Measuring and Characterizing Propagation of Reuse RSA Certificates and Keys across PKI Ecosystem

Fatemeh Nezhadian, Enrico Branca, Anna Barzolevskaia, Andrei Natadze, Natalia Stakhanova

***Abstract*—The insecurities of public-key infrastructure on the Internet have been the focus of research for over a decade. The extensive presence of broken, weak, and vulnerable cryptographic keys has been repeatedly emphasized by many studies. Analyzing the security implications of cryptographic keys' vulnerabilities, several studies noted the presence of public key reuse. While the phenomenon of private key sharing was extensively studied, the prevalence of public key sharing on the Internet remains largely unknown. In this work, we perform a large-scale analysis of public key reuse within the PKI ecosystem. We investigate the presence and distribution of duplicate X.509 certificates and reused RSA public keys across a large collection containing over 314 million certificates and over 13 million SSH keys collected by different sources at different times. We analyze the cryptographic weaknesses of duplicate certificates and reused keys and investigate the reasons and sources of reuse. Our results reveal that certificate and key sharing are common and persistent. Our findings show over 10 million certificates and 17 million public keys are reused across time and shared between our collections. We observe keys with non-compliant cryptographic elements stay available for an extended period of time.**

***Index Terms*—Communication Protocols, Peer-to-Peer Networks, Security Management, Security Services, Data Mining and (Big) Data Analysis.**

I. Introduction

IN 2021, Github revoked RSA keys, generated by the vulnerable GitKraken client's library that created duplicate SSH authentication keys [1]. In 2018, Infineon released a security patch to update the vulnerable Trusted Platform Module in its microcontrollers responsible for generating vulnerable RSA keys [2].

The sad state of public-key infrastructure on the Internet has been the focus of security research for over a decade. In 2011 Holz et al. [3] investigating the deployed X.509 certificates for TLS/SSL certification pointed out that infrastructure is broken, i.e., only one out of five certificates can be counted as valid, and many include cryptographically weak keys.

The presence of broken, weak, and vulnerable cryptographic keys on the Internet has been extensively investigated by a number of studies from different points of view. Some studies traced back the problem to weak random number generators and the lack of entropy [4]–[6], while others claimed a simple misuse of keys [7], and the improper implementation of cryptographic libraries as the main reasons [8]–[11].

Network appliances and services rely on certificates and keys to facilitate secure communication, code signing, authentication, and other security-related functionality. However, mishandling or misuse of certificates or keys poses a significant threat to the overall security of the PKI ecosystem.

Weaknesses in RSA keys allow for faster factorization, i.e., one can efficiently compute the corresponding private keys undermining the security of communication [4], [12]. When keys are reused across different devices and organizations, it exposes the corresponding private key, which can be exploited by attackers. This enables attackers to utilize certificates from benign sources to sign malicious software or impersonate legitimate companies. Consequently, an attacker who gains access to the private key or is able to regenerate it (e.g., in case of vulnerable and weak public keys) can access encrypted content and/or intercept secured network traffic.

Misuse of cryptographic algorithms and a combination of implementation decisions made in software libraries can lead to distinguishable patterns and consequently can be leveraged in identifying a probable origin of a key, e.g., an originating library, its specific version, and operating system [2], [10], [13], [14].

Investigating the reuse of public keys is crucial for understanding high-risk practices within legitimate procedures and for identifying compromised use cases. This phenomenon was initially observed by Heninger et al. [4] in a small study of 6.2 million SSH keys and 5.8 million TLS certificates, and later confirmed by Cangialosi [15]. Although the phenomenon of private key sharing with web hosting providers was extensively explored by Cangialosi et al. [15] and Liu et al. [16], the prevalence of public key reuse across domains remains largely unknown. This includes its occurrence in various applications such as TLS/SSL certificates for web environments, SSH keys for authentication means, and code-signing certificates for ensuring software integrity.

In this study, ① we conduct *the most comprehensive and the largest Internet-wide scan and analysis of TLS/SSL certificates and RSA keys across domains*. ② *We measure and characterize the extent of TLS/SSL certificates and RSA public key sharing across domains* on a diverse set of over 314 million valid certificates and 13 million SSH keys collected from multiple sources. These datasets represent snapshots of the certificates and keys used over time. ③ We develop and publicly offer a key analysis platform *KeyExplorer* that can examine the weaknesses of certificates and keys.

This study represents the most extensive measurement analysis of public RSA key reuse on the Internet to date. Our collection comprises 84 million unique RSA keys, which is 15 times greater than the set used previously in a seminal study conducted by Heninger et al. [4]. We conduct the lifespan analysis of the use of certificates over time and determine the reused certificates and keys across different devices, organiza-

Fatemeh Nezhadian, Enrico Branca, Anna Barzolevskaia, Andrei Natadze, Natalia Stakhanova are with the Department of Computer Science, University of Saskatchewan, Canada (email: flor.nezhadian@usask.ca, enb733@usask.ca, jhu168@usask.ca, esq785@usask.ca, natalia@cs.usask.ca).

tions, and domains. Our analysis shows that 6.5% (10,110,361) TLS certificates are shared across our collection. We find a considerable amount of shared certificates that are used to sign malware while still being served in communication by different hosts. We observe that 28% of certificates used in malicious binaries are also used by Android apps. This emphasizes the existing reuse of keys for different purposes and across domains.

We investigate the cryptographic characteristics of these reused certificates. Through our analysis, we uncover a multitude of security issues. These problems include an alarming number of weak and factorable keys, keys susceptible to the ROCA attack [2], and the use of deprecated and non-compliant signature algorithms. To delve deeper into the underlying causes of these vulnerabilities, we scan IPv4 addresses that serve the reused TLS certificates. Our findings indicate that the majority of reused certificates are associated with embedded network devices running operating systems derived from Linux and BSD distributions. These certificates are primarily utilized by devices manufactured by a small number of companies and have been in active use for an extensive period. For instance, we observed that 48,794 IP addresses, identified as routers, have been serving 950 distinct certificates for over two decades. Alongside duplicate certificates, we discover 14,862,767 identical RSA public keys that appear in multiple distinct TLS certificates, 44% of these shared keys can be considered weak.

In the final phase of our study, we trace the reused keys found in distinct certificates back to their origins. The investigation reveals that these keys are primarily generated by only a couple of versions of OpenSSL and GnuTLS libraries associated with a history of issues related to random number generation implementation.

Our findings emphasize the need for robust mechanisms for the detection and analysis of duplicate and weak certificates and keys. To facilitate this analysis, we implemented a correlation platform to determine duplicate keys and detect TLS certificates and RSA keys' weaknesses. We make the resulting duplicate keys publicly available: https://github.com/thecyberlab/RSA-keys-analysis. Furthermore, we develop a publicly open platform to examine the discussed weaknesses of certificates and keys: https://key-explorer.com/.

This paper is organized as follows: We briefly outline the characteristics of TLS/SSL and SSH protocols in Section II. Section III gives an overview of related work. Next, we explained the data collection in Section IV. Further, we explain our proposed approach in Section V. Section VI discusses our initial analysis. In Sections VII and VIII, we outlined our findings related to shared certificates and keys. Finally, we summarize the major findings and discuss their implications and recommendations in Section IX. We conclude in Section XI.

II. Background

The presence of vulnerable and weak cryptographic elements has a number of critical security implications. If a certificate or the corresponding cryptographic key is vulnerable,

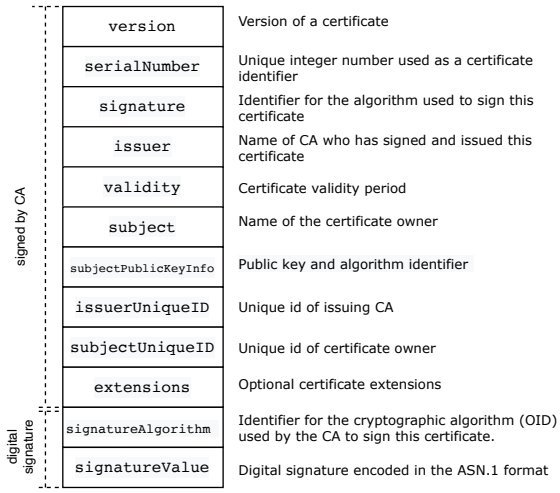

Fig. 1: X.509 certificate layout.

the encrypted content and/or secured network traffic can be accessible by an insider attacker (or an external attacker with access to a compromised machine inside the network) who has access to the private key or is be able to regenerate it. In spite of the strong mathematical foundations of cryptographic algorithms, several studies highlighted weaknesses in practical implementations of cryptographic protocols which rarely relate to the fundamental aspects of the algorithm's theoretic design.

This study focuses on the analysis and measurement of the presence and distribution of cryptographic materials in different sources and hosting environments and to this end, we first need to take a look at some general terms.

*a) **TLS/SSL**:* The Transport Layer Security (TLS) [17] and its predecessor the Secure Sockets Layer (SSL) are the cryptographic protocols that provide point-to-point encryption service to a number of application layer protocols such as HTTPS, DNS, SMTPS, IMAPS, etc. A TLS/SSL connection is initiated with a TLS handshake when a client requests a secure connection. At this stage, a server presents its digital certificate that allows the client to authenticate it. The client can also respond with its certificate. Once both parties agree on the encryption algorithm, further communication happens within the encrypted channel.

*b) **SSH**:* The SSH protocol [18] is a cryptographic communication protocol that ensures secure remote access and network services over potentially insecure networks. The SSH host runs an SSH server, which listens for incoming SSH connection requests from clients. During the initial phase, a client and a server exchange the supported cryptographic algorithms available to both parties, the server then sends its public key to the client to authenticate its identity.

*c) **Digital certificates**:* Digital certificates serve as an attestation of the identity of a certificate's owner (e.g., hostname, organization) bound to its public key. The X.509 format [19] is one of the most widely used standards for digital certificates that, in addition to the public key and owner's identity, contains a period during which the certificate is considered valid, and a digital signature of the issuing certification authority (CA). CA is a trusted third party whose primary function is to validate

the identity of individuals, organizations, or servers requesting digital certificates and to vouch for their authenticity. By signing the leaf certificate (end-entity certificate) with its private key, the CA provides assurance that the certificate's owner is genuine and that the public key contained within the certificate belongs to that owner. To verify this leaf certificate, a client needs to obtain a chain of certificates including a presented leaf certificate, intermediate certificates, and finally, a root certificate. In a web's PKI, when a server presents a leaf certificate, it is expected to include the certificate chain as well. A client then can verify each certificate along the chain using the included CAs' digital signatures. In general, the public key contained in certificates can be used for a specific purpose. The purpose is listed in the "Key Usage" and "Extended Key Usage" extensions field in a certificate such as digital signature, key encipherment, data encipherment, key agreement, TLS web server authentication, TLS web client authentication, code signing, email protection, etc. The structure of an X.509 certificate is given in Figure 1. In addition to the certificate layout, it also shows a fingerprint, which is a computed hash of a certificate in binary format. Typically, leaf TLS certificates are compared by matching these fingerprints [20].

*d) **RSA**:* In this work, we focus on the RSA keys [21] as this is arguably the most popular cryptographic system utilized on the Internet today. RSA is an asymmetric cryptographic algorithm that leverages the fact that while the multiplication of large prime numbers may not be computationally intensive, the factorization of large prime numbers is significantly more complex. An RSA public key is a pair of values (n, e). It is generated based on two prime numbers p and q used to calculate the modulus n, i.e., $n = p * q$. The public key's exponent e is selected at random, so that $e \in 1, 2, ..., \phi(n) - 1$ and the $GCD(e, \phi(n))$ is equal to 1 so that e and n are relatively prime. As a result, an RSA public key $Pub_k = (e, n)$ is represented by an exponent e and a modulus n. RSA public key size is measured by the length of the key's modulus in bits [22].

*e) **PKI**:* Public Key Infrastructure, is a system that manages digital certificates and cryptographic keys. It establishes trust by binding an entity's identity to its public key. PKI enables secure communication and authentication over networks like the Internet. It relies on a trusted Certification Authority (CA) to issue and verify digital certificates.

III. Related Work

There is a significant body of research on TLS/SSL security. Over the years, studies showed weaknesses in TLS/SSL deployments for email [23], consumer IoT devices [24], and industrial IoT devices [25]. Paracha et al. [24] tested 40 TLS-supporting IoT devices across several categories. The study revealed that several of these devices show similarities in TLS fingerprints with other devices from the same manufacturer (e.g., Amazon devices) as well as with other TLS clients (e.g., LG TV). Among other problems, they noted a lack of certificate validation and the use of deprecated certificates.

In 2022, Dahlmanns et al. [25] examined the TLS adoption rate in industrial IoT devices. Their study showed that among 967,551 IoT devices, only 6.5% have implemented the TLS protocol, with 42% of these being configured insecurely. This insecurity arises from various factors such as the reuse of compromised secrets of certificates (30%), certificates relying on deprecated primitives (e.g., recently issued certificates relying on MD5), outdated protocol versions, ciphers, or disabled access control. These findings demonstrate that the progression of industrial protocols toward secure end-to-end communication is not extensively mirrored in real-world deployments.

Several studies examined the weaknesses of TLS certificates. In 2016, Chung et al. [26] showed that on average, 65% of SSL certificates advertised in each IPv4 scan are invalid. Using a set of over 80 million certificates, they found that most invalid certificates originated from a few types of end-user devices associated with a few Autonomous Systems. A concurrent study by Samarasinghe et al. [27], albeit on a smaller scale, analyzed certificates obtained from 299,858 devices. Similar to others, the researchers found a presence of small keys (4% were 512-bit and 768-bit keys) and a use of deprecated RC4 stream cipher (37%). In 2021, Hue et al. [28] evaluated the strength of TLS connections on 3,637 domains in a WPA2-Enterprise ecosystem. The study showed a widespread presence of security issues such as weak algorithms, use of expired certificates, use of short RSA keys, and possible key reuse.

The compromise of cryptographic protocols happens due to errors in protocol implementation, misconfiguration, or improper selection of parameters. In 2012, Heninger et al. [4] studied the presence of vulnerable keys across the Internet by analyzing 6.2 million SSH keys and 5.8 million TLS certificates collected in the wild. Their results showed at least 5.57% of TLS hosts and 9.6% of SSH hosts used vulnerable duplicate keys. Similar results were obtained by Lenstra et al. [29] on the analysis of 11.7 million public RSA keys. The later study performed by Gasser et al. [12] on 56.4 million SSH keys confirmed that the amount of vulnerable keys is declining. While Heninger et al. [4] were able to factor keys for 0.03% of SSH hosts, Gasser et al. [12] found that 0.013%-0.016% SSH hosts use factorable keys.

Several studies showed that key reuse can lead to cross-version [30] and cross-protocol attacks. Felsch et al. [31] showed that a single RSA key pair used to configure different versions of the Internet Key Exchange (IKE) protocol in Clavister and ZyXEL devices can lead to cross-protocol authentication bypasses.

Specifically looking at the reused TLS certificates, Costin et al. [6] analyzed firmware images and was able to identify approximately 35,000 active online devices on the Internet that were utilizing the same self-signed certificates. The use of certificates in digitally signed malware has been studied by [32], [33]. To evade anti-virus programs and bypass system protection mechanisms malware is often signed with valid certificates. Kim et al. [32] found that 88.8% of the signed malware families rely on a single certificate. Kang et al. [34] introduced AndroTracker, a tool that makes use of serial numbers of certificates to detect possible Android malware.

Vulnerabilities in PKI infrastructure undermine the security of the entire ecosystem. To enhance PKI security, various

solutions have been introduced, primarily targeting the core assumption of PKI: the necessity of a trusted CA. Certificate Transparency [35] and Sovereign Keys [36] were introduced in an effort to provide more accountability and make compromises visible. In a sense, both present an after-the-fact solution since they cannot guarantee that a CA issued a correct certificate. Kim et al. [37] developed an Accountable Key Infrastructure that distributes trust among multiple CAs to prevent impersonation of domains by a compromised CA. Larisch et al. [38] introduced the CRLite system for effective and complete dissemination of TLS certificate revocations for web browsers. A more comprehensive solution titled Attack-Resilient Public Key Infrastructure (ARPKI) was proposed by Basin et al. [39]. It is similarly based on multiple CAs to allow even a single truthful entity to prevent attacks. These schemes seek to reduce trust in certification authorities by introducing public logs, multiple CAs, and other alternative solutions. For example, LocalPKI proposed by Dumas et al. [40] enables a local deployment of PKI that allows local authorities to issue certificates.

Several studies showed vulnerabilities of cryptographic keys in both protocols and files are due to improper use of libraries or inherited from their weaknesses, for example, random number generators (RNGs). Heninger et al. [4] confirmed that limited sources for generating appropriate randomness in memory-constrained devices (such as routers, and smart cards). In 2008, a bug in the OpenSSL library made predictable generated random numbers. A follow-up by Yilek et al. [41] confirmed the spread of keys affected by the bug even after six months of disclosure. Slow industry response to the cryptographic bugs was also noted by Hastings et al. [42] which suggests that one of the main causes behind the majority of factored RSA keys may be related to RNG issues.

The various weaknesses in cryptographic keys were later used to attribute keys to the corresponding libraries that generated them. Svenda et al. [10] performed the technical analysis of over 60 million newly generated keys from 22 open and closed-source libraries and from 16 distinct smart card vendors, revealing that various security lapses allow attributing keys to the libraries that generated them based solely on the properties of RSA public keys. Branca et al. [14] took a step further showing that it is feasible to accurately (with 95% accuracy) attribute RSA keys to individual library versions.

We similarly examine the weaknesses present in RSA public keys. However, unlike previous research that concentrates on a specific area of RSA key usage, our study investigates the prevalence of shared certificates and key usage across diverse domains on a much larger scale. Our analysis of over 314 million certificates and 13 million SSH keys collected over a period of several years gives a more comprehensive view of key usage on the Internet.

IV. Collected data

For our measurement study, we collected digital certificates and RSA keys from 6 different sources covering a period of several years. The details of the collected data are given in Table I.

*1) **TLS/SSL certificates**:* To collect certificates, we generated 100 million IP addresses using the IP ranges assigned to the corresponding registrars around the world. We scanned them and selected IPs that appeared alive and responded to our scan. Out of the IPs that responded, we randomly selected 10 million IP addresses, ensuring that at least 1 IP was representing each network segment. This ensured a representation of hosts around the world. The 10 million were selected as a 10% representation sample. These IP addresses were contacted on ports 21 (FTP), 443 (HTTPS), 465 (SMTPS), 993 (IMAPS), and 995 (POP3) to determine the listening hosts. If a host replied, acknowledging our connection (syn-ack packet), we further connected to it using TLS/SSL protocol to obtain a certificate. We made several scans in 2013 collecting certificates using SSL2, SSL3, and SSL23[1], TLS 1, TLS 1.1 protocols. For each successful connection of TLS/SSL, we collected a leaf certificate and discarded the chain of certificates. Our scans conducted between December 2021 and February 2022 reaffirmed the continued presence of 94% of all certificates within this set.

*2) **SSH Keys**:* To collect SSH keys, we scanned the public IPv4 space during May-September, 2021. We used ZMap [43] to perform a single-packet host discovery. For each host discovered by ZMap, we contacted it again on ports 22, 23, 2222, 4444, 5000, and 10001 using the ssh-keyscan tool [44] collecting SSH banner and their public SSH RSA key. In addition to the cryptographic material from RSA keys, we recorded key collection time, an IP address, and a port. Using ssh-keyscan we collected both the public key and server header from the connection.

*3) **Rapid7 certificates**:* We obtained an additional collection of SSL certificates from Rapid7 [45]. We used weekly scans collected by Rapid7 from October 2013 to September 2015, August 2019, and during a period of September 2020 to July 2021. In addition to X.509 certificates, this retrieved set contains a collection of metadata, e.g., time of collection, IP address, protocol and port, and the X.509 certificate's fingerprint. Similar to other sets, we only kept the leaf certificate and discarded the chain of certificates to ensure consistency in analysis. The certificates from 2013-2015 were retrieved from Rapid7 in the past without the corresponding IP addresses. Note, that this set is no longer available from the Rapid7 set in its entirety.

*4) **SBA certificates**:* In our analysis, we also used the subset of certificates offered for analysis by Mayer et al. [46]. The email ecosystem makes wide use of certificates for the secure transmission of messages. The SBA set contains the certificates retrieved over StartTLS protocol from SMTP servers on port 25 during a period of April 2015 - August 2015[2].

*5) **Malware certificates**:* We complemented our data with a set of 40,270,387 malicious files in PE (Microsoft Windows Portable Executable) format retrieved from VX underground's

[1]The OpenSSL header file dealing with the combined use of the SSLv2 and SSLv3 protocols.

[2]The original set is now only available through Wayback machine: https://web.archive.org/web/20160307162719/https://scans.io/study/sba-email. Due to a site problem, we were never able to download the whole set.

TABLE I: The summary of collected certificates and keys

Datasets	Collection period	Responsive hosts/ collected files	Collected certificates and keys	All valid certificates	Valid RSA certificates & keys	Unique RSA keys
Collected TLS/SSL	8/2013-11/2013	1,675,040 IPs	863,872 certs	863,871 (100%)	863,500 certs	863,500 (100%)
Collected SSH	5/2021-9/2021	10,435,118 IPs	13,681,145 keys	n/a	13,681,145 keys	9,747,793 (71.25%)
Rapid7	10/2013-09/2015, 8/2019, 9/2020-7/2021	99,910,085 IPs	295,063,780 certs	234,865,702 (79.6%)	135,823,576 certs	104,904,586 (77.24%)
Malware	2012-2021	40,270,387 files	18,081,501 certs	18,081,501 (100%)	18,081,489 certs	41,282 (0.23%)
Android apks	9/2020-10/2020	72,508 apks	118,743 certs	29,247 (24.63%)	29,247 certs	29,247 (100%)
SBA	4/2015 - 9/2015	n/a	202,381 certs	202,381 (100%)	202,378 certs	121,990 (60.28%)
Total	-	-	314,330,277 certs 13,681,145 keys	254,042,702 (77.45%)	168,681,335 keys 155,000,190 certs	115,708,398* (68.60%)

'*' not deduplicated across sets

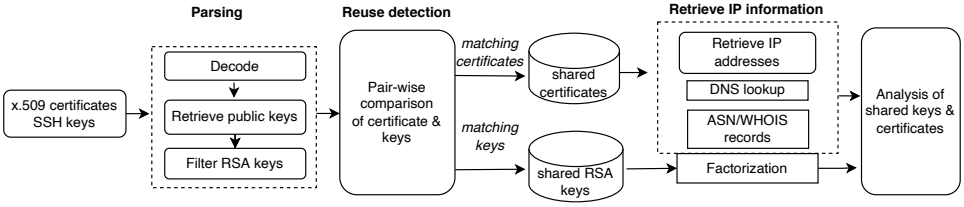

Fig. 2: The flow of the analysis.

APT collection [47] and VirusShare repository [48] from 2012 to 2021.

*6) **Collected APKs certificates**:* For our analysis, we crawled several legitimate Android markets (GooglePlay, Mobile1, Slideme, Xiaomi, Nduo.cn, Mob.org, Anzhi, Fdroid, and ApkGalaxy) to collect 80,000 apk files. Out of the collected apks, 79,652 of them were successfully decompressed and 72,508 apks had at least one cryptographic key.

V. Methodology

a) Challenges: There are many advantages to performing large-scale analysis of RSA certificates and key reuse. In fact, some trends and patterns can only be revealed by looking at the global picture, which individual organizations and CAs may not have. However, large-scale analysis requires an automated approach to obtain certificates and keys, parse them, and analyze them for reuse. While these tasks may appear easy, in reality, there are several challenges that need to be solved.

Challenge 1: Modern cryptographic libraries do not fully support legacy certificates. The evolution of cryptographic standards has resulted in stricter requirements for modern certificates. Consequently, certificates with weaker elements, such as an exponent = 1[3], are no longer produced by libraries, and therefore no longer expected to be parsed [49]. Since legacy certificates are still widely used in practice, valid certificates are accepted by libraries, but only fields necessary for authentication - such as the signature - are verified. This is not sufficient for a large-scale analysis of key reuse. Since modern libraries often fail to parse all individual fields for such certificates, this task requires custom parsing of legacy certificates.

[3]https://github.com/golang/go/blob/master/src/crypto/rsa/rsa.go, line 99 shows harded values of an acceptable key exponent.

Challenge 2: Inconsistent Object Identifiers (OIDs). OIDs are numeric values that uniquely represent various entities and attributes. For example, RFC 5280 [19] and RFC8017 [50] require OID to point to the corresponding encryption and signature algorithms in x.509 certificates. Microsoft uses OIDs to represent certificate extensions [51]. This representation, however, is not consistently implemented across libraries (e.g., OpenSSL does not contain Microsoft OIDs by default [52]) and varies between operating system versions. In many cases, the only source of reliable information is the official vendor documentation. While this is not a problem when manually mapping a handful of OIDs, extending the analysis to millions of certificates is challenging and error-prone.

Challenge 3: Longitudinal analysis of certificates. The goal of our study is not only to analyze the reuse of certificates and keys over time but also to provide insight into the devices and applications associated with these shared certificates. However, this presents several challenges. Some datasets lack corresponding IP addresses, and even when IP addresses are present, we cannot rely on them due to their dynamic nature. There is no guarantee that the IP addresses that previously served these certificates are still active, whether they are assigned to the same devices, or if the same applications using these certificates are running on the same ports. Confirming this information is a tedious task that requires rescanning devices and verifying certificate fingerprints. However, this step does not provide information on the corresponding device, operating system, or library that generated a certificate and key. While we used some of the existing techniques, our analysis proved to be significantly more time-consuming than typically acknowledged for this task. We believe that the development of new techniques is required to accurately address this problem at a large scale.

Challenge 4: Scalability and computational requirements. One of the main advantages of performing a large-scale analysis is the ability of correlating information across different cer-

tificates and keys from various sources. However, capturing interesting insights requires a pairwise comparison of all certificates and keys. For example, in parsed format, our collected data exceeded 4 terabytes. Querying this extensive dataset presents significant computational challenges.

b) Analysis: The conceptual flow of the analysis is illustrated in Figure 2. It includes three main steps.

Parsing At this step, we initially relied on standard libraries. The Android apps were unpacked and analyzed for the presence of cryptographic keys. We used apktool to extract APK files content, and CERT Keyfinder utility[4] to locate and parse key files contained within Android APK files. CERT Keyfinder collects all references to files with standard extensions associated with cryptographic certificates and keys, such as rsa, pem, crt, and cer. We extracted and parsed all certificate files using Keyfinder. Only the keys successfully extracted by Keyfinder were retained for our analysis, ensuring that only correct (non-false positive) certificates and keys were included. Similarly to TLS/SSL certificates, we only kept the leaf certificate in the chain of certificates. These certificate files are intended to be used for signing apk files, i.e., to identify the author of an Android app. We were primarily interested in keys used for code signing, hence, we did not decompile .dex files to extract hard-coded keys or other not-signing certificates in our analysis.

Malware binaries were parsed using GoLang's sigtool [53], which is a PE package designed to extract signing information from PE files, and we only kept the leaf certificate for our analysis. The certificate signing the code is contained in the "Attribute certificate" section of PE files in DER format. The extracted certificates were converted to the PEM format, which was then parsed to extract the certificate and key information. Similarly, the keys that were extracted and parsed successfully by sigtool were included in our analysis ensuring only true positives were used.

The collected certificates were parsed using Python cryptographic libraries: PyOpenSSL, Cryptography, Pyasn1, and Paramiko. These libraries provide the necessary functionality to parse modern certificates in PEM or DER format and extract the corresponding certificate metadata and the public key. All legacy certificates that the modern libraries were not able to process, were handled separately by our custom parsers developed for this study.

While parsing data from our sets, we found that a significant number of IP addresses presented invalid (e.g., containing garbage values, partial keys) or incomplete (i.e., missing critical fields) certificates. We discarded all invalid and incomplete certificates and keys. We further filtered non-RSA keys. In this work, we focus on RSA certificates and keys due to their predominance presence in PKI infrastructure. For our analysis, we retained RSA public keys, i.e., their modulus and exponent, the information extracted from their corresponding certificates, and the IP addresses (when available).

Reuse detection To explore potential certificate and key reuse within our collections, we performed a pairwise comparison of RSA certificates based on their fingerprints (also referred to as thumbprints), i.e., the SHA1 hash of the certificate in DER binary format. In addition to matching certificates, we performed a pairwise comparison of valid RSA keys from all sets irrespective of whether their corresponding certificates are shared or not. In this analysis, shared certificates and keys within each set were removed.

Given the time gap between data collection and analysis, as a final step, we validated shared certificates. For shared certificates, we retrieved available IP addresses. Since the Rapid7 scans from 2013-2015 had no associated IP address or port information, further analysis of the certificates was performed based on the IP information for the matching certificates retrieved from the other sets. To match an IP address/port to the corresponding certificate, we initiated a TLS/SSL connection with the IP addresses of the shared certificates during December 2021 and February 2022 and requested their current certificate.

Retrieving contextual IP information For each of the IP addresses that share certificates, we scanned IP addresses using the nmap application. We also performed a simultaneous DNS query to retrieve the corresponding PTR records, which provided us with a domain name associated with an IP address. We also mapped an IP to ASN and retrieved WHOIS contacts using the Team Cymru service[5]. Through this process, we verified the shared certificates and cross-referenced their fingerprints with those stored in our records. To explore the origin of the shared RSA keys, we have adopted an approach for an RSA key origin attribution proposed by Branca et al. [14].

As the final step of the analysis, we focused exclusively on the shared certificates and the shared RSA public keys. Our correlation analysis was performed on the environment equipped with 64GB of RAM and HighPoint 4-Port M.2 Rocket 1204 PCIe Gen3 NVMe HBA card with 4x2TB NVMEs. This setup allowed us 1500 MB/s in read and write operations which is significantly faster than with a traditional SSD or an NVME along.

VI. Initial Analysis

The results of our initial analysis are presented in Table I. Out of 314 million certificates, we extracted 155,000,190 valid certificates containing RSA keys. Overall, we were left with 168,681,335 valid RSA keys for our analysis.

We observed that there is a significant amount of duplication across certificates and keys used on the Internet. The highest amount of duplication is present in our Malware collection where almost all keys (99%) found in binaries were also seen in other sets. Around 29% of keys served by SSH hosts and 43% of Rapid7 certificates are duplicates.

Some aspects of the key and certificate reuse phenomenon were noted by previous studies. For example, Heninger et al's [4] study conducted in 2011-2012 on a smaller scale (12.8 million TLS hosts and 10.2 million SSH hosts) showed that 61% of TLS hosts and 65% of SSH hosts serve the same key as other hosts. It appears that the usage of repeated keys

[4]https://github.com/CERTCC/keyfinder

[5]https://team-cymru.com/community-services/ip-asn-mapping/

TABLE II: RSA public key size

Key size range (bits)	Total	Frequency						
		Total Unique	TLS/SSL	SSH	Rapid7	Malware	Android apks	SBA
0-1023	4,203,348 (2.49%)	582,823 (0.69%)	9,539 (1.12%)	0 (0.00%)	581,173 (0.78%)	1,126 (3.05%)	10 (0.03%)	240 (0.20%)
1024-2047	47,050,256 (27.89%)	26,223,521 (30.98%)	385,044 (45.30%)	225,357 (2.31%)	25,958,127 (34.65%)	10,554 (28.61%)	9,544 (32.68%)	26,356 (21.69%)
2048-4095	111,732,314 (66.24%)	55,309,352 (65.33%)	444,259 (52.26%)	9,468,431 (97.13%)	45,892,172 (61.26%)	24,405 (66.15%)	18,767 (64.25%)	89,481 (73.64%)
4096-8191	5,684,453 (3.37%)	2,536,655 (3.00%)	11,111 (1.31%)	53,539 (0.55%)	2,480,086 (3.31%)	802 (2.17%)	885 (3.03%)	5,395 (4.44%)
8192-up	10,964 (0.01%)	6,603 (0.01%)	64 (0.01%)	466 (0.005%)	6,122 (0.01%)	8 (0.02%)	2 (0.01%)	38 (0.03%)
Total	168,681,335	84,658,954 (50.19%)	850,017	9,747,793	74,917,680	36,895	29,208	121,510

has decreased since that time, yet remains a considerable issue. To understand the current state of the PKI ecosystem, we investigate the strength of collected keys and certificates individually.

A. Weak key size

We analyzed the strength of the collected RSA keys based on their modulus length. Table II shows the wide presence of weak keys across collections.

Among the collected keys, 30% (51,253,604) of keys are less than 2048 bits in length. They are considered cryptographically weak, and should not be used for cryptographic protections. Since 2015, NIST-compliant RSA keys are required to have a length greater or equal to 2048 bits [54]–[56]. However, our observations reveal that even after 2015 approximately 3 million keys with inadequate key sizes persisted. This is alongside the keys generated prior to 2015, which are still considered valid due to their extended validity period.

NIST also recommended deprecating signing certificates that contained RSA keys of 1024 bits by the end of 2013. However, across all our scans, 2.49% of keys are less than 1024 bits in length and thus have deprecated status. While most of these keys are found in the TLS/SSL set collected in 2013 and the Rapid7 set partially covering 2013, 15,249 keys come from sets collected in 2019-2021 indicating that these legacy keys are still in use.

Among these weak keys, 445,900 (0.26%) are 512-bit RSA keys. Only a portion of these keys (119,612) in our sets is associated with an IP address, so we can see that 119,612 TLS hosts serve these vulnerable RSA keys. As a comparison, in 2012 Heninger et al. [4] showed that 123,038 TLS hosts were using 512-bit keys, which reveals that the numbers have not decreased over time. Since we do not have the corresponding host information for 326,288 keys, it is likely that the actual number of TLS hosts using 512-bit keys is significantly underestimated.

B. Breakable RSA keys

Besides keys' modulus weaknesses, we explore other vulnerabilities that lead to key factorization. The summary of these experiments is given in Table III.

1) ***Weak exponent****:* Out of all collected keys, 28 RSA keys have an exponent equal to one, and 42 keys have an even exponent. The exponent is supposed to be a large coprime number, preferably equal to 65537 as recommended by NIST. In the cases when $e = 1$, deriving a private key is trivial, hence, the corresponding public key is considered weak. When an even exponent is used, by calculating the square root of the ciphertext, it is potentially possible to retrieve the original plaintext without possessing the private key.

2) ***ROCA****:* We have also tested collected keys for the Return of Coppersmith's Attack (ROCA), a security vulnerability that affects the cryptographic keys generated by a specific type of hardware RNG called Infineon RSA library [2]. The cryptographic key originated from the Infineon RSA library allows an attacker to exploit the weak prime numbers generated by the faulty implementation using Coppersmith's algorithm. To test the gathered keys for this vulnerability, we have used the ROCA detection tool[6]. Results show that 231 keys in our all collected data were vulnerable to ROCA.

3) ***GCD-Factorable****:* One of the threats to the RSA cryptosystem is the possibility of factorizing modulus to decompose it to p and q values, and consequently to compute the private key. Theoretically, such factorization is computationally intensive and should be unfeasible for sufficiently large p and q numbers. Yet in practice, the occurrence of weak keys is more common which makes the factorization possible in some cases. Studies by Heninger et al. [4] and Gasser et al. [12] measured the spread of weak factorable keys in 2012 and 2014. To determine the presence of weak factorable keys, we used the *Fastgcd*[7] tool, which was originally developed by Heninger et al. [4]. The tool performs a pairwise computation of GCDs of all moduli to determine if any of them share a prime number with any other modulus, in which case, a computation of the corresponding private key is straightforward. In our study, the GCD's computation was performed on 141,098,520 unique moduli extracted from the collected certificates and keys. We were able to find divisors for 185,731 (0.13%) moduli that were present in 793,694 keys in our sets, i.e., we could factor the moduli corresponding to 793,694 keys (Table III). This is considerably higher (80 times) than the numbers reported by the previous studies, e.g., Heninger et al. [4] reported finding divisors for 2,314 out of 11,170,883 moduli. Since our coverage is significantly larger than the previous studies, we believe our results are more representative of the security state of the RSA keys.

C. Use of certificates over time

The presence of legacy devices with legacy certificates on the Internet can partially provide an explanation of key weaknesses. We therefore also analyzed the lifetime of our collected

[6]https://github.com/crocs-muni/roca

[7]https://factorable.net/resources.html

TABLE III: Factorable RSA keys

Heninger et al. [4]	
SSH hosts using factorable RSA keys	0.03%
TLS hosts using factorable RSA keys	0.5%
Gasser et al. [12]	
SSH hosts using factorable RSA keys	0.013%-0.016%
Our results:	
Factorization analysis:	
GCD factorable moduli	185,731 (0.13%)
Total impacted RSA keys	793,694 (0.47%)
Total impacted certificates	792,246 (0.31%)
Total impacted hosts	35,700
Key dataset:	
TLS/SSL	3,923
SSH	1,448
Rapid7	786,293
Malware	2,030
Android apks	0
SBA	0
Key size	
0-1023	14,279
1024-2047	769,986
2048-4095	9,327
4096-8191	98
8192-up	4
ROCA vulnerability (keys)	231
Exponent = 1	28
Exponent = *even number*	42

certificates. To explore the use of certificates across time, we grouped all certificates based on the time of their collection. Within each time interval, we discarded duplicates based on their fingerprints and checked the presence of the remaining unique certificates in other time intervals. Figure 3 shows the frequency of distinct certificates seen across different time ranges.

The certificates appearing in earlier time ranges (2013-2015) are rarely seen at later times. This is generally expected as some of the older certificates will become obsolete due to evolving cryptographic standards and would require organizations to generate new keys and the corresponding certificates to comply with the new rules and recommendations. For example, the introduction of SHA3 in 2015, and consequently, the requirement to phase out certificate chains with SHA1 by 2017 naturally led to the certificates being reissued.

The drastic shift is clearly visible in recent years. The overwhelming majority (77-90%) of certificates collected in 2020 and 2021 are encountered across multiple time ranges. This fact suggests that organizations are not replacing their certificates as frequently as expected. Many of these certificates are shared between our sets, we thus focus specifically on the shared RSA certificates and keys.

VII. The shared RSA certificates

Out of 155,000,190 valid RSA certificates, 10,110,361 (6.5%) are shared across the collected datasets (Table IV).

There are common scenarios where certificates can be potentially shared across multiple hosts and domains. A certificate may belong to an organization that serves this certificate across several IP addresses that belong to it. Conversely, a certificate may come from a third-party hosting provider supporting multiple clients. Historically, an SSL certificate was issued to a host/domain name indicated in the ”Common Name” field within the ”Subject” field of the certificate, establishing a one-to-one mapping between a certificate and a host. With the growing prevalence of website hosting, in 2000 [57], certificates were permitted to encompass more than one domain name, enabling the utilization of a single certificate across multiple hosts through the use of the ”Subject Alternative Name” extension. This development led to providers commonly employing custom certificates that encompass multiple domains or organizations within a single certificate. These certificates could be served from a single IP address belonging to a hosting provider, by using wildcard certificates[8].

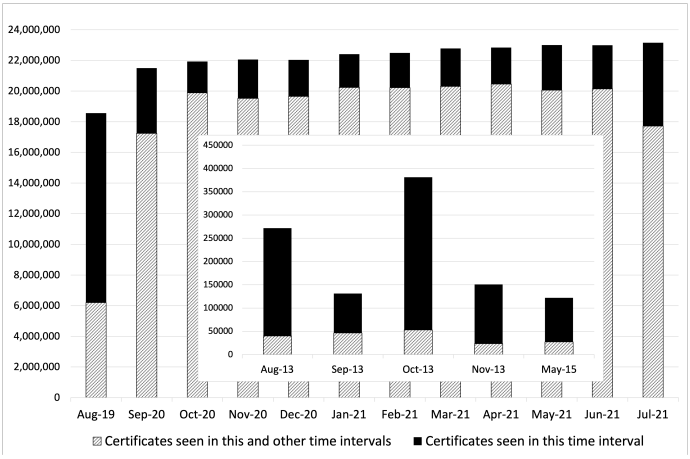

Fig. 3: The use of distinct certificates across time.

In the following subsections, we look at the various aspects of the certificate reuse phenomenon.

A. Reuse of certificates over time

The vast majority of all shared certificates are shared between the Rapid7 set and other sets. A large cluster of SSL certificates is seen in our collected TLS/SSL and Rapid7 set. The presence of these shared certificates is not surprising. Both sets overlap in time (the year 2013), hence, they may potentially include certificates obtained from the same hosts. However, further timeline analysis of these shared certificates shows a significant time gap, i.e., the fingerprint of certificates found in TLS/SSL set match certificates collected by Rapid7 during 2019-2021 (Figure 3). This implies that many certificates have been reused by different hosts for 7-8 years.

For example, out of the certificates that were analyzed, 58 have been consistently used since 2013 and have appeared a total of 6,873 times across our datasets. Interestingly, 31 of them were self-signed with the sha1WithRSAEncryption signature algorithm and were issued by GlobalSign, one of the largest CAs, that is a part of the CA/B (Certification Authority/Browser) Forum, an industry organization that establishes guidelines and best practices for issuance and management of

[8]In RFC 2818, section 3.1 allows for wildcard character * which is considered to match any single domain name.

TABLE IV: The shared RSA certificates

Datasets	Total shared	Unique shared in each set**	Certificates shared with other sets				
			Collected TLS/SSL	Rapid7	Malware	SBA	Android apks
Collected TLS/SSL	752,029 (7.44%)	752,029 (100%)	*	751,993	0	3,948	1
Rapid7	863,792 (8.54%)	863,284 (99.94%)	752,309	*	491	114,733	227
Malware	8,302,431 (82.12%)	377 (<0.01%)	0	8,299,100	*	0	2,338,672
SBA	191,919 (1.90%)	114,734 (59.78%)	17,496	191,879	0	*	0
Android apk	190 (<0.01%)	190 (100%)	1	187	58	0	*
Total	10,110,361	1,730,614	863,323 (8.54%) are distinct across sets				

** duplicates within a set are removed, across sets retained

digital certificates. These 58 certificates are leaf certificates, and their key usage-related extensions indicate that they are intended for various purposes including internal CA use (29 certificates), browser use (6 certificates), device authentication use (16 certificates), and authentication for firewall and security gateways (7 certificates).

Over time, the validity period of certificates varied according to their usage. For example, in cases when renewing certificates was not feasible, RFC 5280 [19] released in 2008 allowed to issue certificates with no expiration date. However, the validity of certificates has since been significantly shortened.

In 2013, the CA/B Forum, in its baseline requirements documents, stated that certificates issued after July 2012 should not exceed a validity period of 5 years. Additionally, certificates issued after April 2015 should have a validity period not exceeding 39 months [58].

In 2017, the CA/B Forum established the maximum validity of certificates to two years (825 days). Prior to this, the maximum validity was three years for most certificates. In 2020, NIST recommended the maximum validity period for certificates should be one year or less [59]. In 2022, the CA/B Forum set the maximum validity period of a TLS certificate to 398 days[9]. In March 2023, Google announced that the maximum certificate validity will be soon reduced to 90 days for all publicly trusted TLS certificates. While recommendations leading to shorter certificate validity emphasize the idea of regularly rotating keys to enhance security and mitigate the potential impact of key compromise, these guidelines do not affect already issued certificates, effectively facilitating the reuse of legacy certificates.

Further investigation revealed that one of these consistently used certificates, which employs the sha1WithRSAEncryption signature algorithm, has been shared 5,989 times, 4 times in the Rapid7 set and 5,985 times in the Malware set. This is a CA signing certificate distributed by GlobalSign, i.e., this certificate can be only used by a CA to sign other certificates and CRLs. Although we retained only leaf certificates in our analysis, it appears to be a root certificate signed with an outdated signature algorithm. Hence, the presence and the reuse of this certificate across malware and other sets is puzzling and dangerous, considering that CA certificates serve as the foundation of trust in the PKI ecosystem. Such incidents underscore the growing concerns to investigate the situation of shared certificates and keys deeper.

[9]https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.8.4.pdf

B. Reuse of certificates for different purposes

While the overlap between the collected TLS/SSL set and Rapid7 is expected, the other cases reveal worrisome patterns. For example, we observed that 28% of certificates used in malicious binaries can also be found in Android apps (Table IV). There have been reports of malicious binaries signed with valid certificates [60], [61] in the past. The reappearance of this problem in Android apps that were collected in 2020, and are supposed to be legitimate, indicates that this phenomenon may be more widespread than initially believed.

We also identified a considerable amount of certificates shared between malware and the Rapid7 sets, i.e., with almost all certificates collected from malware binaries (99.9%) found in the Rapid7 set containing TLS/SSL certificates.

Although both types of certificates are integral to the PKI ecosystem, they have different purposes. The code signing mechanism allows authentication software publisher, while an SSL certificate serves to verify the identity of a server. Typically, the purpose of the certificate can be specified in the ”Key Usage” and ”Extended Key Usage” field of the certificate. Certificates issued for one purpose should not be used for different usage (e.g., a TLS certificate cannot be used to sign code). However, the overlap shows that at some point these malicious certificates were used for both purposes.

C. Weak signatures algorithms

Valid digital certificates must be signed by the certification authority that issued them. While RFC 3279 [62] and its subsequent versions permit the use of any public key signature algorithm in conjunction with a one-way hash function, NIST recommends appropriate hash functions based on the algorithm’s strengths.

To examine the cryptographic algorithms utilized by the certificate issuer to produce their digital signatures, we extracted the object identifiers (OID) of the cipher algorithms from the digital signature section of the certificates. For ease of interpretation, we resolved the OIDs into their corresponding algorithm names. For instance, the OID 1.2.840.113549.1.1.5 was translated to sha1WithRSAEncryption.

The list of the observed signature algorithms in shared certificates is shown in Table V. The table lists a total of 10,110,361 certificates that were associated with 863,323 unique fingerprints (Table IV).

We discovered that the use of obsolete algorithms among certificates is significantly high.

The vast majority of the shared certificates ($>$ 9 million) are signed using the RSA algorithm with SHA-1 and MD5

TABLE V: Signature algorithms seen in shared certificates

Signature algorithm	Total shared	Unique certificate
sha1WithRSAEncryption	8,986,133	770291 (8.57%)
md5WithRSAEncryption	536,080	62830 (11.72%)
sha256WithRSAEncryption	275,416	26451 (9.60%)
sha384WithRSAEncryption	272,627	27 (0.01%)
md2WithRSAEncryption	32,644	15 (0.05%)
sha1WithRSA	5,646	2818 (49.91%)
sha512WithRSAEncryption	1,635	801 (48.99%)
rsaEncryption	116	58 (50.00%)
dsaWithSHA1	20	10 (50.00%)
rsassaPss	20	10 (50.00%)
ecdsa-with-SHA256	8	4 (50.00%)
shaWithRSAEncryption	4	2 (50.00%)
ecdsa-with-SHA1	4	2 (50.00%)
ecdsa-with-SHA512	4	2 (50.00%)
GOST R 34.11-94 with GOST R 34.10-2001	2	1 (50.00%)
sha224WithRSAEncryption	2	1 (50.00%)
Total	10,110,361	863,323 (8.54%)

hashing. Both have not been recommended for use. NIST deprecated the use of SHA-1 in 2011 and disallowed its use for digital signatures in 2013 [63]. Although our TLS/SSL set was collected at the end of 2013, the total number of these certificates (863,871) is several times smaller than the observed number of certificates with the use of SHA-1 algorithm (e.g., dsaWithSHA1, ecdsa-with-SHA1, sha1WithRSA, sha1WithRSAEncryption).

Similarly, the MD5 hashing algorithm has been considered broken, and unacceptable for use in digital signatures since 2008 [64], [65]. Yet, over 500,000 shared certificates using MD5 hashing are still in use. Out of these certificates, the Malware set accounted for the largest proportion, i.e., 75.87% of certificates, followed by Rapid7 (11.73%), TLS/SSL (10.90%), and the SBA dataset (1.50%).

To our surprise, we noticed that roughly 32,000 shared certificates were signed with the MD2 hashing algorithm, which was deprecated in 2011 [66], and thus should not have been present in any of our collections.

Unexpectedly, we also found a certificate, shared two times, that uses GOST hashing functions as the signature algorithm which based on RFC 4491 [67] was a valid algorithm for PKI and CRL profile but has been removed from the OpenSSL family of libraries in 2016[10].

Our numbers indicate that the use of stronger signature algorithms (e.g., SHA-256, SHA-512) is barely noticeable ($\sim$3%) among the shared certificates. We note that in 2015, NIST recommended the minimum use of SHA-256 for any application of hash functions requiring interoperability[11], and in 2020, NIST made it a requirement for all certificates to be signed with an approved signature algorithm and hash algorithm, such as SHA-256 [68].

D. Who uses shared certificates

We identified 863,323 unique certificates across all sets (Table IV), linked to 29,937,166 reachable IP addresses. While not every shared certificate is linked to an IP address in our sets, all corresponding IP addresses are unique.

[10]https://www.openssl.org/news/changelog.txt

[11]https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions

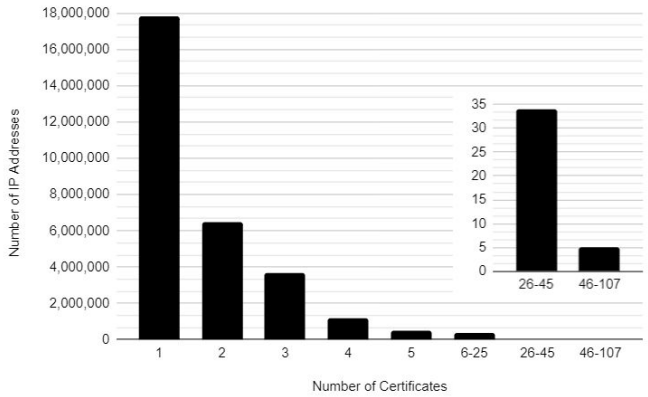

Fig. 4: The distribution of shared certificates among IP addresses.

The majority of these hosts are mainly linked to a single distinct shared certificate, meaning that most hosts in our sets use the same certificate as only one other host. Out of the total, 17,785,693 addresses (59.41%) utilize a single certificate, while 12,151,473 IP addresses (40.59%) are connected to multiple distinct shared certificates, as illustrated in Figure 4.

By focusing our attention on the IP addresses associated with 13 or more certificates, we found a total of 228 distinct certificates duplicated 4,122,199 times and issued by a variety of organizations.

Given a large number of frequently shared certificates, we next analyzed what devices and applications are associated with these certificates.

We scanned IP addresses that share certificates using the nmap application between December 2021 to February 2022. Out of 29,937,166 IP addresses that share certificates, 1,557,199 IP addresses replied with a valid response to the nmap scan. Due to the time gap between collection and analysis, we validated certificates. To unambiguously match an IP address to the corresponding certificate, we initiated a TLS/SSL connection with the IP addresses of the shared certificates. We requested their current certificate and matched their fingerprint with the one we have stored in our records. This step ensured that the host/organization using the IP address is the same and our further analysis is relevant. 51.32% of shared certificates were validated. Our further analysis focuses on IP addresses with validated certificates.

For the majority of these 1,557,199 hosts, nmap was able to identify the operating system (OS) (89.62% hosts) and the device type (88.24% hosts). The remaining cases were categorized as ”None” since nmap could not provide any information on the hosts' software or hardware. The summary of the nmap scan is given in Table VI.

Our scan discovered 169 distinct operating systems within 1,395,662 hosts that served shared certificates. We investigated the top 20 OSs based on the frequency of responded IP addresses (see Table VII).

Our analysis shows that the shared certificates are predominantly used by embedded network devices including routers, mobile phones, printers, and firewalls with the majority of them using OS derived from Linux and BSD distributions.

TABLE VI: The summary of nmap scan

Total shared certificate	10,110,361
Validated shared certificates	5,188,895 (51.32%)
Total IP addresses	29,937,166
Hosts with valid nmap response	1,557,199
Fingerprinted OS	
Hosts with validated certificates	1,395,662
None	161,537
Fingerprinted device	
Hosts with validated certificates	1,374,178
None	183,021

Linux 3.18 OS, released in 2014, is found to be the most common OS serving shared certificates by the majority of hosts. Google and other vendors seem to be utilizing Linux 3.18 on numerous Android-based devices. Some Chromebooks are also operating on the same kernel version as part of Chrome OS. We further investigated the device types of IP addresses associated with this OS and not surprisingly all matched devices are categorized as general-purpose computing systems.

Apart from Linux kernels, the rest of the operating systems indicate network-connected devices, such as HiveOS, SSG firewall, wireless and ADSL routers, webcams, printers, printer servers, terminal servers, and home security controllers, which have embedded computing capabilities and can be remotely managed and monitored. *Most of the devices using shared certificates come from a handful of manufacturers*: ZyXEL Communications Corporation, Aerohive Networks, IPEX Group, Siemens, D-Link Corporation, TRENDnet, Panasonic Corporation, Juniper Networks, Apple, Canon, Vivint Smart Home, Cisco Systems, Microsoft Corporation, Epson Corporation, and Moxa.

We also see a large number of shared certificates used over an extended period of time. For example, the devices identified as the "Efficient Networks 5930 ADSL router" released in 2002, and the "Cisco 7200 router" released in 2005 are among the oldest seen in our scan, pointing that these shared certificates might have been actively used for over two decades.

To validate these observations, we delved deeper into the certificates' validity periods. Our investigation revealed certificates with notably extended validity periods, starting from their manufacturing date and spanning up even to an impressive 50 years. There are a total number of 48,794 IP addresses and 950 distinct certificates associated with these older systems. It appears these instances are indicative of factory-default certificates being in use.

We also analyzed our shared certificates for the presence of certificates generated with default parameters by looking at certificates generated by Linux and BSD systems that still use so-called "Sneak Oil" certificate generation scripts, for automatically generating self-signed certificates. We found that 381 certificates were associated with systems that used default configuration options to establish encrypted TLS/SSL communications.

With the legacy status of the devices, we anticipated finding outdated cryptographic settings. Indeed, the vast majority of certificates (92%) were signed with weak signature algorithms that were introduced before sha256WithRSAEncryption, including md5WithRSAEncryption and sha1WithRSAEncryption present in our set (Table VII). We discovered that around 33% of these certificates relied on weak RSA keys with key lengths less than 2048.

We conducted a similar analysis to identify the hardware hosting the shared certificates which confirmed our conclusions (see Table XIII). The results of this analysis suggest that despite frequent updates and stricter guidelines, embedded network devices continue to persist on the network for extended periods.

VIII. The shared RSA keys

In the previous section, we delved deeper into the phenomenon of shared certificates, yet, we were surprised to discover that multiple distinct certificates were serving identical public keys, i.e., identical modulus and/or exponent.

There are legitimate scenarios for a given organization to retain the same key to re-issue a certain certificate for example in the case of having the private key contained with a hardware security module (HSM) [59] or to reuse the same key for different certificates to ease inner authentication procedures. However, this approach could pose security concerns if the key is weak or vulnerable. Additionally, key reuse across different organizations could become a potential vector for compromise. Therefore, we decided to not only match certificates but also conduct a pairwise comparison of valid RSA keys across all sets.

We found that 10.16% (17,141,441) of all keys are shared across sets, and 87% (14,862,767) of these duplicate keys (116,868 distinct) appear in distinct certificates (see Table VIII).

A. Vulnerable keys

In our initial analysis (Section VI), we discussed several weaknesses associated with all RSA keys, here we specifically examine weaknesses of the shared keys.

*1) **Weak key size**:* Our analysis found that 44% of shared keys can be considered weak, i.e., with key length < 2048 bits (Table IX). This can be interpreted as a strong indicator of the association of duplicate keys with their perceived vulnerability level.

*2) **ROCA**:* We identified one shared key vulnerable to ROCA vulnerability. The key was duplicated 16 times in the Rapid7 dataset with 3 different signature algorithms, i.e., md5WithRSAEncryption, sha1WithRSAEncryption, and sha256WithRSAEncryption. However, our cursory check revealed different issuers and owners corresponding to these certificates.

*3) **GCD-Factorable**:* Among the 793,694 factored RSA keys, 191,236 are duplicate keys that appear in distinct certificates. As expected, almost all of these factored keys are less than 2048 bits in length (Table XI). Interestingly, duplicate keys retrieved from SSH hosts, malware executables, and Android apps, appear to be noticeably absent from GCD factorable keys, i.e., we were able to factor only 12 SSH keys and none of the Android or malware keys.

TABLE VII: The top 20 operating systems seen in hosts sharing certificates

OS Name	Released Year	IP frequency	Unique shared certs	Key Size		Signature	
				≥2048 bits	<2048 bits	Strong*	Weak
Linux 3.18	2014	721,818 (51.72%)	7,970	5,847 (73.36%)	2,123 (26.64%)	502 (6.30%)	7,468 (93.70%)
ZyXEL ZyWALL 70 firewall (ZyNOS 3.65)	2008	173,036 (12.40%)	4,232	2,532 (59.83%)	1,700 (40.17%)	150 (3.54%)	4,082 (96.46%)
Aerohive HiveOS 6.8	2018	136,509 (9.78%)	2,053	1,278 (62.25%)	775 (37.75%)	115 (5.60%)	1,938 (94.40%)
Linux 3.13	2014	59,597 (4.27%)	1,107	737 (66.58%)	370 (33.42%)	136 (12.29%)	971 (87.71%)
iPXE 1.0.0+	2010	56,355 (4.04%)	965	629 (65.18%)	336 (34.82%)	64 (6.63%)	901 (93.37%)
Linux 2.6.32	2009	55,700 (3.99%)	1,309	976 (74.56%)	333 (25.44%)	72 (5.50%)	1,237 (94.50%)
Efficient Networks 5930 ADSL router	2002	43,924 (3.15%)	868	495 (57.03%)	373 (42.97%)	64 (7.37%)	804 (92.63%)
D-Link DWL-624+ or DWL-2000AP or TRENDnet TEW-432BRP WAP	2005-2007	19,926 (1.43%)	750	429 (57.20%)	321 (42.80%)	62 (8.27%)	688 (91.73%)
Panasonic BL-C210A webcam	2009	13,463 (0.96%)	140	104 (74.29%)	36 (25.71%)	35 (25.00%)	105 (75.00%)
Juniper Networks SSG 20 firewall	2006	10,397 (0.74%)	146	94 (64.38%)	52 (35.62%)	35 (23.97%)	111 (76.03%)
Apple iOS 8.0 - 8.1 (Darwin 14.0.0)	2014	9,070 (0.65%)	387	264 (68.22%)	123 (31.78%)	49 (12.66%)	338 (87.34%)
Canon i-SENSYS MF5490dn printer	2008	7,235 (0.52%)	109	78 (71.56%)	31 (28.44%)	33 (30.28%)	76 (69.72%)
Linux 2.6.18 - 2.6.22	2006-2007	7,048 (0.50%)	264	171 (64.77%)	93 (35.23%)	49 (18.56%)	215 (81.44%)
Vivint alarm panel (Linux 2.6.21)	ukn.	4,889 (0.35%)	174	106 (60.92%)	68 (39.08%)	35 (20.11%)	139 (79.89%)
Cisco 7200 router (IOS 12.4)	2005	4,870 (0.35%)	82	73 (89.02%)	9 (10.98%)	38 (46.34%)	44 (53.66%)
Panasonic WV-SP300 or WV-SF330 webcam	2010-2011	4,868 (0.35%)	72	63 (87.50%)	9 (12.50%)	32 (44.44%)	40 (55.56%)
Microsoft Windows Server 2012 R2	2012	4,868 (0.35%)	168	110 (65.48%)	58 (34.52%)	42 (25.00%)	126 (75.00%)
Epson UB-E02 print server	ukn.	4,756 (0.34%)	334	168 (50.30%)	166 (49.70%)	37 (11.08%)	297 (88.92%)
Moxa NPort 5610 terminal server	ukn.	4,540 (0.33%)	154	102 (66.23%)	52 (33.77%)	34 (22.08%)	120 (77.92%)
Linux 4.9	2016	4,040 (0.29%)	82	69 (84.15%)	13 (15.85%)	37 (45.12%)	45 (54.88%)
Total	-	1,346,909 (96.51%)	21,366	14,325 (67.05%)	7,041 (32.95%)	1,621 (7.59%)	19,745 (92.41%)

'*' Strong: signature algorithm is either sha256WithRSAEncryption, sha384WithRSAEncryption, or sha512WithRSAEncryption

TABLE VIII: The shared RSA keys

Datasets	Total number of shared keys	Unique shared in each set**	Shared keys with distinct certificates	Unique shared keys with distinct certificates**	Shared keys found in distinct certificates					
					Collected TLS/SSL	Collected SSH	Rapid7	Malware	SBA	Android apks
Collected TLS/SSL	767,709	754,324 (98.26%)	123,582	110,197 (89.17%)	*	48	123,581	11	805	0
Collected SSH	195,107	169,851 (87.06%)	3,961	2,502 (63.17%)	1,039	*	3,961	0	2	0
Rapid7	7,619,478	1,034,226 (13.57%)	6,556,755	116,867 (1.78%)	6,528,028	8,519	*	898	38,279	594
Malware	8,366,724	320 (<0.01%)	8,165,826	145 (<0.01%)	133	0	8,157,286	*	0	2,899,636
SBA	192,210	114,492 (59.57%)	12,524	4,593 (36.67%)	4,657	3	12,523	0	*	0
Android apks	213	196 (92.02%)	119	102 (85.71%)	0	0	117	58	0	*
Total	17,141,441	1,034,263 (6.03%) distinct across sets	14,862,767 (86.71%)	234,406 (1.58%)	116,868 keys (0.79%) are distinct across sets					

** duplicates within a set are removed, across sets retained

B. Sources of duplicate keys

The majority of the keys are shared between the Collected TLS/SSL and Rapid7 sets. To limit the impact of a potential key compromise, organizations use different keys for distinct purposes when they need more than one certificate. However, in some cases, for example, for consistency in single sign-on mechanisms using the SAML[12], load-balanced environments, multi-domain SSL certificates, and shared hosting situations, organizations may prefer or need to use the same key for different certificates, provided that the certificates are related in terms of the owner.

To understand the reasons behind duplicate keys in our collections, we analyzed the "Subject" field in selected certificates and subsequently parsed the "Organization" field of these shared keys. We observe that only 7,060 (6.04%) out of 116,868 distinct keys have similar "Organization" values. Hence, common practices of legitimate key reuse appear to be responsible for only a small amount of duplicate keys.

Our observations in Table IX also point out that around 99% of unique weak keys are associated with both Rapid7 (100%) and our own collected TLS/SSL (98.42%) datasets.

Interestingly, duplicate keys retrieved from SSH hosts, malware executables, and Android apps, appear to be compliant with NIST regulations (Table IX). When it comes to SSH keys, it is common for clients to be updated, which means that the majority of SSH keys should adhere to current standards. Similarly, keys associated with software compilation, such as malware executables and Android APKs, require compatibility with current standards due to compiler prerequisites. Consequently, these keys are more likely to be stronger.

The most commonly shared key that we could identify in our sets has been seen 3,454,586 times in the Rapid7 set.

[12]Security Assertion Markup Language

TABLE IX: Shared RSA public key size

Key size range(bits)	Total	Frequency						
		Total Unique	TLS/SSL	SSH	Rapid7	Malware	SBA	Android apks
0-1023	832,184 (5.59%)	2,193 (0.26%)	2,182	0	2,193	4	9	0
1024-2047	5,763,788 (38.78%)	35,583 (0.62%)	35,000	208	35,583	36	408	15
2048-4095	8,110,887 (54.57%)	76,695 (0.95%)	71,097	2,245	76,694	92	3,733	80
4096-8191	155,860 (1.05%)	2,383 (1.53%)	1,907	49	2,383	13	440	7
8192-up	48 (<0.01%)	14 (29.17%)	11	0	14	0	3	0
Total	14,862,767	116,868 (0.79%)	110,197	2,502	116,867	145	4,593	102

This key is associated with 3,454,586 distinct TLS certificates, mostly belonging to the same organization, Lancom Systems. We have different records for these certificates. They were served by 11 distinct IPs, and classified once by nmap as a "general purpose" device served by Microsoft Windows Server 2008 SP2. This key is weak, its key modulus length is 1024 bits. The associated certificates were signed with the sha1WithRSAEncryption algorithm, but our TLS scan showed this key still is being served in certificates for an extended period of time despite its obvious weaknesses.

Another source of concern is the presence of shared keys between web (Rapid7 and Collected TLS/SSL) and file (Malware and Android apks) datasets. For example, most RSA keys found in malware samples are served by TLS hosts, and 49% of keys used by Android apps are used to sign malware. This suggests that corresponding private keys are within the reach of malware authors or attackers. This presence of shared keys points towards a broader issue within the PKI ecosystem.

C. Key generators

Duplicated keys, where the same cryptographic key is generated more than once, serve as a warning sign and raise concerns about the security of the key generation process. Consequently, any similar keys produced using the same library should be viewed as questionable. It suggests a lack of proper security measures during key generation, making the generated keys unreliable and vulnerable to exploitation.

To identify the origin of the shared RSA keys, we have adopted an approach for a fine-grained RSA key origin attribution proposed by Branca et al. [14]. The approach is based on spatial characteristics of RSA moduli associated with different library implementations, consequently, allowing for accurate origin attribution.

We have retrained the Random Forest model using their generated set of 6.5 million RSA keys which contains details on the type and version of potential libraries used to generate each key. We utilized this model to deduce the libraries and their versions responsible for generating the shared RSA keys. The resulting set of 17,141,441 public keys contains 1,034,263 distinct RSA moduli. We retraced each modulus back to the original keys to confirm the corresponding dataset (see Table XIV).

We found that the shared keys were predominantly generated by OpenSSL (97%) and GnuTLS (3%) libraries. This is not surprising as OpenSSL is the most widely used open-source cryptographic library installed by default in many Linux kernel-based systems.

The problems with low entropy pool affecting the generation of RSA key prime numbers in OpenSSL 1.0.0 on Linux-based systems were discovered by Heninger et al. [4]. This is again not surprising as OpenSSL library versions 1.0 and 1.1 have a history of issues related to random number generation implementations (e.g., CVE-2015-0285 [69], CVE-2015-3216 [70], CVE-2019-1549 [71]). However, only a few versions of OpenSSL and GnuTLS libraries appear to be responsible for duplicate keys (Table X). For example, the OpenSSL 1.1.x library generated around 97% of duplicate keys. As expected (based on Tables II and VII), more than half of these keys are on the upper end of the key numerical range, i.e., 57% of OpenSSL 1.1.x keys have a length 2048 bits or more. This appears to be common among all affected keys.

Although OpenSSL release notes of the 1.1.1 version state that the random number generator was completely rewritten to address this problem, the presence of weak moduli in a considerable percentage of our shared keys suggests that conditions may persist and be related to the issues of the aforementioned entropy pool. It is worth mentioning that there are 189,405 SSH keys that our analysis predicts were likely generated using older versions of the OpenSSL library. This observation underscores not only the continued usage of outdated cryptographic libraries used in web protocols as well as libraries targeted at infrastructure services such as SSH (Table XIV).

TABLE X: Predicted libraries per modulus

Library	Number of moduli	Affected keys	Modulus size range (bits)				
			0-1023	1024-2047	2048-4095	4096-8191	8192-up
OpenSSL 1.1.x	1,002,727	17,004,035	824,147	6,444,725	9,556,530	178,406	227
GnuTLS 3.6.x	20,989	99,343	14,987	34,218	42,968	7,170	0
GnuTLS 2.2.x	10,451	37,440	5,834	14,747	16,505	354	0
OpenSSL 1.0.x	94	619	294	182	135	8	0
GnuTLS 3.1.x	1	2	0	2	0	0	0
GnuTLS 2.1.x	1	2	0	0	2	0	0
Total	1,034,263	17,141,441	845,262	6,493,874	9,616,140	185,938	227

.x stands for all minor versions

TABLE XI: GCD-factorable shared RSA keys

Impacted distinct shared keys	3,182
Impacted shared keys	191,236
Dataset	
TLS/SSL	3,856
SSH	12
Rapid7	187,368
Malware	0
Android apks	0
SBA	0
Key size	
0-1023	327
1024-2047	190,884
2048-4095	25

TABLE XII: Unresolvable signature algorithms seen in certificates with shared keys

Signature algorithm	Total certificates
1.2.840.113549.1.60.21	26,449
1.2.840.113549.1.60.20	8,613
1.2.840.113549.1.60.27	473
1.2.840.113549.1.60.26	29
1.2.840.113549.1.60.29	3
1.2.832.113549.1.1.4	1
1.2.840.113037.1.1.5	1
1.2.840.113548.1.1.5	1
1.2.840.114573.1.1.5	1
Total	35,571

D. Certificates with shared keys

Furthermore, to identify legacy public key material, we started from 116,868 distinct public keys that we identified to be shared across all sets, this allowed us to detect 14,895,604 certificates that 14,444,022 (96.97%) certificates were linked to either SHA1 (e.g. sha1WithRSAEncryption, sha1WithRSA) or MD5 hashing algorithms, both of which are no longer supposed to be used for general or browsing purposes, according to NIST recommendations [72].

Also, out of 14,895,604 certificates, a total of 35,571 certificates were found to be using not-standard OIDs as signature algorithms, as shown in Table XII.

We also observed that out of 35,571 certificates corresponding to 15,714 keys, 138 keys have been found to be related to 6,114 certificates, collected after 2017, related to hosting companies (OVH, GoDaddy.com, Inc., GANDI SAS), security devices (Fortinet Ltd., SonicWALL), wireless appliances (Ruckus Wireless, Inc., D-LINK), storage appliances (EMC Corporation), SSL libraries using custom formats (AlphaSSL), companies related to EV and DV certificate validation schemes (DigiCert Inc), and collaboration servers (Zimbra Collaboration Server).

IX. Discussion

*a) **Findings**:* The presence of duplicated cryptographic keys can be an alarming indicator when it deviates from established security guidelines. It implies a potential absence of adequate security protocols during the key generation and management, resulting in unreliable keys prone to exploitation. This duplication may stem from various reasons, e.g., errors in the key generation algorithm or weaknesses in the underlying library used for key generation. Regardless of the cause, the presence of duplicates raises red flags and casts doubt on the overall security of the PKI ecosystem.

The results derived through our analysis shed light on:

- *The persistently weak state of RSA keys:* Our analysis extends over a considerable length of time, spanning up to nine years in certain datasets (Rapid7 and Malware) which gives us a unique historical view of RSA key security. While numerous previous studies have suggested improvements in the quality of RSA keys, our analysis reveals a different outcome. For example, we factored 185,731 unique moduli corresponding to 793,694 RSA keys (181,784 unique). In 2012, Heninger et al. [4] reported finding divisors for 2,314 moduli for 16,717 distinct public keys, and in 2016, Hastings et al. [42] factored 313,000 RSA keys.
  Despite conducting our analysis almost 10 years after the initial report, we still observed significantly higher numbers, which is a disheartening outcome. The comprehensive and large-scale nature of our study, however, gives a more realistic view security state of the PKI ecosystem. Although the solution to this situation appears to be simple—enforcing strong key generation - many cryptographic libraries, such as OpenSSL, still allow for the generation of weak keys essentially weakening the PKI environment.
- *Shared RSA keys are weak:* We found that 87% (14,862,767) shared keys appear in distinct certificates, out of which we found that 44% are weak in terms of key length and 1.3% (191,236) were factored due to weak prime number selection (See Subsection VIII-A). We also noticed that 92% of all certificates shared among devices were found to be non-compliant with the NIST standards.
- *Origins of shared RSA keys are predictable:* We found that shared keys were predominantly generated using outdated and vulnerable libraries, resulting in inherently weak and vulnerable keys. Predicting the cryptographic library responsible for generating specific keys, while emphasizing the importance of key attribution, plays a crucial role in forensic investigations. It allows forensic analysts to untie the origins and traceability of cryptographic keys used in various security contexts, enabling a deeper understanding of the nature and scope of breaches or unauthorized access.
- *TLS certificates are widely reused for different purposes:* Aside from the presence of vulnerable cryptographic attributes, another factor contributing to the weaknesses in the PKI is the utilization of certificates beyond their intended scope. Certificates issued for a specific purpose should not be used for any other usage. In reality, the certificates are being used interchangeably. Almost all certificates found in malicious binaries, 8,299,100 (99.9%), were served by TLS hosts in the Rapid7 set. 2,338,672 (28%) certificates used in malicious binaries were also used for signing Android apps.
- *Overwhelmingly high use of obsolete algorithms among shared certificates:* 95% of the shared certificates are signed using deprecated hashing algorithms, which were already considered obsolete by the time of our certificate collection. Such a significant presence of non-compliant certificates points to the bigger problem within the PKI ecosystem.
  Weak hashing algorithms can undermine the integrity and authenticity of TLS certificates and allow attackers to create fraudulent certificates leading to impersonation and unauthorized access. It becomes easier for attackers to tamper with the certificate data without detection. This compromises the trustworthiness of the certificates and opens the door to various security risks, such as man-in-the-middle attacks or the interception of sensitive

information. Finally, weak hashing algorithms hinder the long-term security and validity of TLS certificates. As cryptographic attacks evolve and computational power increases, weak hashing algorithms become even more susceptible to brute-force and collision attacks. Although the certificate community (CA/B Forum) appears to be moving towards reduced longevity of certificates, the lack of enforcement for revocation of legacy certificates and re-issuance for devices using them leads to continued use of these certificates and consequently weakens the security of the public-key infrastructure.

- *The shared certificates are predominantly used by legacy embedded network devices:* Many shared certificates appear to be served by legacy devices. Over half of validated shared certificates (55%) come from devices that pre-date our data collection period. The rest are served by devices released in 2014. All these certificates are currently in use.

b) ***Recommendations****:*

- *Legacy devices*. An obvious solution to many of the discussed problems is adopting stricter practices and guidelines for key generation, storage, distribution, and revocation. It can mitigate risks associated with compromised or leaked keys. This is the path that the Certificate authority community has been following, i.e., gradually phasing out support for older standards and implementing shorter validity periods for certificates and keys. This unfortunately leaves often unupdateable legacy devices in an even more isolated state creating inherit security gaps. There are generally no guidelines for the treatment of legacy devices, and the approaches their owners take vary drastically from ignoring the problem, to downgrading security standards, and running several PKI infrastructures in parallel.
  Given the rapidly evolving technological capabilities and constant improvements in security policies, a more security cautious approach for embedded devices would be 1) creating consistent guidelines for the treatment of all devices across all libraries; 2) adopting automated updates of the device factory-default certificates, similar to how web hosting environments regularly update their certificates; 3) for devices maintaining outdated certificates and encountering obstacles in the renewal process, to minimize their exposure within broader networks as a means of reducing the attack surface.
- *The development of domain-specific policy engines*, which can interpret the requirements of a particular domain regarding certificate attributes and extensions while complying with the latest NIST recommendations, presents a potential approach to reducing the reuse of generic certificates for various purposes.
- *Client applications of certificate-using systems* can also contribute to the gradual improvement of certificate security by conducting compliance checks. These checks involve verifying the service provider's adherence to regulations and industry standards to ensure effective cryptographic key management, as well as monitoring certificate revocations.
- *Maintaining a large-scale view of PKI*. The analysis conducted in our study highlights the extent of sharing of certificates and keys across the PKI ecosystem. While individual CAs may not have a comprehensive view of the Internet, our developed *KeyExplorer* platform can be utilized for identifying instances of certificate reuse.

X. Ethics Considerations

Internet-wide network scans may be perceived as an adversarial and may trigger alerts from intrusion detection systems. In our study, we were careful to work within ethical boundaries. First, we ensured that no data was collected from hosts that necessitated credentials and no personal information was gathered during our research process. Second, we avoided establishing application-level connections with hosts to traverse domain boundaries.

XI. Conclusion

In this study, we conducted the most comprehensive Internet-wide scan and analysis of TLS/SSL certificates and RSA keys to date. Our study focused on the less explored phenomenon of RSA public key reuse on the Internet. Collecting a diverse set of over 254 million valid certificates and almost RSA 170 million keys from multiple sources across different points in time allowed us to not only investigate the cryptographic characteristics of reused certificates and keys but also confirm the persistence of their persistent spread in network devices over time.

Although browser vendors tend to follow the latest standards of cryptographic elements, we showed that deprecated hashing algorithms and non-compliant key sizes are still widespread in embedded network devices. The RSA keys found in malware samples are currently served by TLS hosts and Android apps. Known vulnerabilities remain unpatched on still accessible devices and files. Owners and issuers of certificates and keys should pay careful attention to the excessive use of duplicated keys. The cross-domain use of certificates necessitates the implementation of more domain-specific practices for certificate management systems. Our study provides a valuable tool for identifying vulnerabilities in cryptographic implementations, and we hope it will be utilized by the owners and issuers of the certificates and keys.

References

[1] M. Hanley, "Github security update: revoking weakly-generated ssh keys," Oct. 2021. [Online]. Available: https://github.blog/2021-10-11-github-security-update-revoking-weakly-generated-ssh-keys/

[2] M. Nemec, M. Sys, P. Svenda, D. Klinec, and V. Matyas, "The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli," in *24th ACM Conference on Computer and Communications Security (CCS'2017)*. ACM, 2017, pp. 1631–1648.

[3] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "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*, ser. IMC '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 427–444.

[4] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your ps and qs: Detection of widespread weak keys in network devices," in *Proceedings of 21st USENIX Security Symposium (USENIX Security 12)*. Bellevue, WA: USENIX, 2012, pp. 205–220.

TABLE XIII: Key properties of devices seen in shared certificates

Device Name	IP frequency	Unique shared certs	Key Size		Signature	
			≥2048 bits	<2048 bits	Strong*	Weak
general purpose	868,432 (63.20%)	9,703	7,001 (72.15%)	2,702 (27.85%)	578 (5.96%)	9,125 (94.04%)
firewall	189,588 (13.80%)	4,398	2,631 (59.82%)	1,767 (40.18%)	159 (3.62%)	4,239 (96.38%)
WAP	140,013 (10.19%)	2,123	1,299 (61.19%)	824 (38.81%)	118 (5.56%)	2,005 (94.44%)
specialized	61670 (4.49%)	1,057	671 (63.48%)	386 (36.52%)	66 (6.24%)	991 (93.76%)
broadband router	46349 (3.37%)	887	501 (56.48%)	386 (43.52%)	65 (7.33%)	822 (92.67%)
webcam	18339 (1.33%)	156	118 (75.64%)	38 (24.36%)	38 (24.36%)	118 (75.64%)
phone	9321 (0.68%)	398	270 (67.84%)	128 (32.16%)	50 (12.56%)	348 (87.44%)
printer	7618 (0.55%)	135	91 (67.41%)	44 (32.59%)	36 (26.67%)	99 (73.33%)
router	5731 (0.42%)	104	78 (75.00%)	26 (25.00%)	38 (36.54%)	66 (63.46%)
print server	4756 (0.35%)	334	168 (50.30%)	166 (49.70%)	37 (11.08%)	297 (88.92%)
terminal server	4545 (0.33%)	156	103 (66.03%)	53 (33.97%)	35 (22.44%)	121 (77.56%)
load balancer	3549 (0.26%)	117	80 (68.38%)	37 (31.62%)	40 (34.19%)	77 (65.81%)
VoIP adapter	3325 (0.24%)	185	117 (63.24%)	68 (36.76%)	43 (23.24%)	142 (76.76%)
switch	3062 (0.22%)	91	66 (72.53%)	25 (27.47%)	24 (26.37%)	67 (73.63%)
proxy server	2933 (0.21%)	118	82 (69.49%)	36 (30.51%)	36 (30.51%)	82 (69.49%)
media device	1565 (0.11%)	157	98 (62.42%)	59 (37.58%)	35 (22.29%)	122 (77.71%)
storage-misc	1496 (0.11%)	330	190 (57.58%)	140 (42.42%)	30 (9.09%)	300 (90.91%)
terminal	1111 (0.08%)	79	65 (82.28%)	14 (17.72%)	30 (37.97%)	49 (62.03%)
VoIP phone	413 (0.03%)	105	51 (48.57%)	54 (51.43%)	27 (25.71%)	78 (74.29%)
remote management	349 (0.03%)	48	41 (85.42%)	7 (14.58%)	24 (50.00%)	24 (50.00%)
bridge	13 (<0.01%)	5	5 (100.00%)	0 (0.00%)	5 (100.00%)	0 (0.00%)
Total	1,374,178	20,686	13,726 (66.35%)	6,960 (33.65%)	1,514 (7.32%)	19,172 (92.68%)

'*' Strong: signature algorithm is either sha256WithRSAEncryption, sha384WithRSAEncryption, or sha512WithRSAEncryption

TABLE XIV: Predicted libraries per dataset

Library	Number of moduli	Affected keys	Frequency					
			TLS/SSL	SSH	Rapid7	Malware	SBA	Android apks
OpenSSL 1.1.x	1,002,727	17,004,035	744,543	189,405	7,534,379	8,350,100	185,408	200
GnuTLS 3.6.x	20,989	99,343	15,474	3,794	58,950	16,607	4,509	9
GnuTLS 2.2.x	10,451	37,440	7,623	1,891	25,632	17	2,273	4
OpenSSL 1.0.x	94	619	67	17	515	0	20	0
GnuTLS 3.1.x	1	2	1	0	1	0	0	0
GnuTLS 2.1.x	1	2	1	0	1	0	0	0
Total	1,034,263	17,141,441	767,709	195,107	7,619,478	8,366,724	192,210	213

$.x$ stands for all minor versions

[5] A. Everspaugh, Y. Zhai, R. Jellinek, T. Ristenpart, and M. Swift, "Not-so-random numbers in virtualized linux and the whirlwind rng," in *2014 IEEE Symposium on Security and Privacy*, May 2014, pp. 559–574.

[6] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, ser. SEC'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 95–110.

[7] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in Android applications," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 73–84.

[8] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman, "The matter of heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14. New York, NY, USA: ACM, 2014, pp. 475–488.

[9] J. Li, Z. Lin, J. Caballero, Y. Zhang, and D. Gu, "K-hunt: Pinpointing insecure cryptographic keys from execution traces," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 412–425.

[10] P. Svenda, M. Nemec, P. Sekan, R. Kvasnovsky, D. Formanek, D. Komarek, and V. Matyas, "The million-key question—investigating the origins of RSA public keys," in *Proceedings of 25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 893–910.

[11] I. Muslukhov, Y. Boshmaf, and K. Beznosov, "Source attribution of cryptographic api misuse in android applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 133–146.

[12] O. Gasser, R. Holz, and G. Carle, "A deeper understanding of ssh: Results from internet-wide scans," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[13] A. Janovsky, M. Nemec, P. Svenda, P. Sekan, and V. Matyas, "Biased rsa private keys: Origin attribution of gcd-factorable keys," in *Computer Security – ESORICS 2020*, L. Chen, N. Li, K. Liang, and S. Schneider, Eds. Cham: Springer International Publishing, 2020, pp. 505–524.

[14] E. Branca, F. Abazari, R. R. Carranza, and N. Stakhanova, "Origin attribution of rsa public keys," in *Security and Privacy in Communication Networks*, J. Garcia-Alfaro, S. Li, R. Poovendran, H. Debar, and M. Yung, Eds. Cham: Springer International Publishing, 2021, pp. 374–396.

[15] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Measurement and analysis of private key sharing in the https ecosystem," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 628–640.

[16] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's pki," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 183–196.

[17] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[18] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Transport Layer Protocol," RFC 4253, Jan. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4253

[19] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, May 2008.

[20] J. Lennox, "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)," RFC 4572, Jul. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4572

[21] B. Kaliski and J. Staddon, "PKCS #1: RSA Cryptography Specifications

Version 2.0," RFC 2437, Oct. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2437

[22] NIST, "Digital Signature Standard (DSS)," National Institute of Standards and Technology, Tech. Rep., Feb 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf

[23] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. Ali Kaafar, "Tls in the wild: An internet-wide analysis of tls-based protocols for electronic communication," in *Proceedings 2016 Network and Distributed System Security Symposium*, ser. NDSS 2016. Internet Society, 2016.

[24] M. T. Paracha, D. J. Dubois, N. Vallina-Rodriguez, and D. Choffnes, "Iotls: understanding tls usage in consumer iot devices," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 165–178. [Online]. Available: https://doi.org/10.1145/3487552.3487830

[25] M. Dahlmanns, J. Lohmöller, J. Pennekamp, J. Bodenhausen, K. Wehrle, and M. Henze, "Missed opportunities: Measuring the untapped tls support in the industrial internet of things," *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247007445

[26] T. Chung, Y. Liu, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Measuring and applying invalid ssl certificates: The silent majority," in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 527–541.

[27] N. Samarasinghe and M. Mannan, "Tls ecosystems in networked devices vs. web servers," in *International Conference on Financial Cryptography and Data Security*, 2017.

[28] M. H. Hue, J. Debnath, K. M. Leung, L. Li, M. Minaei, M. H. Mazhar, K. Xian, E. Hoque, O. Chowdhury, and S. Y. Chau, "All your credentials are belong to us: On insecure wpa2-enterprise configurations," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1100–1117. [Online]. Available: https://doi.org/10.1145/3460120.3484569

[29] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter, "Ron was wrong, whit is right," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 64, 2012.

[30] T. Jager, J. Schwenk, and J. Somorovsky, "On the security of tls 1.3 and quic against weaknesses in pkcs1 v1.5 encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1185–1196.

[31] D. Felsch, M. Grothe, J. Schwenk, A. Czubak, and M. Szymanek, "The dangers of key reuse: practical attacks on ipsec {IKE}," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 567–583.

[32] D. Kim, B. J. Kwon, and T. Dumitraş, "Certified malware: Measuring breaches of trust in the windows code-signing pki," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1435–1448.

[33] F. Nezhadian, E. Branca, and N. Stakhanova, "Certificate reuse in Android applications," in *Proceedings of the Information Security Conference (ISC2023)*. New York, NY, USA: Association for Computing Machinery, 2023.

[34] H. Kang, J. wook Jang, A. Mohaisen, and H. K. Kim, "Androtracker : Creator information based android malware classification system," 2014.

[35] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962, Jun. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6962

[36] P. Eckersley. Sovereign Key Cryptography for Internet Domains. [Online]. Available: https://git.eff.org/?p=sovereign-keys.git;a=blob;f=sovereign-key-design.txt;hb=master

[37] T. H.-J. Kim, L.-S. Huang, A. Perrig, C. Jackson, and V. Gligor, "Accountable key infrastructure (aki): a proposal for a public-key validation infrastructure," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 679–690.

[38] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Crlite: A scalable system for pushing all tls revocations to all browsers," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 539–556.

[39] D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, and P. Szalachowski, "Design, analysis, and implementation of arpki: An attack-resilient public-key infrastructure," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 393–408, 2018.

[40] J.-G. Dumas, P. Lafourcade, F. Melemedjian, J.-B. Orfila, and P. Thoniel, "Localpki: An interoperable and iot friendly pki," in *E-Business and Telecommunications*, M. S. Obaidat and E. Cabello, Eds. Cham: Springer International Publishing, 2019, pp. 224–252.

[41] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage, "When private keys are public: Results from the 2008 debian openssl vulnerability," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 15–27.

[42] M. Hastings, J. Fried, and N. Heninger, "Weak keys remain widespread in network devices," in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 49–63.

[43] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *22nd USENIX Security Symposium (USENIX Security 13)*. Washington, D.C.: USENIX Association, Aug. 2013, pp. 605–620. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/durumeric

[44] OpenBSD, "ssh-keyscan(1)," https://man.openbsd.org/ssh-keyscan.1.

[45] Rapid7, "Rapid7 Sonar SSL/TLS Datasets," https://opendata.rapid7.com/sonar.ssl/.

[46] W. Mayer, A. Zauner, M. Schmiedecker, and M. Huber, "No need for black chambers: Testing tls in the e-mail ecosystem at large," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*. IEEE, 2016, pp. 10–20.

[47] VX-UNDERGROUND, "VX-UNDERGROUND APT Information," https://vx-underground.org/apts.html.

[48] VirusShare, "VirusShare," https://virusshare.com.

[49] PHPWatch, "PHP 8.4: OpenSSL: Minimum required OpenSSL version increased to 1.1.1." [Online]. Available: https://php.watch/versions/8.4/openssl-min-111

[50] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "Rfc 8017: Pkcs 1: Rsa cryptography specifications version 2.2," USA, 2016.

[51] Microsoft, "Object identifiers (ad ds)," 2019. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/ad/object-identifiers

[52] Z. Durumeric, "Microsoft object identifiers," 2012. [Online]. Available: https://zakird.com/2012/12/09/microsoft-oids

[53] D. Lee, "sigtool," https://github.com/doowon/sigtool.

[54] E. Barker and Q. Dang, "Recommendation for key management: Part 3," National Institute of Standards and Technology, Tech. Rep., Jan 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf

[55] E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, and S. Simon, "Recommendation for pair-wise key establishment using integer factorization cryptography," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., mar 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br2.pdf

[56] E. Barker, "Recommendation for key management: Part 1," National Institute of Standards and Technology, Tech. Rep., May 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[57] E. Rescorla, "HTTP Over TLS," RFC 2818, May 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2818

[58] CA/Browser Forum. (2013) Baseline requirements for the issuance and management of publicly-trusted certificates. [Online]. Available: https://cabforum.org/wp-content/uploads/Baseline_Requirements_V1_1_3.pdf

[59] NIST. (2020, April) Securing web transactions tls server certificate management. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-16.pdf

[60] Chronicle, "Abusing code signing for profit," May 2019. [Online]. Available: https://chroniclesec.medium.com/abusing-code-signing-for-profit-ef80a37b50f4

[61] D. Kim, B. J. Kwon, and T. Dumitraş, "Certified malware: Measuring breaches of trust in the windows code-signing pki," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1435–1448.

[62] R. Housley, T. Polk, and L. E. B. III, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3279, May 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3279

[63] NIST, "Research Results on SHA-1 Collisions ," Feb. 2017. [Online]. Available: https://csrc.nist.gov/News/2017/Research-Results-on-SHA-1-Collisions

[64] M. Stevens, A. Sotirov, J. Appelbaum, A. K. Lenstra, D. A. Molnar, D. A. Osvik, and B. de Weger, "Short chosen-prefix collisions for md5 and the creation of a rogue ca certificate," in *CRYPTO*, 2009.

[65] C. R. Dougherty, "MD5 vulnerable to collision attacks ," Dec. 2008. [Online]. Available: https://www.kb.cert.org/vuls/id/836068
[66] S. Turner and L. Chen, "MD2 to Historic Status," RFC 6149, Mar. 2011. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6149
[67] D. Shefanovski and S. Leontiev, "Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile," RFC 4491, May 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4491
[68] M. Souppaya, W. Haag, M. Akram, W. Barker, R. Clatterbuck, B. Everhart, B. Johnson, A. Kapasouris, D. Lam, B. Pleasant, M. Raguso, S. Symington, P. Turner, C. Wilson, and D. Dodson, "Securing web transactions tls server certificate management," 2020-06-16 2020.
[69] "Cve-2015-0285." [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0285
[70] "Cve-2015-3216." [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3216
[71] "Cve-2019-1549." [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-1549
[72] K. A. McKay and D. A. Cooper, "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) implementations," National Institute of Standards and Technology, Tech. Rep., aug 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br2.pdf