IoTLS 2.0: How Far Has IoT Industry Come in Securing Communications with TLS?

Chakshu Gupta
University of Twente
Enschede, The Netherlands

Nilufer Guldali
University of Twente
Enschede, The Netherlands

Marc Damie
University of Twente
Enschede, The Netherlands

Abstract

Transport Layer Security (TLS) is commonly used to secure communications over the Internet. However, since the rise of Internet of Things (IoT) devices, there have been concerns regarding the security practices adopted in the development of these devices. In particular, the study IoTLS (IMC'21) analyzed TLS practices in consumer IoT devices and revealed widespread use of insecure protocol versions, weak cipher suites, and improper certificate validation.

Our work reproduces the IoTLS analysis on a new set of 22 comparable consumer IoT devices to assess the advancement of security practices five years later. Our findings indicate significant improvements; for example, support for deprecated TLS versions has decreased from 45% to 9.1%. However, insecure configurations remain prevalent: 86.4% of devices still accept weak cipher suites. These results indicate that while manufacturers have made progress in adopting secure TLS practices, further efforts are needed to achieve a consistently secure IoT landscape.

Keywords

Transport Layer Security, Internet of Things, Network Security

ACM Reference Format:
Chakshu Gupta, Nilufer Guldali, and Marc Damie. 2026. IoTLS 2.0: How Far Has IoT Industry Come in Securing Communications with TLS?. In *19th European Workshop on Systems Security (EuroSec '26), April 27–30, 2026, Edinburgh, Scotland Uk.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3803525.3804979

1 Introduction

Transport Layer Security (TLS) is a protocol designed to secure communications between two devices by providing authentication, confidentiality, and data integrity. Operating independently of the application layer, TLS establishes a protected channel for data transmission over a network, thereby preventing eavesdropping, tampering, and message forgery [21, 28].

TLS establishes a secure connection through a handshake process, starting with the ClientHello and ServerHello messages. The ClientHello message informs the server about the client's capabilities, including the highest supported protocol version and the available cipher suites. A cipher suite is a defined set of algorithms that manage key exchange, authentication, digital signatures, and encryption [26]. Among these, key exchange (for establishing a shared secret) and encryption (for securing message contents) are significant for security analysis. In response, the server chooses its preferred options from the client's list and replies with a ServerHello message, specifying the selected protocol version, cipher suite, and additional parameters [18].

Despite TLS's security guarantees, older versions and misconfigurations pose significant risks. Prior studies [8, 11, 27] have demonstrated that many devices, particularly within the Internet of Things (IoT) ecosystem, are vulnerable to insecure TLS configurations. Common issues include the use of self-signed or expired certificates, weak cipher suites, and support for deprecated protocol versions. These vulnerabilities open the door to various attacks, such as Lucky 13, POODLE, and BEAST [20]. For example, the POODLE attack exploits a client's fallback to SSL 3.0 (Secure Sockets Layer), enabling an attacker to intercept communications.

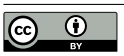
IoT devices, which often operate with limited resources and inconsistent update practices, are especially vulnerable. TLS is widely adopted in the IoT landscape to secure communications, but many devices still rely on outdated and insecure configurations [8, 11, 27]. For example, Dahlmanns et al. showed that 42% of TLS deployments in industrial appliances are insecurely configured [10], underscoring the broader concern for IoT security.

A notable study in this area is IoTLS [27], which evaluated TLS configurations in 40 consumer IoT devices (with traffic captured from 2018 to 2020). Although some devices supported recent protocol versions, the study found that more than half still permitted insecure cipher suites or failed to validate server certificates correctly. The authors concluded that TLS implementations in IoT devices require significant improvement and recommended uniform adherence to secure industry standards.

Our Contribution. Five years after the initial IoTLS study [27], our work revisits and reproduces its analysis using a new but comparable set of IoT devices. We examine 22 consumer IoT devices to assess whether manufacturers have meaningfully addressed previously identified weaknesses. Our results show measurable progress: support for deprecated TLS versions has dropped sharply from 45% to just 9.1%. Nevertheless, significant security gaps remain, as 86.4% of the tested devices still accept weak cipher suites. In addition to replicating the IoTLS analysis, we also evaluate TLS extension support and assess compliance with NIST guidelines. Our findings indicate that adoption of these extensions continues to lag behind recommended practices. Overall, while the security posture of TLS in IoT devices has improved, sustained and coordinated efforts are still needed to achieve a consistently secure IoT ecosystem.

2 Secure TLS practices

While TLS is theoretically secure, improper implementation or configuration choices can introduce vulnerabilities that undermine its security properties. Numerous attacks have exploited such



This work is licensed under a Creative Commons Attribution 4.0 International License.
EuroSec '26, Edinburgh, Scotland Uk
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2603-3/2026/04
https://doi.org/10.1145/3803525.3804979

implementation- and configuration-induced weaknesses in TLS. To mitigate these risks, the National Institute of Standards and Technology (NIST) has established configuration guidelines [21] which are globally considered the "gold standard" for secure communications. Compliance with these guidelines is commonly used as a metric for assessing the security of a TLS deployment, as deviations typically result in vulnerabilities that attackers can exploit.

This section first presents the main attacks against TLS implementations and then reviews the commonly accepted guidelines published by NIST that are established to prevent such attacks.

2.1 Main attacks on the TLS protocol

To highlight the significance of the NIST guidelines, this subsection presents an overview of notable attacks on TLS and their consequences. For a more detailed presentation of TLS attacks, we recommend referring to the study by Kotzias et al. [18].

BEAST [13] exploits predictable initialization vectors (IVs) in TLS 1.0 to decrypt messages encrypted using CBC cipher suites. The disclosure of this weakness was a first motivation for the transition toward TLS 1.2.

Lucky 13 [6] is a timing attack targeting cipher suites that rely on the CBC encryption mode. At the time of disclosure, the use of the RC4 cipher was initially promoted as a simple mitigation compatible with existing implementations. RC4 was later shown to be vulnerable to practical attacks, making authenticated encryption with associated data (AEAD) the preferred mitigation.

RC4 attacks [7] have identified statistical biases in the RC4 cipher. The initial attack presented by AlFardan et al. [7] was progressively improved, demonstrating that no effective mitigation could render RC4 secure. These attacks contributed to the deprecation of RC4 and encouraged the adoption of TLS 1.2 with AEAD cipher suites.

POODLE [23] exploits client fallback mechanisms to SSL 3.0 to mount attacks based on weaknesses in CBC cipher suites. The primary mitigation consists of disabling SSL 3.0 and preventing downgrades to deprecated protocol versions.

CRIME [30] exploits TLS compression mechanisms to recover secret information from authentication cookies. The commonly accepted mitigation is to disable compression in TLS.

FREAK exploits weaknesses in protocol negotiation to downgrade connections to export-grade RSA parameters, making the resulting encryption vulnerable to modern RSA attacks. Adrian et al. [5] demonstrated a similar downgrade attack targeting DHE cipher suites. Mitigations require clients and servers to prevent downgrades to weak cipher suites and cryptographic parameters.

SHA-1 collision attacks [33] exposed weaknesses in cipher suites relying on SHA-1, motivating its deprecation in favor of stronger hash functions such as SHA-256.

2.2 TLS configuration

Motivated by the forementioned attacks and other breakthroughs in cryptanalysis, NIST has published a set of guidelines [21] to ensure secure TLS deployments. This subsection outlines the main TLS configuration components along with the corresponding recommendations provided by NIST.

Protocol Versions. At the time of writing, NIST guidelines recommend the use of TLS 1.3. Devices that support TLS 1.3 are encouraged to support TLS 1.2 as well for backward compatibility. While legacy TLS versions were previously tolerated for interoperability, the current NIST guidelines [21] mandate the use of TLS 1.2 or 1.3 and explicitly forbid the use of TLS 1.0 and 1.1, except under highly restricted and documented legacy exceptions [24].

Certificate Authority (CA). During the TLS handshake, the server presents its certificate to the client to authenticate itself [21]. After the successful authentication, the communication channel with the server is secured against man-in-the-middle attacks. The server's Certificate message must include a valid certificate chain that ends at a **trusted certificate authority (CA)**. A certificate is deemed untrustworthy if it has expired or if its digital signature fails verification. Signature verification may fail if the certificate is self-signed or if it is signed by a CA that the client does not trust.

Key Exchange Algorithms. In TLS cipher suites, the key exchange mechanism can utilize either pre-shared keys (PSKs) or public-key cryptography. PSK-based key exchange does not rely on server certificates; instead, it uses a secret key that is established between the communicating parties before the TLS handshake. This approach is particularly well suited to performance- and resource-constrained environments [34].

When using public-key-based key exchange, algorithms such as Diffie-Hellman (DH), ephemeral Diffie-Hellman (DHE), elliptic curve Diffie-Hellman (ECDH), and ephemeral elliptic curve Diffie-Hellman (ECDHE) are employed. These algorithms are typically combined with a digital signature algorithm, which is applied either by a Certificate Authority (CA) to sign the server's certificate or by the server itself to authenticate during the handshake [21]. Among these options, DHE and ECDHE provide forward secrecy, thus improving the confidentiality of past sessions in the event of key compromise. Consequently, NIST recommends against the use of plain RSA for key exchange due to its lack of forward secrecy [21].

In TLS 1.3, the cipher suites no longer specify the key exchange algorithm. Instead, TLS 1.3 defines a small, fixed set of cipher suites, with key exchange negotiated separately.

Encryption Algorithms. The encryption algorithm defines how messages are encrypted and authenticated. NIST recommends the use of the Advanced Encryption Standard (AES) in authenticated encryption modes—specifically Galois/Counter Mode (GCM) or Counter with CBC-MAC (CCM)—rather than the legacy Cipher Block Chaining (CBC) mode [21]. Cipher suites that incorporate AES remain acceptable, whereas those relying on older algorithms such as the Data Encryption Standard (DES) and the International Data Encryption Algorithm (IDEA) are discouraged and have been removed from TLS 1.2 [15]. Furthermore, NIST no longer permits cipher suites based on Triple DES (TDEA/3DES).

In addition, ChaCha20-Poly1305 [19] has been verified for its security and is designed for high efficiency, particularly on platforms without hardware acceleration for AES. Although it has not yet been standardized by NIST, its adoption is increasing following its verification and endorsement by the IETF [25].

Extension Support. The TLS protocol supports the use of extensions to enhance its functionality. These extensions can be configured on both the client and server sides, based on the necessary features. Only the extensions that are needed for interoperability or enhanced security should be enabled. NIST identifies five TLS extensions as mandatory: (1) Renegotiation Indication, (2) Server Name Indication, (3) Extended Master Secret, (4) Signature Algorithms, and (5) the Certificate Status Request extension [21].

The Renegotiation Indication extension cryptographically binds the initial handshake and any subsequent renegotiation, protecting against renegotiation attacks. This can be configured in two ways: either by including the extension in the initial ClientHello message or by adding the "TLS_EMPTY_RENEGOTIATION_INFO_SCSV" signaling cipher suite value. The Server Name Indication (SNI) extension enables a client to specify the intended server when multiple virtual hosts share the same network address. The Extended Master Secret extension binds the master secret to the hashed transcript of the TLS handshake, thereby preventing certain man-in-the-middle attacks. Through the Signature Algorithms extension, clients advertise an acceptable set of hashing and signature algorithm pairs. Finally, the Certificate Status Request extension—commonly known as OCSP Stapling—allows the client to request that the server include a time-stamped, digitally signed proof of certificate validity directly in the handshake, eliminating the need to contact a third-party certificate authority and improving both privacy and connection performance.

3 Related Work

Since the standardization of TLS, many studies have analyzed the security of real-world deployments [16, 18, 22, 31]. For example, in 2018, Kotzias et al. [18] conducted a comprehensive study of TLS configurations and evaluated their compliance with accepted security guidelines.

Building on [18], subsequent works examined TLS deployments across different sectors, including IoT [11, 27]. As our work focuses on IoT, we discuss in detail the two main studies on TLS usage in this domain, namely the IoTLS study [27], which we replicate, and the study by Dong et al. [11].

The IoTLS study [27] analyzed TLS usage in consumer IoT devices, including supported TLS versions, cipher suites, certificate validation, and TLS fingerprinting. The authors evaluate 40 TLS-enabled devices deployed in an isolated environment comparable to the controlled IoT Lab setting used in our study. They combine active experimentation with passive traffic analysis collected over approximately two years from 2018 to 2020, which enables longitudinal observations of TLS usage.

Dong et al. [11] extended prior work by leveraging crowdsourced data collected from 2,014 IoT devices. Their study analyzes data gathered with IoT Inspector [17] between April 2019 and August 2020. In addition to client-side behavior, the authors examine TLS practices by analyzing server-side certificate handling.

Although these studies provide detailed analyses of TLS implementations in IoT devices, both in controlled environments and in real-world deployments, they are now more than five years old. In this work, we replicate the IoTLS study using newly collected data to assess whether TLS security in IoT ecosystems has improved over time.

Beyond TLS configuration, other works have evaluated IoT security from complementary perspectives. For example, Alrawi et al. [8] analyzed security vulnerabilities in 45 home IoT devices including weak authentication methods.

4 Experimental results

4.1 Setup

We conduct our experiment using 22 IoT devices to emulate a typical smart home environment, with all devices connected to a single router. To initiate network communication, we interact with each device either directly or through its companion app on the mobile phone. This interaction causes the devices to communicate with their respective backend servers, which requires TLS handshakes. We capture all network traffic at the router and extract the TLS handshake messages corresponding to each device. To facilitate analysis, we categorized the devices according to the IoTLS classification scheme into five groups—Assistants & Hubs, Automation, Security, Entertainment, and Appliances—as shown in Table 1.

Comparison with IoTLS [27] devices. The IoTLS study [27] examined 40 consumer IoT devices across various categories, including cameras, smart hubs, home automation, TVs, audio systems, and appliances. While our dataset is smaller in scale (22 devices compared to 40), it covers similar categories, including smart assistants, security cameras, and smart appliances. There is also a brand overlap with devices from Amazon, Philips, Samsung, TP-Link, Nest, and Ring appearing in both studies. However, differences exist in both brand and device selection. For example, the IoTLS study included an LG TV, whereas our dataset features a Philips TV. Overall, although there are some differences, the device categories in the two studies are comparable.

Although our dataset partially overlaps with that used in the original IoTLS study [27], one might ask why we did not conduct experiments on the same devices. This choice is motivated by the rapid evolution of the IoT ecosystem. Manufacturers frequently introduce new products while discontinuing older models, often limiting the production lifespan of individual devices. Consequently, procuring legacy IoT devices is challenging, as retailers typically make only the most recent versions available for purchase.

In addition to the difficulty of acquiring older devices, such devices often no longer receive software or security updates. Conducting experiments on the same devices used in IoTLS [27] would therefore risk primarily capturing the effects of software obsolescence rather than reflecting contemporary industry practices. To accurately evaluate whether the IoT industry has made meaningful progress, it is essential to examine newer devices that are actively maintained and supported.

The next subsections summarize the TLS security analysis of 22 devices, following the NIST guidelines detailed in Section 2. Table 2 highlights TLS misconfigurations identified in our dataset. Additionally, Table 4 compares our findings with those from the IoTLS study [27].

Table 1: List of the 22 TLS-supporting devices in our study.

Assistants & Hubs (4/22)	Automation (7/22)	Security (3/22)	Entertainment (4/22)	Appliances (4/22)
Amazon Echo	Calex Floor Lamp	Nest Camera Indoor	Philips TV	Philips Air Purifier
Amazon Echo Dot	Calex Motion Sensor	Ring Doorbell	Google Chromecast v3	Princess Smart Aerofryer
Google Nest Mini	TP-Link HS110	PNI Home Security System	Sony PlayStation 5	Samsung Smart Fridge
Philips Hue Bridge	TP-Link KP105		TCL Soundbar	Bosch Dishwasher
	Tuya Water Sensor			
	WellToBe Pet Feeder			
	iLife 10s			

Table 2: Notable TLS malpractices across the analyzed devices

Malpractice	Number of Devices
Offers cipher suite with 3DES	4/22
Offers version TLS 1.0	2/22
Offers version TLS 1.1	2/22
Offers plain RSA key exchange	13/22
Offers cipher suite with CBC	19/22
Offers cipher suite with SHA-1	17/22

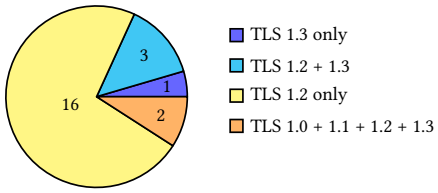


Figure 1: Proportion of TLS versions supported across analyzed IoT devices.

4.2 Protocol Versions

We begin by examining the TLS versions utilized by the devices and their backward compatibility support. Most of the devices (21 out of 22) supported TLS 1.2, while only the Amazon Echo supported TLS 1.3 exclusively.

Out of 22 devices, six supported TLS 1.3, and all six successfully established connections using that version. These devices were the Amazon Echo, Google Chromecast v3, Google Nest Mini, Nest Indoor Camera, Samsung Smart Fridge, and TCL Soundbar. If a downgrade were required, all of these devices except the Amazon Echo would have the capability to fall back to an earlier TLS version.

Of the 21 devices that supported TLS 1.2, 5 also supported version 1.3, while 16 supported version 1.2 exclusively. Notably, some of the cipher suites offered by these devices –such as 3DES, AES in CBC mode, and SHA-1– suggest potential compatibility with deprecated TLS versions 1.1 and 1.0 [24]. These cipher suites are no longer recommended for use with TLS 1.2, but they have not been removed from the TLS 1.2 registry to maintain backward compatibility.

Furthermore, two devices that support TLS 1.3—the Samsung Smart Fridge and TCL Soundbar—explicitly offer support for TLS versions 1.3, 1.2, 1.1, and 1.0. Supporting TLS 1.1 and 1.0 introduces potential security risks due to known vulnerabilities and deprecated cipher suites. Given that these devices already support the latest version of TLS, best practice would dictate disabling deprecated protocol versions along with their associated insecure cipher suites.

The Figure 1 provides an overview of the supported TLS versions among the analyzed IoT devices.

4.3 Encryption Algorithms

The majority of the devices (15 out of 22) established communication using the Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM). These devices employed key lengths of either 128 or 256 bits, in line with NIST recommendations [14]. AES is widely regarded as resistant to a range of cryptographic threats and is practical to implement across diverse platforms, including resource-constrained IoT devices [29].

In contrast, seven devices—the iLife T10s, TP-Link KP105, Calex Floor Lamp, Calex Motion Sensor, PNI Home Security System, Tuya Water Sensor, and WellToBe Pet Feeder—established communication using AES-128 in Cipher Block Chaining (CBC) mode. While AES itself remains secure, CBC mode is no longer recommended for use with TLS [20], as it is vulnerable to attacks such as Lucky 13 and POODLE.

The clients, referred to here as devices, typically offer a broader set of cipher suites than those ultimately selected by the server for the connection. This behavior can be exploited by an attacker through cipher-suite downgrade or selection of a suite with known vulnerabilities. Therefore, it is important to analyze not only the negotiated cipher suite but also the full list of cipher suites offered by the client. Four devices—the Google Chromecast v3, Nest Camera Indoor, Philips TV, and Sony PlayStation 5—advertised support for 3DES, a cipher suite that is disallowed by NIST due to well-known attacks [9, 21]. Although only 7 out of 22 devices actually used CBC mode during communication, 19 out of 22 offered cipher suites that rely on CBC mode for encryption. In addition, SHA-1 has been deprecated for use in TLS 1.2 and DTLS 1.2 [2]; however, 17 out of 22 devices still offered at least one cipher suite that uses SHA-1 as the message authentication algorithm.

4.4 Certificates

The devices can be categorized into three groups based on their authentication mechanisms: devices using TLS 1.3, devices using pre-shared keys (PSKs), and devices requiring certificates. Devices that support TLS 1.3 (6 out of 22) encrypt certificates during the initial handshake and subsequently rely on PSKs for continued communication with the server. Devices using PSKs in TLS 1.2 (5 out of 22) omit certificate messages altogether [34]. The remaining

devices (11 out of 22) rely on TLS 1.2 with certificate-based authentication. Out of these 11 devices, we note that three utilize mutual TLS (mTLS), where the client also sends a certificate to the server for authentication. In this section, we analyze the server certificates used by these 11 devices. For those using mTLS, we also analyze the client certificates.

4.4.1 Certificate Authority. A certificate is responsible for verifying the identity of the communicating device, while a Certificate Authority (CA) serves as a trusted third party that validates the certificate. We verify the validity of these CAs using the lists of trusted CAs maintained by Apple [4] and Mozilla [1].

We note that only 2 out of 22 devices used server certificates verified by (untrusted) private CAs. The Bosch Dishwasher used certificates issued by HCB CA 1 and HCB Server CA 2, neither of which is included in the trusted CA lists maintained by Apple or Mozilla. Similarly, the TP-Link HS110 used certificates issued by TP-LINK CA P1 and TP-Link CA, which are also absent from both trusted lists. In contrast, the TP-Link KP105—another device from the same manufacturer—used certificates issued by GoDaddy, which is a trusted public CA. The use of proprietary or non-public server certificate authorities introduces the risk of devices accepting spoofed server identities, particularly if the CA is mismanaged or becomes compromised.

All three devices using mTLS sent client certificates that were verified by untrusted private CAs, which is consistent with the findings of Dong et al. [12]. The Bosch dishwasher sent a client certificate signed by SMM CA B2 and Appliance CA 1, both issued by the organization BSH Hausgeräte GmbH, and thus effectively self-signed by the manufacturer. The Philips air purifier sent a client certificate signed by Amazon Web Services' private CA service. Finally, the Princess Aerofryer sent a client certificate signed by Appliance Access CA. As IoT devices typically communicate only with servers configured by their manufacturers, these client certificates need to be verified solely by the manufacturers' own servers. Hence, this practice may not pose the same level of risk as server certificates signed by private CAs.

4.4.2 Certificate validity. A common vulnerability in TLS deployments is the use of expired certificates, which are rejected by compliant TLS clients and strongly discouraged.

We identified one device (Princess Smart Aerofryer) that accepted an expired server certificate (by a few days); the certificate has since been renewed. This finding echoes concerns raised by IoTLS study [27] that many IoT devices do not properly verify certificate validity. Future work using active measurement techniques could help identify devices that accept expired certificates.

Industry standards recommend limiting certificate validity periods. While NIST historically recommended a maximum validity of three years for end-entity certificates, the CA/Browser Forum (an industry standards body including major certificate authorities and browser vendors) has more recently reduced this limit to 13 months for publicly trusted certificates.

In our study, all device server certificates except two complied with this guideline. The two non-compliant devices, both produced by TP-Link, used server certificates with a validity period of 15 years. In contrast, all three devices using mTLS exhibited very long validity periods for the client certificates: the Bosch dishwasher used a 10-year certificate, the Philips air purifier a 24-year certificate, and the Princess aerofryer a 15-year certificate. Shorter certificate lifetimes are encouraged to improve cryptographic agility and reduce exposure in the event of key compromise.

4.5 Key Exchange

For key establishment, 11 out of 22 devices use ECDHE. An additional six devices establish connections using TLS 1.3, which does not expose the key exchange algorithm through the negotiated cipher suite. Instead, TLS 1.3 supports only three key exchange mechanisms: (EC)DHE , pre-shared keys (PSKs) only, or PSKs with (EC)DHE [28]. Notably, key exchange methods used in earlier TLS versions that did not provide forward secrecy, such as plain RSA, have been removed in TLS 1.3.

Five devices depend solely on PSKs for key establishment. While the use of PSKs is generally discouraged due to the risk of key compromise—which would expose all past sessions that used the same key—they are still considered acceptable in closed or controlled environments, such as for IoT devices.

One device (TP-Link KP105) uses static RSA for key exchange, a method discouraged by NIST because it does not provide forward secrecy. A total of 13 out of 22 devices proposed cipher suites with static RSA for key exchange. If a server supports static RSA key transport in TLS versions earlier than 1.3, it may be vulnerable to server impersonation attacks. This risk can be mitigated by disabling support for static RSA key exchange [28].

4.6 Extension support

Paracha et al. [27] did not study extension support, so no direct comparison with the initial IoTLS study can be made on this aspect. Only 2 out of 22 devices (Google Nest Mini and TCL Soundbar) configure all five mandatory TLS extensions defined by NIST [21].

Among these mandatory extensions, the *Certificate Status Request* is the least commonly configured, with 19 of the 22 devices failing to support it. This results in less secure TLS handshakes for most devices, increasing their susceptibility to man-in-the-middle and privacy attacks. Additionally, 8 of the 22 devices do not configure the *Server Name Indication* extension, which can expose servers to attacks during connection initiation with these devices. Furthermore, 5 devices lack support for the *Signature Algorithms extension*, creating a high risk of handshake failure or, in the worst case, a downgrade to weak cryptographic algorithms.

Three devices are missing the *Extended Master Secret* extension. Without this extension, the master secret of a TLS session is not cryptographically bound to the specific handshake, allowing an attacker to synchronize keys across different sessions. Finally, 2 of the 22 devices do not configure the Renegotiation Indication extension or add it to the cipher suites list. As a result, these devices do not support secure renegotiation, leaving their TLS connections vulnerable to man-in-the-middle (MitM) injection attacks [3]. Table 3 summarizes these observations.

4.7 Compression methods

Similar to extension support, IoTLS [27] did not report the compression methods used by their devices. In our sample, all devices disabled TLS compression, in line with NIST recommendations.

Table 3: List of the missing mandatory TLS extensions across the 22 analyzed devices, with *None* showing devices that do not support any mandatory extension. Some devices miss multiple extensions, hence the total exceeds 22.

Missing Extension	Number of Devices
Renegotiation Indication	2
Server Name Indication	8
Extended Master Secret	3
Signature Algorithms	5
Certificate Status Request	19
None	2

Therefore, all devices are compliant with current best security practices with respect to compression methods.

5 Have TLS practices improved in the IoT industry?

Our study highlights several significant improvements compared to the IoTLS study [27]. One key observation is the wider adoption of TLS 1.3, which increased from 17% in IoTLS to 27.3% in our study. Another notable improvement is the substantial reduction in the proportion of distrusted certificate authorities (CAs), which decreased from 20% to 0.9%.

We also observe meaningful improvements in the use of cipher suites. In IoTLS [27], although most devices supported cipher suites that offered forward secrecy, connections were frequently established without it. In contrast, nearly all devices in our analysis established connections using forward secrecy, except for a single device. This improvement may be attributed to clients supporting more up-to-date cipher suites; however, it is more likely due to servers prioritizing cipher suites that provide forward secrecy.

Despite these advances, some insecure practices remain prevalent. Specifically, 9.1% of devices still offer deprecated TLS versions, such as TLS 1.0 and TLS 1.1. Moreover, although insecure cipher suites are not used to establish connections, 86.4% of devices continue to offer them. This may be because such cipher suites remain listed in the TLS registry despite being deprecated or no longer recommended. While the occurrence of these malpractices has declined in the last five years, their continued presence remains problematic, indicating that further security improvements are necessary.

An overview of these improvements, comparing our results with those of IoTLS [27], is presented in Table 4.

In addition to the analyses performed in the IoTLS study, we examined the use of TLS extensions by IoT devices. Because this aspect was not evaluated in IoTLS, we cannot assess improvements or declines in extension adoption over the past five years. Nevertheless, our findings indicate that the use of TLS extensions still falls short of the guidelines set by NIST [21]. Notably, four devices using TLS 1.3 still failed to configure all required extensions, leaving certain vulnerabilities unmitigated. Furthermore, the majority of devices did not implement the *Certificate Status Request* extension. As a result, these devices lack an efficient early-warning mechanism for detecting compromised or revoked certificates, increasing their exposure to MitM attacks. On a positive note, our analysis shows

Table 4: Comparison of TLS practices observed in IoTLS [27] (2021) and IoTLS 2.0 [Our study] (2025)

	IoTLS	IoTLS 2.0
Deprecated TLS Versions (1.0/1.1)	45% (18/40)	9.1% (2/22)
Devices Establishing Connections Without Forward Secrecy	55% (22/40)	4.5% (1/22)
Devices offering Insecure Cipher Suites	85% (34/40)	86.4% (19/22)
Devices w/ Distrusted server certificates	20% (8/40)	0.9% (2/22)
TLS 1.3 Adoption Rate	17.5% (7/40)	27.3% (6/22)

that all devices correctly configured the *Compression Methods* extension to *NULL*, as recommended by NIST, thereby protecting against compression-based side-channel attacks [32].

5.1 Post-quantum cryptography transition

While our primary goal is to assess improvements observed over the past five years, another topic has become increasingly central in the security community: post-quantum cryptography. As quantum computing capabilities advance, both public and private organizations are promoting the adoption of cryptographic standards that remain secure against quantum attacks [35].

Some companies, such as Google [35] and Cloudflare [36], have begun transitioning parts of their TLS infrastructure to post-quantum cryptography, for example through the adoption of ML-DSA signatures. These organizations are leading the effort, and eventually, IoT devices will also need to incorporate quantum-safe primitives.

Currently, none of the 22 devices in our sample support post-quantum cryptography. This is not particularly concerning, given that large-scale transitions have only recently begun and standardization is still ongoing. Nevertheless, it will be important to monitor the IoT sector in the coming years to evaluate adoption and progress toward post-quantum migration. Addressing the vulnerabilities discussed earlier remains the immediate priority, but preparing for the post-quantum transition should follow closely.

6 Conclusion

We evaluated the current state of TLS usage in IoT devices through passive traffic analysis and a reproduction of the IoTLS study [27]. Our results demonstrated that the IoT industry has made tangible progress in addressing previously identified TLS vulnerabilities. Support for TLS 1.3 has increased from 17.5% to 27.3%, while reliance on untrusted certificate authorities has dropped from 20% to 0.9%. Despite these gains, many devices still expose insecure handshake options, with 86.4% of devices continuing to offer weak cipher suites, indicating an incomplete transition away from legacy settings. Overall, the implementation of TLS in IoT devices lags behind the recommendations set by NIST. Although security practices are improving, further efforts are needed to fully remove legacy settings and achieve consistently secure TLS implementations across the IoT ecosystem.

References

[1] [n. d.]. Mozilla Included CA Certificates - MozillaWiki. https://wiki.mozilla.org/CA/Included_Certificates

[2] [n. d.]. RFC 9155: deprecating MD5 and SHA-1 signature hashes in TLS 1.2 and DTLS 1.2. https://datatracker.ietf.org/doc/rfc9155/

[3] 2012. Transport layer security renegotiation Remote Man-in-the-Middle attack vulnerability 2012. https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/Cisco-SA-20091105-CVE-2009-3555

[4] 2025. Available root certificates for Apple operating systems - Apple Support 2025. https://support.apple.com/en-us/103272

[5] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, et al. 2015. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 5–17.

[6] Nadhem J. Al Fardan and Kenneth G. Paterson. 2013. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *2013 IEEE Symposium on Security and Privacy*. 526–540. doi:10.1109/SP.2013.42

[7] Nadhem AlFardan, Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob CN Schuldt. 2013. On the security of {RC4} in {TLS}. In *22nd USENIX Security Symposium (USENIX Security 13)*. 305–320.

[8] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1362–1380. doi:10.1109/SP.2019.00013

[9] Karthikeyan Bhargavan and Gaëtan Leurent. 2016. On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 456–467.

[10] Markus Dahlmanns, Johannes Lohmöller, Jan Pennekamp, Jörn Bodenhausen, Klaus Wehrle, and Martin Henze. 2022. Missed Opportunities: Measuring the Untapped TLS Support in the Industrial Internet of Things. In *Proceedings of the 17th ACM Asia Conference on Computer and Communications Security (ASIA CCS '22)*. 1–14. doi:10.1145/3488932.3497762

[11] Hongying Dong, Hao Shu, Vijay Prakash, Yizhe Zhang, Muhammad Talha Paracha, David Choffnes, Santiago Torres-Arias, Danny Yuxing Huang, and Yixin Sun. 2023. Behind the Scenes: Uncovering TLS and Server Certificate Practice of IoT Device Vendors in the Wild. In *Proceedings of the 2023 ACM on Internet Measurement Conference* (Montreal QC, Canada) *(IMC '23)*. Association for Computing Machinery, 457–477. doi:10.1145/3618257.3624815

[12] Hongying Dong, Yizhe Zhang, Hyeonmin Lee, Kevin Du, Guancheng Tu, and Yixin Sun. 2024. Mutual TLS in Practice: A Deep Dive into Certificate Configurations and Privacy Issues. In *Proceedings of the 2024 ACM on Internet Measurement Conference* (Madrid, Spain) *(IMC '24)*. Association for Computing Machinery, New York, NY, USA, 214–229. doi:10.1145/3646547.3688415

[13] Thai Duong and Juliano Rizzo. 2011. Here come the ⊕ ninjas.

[14] Morris J. Dworkin. 2007. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Technical Report NIST SP 800-38D. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-38D

[15] Pasi Eronen. 2009. DES and IDEA Cipher Suites for Transport Layer Security (TLS). RFC 5469. doi:10.17487/RFC5469

[16] Ralph Holz, Jens Hiller, Johanna Amann, Abbas Razaghpanah, Thomas Jost, Narseo Vallina-Rodriguez, and Oliver Hohlfeld. 2020. Tracking the deployment of TLS 1.3 on the web: a story of experimentation and centralization. *SIGCOMM Comput. Commun. Rev.* 50, 3 (July 2020), 3–15. doi:10.1145/3411740.3411742

[17] Danny Yuxing Huang, Noah Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. 2020. IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 2, Article 46 (June 2020), 21 pages. doi:10.1145/3397333

[18] Platon Kotzias, Abbas Razaghpanah, Johanna Amann, Kenneth G. Paterson, Narseo Vallina-Rodriguez, and Juan Caballero. 2018. Coming of Age: A Longitudinal Study of TLS Deployment. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*. 415–428. doi:10.1145/3278532.3278568

[19] Adam Langley, Wan-Teh Chang, Nikos Mavrogiannopoulos, Joachim Strombergson, and Simon Josefsson. 2018. RFC 7905: CHACHA20-Poly1305 cipher suites for Transport Layer Security (TLS). https://datatracker.ietf.org/doc/rfc7905/

[20] Olivier Levillain, Baptiste Gourdin, and Hervé Debar. 2015. TLS Record Protocol: Security Analysis and Defense-in-depth Countermeasures for HTTPS. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '15)*. 225–236. doi:10.1145/2714576.2714592

[21] Kerry A. McKay and David A. Cooper. 2019. *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations*. Technical Report NIST SP 800-52 Rev. 2. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-52r2

[22] Georg Merzdovnik, Klaus Falb, Martin Schmiedecker, Artemios G. Voyiatzis, and Edgar Weippl. 2016. Whom You Gonna Trust? A Longitudinal Study on TLS Notary Services. In *Data and Applications Security and Privacy XXX*, Silvio Ranise and Vipin Swarup (Eds.). Springer International Publishing, Cham, 331–346.

[23] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. 2014. This POODLE bites: exploiting the SSL 3.0 fallback. *Security Advisory* 21 (2014), 34–58.

[24] Kathleen Moriarty and Stephen Farrell. 2021. Deprecating TLS 1.0 and TLS 1.1. RFC 8996. doi:10.17487/RFC8996

[25] Y. Nir and A. Langley. 2018. Chacha20 and Poly1305 for IETF protocols. https://www.rfc-editor.org/rfc/rfc8439

[26] Ramzi A. Nofal, Nam Tran, Carlos Garcia, Yuhong Liu, and Behnam Dezfouli. 2019. A Comprehensive Empirical Analysis of TLS Handshake and Record Layer on IoT Platforms. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Association for Computing Machinery, 61–70. doi:10.1145/3345768.3355924

[27] Muhammad Talha Paracha, Daniel J. Dubois, Narseo Vallina-Rodriguez, and David Choffnes. 2021. IoTLS: understanding TLS usage in consumer IoT devices. In *Proceedings of the 21st ACM Internet Measurement Conference (IMC '21)*. Association for Computing Machinery, 165–178. doi:10.1145/3487552.3487830

[28] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. doi:10.17487/RFC8446

[29] J Shanthalakshmi Revathy, M. Surya Prakash, K Lingeshwaran, J Davidv Dhinakaran, and V Harish. 2025. A Comprehensive Study of the Advanced Encryption Standard (AES) for Secure Communications. In *2025 3rd IEEE International Conference on Industrial Electronics: Developments & Applications (ICIDeA)*. 1–6. doi:10.1109/ICIDeA64800.2025.10963018

[30] Juliano Rizzo and Thai Duong. 2012. The CRIME attack. In *ekoparty security conference*, Vol. 2012.

[31] Richard Roberts, Yaelle Goldschlag, Rachel Walter, Taejoong Chung, Alan Mislove, and Dave Levin. 2019. You Are Who You Appear to Be: A Longitudinal Study of Domain Impersonation in TLS Certificates. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2489–2504. doi:10.1145/3319535.3363188

[32] Amichai Shulman. 2013. A Perfect CRIME? Only TIME Will Tell. https://api.semanticscholar.org/CorpusID:51760263

[33] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. 2017. The first collision for full SHA-1. In *Annual international cryptology conference*. Springer, 570–596.

[34] Hannes Tschofenig and Pasi Eronen. 2005. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279. doi:10.17487/RFC4279

[35] Kent Walker and Hartmut Neven. 2026. *The quantum era is coming. Are we ready to secure it?* https://blog.google/innovation-and-ai/technology/safety-security/the-quantum-era-is-coming-are-we-ready-to-secure-it/ Google Blog: Innovation & AI, accessed 2026-02-09.

[36] Bas Westerbaan. 2025. *State of the post-quantum Internet in 2025*. https://blog.cloudflare.com/pq-2025/ Cloudflare Blog, accessed 2026-02-09.