TLS 1.3 CIP Security Impacts

Jack Visoky
Security Architect and Sr. Project Engineer
Rockwell Automation

Joakim Wiberg
Team Manager Technology and Platforms
HMS Industrial Networks

Nancy Cam-Winget
Distinguished Engineer
Cisco Systems

Presented at the ODVA
2018 Industry Conference & 18th Annual Meeting
October 10, 2018
Stone Mountain, Georgia, USA

Abstract

CIP Security is currently built on TLS 1.2. The next major update of TLS (1.3) was recently published in mid-2018. TLS 1.3 brings several new features and functions, as well as eliminating some of the existing functionality for TLS 1.2. The changes in TLS 1.3 provide potential opportunities for improvement in both performance and security of CIP Security. However, besides these potential benefits there are also important functions of TLS 1.2 that will likely be removed in TLS 1.3. This paper explores the potential benefits and pitfalls of TLS 1.3 with regard to CIP Security, and recommends mitigating actions for the potential issues identified in TLS 1.3.

Keywords

CIP Security, Cybersecurity, Cryptography, TLS

Definition of terms (optional)

Acronym/Term	Description
3DES	Triple DES: a legacy encryption algorithm that applies the DES algorithm to each data block three times. See also: DES.
AEAD	Authenticated Encryption with Associated Data: Authenticated Encryption refers to a mechanism where the encryption of data and the verification of the authenticity of that data are combined, providing both confidentiality and authenticity via one algorithm. Associated Data refers to the ability to add extra data which is not encrypted but is still verified for authenticity, allowing for a portion of the message to be encrypted, and another portion to be not encrypted but still have guaranteed cryptographic authenticity. See also: AES GCM for an example of an algorithm that implements this.

AES	Advanced Encryption Standard: a symmetric encryption algorithm that is in wide use today and endorsed by several standards organizations. AES is a block cipher, and as such there are several modes of performing AES, which allow for various trade-offs.
AES GCM	AES Galois Counter Mode: A mode of performing AES which provides AEAD. That is, data is encrypted and authenticated, and there is the potential for additional authenticated data that is not encrypted.
CA	Certificate Authority: the service that signs certificates to vouch for their validity. This includes keeping the private keys used to sign the certificates private, as well as publishing the public keys used to verify the certificate
ChaCha20	A stream cipher developed by cryptographer Daniel J. Bernstein. This cipher is significant because it is included in one of the few cipher suites initially defined for TLS 1.3. See also: Cipher Suite, TLS
Certificate	Also known as a “Digital Certificate”, this is a piece of data signed by a Certificate Authority that is associated with a public/private keypair. The certificate can be used to prove the identity of a given party, and is often used for authentication within a secure connection, such as a TLS session. See also: TLS
Cipher Suite	A specification for protection of data in communication. Cipher suites for TLS specify the endpoint authentication mechanism, key agreement mechanism, and subsequent data encryption and data authentication mechanisms. See also: TLS
ClientHello	Initial message sent in a TLS connection to begin the TLS handshake. See also: TLS
DES	Data Encryption Standard: a (now legacy) symmetric encryption mechanism developed in the 1970’s. Although currently considered insecure, it paved the way for the development of the modern AES symmetric encryption mechanism. See also: AES
DHE	Diffie Hellman Exchange: A mechanism for key agreement that relies on modular multiplication. This was first published by two cryptographers, Whitman Diffie and Martin Hellman. Variations of this key exchange are still used today, most notably when combined with Elliptic Curve Cryptography. See also: ECC
DoS	Denial of Service: a class of attack where a resource is made unavailable by an attacker. This often, though not always, is performed via some sort of resource exhaustion attack. A very simple DoS attack example would be to continually send TCP handshake requests to a server but never complete the request. This will, in many cases, cause the server to reserve resources for the new TCP connection, eventually using up a significant amount of the resources on the server and likely affecting other functionality.
DSA	Digital Signature Algorithm: A variant of the ElGamal signature scheme that allows for digital signing of data. It relies on a public-private key pair for signing and verification.

Early Data	Initial data sent in a TLS 1.3 connection before the full key agreement has taken place. This data is protected by a Pre-Shared Key. See also: PSK.
ECC	Elliptic Curve Cryptography: a cryptographic scheme that relies on mathematics performed on an elliptic curve. ECC can be used for key agreement, digital signatures and data encryption/decryption (when coupled with a symmetric encryption algorithm).
ECDSA	Elliptic Curve Digital Signature Algorithm: a method for using ECC to generate and verify digital signature. See also: ECC
HMAC	Hashed Message Authentication Code: a MAC that is generated using a cryptographic hash. See also: MAC.
HTTP	Hyper Text Transfer Protocol: a protocol that is commonly used to transmit web data over the Internet. This protocol is ubiquitous in modern Internet communications.
IANA	Internet Assigned Number Authority: an international organization that is responsible for assigning various numbers to items related to Internet standardization, including things like RFC number, port number assignments, and others. See also, RFC.
IESG	Internet Engineering Steering Group: a group that is responsible for technical management of IETF activities and the Internet standards process. See also: IETF
IETF	Internet Engineering Task Force: the most widely recognized, participated in, and used Internet standards body which develops open standards through open processes.
MAC	Message Authentication Code: a piece of data that can be used to verify the authenticity of a message. MACs are used extensively in secure communications to ensure the cryptographic authenticity of the data being transmitted.
MD5	Message Digest 5: a legacy cryptographic hashing algorithm that was popular in the 90's but has since been shown to be forgeable.
PFS	Perfect Forward Secrecy: a property of secure communications that allows for the assurance that the compromise of one or both communicating entities private keys will not automatically result in the compromise of the session key, and subsequently all of the data exchanged during that session.
Poly1305	A MAC developed by cryptographer Daniel J. Bernstein. This cipher is significant because it is included in one of the few cipher suites initially defined for TLS 1.3. See also: MAC
PSK	Pre-Shared Key: a value which can be used to bootstrap authentication and key agreement within a TLS session. This value is symmetric between the two communicating parties (originator and target) and must be securely shared before it is used. It might be shared through a separate secure TLS session, or possibly through an out-of-band mechanism.
RC4	Rivest Code 4: A symmetric stream cipher popular in the 1990's for encryption of data. It has since been shown to contain security flaws and therefore is no longer recommended for use.
RFC	Request For Comment: the de-facto Internet standards documents produced and managed by the IETF. Not all of these documents serve as normative standards, but many of

	the technologies that define how the Internet works are specified through IETF RFCs. See also: IETF.
RSA	Rivest Shamir and Adleman: an asymmetric cryptosystem that relies on the hardness of the factoring large numbers problem for its cryptographic properties. This cryptosystem is in wide use today and can be used for digital signatures, encryption/decryption, and key agreement.
SHA	Secure Hash Algorithm: a family of cryptographic hash functions that are widely used and endorsed by the US National Institute of Standards and Technology (NIST). Although use of the SHA-1 version of the SHA family has been deprecated due to insecurity, the SHA-2 family and SHA-3 family are still widely considered to be robust and secure cryptographic hash algorithms.
TLS	Transport Layer Security: the most ubiquitous secure communications protocol in use today. This protocol is defined by the IETF in a series of RFCs. Although TLS 1.2 is the most widely used, TLS 1.3 has recently been published and is beginning to gain traction and adoption. See also: IETF, RFC.
Vendor Certificate	A certificate issued to a device by the device vendor. Within CIP Security, this certificate serves as the default certificate that can be used to bootstrap a secure connection for the purpose of provisioning the device with CIP Security configuration which includes either a new certificate or a Pre-Shared Key. See also: PSK

Introduction

TLS 1.3 was approved by the IESG in March of 2018. This approval introduced a new version of TLS, with some important and noticeable changes over the previous version. Note that TLS 1.3 is an evolution of TLS 1.2 in that it still builds on the same basic structures and functions that were put in place in TLS 1.2. Improvements in TLS 1.3 focus on:

- Improving privacy considerations by enforcing perfect forward secrecy
- Separation of authentication and key management functions and simplification of cryptographic functions to ease security analysis and review
- Performance improvements by optimizing the time needed to complete the handshake

At its core, TLS 1.3 retains its goals with the same basic structures and capabilities remaining. TLS 1.3 still provides authenticated and confidential communication at the transport layer, providing security for communicating over an untrusted network.

The TLS 1.3 RFC points out several notable changes from TLS 1.2. An abbreviated list of these changes follows:

- Removal of legacy symmetric cryptography algorithms
- 0-Round Trip Time handshake mode
- Removal of static RSA and Diffe-Hellman cipher suites
- Encryption of handshake messages after ServerHello
- Re-design of key derivation
- Restructuring of the handshake state machine
- Inclusion of ECC in the base specification
- Removal of compression, DSA, and custom DHE groups

- Deprecation of TLS 1.2 style version negotiation
- Replacement of previous session resumption with a simplified PSK exchange
- Updated references within the RFC

Notable impacts of these changes to CIP Security will be discussed within this paper.

CIP Security is heavily tied to TLS and DTLS, and as such support for the latest version of TLS is particularly important. As TLS evolves, the CIP Security specification must necessarily follow. It is unlikely that support for TLS 1.2 will evaporate in the short term, although from a long-term view TLS 1.3 will come into prominence as TLS 1.2 moves to the background. In order for CIP Security to maximally benefit from TLS it will need to eventually support TLS 1.3. The benefits derived are essentially the same benefits that originally motivated the usage of TLS 1.2. The widespread usage and testing of TLS provides a high level of confidence in the security and robustness of this technology. As issues in TLS are discovered and fixed CIP Security is improved. However, TLS 1.2 will receive less and less testing and updating as TLS 1.3 gains wider deployment. Furthermore, commercial and open source libraries will also move to support TLS 1.3 and provide reduced support for TLS 1.2. The timeframe for supporting TLS 1.3 is not immediate, although it is important to investigate. As a rough timeframe, it is important that Volume 8 of the CIP specification is enhanced to support TLS 1.3 within three to six years. This paper will provide an investigation of TLS 1.3 and potential impacts to CIP Security.

Handshake Changes

Most of the major changes from TLS 1.2 to TLS 1.3 are within the handshake. One major change in TLS 1.3 is the decoupling of the key establishment and authentication during the TLS handshake. In the initial `ClientHello` message, the client indicates a list of (EC)DHE groups, AEAD algorithms, and signature algorithms. The server then chooses one from each of these lists; any combination offered can be supported. This allows for a greater amount of flexibility in algorithms for authentication and key establishment. However, in general this will not have a major impact to CIP Security, as the authentication and key establishment occur largely transparently to the CIP and EtherNet/IP layer.

Another change for TLS 1.3 is that renegotiation is no longer supported. This was an optional feature within CIP Security to deal with 64 bit sequence number rollover, although the renegotiation could also be triggered at the discretion of the user. As this feature was optional the removal of renegotiation from TLS 1.3 is unlikely to have a large impact on CIP Security, although it could mean minor changes are needed in implementations wishing to support TLS 1.3. For TLS 1.3, this feature should be deprecated within CIP Security as it is no longer supported. However, note that TLS 1.3 does support the idea of a `KeyUpdate` message. This allows for new keys to be used for protection of the transmitted data. The `KeyUpdate` message is protected with the previously used session keys, so there is no need to go through the entire cryptographic handshake again. When CIP Security implements support for TLS 1.3 this `KeyUpdate` mechanism should be used to ensure that a given set of cryptographic keys does not protect more data than it is capable, which includes managing the sequence count rollover. Devices must support this mechanism, although its usage can be configured by the user.

Possibly the largest and most impactful change to the TLS handshake has to do with support for a zero round trip time (0-RTT) handshake. Although this is detailed in the TLS 1.3 RFC, as a brief overview, the server and client can establish a PSK and cryptographic parameters that can be used when re-connecting. Besides making use of a PSK, the client also encrypts application data using the PSK within the handshake; this data is referred to in the RFC as “Early Data”. The presence of this Early Data is what allows application data to be sent without any communication round trips within the handshake; hence the name 0 Round Trip Time. Of course the re-connection must be within a certain time limit, and is subject to certain risks. However, the benefit is that connections occur significantly quicker than with the standard TLS handshake. The diagram below shows the 0-RTT handshake. Note the presence of application data from the very first handshake message:

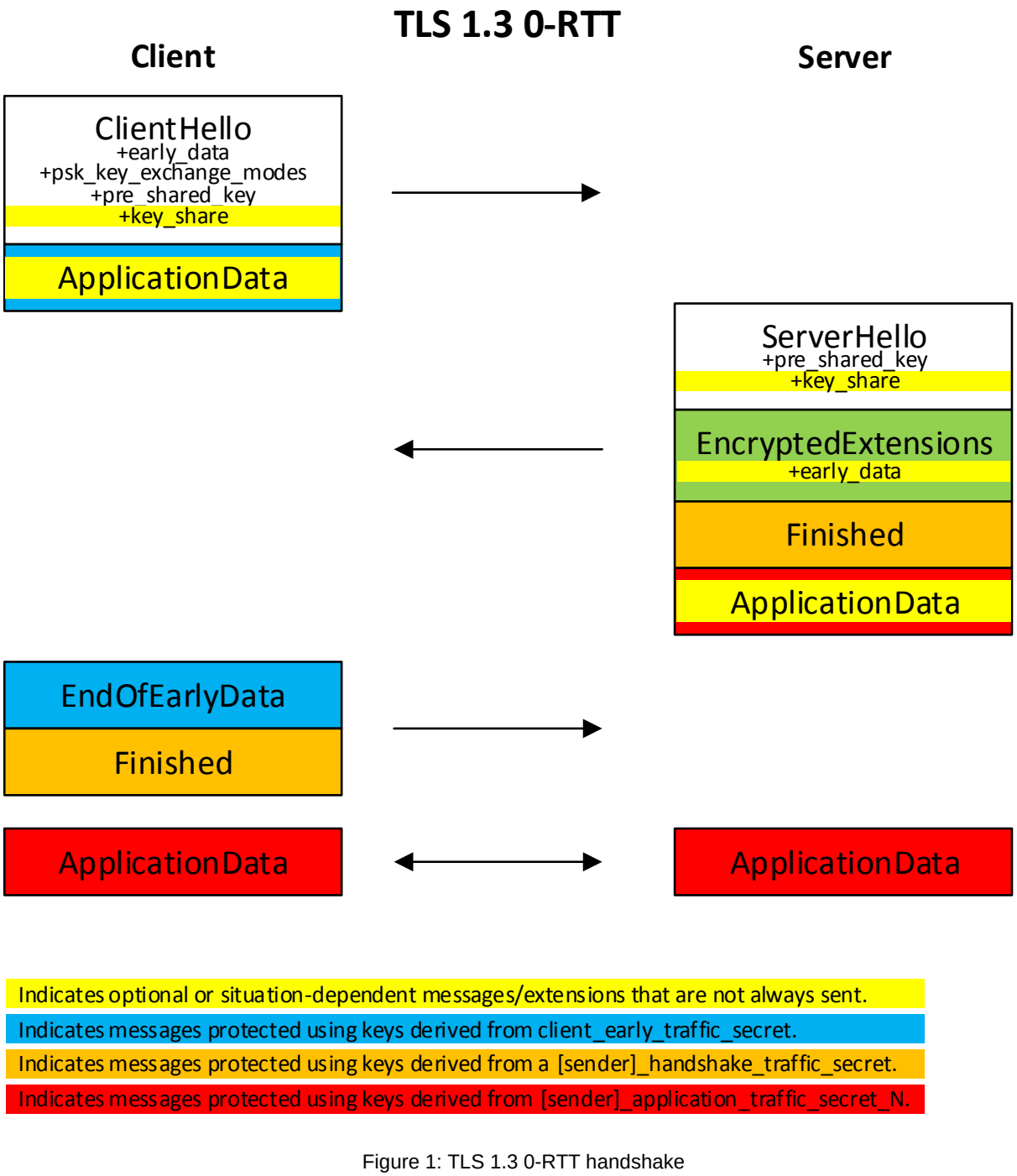

Figure 1: TLS 1.3 0-RTT handshake

Contrast the above figure with the existing TLS 1.2 handshake:

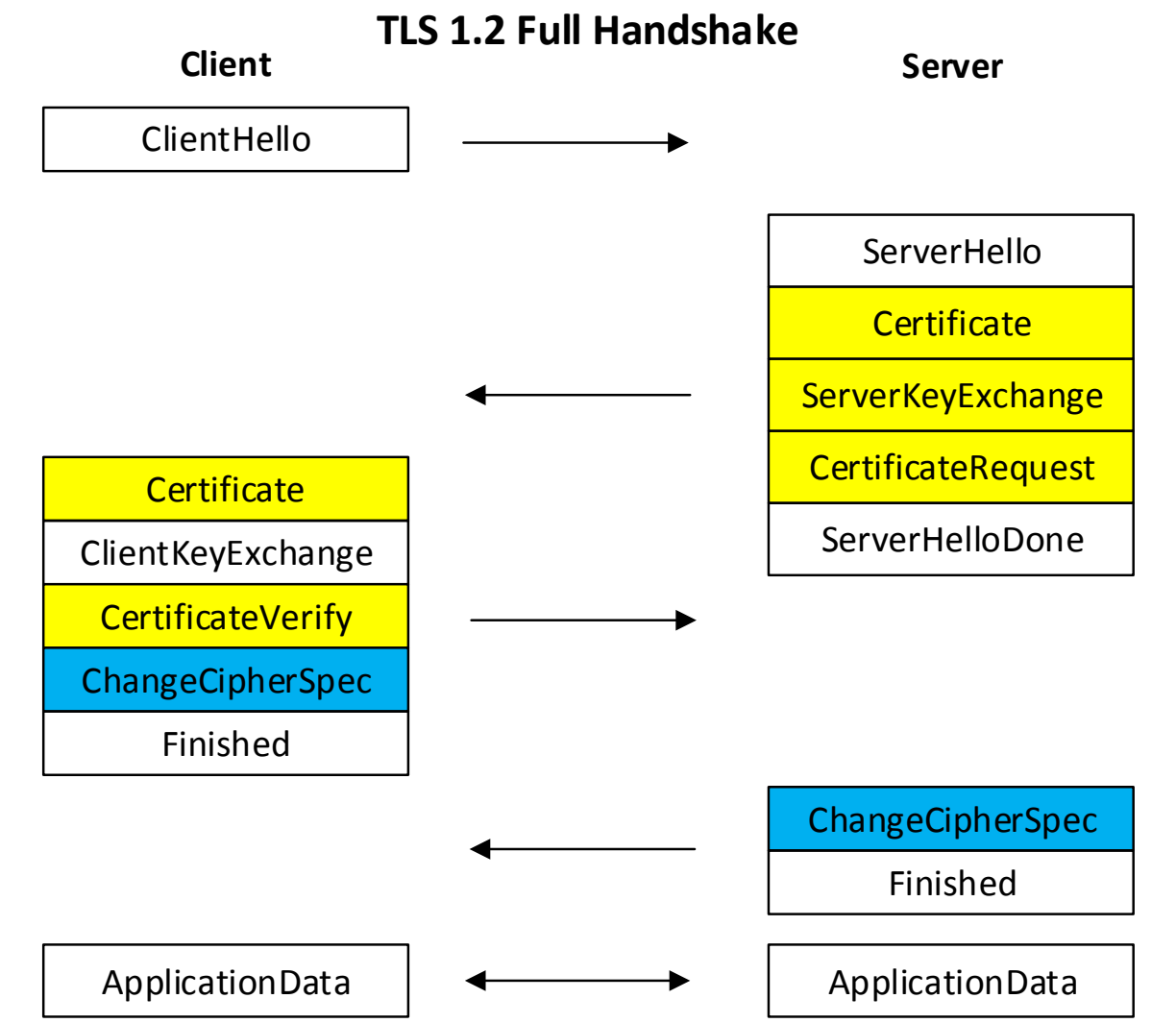

Figure 2: TLS 1.2 handshake

The main risk around 0-RTT is that of replaying Early Data. That is, because application data arrives before full endpoint authentication has occurred there is a potential that the data is being re-played by an attacker. The target will act on this data; if the data changes state then an attacker could affect the system in an undesirable way. For HTTP connections using TLS 1.3 there are mechanisms to mitigate the impact of this (for example, ensuring that the early data is idempotent). With CIP Security, it is not as straightforward to provide these guarantees. With a CIP application, not many assumptions can be made as to what type of messages are sent. The first message sent could easily be a delete service, or a reset service, something that might have significant effect on the target that could be exploited by a malicious actor. At the same time, there are plenty of actions that could be taken which would not have this type of effect; get attributes, or listen-only connections for example. Therefore, this is a useful feature to support for CIP Security, although it should be disabled by default and only used if configured to do so, with the recommendation that its use is limited to communications that do not change device state. There are likely several details that would need to be worked out for this to be supported. One potential option would be for the originator to set a PSK on the target to support 0-RTT. The EtherNet/IP Security Object

already supports a PSK attribute; this could be extended to be used for PSKs that support a 0-RTT handshake. This would allow the originator to opt-in to a 0-RTT, presumably only doing this if the risk of early data replay was acceptable within the given application.

TLS 1.3 has extensions for negotiated groups and key shares. Interoperability testing events with TLS 1.2 have shown that using the TLS 1.2 elliptic curve extensions to be useful. Although this was an optional extension some libraries expected it when ECC was used. Therefore, supporting these extensions in TLS 1.3 could further help interoperability and help to diagnose and debug issues when they arise.

Cipher Suites

With the decoupling of authentication and tunnel key establishment as well as the promotion of confidentiality and integrity, only Authenticated Encryption with Associated Data (AEAD) cipher suites are specified within TLS 1.3.
In particular, TLS 1.3 only supports newer cipher suites that are combined as follows:

Encryption Cipher	HMAC for key derivation	Notes
AES-128 GCM	SHA-256	Mandatory
AES-256 GCM	SHA-384	Optional
CHACHA20 Poly1305	SHA-256	Optional
AES-128 CCM	SHA-256	Optional
AES-128-8	SHA-256	Optional

Table 1 TLS 1.3 Cipher Suites

The following cryptographic primitives are no longer supported:

- SHA-1 Hash Function
- RC4 Steam Cipher
- DES
- 3DES
- AES-CBC
- MD5 Algorithm
- Various Diffie-Hellman groups
- EXPORT-strength ciphers
- RSA Key Transport

In specifying AEAD only cipher suites, the TLS Record Layer structure and workflow (for how to protect TLS data and metadata) becomes more straightforward and improves on the ability to analyze its security. This provides an indirect benefit of further assurance and confidence in the ability of cryptographic experts to analyze the protocol for potential weaknesses and implement improvements.

Cipher Suite Negotiation

Cipher suites in TLS 1.2 were backwards compatible with cipher suites defined in earlier versions. TLS 1.2 defines 37 cipher suites, adding the cipher suites from previous versions this sum up to 319 cipher suites in total. The cipher suite is basically a complete set of cryptographic algorithms needed for the secure network connection through TLS; these deal with key derivation and endpoint authentication, as well as the protection of the data in transit. The cipher suites follow a naming convention representing the algorithms comprising it. For TLS 1.2 the naming convention in Figure 3 is used.

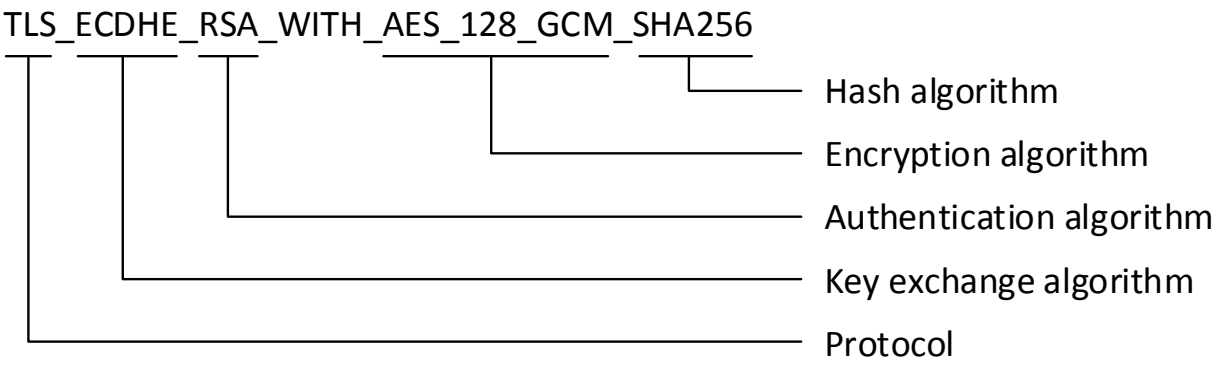

Figure 3 TLS 1.2 cipher suite naming convention

TLS 1.3 does eliminate all of the cipher suites defined and available under TLS 1.2 and earlier. Instead a set of just five new cipher suites have been defined (shown in Table *1*). The naming convention of the cipher suites has also changed, the new naming convention is shown in Figure 4. The rationale behind this is that TLS 1.3 only defines two authentication methods, pre-shared keys and certificate based, and the authentication method is chosen via a TLS extension. In a similar fashion as the authentication method the key exchange method is also negotiated via TLS extension.

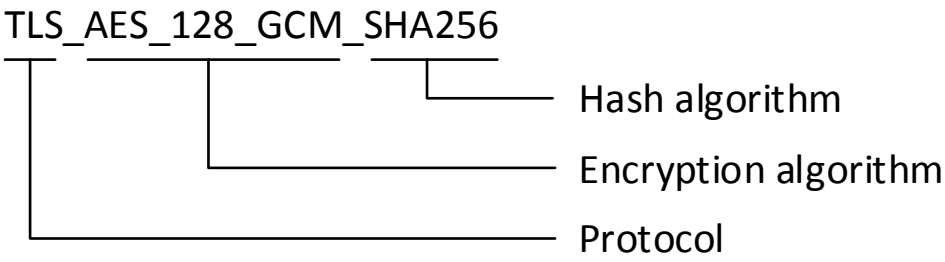

Figure 4 TLS 1.3 cipher suite naming convention

Since both the authentication method and the key exchange method are chosen via TLS extension instead of the cipher suite the number of cipher suites for TLS has been reduced to the following:

- 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 cipher suites are registered and maintained by IANA in the TLS Cipher Suites Registry. In this registry each cipher suite is given a unique number used for identification. Even if TLS 1.2 and TLS 1.3 used the same cipher suite number space cipher suites defined for TLS 1.2 cannot be used with TLS 1.3. Likewise cipher suites defined for TLS 1.3 cannot be used in TLS 1.2 and lower.

The encryption algorithm defined as a part of the TLS 1.3 cipher suite naming convention must be an Authenticated Encryption with Additional Data (AEAD) algorithm. In many cryptographic applications it's preferable to have both confidentiality and message authentication. Confidentiality is the cryptographic function that ensures that the data is only available to the parties who are authorized to obtain it, normally this is realized using encryption. Message authentication on the other hand is the primitive that ensures that the data hasn't been altered or forged by any untrusted parties, this is normally archived using a Message Authentication Code (MAC). The confidentiality and message authentication services are

usually used together and in many cases the two algorithms use independent keys. With AEAD both of those security services use a single crypto algorithm. In this concept the encryption and integrity functions have been replaced by the AEAD algorithm.

With the reduction of supported cipher suites and introduction of AEAD in TLS 1.3 it is no longer possible to send unencrypted data, which used to be an option with TLS 1.2. In TLS 1.2 is was possible to send unencrypted data by using one of the NULL encryption cipher suites. Removing the option of sending unencrypted data was in general done for good reasons since it reduce the chances of misconfiguration and thus unintentionally exposing confidential data. One of the main goals of TLS 1.3 was to reduce the large number of configuration options provided in TLS 1.3, simplifying configuration for the most common use cases. However in the case of resource constrained devices requiring encryption may be an expensive and burdensome feature. Furthermore this is even more of an issue for industrial control devices implementing CIP Security, as in this case the control data is sent at a high rate. During the development of TLS 1.3 the question of dropping support to send unencrypted data has been raised a couple of times. One of the reasons this was questioned was to make it possible to capture data and see what is sent on the wire, and by doing this ease the debugging while tracking down issues in installations. Another request to allow sending unencrypted data came by similar reasons as for resource constrained devices and sending data at a high rate. The summary of the discussions on the IETF TLS 1.3 mail archive is that the TLS 1.3 working group felt that supporting unencrypted data wasn't necessary, due to the perception that this use case is rare and that it opens up the possibility for misconfiguration. Due to the bias for simplification (and subsequent worry that complexity leads to security issues the group felt that removing as much complexity as possible was generally beneficial, and thus did not proceed with including cipher suites that support data authentication without encryption.

Since many of the devices implementing CIP Security are low end devices without a lot of processing power it is often desirable to use the authentication only cipher suites sending unencrypted data, thus reducing the packet processing required. In this case the end-points will still be authenticated, meaning that just trusted devices can communicate, and that the integrity of all data sent will be guaranteed. In most cases this is more than good enough since it will prevent rogue devices from connecting and altering the data sent. The fact that the control data can be seen isn't generally an issue since it usually doesn't carry any secrets. If CIP Security is to adopt TLS 1.3 this option won't exist anymore since all data must be encrypted, which will put a lot more burden on the devices and likely increase the bill of material cost since more expensive CPUs and memories will be required to achieve acceptable performance levels. A draft RFC that adds the option to send unencrypted but authenticated data has been put together by ODVA member companies to address the issue. The internet draft proposes new cipher suites, following the TLS 1.3 naming convention, that only provide for Authentication and Integrity.

At the time of this writing, ODVA is also pursuing the ability to affect integrity only at the TLS Record Layer while maintaining all the other properties, with the exception of obviating confidentiality. The specification is detailed in the draft RFC titled **TLS 1.3 Authentication and Integrity Only Ciphersuites** [3]. This draft RFC has gone through initial discussion with the IETF TLS Working Group. Through this discussion the authors received much positive feedback as well as some constructive suggestions. As of this time work is taking place to incorporate the suggestions before the draft is brought to IANA for assignment of an RFC number. The goal of this effort is to publish this draft under the "Informational" category, which will make these cipher suites and associated changes an optional part of TLS 1.3. In this case it would be up to ODVA members (and other vendors in the broader IoT space who find this functionality useful) to put pressure on TLS library vendors to include support for this optional TLS 1.3 functionality.

One of the cipher suites suggested by TLS 1.3 uses the ChaCha20 stream cipher and the Poly1305 authenticator. For the AEAD algorithms used in TLS 1.3 the two are combined into something called `CHACHA20-POLY1305`. Using the ChaCha20 instead of Advanced Encryption Standard (AES), which is considered as the industry standard, may be beneficial for devices implementing CIP Security. For software implementations ChaCha20 is generally significantly faster than AES, and in the case that the device hardware doesn't support AES acceleration the processor requirements for support of ChaCha20

are fairly lightweight. This of course assumes that CIP Security adopts TLS 1.3 and defines `TLS_CHACHA20_POLY1305_SHA256` as one of the required cipher suites.

Authentication and Key Establishment

Beyond the culling of cipher suites for protecting the TLS data, TLS 1.3 has also removed the use of static RSA and ECC keys to provide Perfect Forward Secrecy (PFS).

Perfect Forward Secrecy is a functionality that is part of the key exchange protocols, and it gives assurances that the session keys will not be compromised even if the private key of the server is compromised. This is accomplished via generating a unique session key for every session initiated using a non-deterministic algorithm. Perfect Forward Secrecy also protects against compromise of a single session key ensuring that the loss won't affect any data other than that exchanged in the specific session. Perfect Forward Secrecy is not functionality that comes at no cost. It will make it significantly more difficult to capture and decrypt data for debugging purposes, as well as putting higher processing burden on the end nodes.

The removal of static keys in TLS 1.3 impacts the CIP Security specification. While RSA or ECC based certificates can still be used for authentication, the key derivation for the initial session establishment requires the use of an ephemeral Diffie-Hellman exchange. More specifically, TLS 1.3 only supports three basic key exchange modes:

- (EC)DHE : Diffie-Hellman over finite fields or elliptic curves
- Pre-shared Key (PSK) only
- PSK with EC(DHE)

A "`key_share`" is now used to negotiate the ephemeral parameters for the session key. Alternatively, the PSK identity is provided if a pre-shared key was externally pre-provisioned.

Eventually CIP Security will adopt TLS 1.3, and as a part of this effort RSA keys needs to be removed for key established in the case when using TLS 1.3. At the same time TLS 1.2 likely needs to be supported for backwards compatibility. For an easier migration path it is recommended for device vendors to use ECC keys within the Vendor Certificate. The CIP Security specification today already discusses the option of creating the Vendor Certificates using RSA and ECC keys and already today recommends using ECC keys for performance reasons.

The authentication between the TLS Client and TLS Server is still expected to occur using certificate based mechanisms. The certificate validation is achieved through the `CertificateVerify` message which also provides integrity of the handshake messages (to the `CertificateVerify` point). The mandatory signature algorithms are:

- `RSA_PKCS1_SHA256`: for certificates
- `RSA_PSS_RSAE_SHA256`: for certificates and `CertificateVerify` message
- `ECDSA_SECP256r1_SHA256`: for certificates and `CertificateVerify` message

Note that within TLS 1.3 the server can still request for authentication of the client via certificate verification. This functionality is featured prominently in CIP Security, although is not as widely used in the Internet use case.

Extensions

Within TLS 1.3 there are several mandatory extensions. Although these must be implemented by TLS 1.3 libraries, they do not necessarily need to have support for configuration within the CIP object model. The following gives a brief discussion of these extensions and some commentary on what support, if any, is

likely to be helpful within the CIP object model. TLS 1.3 relies on and requires the following extensions be mandatory to implement:

- **Supported Versions**: as TLS 1.3 is not backwards compatible with TLS 1.2 (and older versions, although CIP Security does not support any version before TLS 1.2), new rules and assignments for legacy support are specified. More importantly, TLS 1.3 provides this extension and its designated logic to address interoperability with older versions and address potential version negotiation attacks. It is likely that within CIP Security TLS 1.2 and TLS 1.3 will both need to be supported for an extended period of time. However, it might be the case that a given user wishes to disable support for a given version of TLS within their CIP Security endpoints. Assuming this is true it is important that this extension can be accessed and connection decisions made based on this value. A simple attribute to denote what version of TLS the user wishes to allow is likely sufficient.

- **Cookie**: provides a measure of Denial of Service (DoS) protection by forcing the client to demonstrate reachability at their network as well as allowing the TLS server to be stateless. The TLS server may present the TLS Client a "cookie" that the client must then provide in its subsequent `ClientHello`. Providing some of the DoS protections for CIP Security endpoints could be useful. Users with concerns around this may want to optionally enable the use of cookies in the retry of the Hello message. As this is an optional feature it should be something that the user chooses to opt-in to, rather than being mandated. It is however important that all CIP Security endpoints that support TLS 1.3 also support this cookie extension. Users could enable its use via setting a new attribute in the EtherNet/IP Security Object.

- **Signature Algorithms:** specifies the algorithms to be used to verify the certificate (and possibly the `CertificateVerify` message). This is closely related to the "Allowed Cipher Suites" attribute of the EtherNet/IP Security Object. This attribute could likely be extended to allow/disallow certain signature algorithms from being used.

- **Negotiated Groups:** enables the TLS client to specify the supported groups to use for key establishment. Again, although this is not the same as a cipher suite it follows closely the idea of Allowed Cipher Suites, and therefore there may be enhancements possible for that attribute to more closely support this extension.

- **Server Name Indication (SNI):** in TLS 1.3 this extension must be provided in the resumption handshake. This is a somewhat interesting extension, as most CIP Security endpoints are addressable on TCP/IP only via IP address. Without the presence of the IP address within the certificate there are situations in which differentiating between more than one trusted endpoint could prove difficult. Therefore it could be useful to include support for this at some point, although again this would likely be an optional feature that the user would need to configure. Alternatively, CIP identity information could be placed within an X.509 certificate used for authentication in a TLS handshake. Either the certificate mechanism or the extension mechanism would provide the same level of risk mitigation in these scenarios. Despite this, it is likely worthwhile to consider adding explicit support for configuring the SNI within the CIP object model.

- **Certificate Authorities:** this allows the client to specify the certificate authorities that it will trust for the purpose of authentication within the handshake. This extension is potentially useful for the target to know whether or not the handshake will be successful. Furthermore, this could open up a possibility for a target to support multiple certificate-based identities. The target could use the certificate/identity that is signed by the certificate authority sent within this extension. Currently multiple certificate based identities is not supported within CIP Security, although this could be useful to allow for more granular control of security configuration within CIP Security.

Conclusions and Recommendations

As discussed, TLS 1.3 brings a host of new features and benefits. Assuming that efforts to include NULL encryption are successful, then it is highly recommended to include support for TLS 1.3 in CIP Security within the next few years. Even if these efforts are not successful, it is still recommended that support be included, although the timeframe for support could be longer in this case. It is expected that TLS 1.2 will need to be supported for several years, so CIP Security will need to include support in such a way as to allow for TLS 1.2 and TLS 1.3 to both coexist, potentially allowing the user to choose if only one version is to be used within a given endpoint. With support for NULL encryption from the IETF, it is recommended that the CIP Specification be enhanced to support TLS 1.3 within the next three years. Without NULL encryption support, the timeframe for including TLS 1.3 should be extended to six years, allowing hardware designs more time to catch up with the mandatory encryption.

References

[1] ODVA, Inc. The CIP Networks Library, Volume 8: CIP Security™, PUB00299

[2] TLS 1.3 RFC https://datatracker.ietf.org/doc/rfc8446/

[3] TLS 1.3 Authentication and Integrity only Ciphersuites RFC Draft https://tools.ietf.org/html/draft-camwinget-tls-ts13-macciphersuites-00

[4] TLS 1.2 RFC https://www.ietf.org/rfc/rfc5246.txt