Erik Voldstad, and Niklas Carlsson. 2023. Longitudinal Analysis of Wildcard Certificates in the WebPKI. In *2023 IFIP Networking Conference (IFIP Networking)*. IEEE, 1–9.
[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*. 1289–1306.
[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 2011 ACM SIGCOMM conference on Internet measurement conference*. 427–444.
[24] Deepak Kumar, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, J Alex Halderman, and Michael Bailey. 2018. Tracking certificate misissuance in the wild. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 785–798.
[25] Ben Laurie, Eran Messeri, and Rob Stradling. 2021. Certificate Transparency Version 2.0. RFC 9162. https://doi.org/10.17487/RFC9162
[26] Yabing Liu, Will Tome, Liang Zhang, David Choffnes, Dave Levin, Bruce 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 Internet Measurement Conference*. 183–196.
[27] Microsoft. 2024. Release notes - Microsoft Trusted Root Certificate Program. https://learn.microsoft.com/en-us/security/trusted-root/release-notes (accessed Oct 29, 2024).
[28] Mozilla Wiki. 2024. Mozilla's CA Certificate Program. https://wiki.mozilla.org/CA (accessed Oct 29, 2024).
[29] Michael Norman, Vince Kellen, Shava Smallen, Brian DeMeulle, Shawn Strande, Ed Lazowska, Naomi Alterman, Rob Fatland, Sarah Stone, Amanda Tan, Katherine Yelick, Eric Van Dusen, and James Mitchell. 2021. CloudBank: Managed Services to Simplify Cloud Access for Computer Science Research and Education. In *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions* (Boston, MA, USA) *(PEARC '21)*. Association for Computing Machinery, New York, NY, USA, Article 45, 4 pages. https://doi.org/10.1145/3437359.3465586
[30] OpenSSL. 2024. OpenSSL: Cryptography and SSL/TLS Toolkit. https://www.openssl.org/ (accessed Nov 20, 2024).
[31] Eric Rescorla and Tim Dierks. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. https://doi.org/10.17487/RFC5246
[32] Sectigo. [n. d.]. Sectigo Chain Hierarchy and Intermediate Roots. https://support.sectigo.com/articles/Knowledge/Sectigo-Chain-Hierarchy-and-Intermediate-Roots (accessed Nov 18, 2024).
[33] Zeek. 2023. An Open Source Network Security Monitoring Tool. https://zeek.org/ (accessed Oct 29, 2024).

A Ethics

The data logging procedure of this research has been closely supervised by the university's InfoSec department and ethically approved by the Institutional Review Board (IRB). The study is limited to specifically authorized fields within the processed Zeek logs, with no access to raw network traffic. All data storage and computations are securely conducted within the university's protected cluster, accessible solely through a restricted network. Additionally, data access is limited to authorized personnel who have completed the university's comprehensive security training. These measures effectively mitigate the risk of data leakage and ensure the protection of sensitive information.

B Identified TLS Interception Issuers

We observe that the vast majority of TLS interception activity—94.74% of connections—is attributed to issuers in the Security & Network category, involving 17,915 unique client IPs, indicating widespread deployment of security appliances and network monitoring tools across many endpoints. The Business & Corporate category accounts for only 4.99% of connections, yet still involves a substantial number of client IPs (4,787), suggesting that corporate environments also deploy various interception measures, though at a smaller scale. Other categories collectively account for less than 0.3% of all connections and involve very few clients (fewer than 100 IPs combined).

Other scenarios. Our TLS interception identification method cannot detect cases where the original certificate was issued by and chained to a non-public-DB issuer and is therefore absent from CT logs. Additionally, it cannot identify interception occurring through local proxies within sub-networks or encrypted VPN tunnels. In addition, there may be cases where a self-signed certificate falsely claiming to represent a given domain is served by a web server not actually associated with that domain. In such scenarios, the mismatch between the certificate's subject and the server's domain name would typically cause the TLS client to reject the connection. Successfully deceiving the client into accepting the certificate would require a compromised network environment that redirects traffic to the attacker's server, such as via DNS cache poisoning or BGP hijacking.

C Certificate Chain Port Distribution

Table 4 shows the port distribution for certificate chains across all non-public-DB issuer-associated categories. For *hybrid* chains and *non-public-DB-only* chains with multiple certificates, port 443 dominates, accounting for 97.21% and 83.51% of connections, respectively—indicating a typical HTTPS usage. In contrast, *non-public-DB-only* single-certificate chains and *TLS interception* chains exhibit more diverse port usage. While port 443 remains relevant (46.29% and 13.36%, respectively), a substantial portion of traffic uses non-standard ports such as 8888, 33854, and 8013. Notably, in the case of *TLS interception* traffic, port 8013 is frequently used by the *Fortinet* security software [3], which turns out to be a TLS interception use pattern, as the provider *Fortinet* is identified as a TLS interception issuer in our study. These trends suggest that many non-public-DB- and interception-related TLS deployments extend beyond conventional HTTPS channels.

Hybrid		Non-public-DB-only				Interception	
		Single Certs		Multiple Certs			
Port	%	Port	%	Port	%	Port	%
443	97.21	443	46.29	443	83.51	8013	35.40
8443	1.36	8888	21.52	8531	4.18	4437	25.14
8088	1.22	33854	19.08	9093	2.85	14430	16.34
25	0.18	13000	4.22	38881	1.81	443	13.36
9191	0.01	25	1.30	6443	1.45	514	3.53
Other	0.02	Other	7.59	Other	6.2	Other	6.23

Table 4: Port distribution of connections associated with *hybrid*, *non-public-DB-only*, and *TLS interception* certificate chains.

D Certificate Chain Validation Methodology and Evaluation

In this section, we first describe our proposed certificate chain validation methodology in details. We next evaluate our approach in comparison with the public key-signature-based certificate chain validation.

D.1 Issuer-Subject Validation Methodology

Due to the lack of public key and signature data in our X.509 logs, we focus our validation on two key certificate fields: the issuer, denoting the entity that issued the certificate, and the subject, denoting the entity to which the certificate was issued. For each certificate chain, we traverse certificates sequentially from the leaf upward, checking whether the issuer field of each certificate matches the subject field of the next certificate in the chain. When mismatches are detected, we record the index (i.e., position) of the conflicting issuer–subject pair.

It is important to note that certificate cross-signing can cause such issuer–subject mismatches to appear, even when the chain is technically valid. This is because cross-signed certificates may not follow a direct issuer–subject match, yet still form a valid chain trusted by root stores. To account for this, we identify potential cross-signing cases by comparing our results with those generated by Zeek's validation logic and cross-signing disclosures from CAs [32]. These cross-signing relationships, while not always evident from the certificate fields alone, are typically recognized by trusted root stores and must be explicitly considered to avoid false positives in the issuer-subject mismatch detection.

D.2 Methodology Evaluation

We evaluate our methodology in comparison with the public key-signature certificate chain validation approach.

Dataset. We build our validation dataset using certificate chains collected in November 2024. Specifically, we setup our client and connect to servers previously identified as delivering certificate chains involving non-public-DB issuers (i.e., *hybrid* and *non-public-DB-only* chains). As a result, we successfully obtain 12,676 certificate chains. We note that, because these chains were collected directly rather than through the campus network data collection infrastructure, we obtained the full PEM data, including public keys and signatures.

	Issuer-subject	Key-signature
#. Single-certificate chains	2,568	2,568
#. Valid chains	9,825	9,821
#. Broken chains	283	284
#. Chains with unrecognized keys	-	3

Table 5: Validation results of 12,676 certificate chains using the issuer-subject approach, in comparison with the key-signature method.

Certificate chain validation result. We validate these 12,676 certificate chains using the issuer–subject methodology adopted in our study, and compare the results with those obtained using the key–signature validation method implemented via the Python `cryptography` package [7]. Each certificate chain is traversed sequentially from the leaf upward, verifying whether the issuer field of each certificate matches the subject field of the next certificate (in issuer–subject method), or whether the certificate's signature can be verified using the public key of the next certificate in the chain (in key–signature validation method). Table 5 shows the validation results for both methods. Our issuer–subject method classifies 2,568 chains as single-certificate chains, 9,825 as valid multi-certificate chains, and 283 as broken certificate chains with issuer-subject mismatches. The key–signature method, meanwhile, identifies 2,568 single-certificate chains, 9,821 valid chains, 284 broken chains due to key–signature validation failures, and 3 certificate chains involving public keys that are not recognized by the Python `cryptography` package (these 3 certificate chains are identified as valid chains by the issuer-subject approach).

Notably, there is one certificate chain deemed valid by the issuer–subject method but invalid by the key–signature approach. Further inspection reveals that this discrepancy arises from an ASN.1 parsing error raised by the Python `cryptography` package, which causes the key–signature verification to fail. This case highlights a limitation of the issuer–subject methodology: it cannot detect malformed certificates or public key integrity issues internal to certificates. Nonetheless, such cases are rare. Apart from the three certificate chains with unrecognized public keys and the one with the formatting error, the issuer–subject validation results are fully consistent with those of the key–signature method. Additionally, our approach accurately identifies the position of each issuer-subject mismatch within broken chains, and these positions align with those identified by the key–signature validation.

Limitation. As discussed, our issuer–subject-based certificate chain validation approach is limited in that it cannot detect issues such as malformed certificates or public key integrity problems that occur at the formatting level. Beyond formatting errors, there is also the possibility of cases where all issuer–subject pairs within a chain appear consistent, yet the digital signature of a certificate cannot be validated using the public key of its issuer—due to either an incorrect key or potential certificate impersonation.

E Hybrid Certificate Chain Overview

Figure 5 shows the certificate structure and relationships within hybrid chains. Different colors represent certificates issued by public-DB issuers (blue) and non-public-DB issuers (red). Node sizes indicate the certificate's position in the chain: leaf certificates (smallest), intermediate certificates (medium), and root certificates (largest). Two nodes are connected by an edge if they are ever observed together in at least one certificate chain.

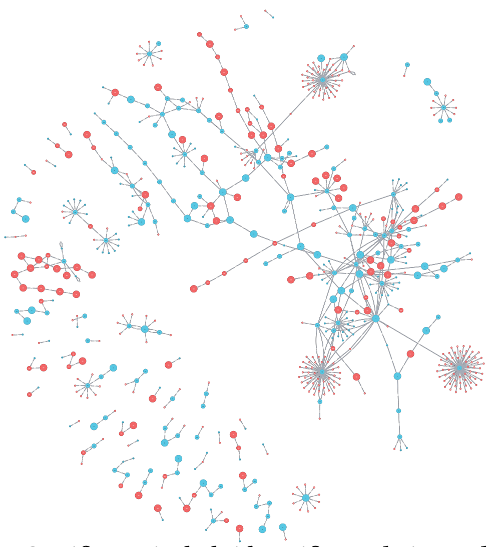

Figure 5: Certificates in *hybrid* certificate chain. Nodes with colors represent different certificate types: public-DB issuers (blue) and non-public-DB issuers (red).

F Issuer and Use Pattern of Hybrid Chains.

In this section, we explore the certificate issuers and the actual use pattern of those hybrid certificate chains.

F.1 Chain is a complete matched path

Category	Entity	#. Chains
Corporate	Symantec, SignKorea and others	10
Government	Korea, Brazil, USA	16

Table 6: Non-public-DB issuer-issued certificate chained to public trust anchors.

We identify 26 hybrid certificate chains in which non-public-DB issuer-issued leaf certificates are anchored to public trust roots. As summarized in Table 6, these chains fall into two categories, each involving a non-public-DB issuer linked to a publicly trusted intermediate or root CA. These configurations appear to serve domain-specific or localized applications. For instance, a chain used by the U.S. Department of Veterans Affairs includes a leaf certificate issued by "Veterans Affairs CA B3," chained through "Verizon SSP CA A2" to a root certificate within the U.S. Federal PKI. Similar patterns are observed in chains associated with the Government of Korea (KLID) and Brazil's national PKI authority (Instituto Nacional de Tecnologia da Informação, ITI), where non-public-DB issuer-issued certificates are anchored to publicly trusted roots. Comparable behavior is also found in corporate deployments such as Symantec, where "Symantec Private SSL SHA1 CA" issues leaf certificates chained to Symantec's own trusted root.

We also identify 10 certificate chains that exhibit an unusual structure: each is a *complete matched path* issued by both public-DB and non-public-DB issuers, where the public-DB issuer-issued leaf and two intermediate certificates are followed by an additional non-public-DB issuer-issued certificate. In these cases, the certificate issued by a non-public-DB issuer has a distinct issuer and subject, with the subject matching the issuer of the preceding public-DB issuer-issued certificate. These chains are served by hosts operated by two organizations: *Scalyr* and *Canal+*. For example, the domain *app.scalyr.com* presents a leaf certificate issued by the public-DB issuer *Sectigo*, which chains correctly to a public-DB issuer *AAA Certificate Services*. However, this intermediate certificate is then followed by a certificate issued by a non-public-DB issuer *Scalyr* itself, with the subject referencing the public-DB issuer *AAA Certificate Services*. These chains are associated with several backend service domains, such as **.canal-plus.com* and *app.scalyr.com*, suggesting that this may reflect an internal infrastructure deployment. Additionally, over 98.49% of connections to these servers were successfully established.

F.2 Chain Contains a Complete Matched Path

Among the 70 chains containing a *complete matched path*, 14—each from a distinct domain—exhibit misconfigurations involving the software toolkit provided by the public-DB issuer Let's Encrypt. These chains contain a valid path that terminates at a Let's Encrypt root certificate, but are incorrectly followed by an additional certificate with issuer `Fake LE Root X1` and subject `Fake LE Intermediate X1`. This certificate is a default placeholder generated by Let's Encrypt's staging environment when the `-test-cert` or `-dry-run` option is used during certificate renewal [5]. The presence of this certificate suggests that domain operators inadvertently deployed staging test certificates in production environments, potentially disrupting server connectivity due to differences in client-side certificate validation behavior.

The remaining 56 certificate chains, spanning 41 distinct domains, exhibit varying degrees of misconfiguration. Although each includes a *complete matched path* issued by public-DB issuers (e.g., *DigiCert*, *COMODO*, *Sectigo*, *GoDaddy*), unnecessary certificates are appended at different points within the chains. Many such additions appear to stem from corporate environments. For example, one domain [4] appends a self-signed internal HP certificate—with both issuer and subject common name (`CN`) set to "tester"—to an otherwise valid chain (i.e., a *complete matched path*). In some cases, domains attach multiple root certificates from different public-DB issuers to the end of valid chains (i.e., *complete matched path*) across

[4] *webauth.hpconnected.com*

different connections. We also identify misconfigurations related to certificate management software. For example, we observe multiple certificate chains include a self-signed certificate issued by *Athenz* [2] appended to otherwise valid public-DB-only chains. These observations highlight how improper software configurations can lead to noncompliant and potentially incorrect certificate chain structures.

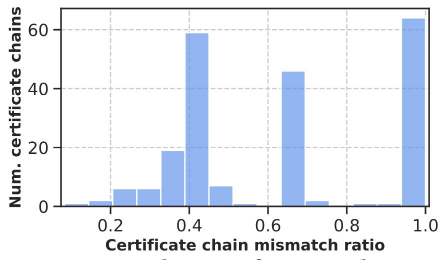

Figure 6: Distribution of mismatch ratios.

F.3 No Complete Matched Path

Table 7 shows our categorization of certificate chains that lacks a *complete matched path*. We observe various types of misconfigurations. The category "Non-public-DB self-signed leaf followed by mismatched issuer–subject pairs" reflects highly customized certificate setups. In particular, 100 out of 108 chains in this group use identical issuer and subject fields in the leaf certificate. [5] We also identify 13 chains (used by 13 distinct domains) where a self-signed certificate replaces the original leaf certificate in an otherwise valid public-DB-only chain. Additionally, 61 chains are completely broken, with no matching issuer–subject pairs, and 27 chains contain partial matches but fail to construct a *complete matched path*. Interestingly, we observe 5 chains that append a non-public-DB issuer-issued root after a truncated public-DB-only sub-chain. These misconfigurations likely stem from improper certificate management or erroneous TLS deployment practices.

Category	#. Chains
Non-pub-DB self-signed leaf followed by mismatched {issuer-subject} pairs	108
Non-pub-DB self-signed leaf followed by a valid sub-chain	13
All {issuer-subject} pairs are mismatched	61
Partial {issuer-subject} pairs are mismatched	27
Non-pub-DB root appended to a valid public-issued sub-chain	5
Non-pub-DB root and mismatched {issuer-subject} pairs	1

Table 7: Categorization of certificate chains without a complete matched path.

G Hybrid Certificate Chain Mismatch Ratios

We show the mismatch ratios of *hybrid* certificate chains in Figure 6.

H Non-Public-DB-Only and Interception Chains

We show statistics for *non-public-DB-only* and *TLS interception* chains with more than one certificates in Table 8. It is noteworthy that we observe a high proportion of *matched paths* in 99.76% of *non-public-DB-only* chains and 98.94% of *TLS interception* chains.

[5] `emailAddress=webmaster@localhost, CN=localhost, OU=none, O=none, L=Sometown, ST=Someprovince, C=US`

	Non-public-DB-only	TLS int.
Is a *matched path* (%)	99.76	98.94
Contains a *matched path* (#)	142	56
No *matched path* (#)	87	2,764

Table 8: Statistics of non-public-DB-only and TLS interception(i.e., TLS int-) chains.

I Complex PKI Structures in Non-Public-DB-Only and Interception Certificate Chains

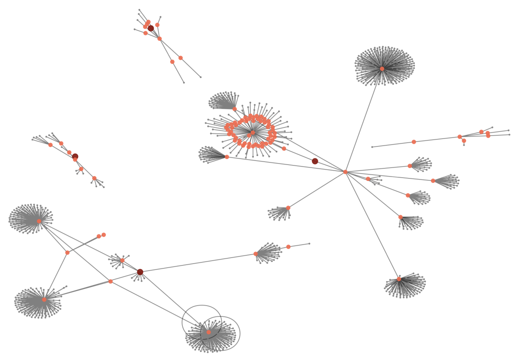

Figure 7: Certificates in non-public-DB-only certificate chain.

Figure 7 shows complex PKI structures observed in *non-public-DB-only* chains, where intermediate certificates are ever linked to at least three different intermediate certificates; in the figure, black nodes represent leaf certificates, orange nodes denote intermediate certificates, and dark red nodes indicate root certificates.

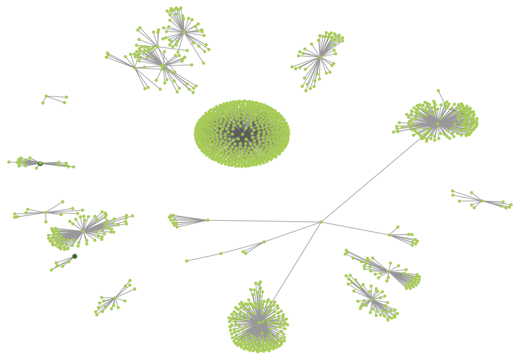

Figure 8: Certificates in TLS interception certificate chain. Leaf certificates are omitted.

Figure 8 illustrates the complex PKI structures observed in *TLS interception* chains, where intermediate certificates are ever linked to at least three different intermediate certificates.; in the figure, light green nodes represent intermediate certificates, and dark green nodes indicate root certificates. Note that leaf certificates are omitted.