CONVERGE
TECHNOLOGY SOLUTIONS
CYBERSECURITY

Transport Layer Security (TLS)

Comparing TLS 1.2 and 1.3

By:

Jeffry Natzke, Senior Engineer

Jeffry.Natzke@convergetp.com
March 23, 2023

Version 1.0

Overview

Transport Layer Security (TLS) is the protocol used between client and server for encrypted communication on the internet today. The latest version of the protocol, TLS 1.3, is a big step forward in the security and performance evolution of TLS. However, it has downsides for organizations.

TLS 1.3 offers greater privacy and data security, but the encryption that provides this security impedes network monitoring and presents challenges for network security. What's more, moving to and supporting TLS 1.3 may pose challenges related to deployment, architecture, troubleshooting, and policy. To accommodate TLS 1.3, organizations must rearchitect their security stacks appropriately. Because of this, adoption has slowed.

This document describes how the TLS protocol functions and how TLS 1.3 differs from TLS 1.2 and prior versions. It also discusses the development, workings, risks, and vulnerabilities of the latest TLS version. Though TLS secures many applications, this analysis focuses on the Hypertext Transfer Protocol Secure (HTTPS) application.

With a better understanding of how TLS 1.3 works and how it differs from prior versions, organizations can be in a better position to move to the current version more smoothly.

The history of TLS

In 2013 the Internet Engineering Task Force (IETF), the standards body that defines internet protocols, began standardizing the latest version of the Transport Layer Security (TLS) protocol. The newest version of TLS, TLS 1.3 IETF RFC8446, was published by the IETF in 2018. Today, TLS provides encryption and ensures the authenticity of HTTPS websites. TLS 1.3 incorporates significant changes to improve data security and performance.

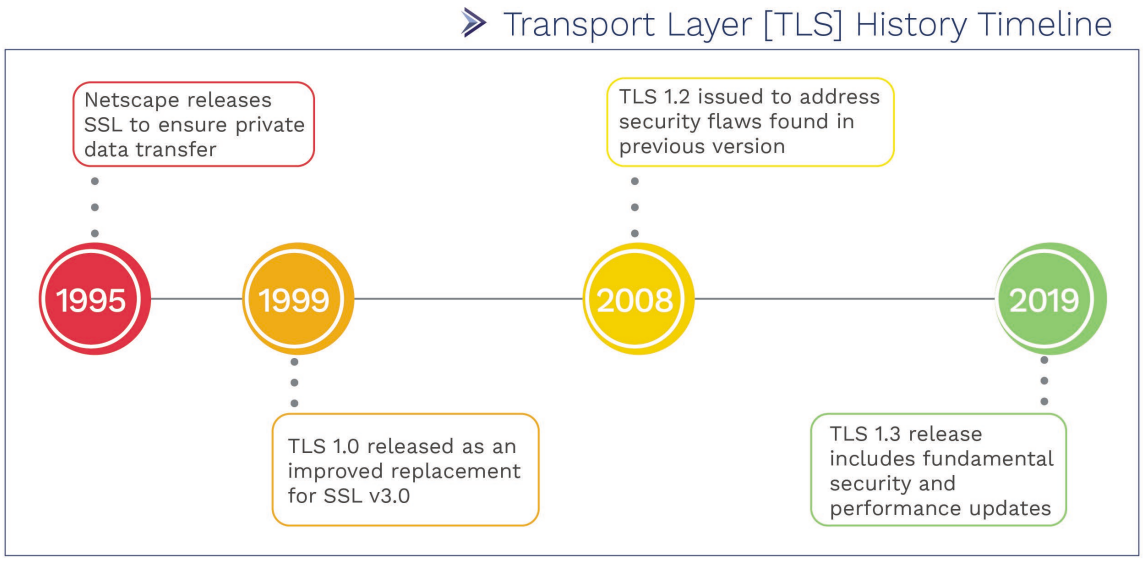


Figure 1: History of Transport Layer Security (TLS)

TLS Evolution

The predecessor to TLS is the Secure Socket Layer (SSL) protocol. Netscape developed the SSL protocol in the mid-1990s to provide secure communications for users of its Netscape browser. The IETF embraced and standardized this proprietary Netscape SSL version in 1999 as TLS 1.0. TLS has continued to evolve to address security flaws and issues over the years, as shown in Figure 1. In its latest update, TLS 1.3, the IETF made fundamental changes from TLS version 1.2.

TLS Technical Backgrounder

Simply put, TLS provides a way to exchange sensitive information through an insecure medium. Specific mechanisms in the TLS protocol provide secure communications between a client and server from a pool of available options. Combining the cryptographic techniques below, allowing keys to lock or unlock functions is crucial to providing a secure communication channel:

- Authentication
- Authorization
- Encryption

More detailed information on encryption and cryptographic key types is available here.

The primary benefits of using the TLS protocol are:

- Confidentiality
- Authentication
- Integrity

Providing **confidentiality** for transferred data ensures that the data between the client and server cannot be deciphered by anyone intercepting the traffic flow. Encryption has been necessary for online internet connectivity, typically used for sending credit card information or remote logins. In recent years, HTTPS has been used for more and more applications on the internet to protect user traffic from eavesdropping and various injection attacks.

Authentication of the destination server the client is talking to ensures that a client connects and interacts with the intended server.

Verifying the **integrity** of messages and communication is extremely important in confirming that no tampering or corruption of data has occurred.

Encryption and key exchanges

There are two types of encryption/decryption used to provide secure communications:

- Asymmetric
- Symmetric

Asymmetric encryption uses different keys for encryption and decryption. Asymmetric keys are tied to certificates generated by certificate authorities (CAs). This authority signs these certificates with an associated private and public key. The public key is contained in the publicly attainable certificate (shared with anyone), and the private key is only known to the certificate owner. A public key encrypts traffic from a client to a server (certificate owner), and the private key can only decrypt this traffic. Asymmetric encryption is slower and more complex, but it allows the derivation of the associated keys over an insecure medium.

Symmetric encryption uses the same key for traffic encryption and decryption. Symmetric encryption is faster, requiring the sender and receiver to exchange keys outside the symmetric encryption channel or tunnel.

TLS uses both asymmetric and symmetric encryption. The most significant difference between TLS 1.3 and previous TLS versions is the methods used to negotiate how traffic is encrypted.

TLS 1.2 Backgrounder

The standard defining the TLS 1.2 protocol is RFC 5246, released in August 2008. This RFC has been made obsolete by RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3.

TLS 1.2 handshake

For a browser (client) and web server to agree on a symmetric encryption key, they must exchange cryptographic data. This exchange is the TLS handshake. TLS 1.2 requires two round trips to complete the handshake for HTTPS traffic. The steps below describe the basic message flow for TLS 1.2:

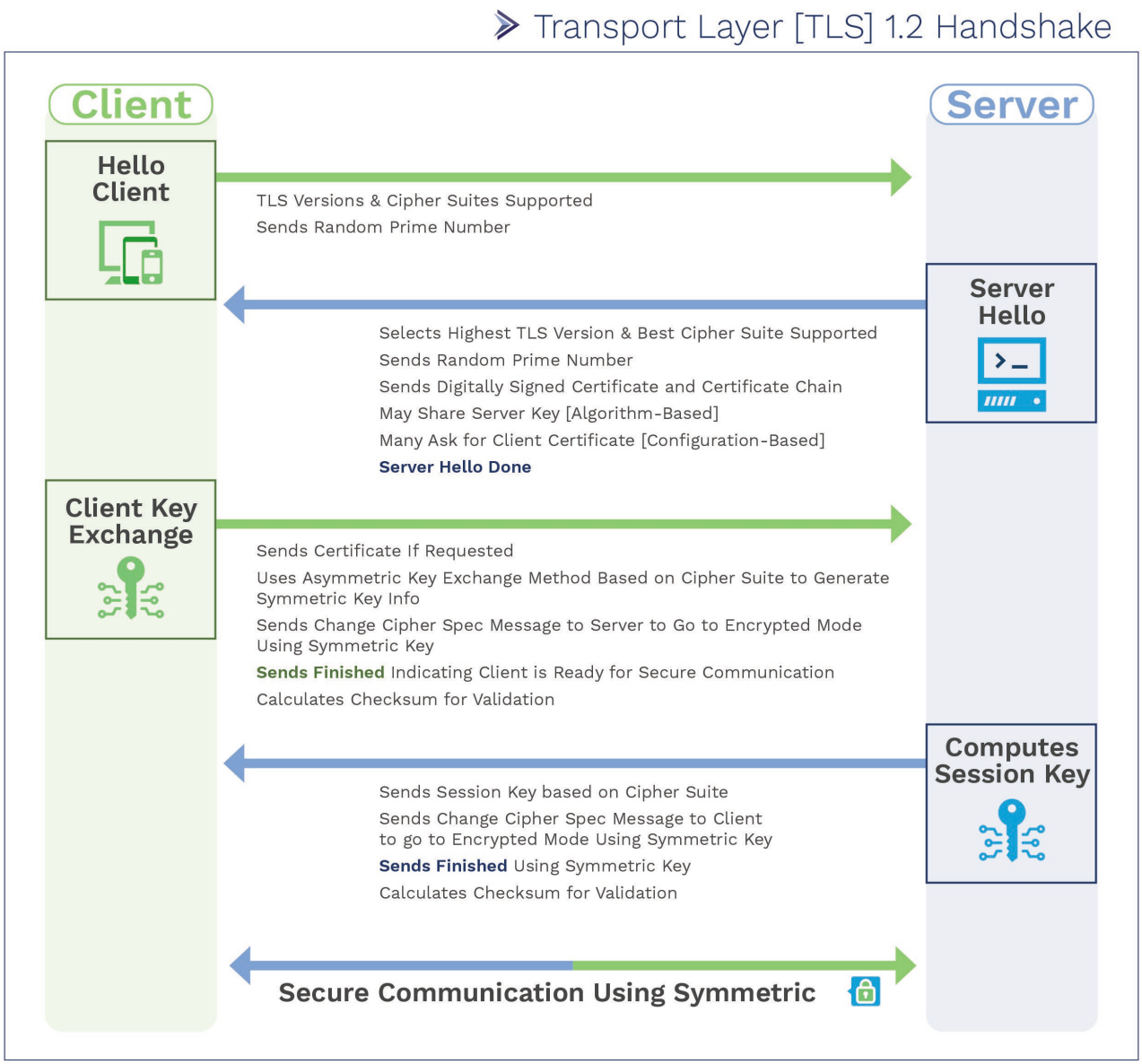
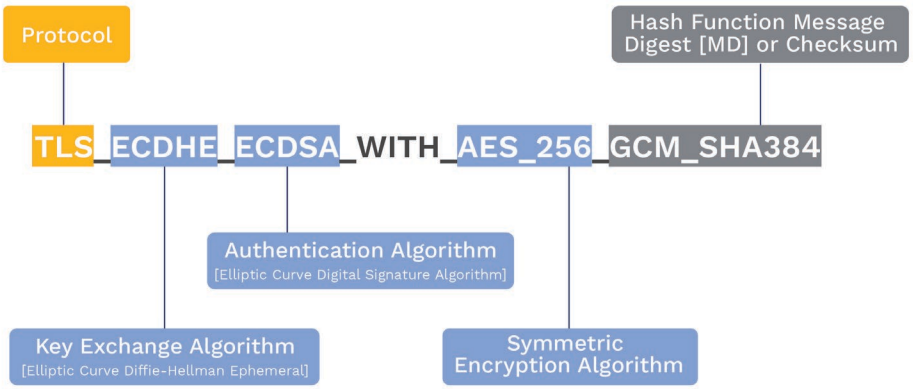


Figure 2: TLS 1.2 Handshake

TLS Hello messages

The client browser begins the process with the Client Hello message directed to the HTTPS server. The TLS version and cipher suites supported by the browser are sent in the Client Hello message. The browser also sends a random prime number.

A cipher suite is a formatted text string that specifies all the TLS-related ciphers the browser supports. Here's an example of a cipher suite embedded in the Client Hello message:



Since browsers typically support many cipher suites, multiple cipher suites are sent to the server. The server chooses one for the TLS session. For TLS version 1.2, the number of supported cipher suites is more than 35.

After receiving the Client Hello message, the server responds with the Server Hello message. Server Hello messages may consist of single or multiple TCP packets. The server chooses the highest TLS version and best cipher suite supported by both the client and server. The server sends this information to the client and sends its own randomly selected prime number to the client. The connection is terminated if the client and server do not share TLS versions or cipher suites in common.

Additionally, the server typically sends its digitally signed certificate and certificate chain. The certificate chain includes the leaf certificate and any intermediate certificates. As part of certificate authentication, the certificate is signed by a certificate authority (CA). Because the CA signs the certificate, the client can verify its legitimacy. The client verification steps include the following:

- Checking the digital signature.
- Verifying the certificate chain.
- Checking for other potential issues like expired or invalid domain names.
- Verifying the server has the certificate's private key (via key exchange).

Depending on the key exchange cipher, the server may send the client key exchange information as an optional message. If the Diffie-Hellman key exchange method is used, the server is required to provide additional data.

The server sends a Server Hello Done message to let the client know it has sent all messages related to the Client/Server Hello message exchange, as depicted in Figure 2 above, steps one and two. This exchange requires one round trip between the client and the server.

It's important to point out a significant security flaw regarding the TLS protocol up to and including version 1.2: The Client and Server Hello messages are transferred in clear text. A man-in-the-middle attack could easily capture this information. Under the right conditions, a replay attack would allow an attacker to capture and hijack the TLS session. More discussion on this is in the following TLS 1.3 sections.

Key exchange and session key generation

A second round trip is required to complete the remainder of the TLS protocol interaction. From Figure 2, steps three and four complete the exchange of key information and the generation of the symmetric encryption keys used to encrypt client/server traffic for the remainder of the TLS session.

Step three begins the client key exchange process. However, the client should reply with its CA-signed certificate if the server requests this information during the Server Hello. HTTPS servers requiring a client certificate are rare. If the server requests a client certificate and the client does not provide one, the connection is terminated.

The client generates information to create the symmetric encryption key and sends it to the server (step three). This is the client's contribution to the session key. The steps used depend on the key exchange method selected by the server and sent to the client in the Server Hello message. If the RSA key exchange algorithm is used, this key information is encrypted with the server's public key and sent to the server by the client. When a Diffie-Hellman key exchange is used, this message contains the client's Diffie-Hellman public key.

Next, the client sends the Change Cipher Spec message. This message lets the server know the client has generated the session key. From this point, the client will use this key to encrypt traffic to the server. The client then sends a Finished message encrypted using the newly created symmetric session key. A checksum is also generated to verify the integrity of the handshake.

In step four, the server performs the same process. It decrypts the key information sent by the client and computes the session key. The server then sends its Change Cipher Spec message, indicating it is moving to encrypted communication. A Finished message using the same checksum is sent to the client.

The TLS handshake is now complete (step five). The client and server each have symmetric session keys and MD keys (hash-based message authentication code, or HMAC, checksums). Referred to as key pairs, they effectively form two encryption tunnels to support the symmetric encryption sessions, as shown below:

Tunnel 1 – Client symmetric and HMAC keys to encrypt/decrypt **client-to-server** session traffic:

- Both the client and server have these keys.
- The client encrypts traffic and generates the HMAC MD with this key pair.
- The server decrypts traffic and calculates the HMAC MD checksum with this key pair.

Tunnel 2 – Server symmetric and HMAC keys to encrypt/decrypt **server-to-client** session traffic:

- Both the client and server have these keys.
- The server encrypts traffic and generates the HMAC MD with this key pair.
- The client decrypts traffic and calculates the HMAC MD checksum with this key pair.

Because this is symmetric encryption, the client and server have both key pairs (all four keys).

Encrypted application traffic between the client browser and web server now flows. As the session layer is now encrypted (HTTP, HTML, and JavaScript), visibility above the TCP layer for any packet captures used for troubleshooting will be hindered. TLS/SSL traffic inspection requires specialized appliances or cloud-based systems to intercept and participate in the TLS setup process so that encrypted traffic may be unencrypted to allow various cybersecurity tools to inspect this traffic. Cybersecurity tools include:

- IDS/IPS appliances/services
- DLP appliances/services
- Malware/sandbox appliances/services

See RFC 5246 for more details on the TLS 1.2 protocol.

TLS 1.3 Backgrounder

The standard defining the TLS 1.3 protocol is RFC 8446, released in August 2018. This updated version of the TLS protocol provides better data security and privacy and improves speed and efficiency. Supporting TLS 1.3, however, may require infrastructure changes and browser and web server software upgrades.

Significant changes to TLS 1.3 compared to TLS 1.2:

- Fewer supported cipher suites (less secure ciphers removed)
- The new standard removes cipher suites using static keys like RSA and Diffie-Hellman.
- All cipher suites use ephemeral keys.
- Round trips for TLS negotiation are reduced from 2 to 1, making TLS 1.3 faster.
- Perfect Forward Secrecy (PFS) protocol is mandated in TLS 1.3.
- Methods to prevent TLS downgrade attacks are included, providing enhanced security.
- Zero round-trip time supported (0RTT).

Changes to the TLS protocol specified in the TLS 1.3 standard protect against the following cybersecurity attacks. This is not a comprehensive list:

- DROWN: RSA encryption targeted
- SLOTH: Related to hashing
- POODLE: TLS version fallback
- FREAK: RSA encryption key exploit
- Logjam: Diffie-Hellman key exploit

Additional details on supported TLS 1.3 cipher suites

The number of supported cipher suites was reduced from more than 35 in TLS 1.2 to five in TLS 1.3, listed below:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_CCM_SHA256
- TLS_AES_128_CCM_8_SHA256

The listed cipher suites use Diffie-Hellman ephemeral (DHE)-based asymmetric key exchange schemes rather than static key exchange algorithms like RSA and non-ephemeral DH. Each cipher suite is classified as an authenticated encryption with associated data (AEAD) cipher. The TLS 1.3 protocol standard now mandates this class of cipher. AEAD is defined in the IETF standard RFC 5116. This standards document establishes a uniform interface and a registry for such algorithms. Authenticated encryption is a form of encryption providing both plaintext encryption and a means to check its authenticity.

See RFC 5116 for more details on AHED ciphers.

As can be seen below in Figure 3, the handshake is significantly shorter than previous TLS versions of the protocol:

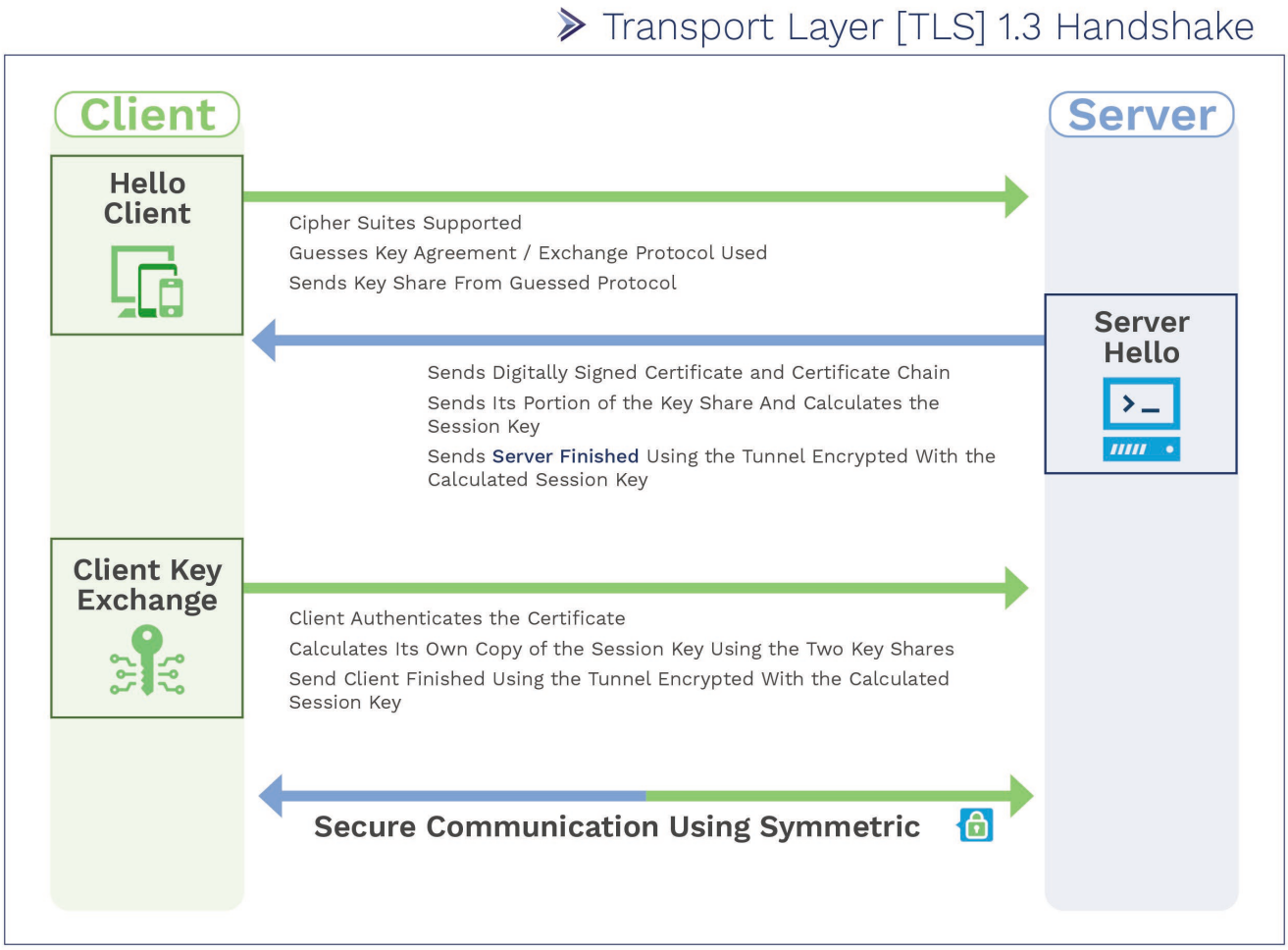


Figure 3: TLS 1.3 Handshake

TLS 1.3 handshake

Like prior versions of the TLS protocol, version 1.3 requires exchanging cryptographic data to enable symmetric key encryption between a client and server. A handshake facilitates the exchange and generation of key information. TLS 1.3 reduces the round trips from 2 to 1 when moving from version 1.2 to 1.3, as this latest version dramatically improves the efficiency of the TLS protocol. See Figure 3 above.

TLS 1.3 Hello messages

Figure 3 shows a Client Hello message (step one) initiates the process. However, this Client Hello message sends significantly more information than prior TLS versions. Again, the number of supported cipher suites in TLS 1.3 is five compared to more than 35 for TLS 1.2. This reduction in cipher suite choices allows the client to move to the asymmetric key exchange phase during the Client Hello message phase. Supported cipher suites are transferred

to the server first, then the client guesses what key agreement/exchange protocol will be used and sends its key share from the guessed protocol, as shown in step one of Figure 3.

In step two, the server responds with its Server Hello message. This response includes the following information:

- Certificate and certificate chain (assuming the client guessed the correct AHEAD cipher)
- The server's portion of the key share (server calculates the session key)
- Server Finished message (encrypted using the server's session key)

Finally, the client will authenticate the server certificate and use the two key shares to calculate its copy of the session key since the client has all the relevant information. The client then sends its encrypted Finished message to the server (step three).

The TLS 1.3 handshake is now complete (step four). Again, two encryption tunnels are formed to support the symmetric encryption sessions between the client and server.

Handshake improvements in TLS 1.3

Comparing Figure 2 with Figure 3, we can see that the legacy TLS protocol requires a client/server Hello message exchange (one round trip) followed by the Client/Server Key Exchange message exchange (second round trip). Both message phases (steps one through four in Figure 2) are optimal for TLS 1.2, as additional round trips may be necessary for specific negotiation scenarios.

The TLS 1.3 negotiation process, Figure 3, requires a single round trip to complete most of the process (though arguably, it could be considered a round trip and a half). This reduction represents an efficiency improvement resulting in a significantly faster negotiation process. It is also noteworthy that now only the Client Hello message is sent in the clear, and all Server Hello message reply traffic (such as the server certificate) is encrypted. This new aspect of TLS 1.3 is a significant security improvement over TLS 1.2, where both the Client and Server Hello message traffic is sent in the clear.

The reduction in required round trips in TLS 1.3 is due to the reduced number of cipher suites supported. One clear security issue with supporting such a high number of cipher suites in TLS 1.2 (over 35) is the complexity and possible confusion associated with potential server-side misconfigurations, making internet users vulnerable to known exploits.

TLS 1.3 reduces not only the number of cipher suites but also the number of negotiated elements. In TLS 1.2, cipher suites must handle and process the information below:

- Types of supported certificates
- Hash function (SHA1, SHA2, etc.)
- Message authentication code (MAC)
- Digital signatures
- Key exchange algorithm
- Symmetric encryption cipher
- Cipher mode

Completing TLS session negotiations requires TLS 1.2 cipher suites to use four algorithms:

- Key exchange
- Authentication
- Bulk cipher
- Hashing algorithm

The number of negotiated ciphers drops to two when using TLS 1.3:

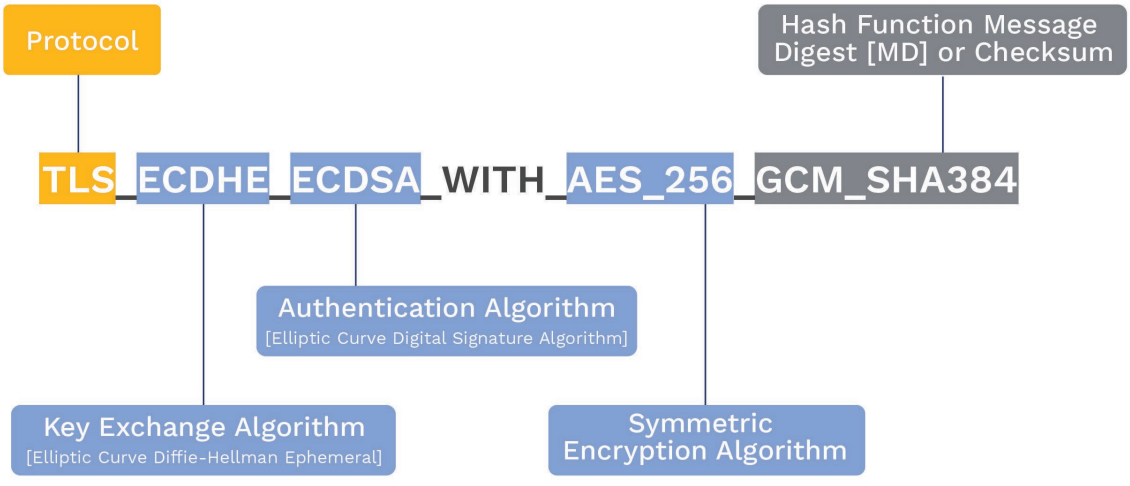
- Bulk cipher
- HKDF (HMAC-based extract-and-expand key derivation function) hash

Removing confusion by reducing the number of cipher suites and supporting the most secure, efficient algorithms was the goal of the IETF in putting forward the TLS 1.3 standard in RFC 8446. These key exchange algorithms belong to the Diffie-Hellman ephemeral (DHE) schemes. Thus, the client can now send its key share information in the Client Hello message, which helps improve negotiation efficiency. All cipher suite schemes using static keys like RSA and non-ephemeral DH have been completely removed. Perfect Forward Secrecy (PFS) is also mandated in the TLS 1.3 standard. A subsequent section covers details of PFS. These changes significantly improve the security and efficiency of the TLS 1.3 protocol.

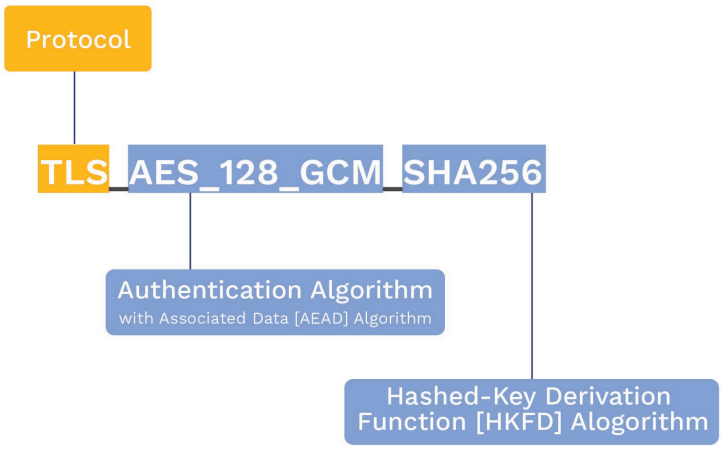
TLS 1.3 cipher suites – more details

The IETF removed support for older, less secure algorithms. Streamlining the specified cipher suites shortens the handshake from two round trips to one. Negotiated ciphers are reduced from four negotiations/algorithms to two.

From this:



To this TLS 1.3 cipher suite example:



Using a version of the Diffie-Hellman ephemeral key exchange algorithm makes it unnecessary for the first two algorithms in the TLS 1.2 cipher suite to be included in the cipher suite name. Those functions still operate but are no longer negotiated during the handshake. The cipher suites for TLS 1.3 have a shortened format.

TLS 1.3 zero round-trip resumption

Zero round trip time resumption (0RTT) is a TLS 1.3 mechanism that allows sessions to resume without going through the entire TLS handshake. In TLS version 1.2 and prior, session resumption is enabled using session IDs and tickets. With TLS 1.2, the server can decide not to honor a ticket so that a full TLS handshake can be initiated with the client as usual. In the TLS 1.2 implementation of 0RTT, specific encrypted session information is stored. These static elements introduce vulnerabilities and violate PFS (more on PFS later). TLS 1.3 uses a more secure solution to remedy this issue.

TLS 1.3 uses pre-shared keys (PSKs) specific to 0RTT. These keys can be negotiated as part of the initial handshake. A subsequent handshake is required to perform resumption using the PSKs. Additionally, the server side can implement a policy to force the connection to re-negotiate the entire handshake if the session resumption time is too long.

See RFC 5077 for TLS 1.2 details. See RFC 8446 for TLS 1.3 details.

0RTT does present some cybersecurity risks. For example, if an attacker can access an encrypted session and capture the client's first request, including the 0RTT data, it would be possible to "replay" the request later to gain access. Secondly, the 0RTT handshake might ultimately compromise Perfect Forward Secrecy (PFS) by providing the means to decrypt prior sessions. Reducing the session resumption time in the server policy mentioned above would effectively minimize this risk.

Perfect Forward Secrecy (PFS)

As stated previously, TLS 1.3 mandates PFS. This requirement in the protocol led to the removal of anything but ephemeral-based Diffie-Hellman cipher suites. Ephemeral means "something that lasts for a very short time."

PFS is a feature of all supported cipher suites in TLS 1.3. The idea of PFS is "to guarantee that decryption of past network communications is not possible." In other words, PFS ensures that older communications cannot be deciphered using a current, compromised key. Thus, all supported TLS 1.3 cipher suites use some version of the Diffie-Hellman ephemeral (DHE) key exchange protocol.

See a primer on how the Diffie-Hellman key exchange algorithm works.

Downgrade attacks — version negotiation improvement

TLS 1.3 changes version negotiation, improving it over TLS 1.2. As discussed earlier, the TLS Client Hello message includes a list of supported versions of the TLS protocol. The server selects the highest supported version and indicates its choice in the Server Hello message. If both parties support it, new fields in the initial client/server exchange force TLS 1.3 negotiation. Specifically, in TLS 1.3, the Client Hello message indicates TLS 1.3 support using a new supported_versions extension. The session terminates if a party supporting TLS 1.3 attempts to downgrade the session.

Backward compatibility with TLS 1.2 is ensured by having the Client Hello message indicate TLS 1.2 using the legacy_version field. TLS 1.3 servers know when TLS 1.3 is negotiated because of the supported_versions extension, whereas a TLS 1.2 server will see a TLS 1.2 Client Hello message and proceed with TLS 1.2 negotiation.

Additionally, the random number in the Server Hello message in TLS 1.3 allows a specific value to be written in the last eight bytes when TLS 1.2 or below is negotiated. This mechanism enables a TLS 1.3 client to detect a downgrade attack if the client initially reaches a server supporting TLS 1.3.

Compared to the TLS1.2 protocol, the above TLS 1.3 techniques, and the fact that only five highly secure cipher suites are supported, reduces the risk of downgrade attacks.

See appendices C, D, and E of RFC 8446 for details regarding security and downgrade attacks.

The pros and cons of TLS 1.3

TLS 1.3 improves security, increases speed and efficiency, and offers better privacy over previous protocol versions:

- **Improved security**
  - Traffic is harder to intercept because all secure session setup is completed in one round trip.
  - The TLS 1.3 protocol design supports a method to prevent downgrade attacks.
  - Encryption is improved because only the initial client handshake request is unencrypted.
  - Perfect Forward Secrecy (PFS) is used with all cipher suites, making it impossible to decrypt session traffic with prior compromised session keys.
- **Better privacy**
  - More of the handshake is encrypted than previous TLS versions, making it more difficult to intercept traffic.
- **Improved speed and efficiency**
  - The TLS Handshake is streamlined, requiring a single round trip to protect traffic.
  - Supports zero RTT for session resumption.
  - Reduces the number of supported cipher suites to five which makes cipher suite selection more efficient.
  - Algorithms are faster (elliptical curve) at calculating public and shared key values.

Increased security provided by the TLS protocol adds challenges for enterprise security teams wanting to integrate or adopt TLS 1.3.

General adoption challenges of TLS 1.3:

- **Deployment**
  - Infrastructure changes may be necessary to address improved security measures.
  - Modifications to applications are necessary to enable the TLS 1.3 protocol.
  - New or updated licenses may be necessary for related software or devices to support TLS 1.3.
- **Policy and practices**
  - Policies must be reviewed prior to adoption.
  - Policy modifications might be necessary to incorporate the higher security level of TLS 1.3.
  - Use case-by-case evaluation to understand options and required outcomes.

What to consider before deploying TLS 1.3

A leading cloud security vendor reports blocking 24 billion threats over HTTPS between October 2021 and September 2022. This represents a 20% increase from the 20.7 billion threats blocked in 2021, which was a 314% increase over the prior year. This accelerating trend by threat actors of using encrypted traffic to obfuscate malware payloads and attack traffic, in general, requires enterprises to inspect (SSL inspection) encrypted network traffic.

Security tools and infrastructure participating in the SSL inspection strategy are impacted; evaluating this infrastructure is essential.

Consider these characteristics in evaluation:

- Inbound and outbound traffic session CPU processing requirements.
- SSL inspection architecture related to visibility into encrypted traffic. Must be implemented using inline architecture to detect malicious payloads and activities, including:
  - Malware
  - Command and control activities
  - Data exfiltration
- Tools requiring traffic decryption, to include:
  - Firewalls
  - Intrusion detection systems (IPS)
  - Data loss protection systems (DLP)

Address these implementation requirements:

- Centralization
- Firewall scaling to handle the increased CPU resources TLS 1.3 decryption requires
- Architectural trend (simplifies implementation)
- Traffic-mirroring inspection architectures do not work with TLS 1.3
- Analysis:
  - Out-of-path traffic analysis is no longer possible.
  - Inline traffic analysis is the only architecture that works.
  - Requires maintaining open sessions for inspection duration because of the use of ephemeral keys.
  - Requires additional endpoint security capabilities.

How TLS 1.3 impacts inspection

Version 1.3 is a major overhaul of TLS and could negatively impact inspection techniques. These common SSL inspection methods work with the current non-PFS implementations of TLS 1.2:

- Dedicated appliance in TAP Mode (port mirror connection)
- Nex-gen firewalls do not terminate TCP connections
- Proxy (true inline with separate connections to client and server)

When implementing TLS 1.3:

- TAP Mode will no longer work because of PFS (non-static encryption keys). This method will also fail in TLS 1.2 when stronger cipher suites supporting PFS are available.
- Next-gen firewalls will still work, but performance will suffer on existing platforms. Higher performance is required for TLS 1.3 and new TLS 1.2 cipher suites. An appliance hardware refresh may be needed to scale to the more CPU-intensive cipher suites being used.
- True inline proxy appliance hardware refresh required to scale to the more CPU-intensive cipher suites used by TLS 1.3 and new TLS 1.2 cipher suites.
- True inline, cloud-based proxy solutions, such as Security as a Service (SECaaS), should be well-positioned to continue providing TLS inspection. Most cloud-based proxy implementations are built to allow easy scaling for the additional decryption loading.

When SSL inspection breaks or cannot scale, downstream threat-detection engines that depend on it will be blind to encrypted TLS traffic. This includes the following security devices:

- IDS and IPS
- DLP
- Sandbox

With TLS 1.3, the server certificate (transferred in the TLS Server Hello message) is encrypted. SSL inspection solutions must be capable of extracting and using the Server Name Indication (SNI) TLS extension and the destination server IP address since this information is transferred to the server in the non-encrypted Client Hello message. This is sufficient information to allow SSL inspection solutions to establish connections to the server. Solutions that don't support this capability cannot complete the secondary TCP connection to the origin server.

As part of the TSL 1.3 specification, Perfect Forward Secrecy (PFS) is a mechanism that ensures that the keys exchanged are temporary, not static. Ephemeral exchange algorithms like ECDHE generate their public-private key pair on-the-fly when a TLS connection is established. TLS 1.3 eliminates static key exchange implementations like RSA. This significantly increases key exchange security.

Recommended actions for moving to TLS 1.3

The following considerations and actions are recommended for a smooth TLS 1.3 implementation:

- Conduct planning and strategy sessions with stakeholders to discuss:
  - Budget
  - Projects
  - Licensing
  - Upgrade
  - Maintenance contracts

- Modify infrastructure to align with required architectural and system scaling adjustments:
  - Deploy new hardware.
  - Plan any network changes.
  - Evaluate device and traffic inspection requirements.

- Validate that the following can support TLS 1.3:
    - Website servers and services
    - Internal browsers

Closing thoughts

The preceding sections discussed and explained the latest version of the TLS protocol, TLS 1.3, compared to its predecessor TLS 1.2. Changes made in version 1.3 significantly improve security and performance and reduce overall complexity in TLS. Improved security and performance were the main goals of the IETF when the organization began work on the standard in 2013.

The standard will continue to evolve to improve performance and minimize cybersecurity vulnerabilities. For example, both TLS 1.2 and 1.3 send the Client Hello message in the clear, leaving a security weakness to be addressed in future TLS revisions or non-TLS encryption methods currently being explored.

The shift to TLS 1.3 provides upgraded security, privacy, and performance benefits, but it can be complex. Organizations without the needed in-house expertise or time should connect with an experienced third-party partner to guide successful implementation. When reviewing potential partners, look for firms with security and network architectural design experience, proficiency with the tools and infrastructure involved in TLS encryption processing, and expertise in implementing and troubleshooting downstream inspection tools.

Converge Technology Solutions is a services-led, software-enabled, IT & Cloud Solutions provider focused on delivering industry-leading solutions. Converge's global approach delivers advanced analytics, application modernization, cloud platforms, cybersecurity, digital infrastructure, and digital workplace offerings to clients across various industries. Converge supports these solutions with advisory, implementation, and managed services expertise across all major IT vendors in the marketplace. This multi-faceted approach enables Converge to address the unique business and technology requirements for all clients in the public and private sectors.