FIX-over-TLS (FIXS)
Technical Specification

Version 1.0 – Technical Standard – February 12, 2021

THIS DOCUMENT IS THE FINAL VERSION OF A FIX TECHNICAL STANDARD. THIS VERSION HAS BEEN APPROVED BY THE GLOBAL TECHNICAL COMMITTEE AS THE FINAL STEP IN CREATING A NEW FIX TECHNICAL STANDARD OR A NEW VERSION OF AN EXISTING FIX TECHNICAL STANDARD. POTENTIAL ADOPTERS ARE STRONGLY ENCOURAGED TO USE ONLY THE FINAL VERSION. EXISTING ADOPTERS ARE STRONGLY ENCOURAGED TO UPGRADE TO THE FINAL VERSION.

DISCLAIMER

Document History

Revision	Date	Author	Revision comments
Release Candidate 1	Aug. 18, 2017	FIX Global Technical Committee	Published for public review on the FIX Trading Community website.
Draft Standard	Dec. 21, 2017	FIX Global Technical Committee	Promoted from Release Candidate 1 without changes.
	Jan. 2, 2018	Charles Kilkenny, Don Mendelson	Minor corrections
Technical Standard	Feb. 12, 2021	FIX Global Technical Committee	Change from Draft Standard to Technical Standard. Update of TLS references. Change of FIXUA to FIXA.

Table of Contents

1 Introduction

FIX-over-TLS (FIXS) is a technical standard that specifies how to use the Transport Layer Security (TLS) protocol with FIX. It provides some standardisation and ensures a minimum level of security is applied. We believe FIXS will make it easier for FIX participants to employ TLS, and hope that this will help to improve security across the industry.

TLS is a rich protocol with many features and options. The protocol, for example, allows for new security functions to be added and vulnerable functions to be dropped. Additionally, information security is wide and varied. Understanding the TLS protocol features and options is complex and time consuming, and incorrect configuration or management of TLS can result in insecure linkage or no security at all. The FIXS standard therefore aims to make employing TLS simpler, and further provides guidance and best practice that is valid at the time of writing.

FIXS is primary focused on how to use TLS reliably with a minimum level of standardisation across the FIX community. The standard first concentrates on possible methods to authenticate the parties connecting to one another. It then goes into the different aspects of each authentication method as well as the different protocol options and what is recommended. This includes the different available cipher suites as well as certificate properties and validation.

FIXS optionally includes authentication of clients as part of the FIX session. This is termed using FIX User Authentication (FIXA) and it can be used to authenticate FIX clients at the FIX session level rather than authenticating clients at the TLS level.

FIXS does not prevent participants using additional security controls. FIXS defines a minimum set of requirements, which are needed for common use cases and interoperability. Participants may choose to use security controls beyond what is specified in FIXS for extra security or to address the latest vulnerabilities.

Security is only one aspect of using TLS. Another aspect is performance and a further consideration is compatibility with out-of-band monitoring solutions. We therefore try to balance security with the needs of performance and compatibility, in order to keep FIXS suitable for trading and other activities within banking and finance.

This is the first publication of FIXS and we are still learning. As always, we welcome your feedback and hope that you will share your views with us.

1.1 Scope

This guide is targeted at companies wanting to secure FIX sessions generally. This is whether FIX is being run over the public Internet, across other networks or internally within a firm.

We hope to address general concerns about performance through proof-of-concepts. FIXS is not yet intended to be a substitute for other security controls in ultra-low latency environments.

This guide considers TLS within the context of FIX and provides guidance independent of particular systems. We intend to provide a separate document, leading on from this one, for guidance on the open source Stunnel program. We also refer to some implementations in this guide when clarifying support for particular options. However, the aim of this document is not to go into the detail of TLS implementations.

This guide covers the use of TLS to protect application and session level communications exposed by FIX session protocols. This is the communication between FIX initiator and FIX acceptor over a TCP/IP socket. It does not consider the use of TLS or other security protocols such as IPSec for router-to-router or virtual private network (VPN) communications. It does not consider other protocols such as IPSec for application level security. Additionally, this guide does not address other FIX message transports such as Advanced Message Queuing Protocol (AMQP), IBM WebSphere MQ or multicast UDP.

A notable point about site-to-site VPNs between routers and or firewalls using IPSec and perhaps TLS/SSL is that connections can be two-way. That is to say either side can connect to the other (i.e. acting as client or server), and maybe both can be connected to the other at the same time (i.e. each side acting as both client and server simultaneously). In this guide, however, we are only concerned with a one-way connection from the FIX initiator to the FIX acceptor.

This document is not intended to be a complete guide to TLS or SSL. Instead, there are many publicly available resources on the Internet which provide in-depth information on TLS/SSL. We have provided links to a number of these resources at the end of this document.

Please note TLS may not be appropriate for time critical data streams. For example, TLS is not used to secure Real-time Transport Protocol (RTP) communications that are used for voice and media streaming. Instead, the Secure Real-time Transport Protocol (SRTP) is defined for that purpose. If one is looking to encrypt similar time critical data streams where completeness of the information is less important, TLS may be inappropriate.

We have a few FIX session protocols today. The session protocols include the FIX 4.0 and above FIX Session Protocol (FIX4), the FIX Session Transport (FIXT) Protocol and the FIX Performance Session Layer (FIXP) Protocol. All of these FIX session protocols are in scope for FIXS.

1.2 An overview of TLS

1.2.1 History of TLS/SSL

The TLS protocol has evolved over a number of years, in order to better protect against risks and vulnerabilities.

The current version of TLS is defined by the Internet Engineering Task Force (IETF).

The history can be summarised as follows:

- TLS v1.3 – Defined by RFC 8446 (August 2018); out of scope for this version of FIXS
- TLS v1.2 – Updated by RFC 5246 (August 2008)
- TLS v1.1 – Updated by RFC 4346 (April 2006)
- TLS v1.0 – Defined by RFC 2246 (January 1999)
- SSL v3 – Deprecated by RFC 7568 (June 2015) and vulnerable to POODLE, specified by RFC 6101 (August 2011)
- SSL v2 – Prohibited in TLS by RFC 6176 (March 2011),
- SSL – Originally from Netscape

1.2.2 Basics of the TLS protocol

The TLS protocol comprises two protocols: the *Handshake Protocol* and the *Record Protocol*.

The Handshake Protocol is used to authenticate the server and optionally the client using Public Key Infrastructure (PKI) technology or other techniques such as pre-shared keys (PSKs). The Handshake Protocol is also used to agree the parameters of the Record Protocol. If agreed, the Record Protocol follows.

The parameters of the Record Protocol include a shared secret and a data encryption algorithm, known as a cipher. The shared secret may be ephemeral, meaning it will not be the same for different sessions, resulting in Forward Secrecy. Forward Secrecy is the property of sessions where, if a key from a later session becomes compromised, it cannot be used to decrypt earlier sessions.

The Handshake Protocol at the start of TLS communication typically employs costly asymmetric cryptography, whilst the Record Protocol running for the majority of the communication employs more efficient symmetric cryptography for encryption of ongoing data streams. The Record Protocol additionally provides a reliable connection for the application level protocol (the FIX session protocol in our case) as if it were a TCP/IP socket.

1.3 Network topologies and the perspectives of FIX participants

FIX does not mandate a particular network topology. However, across the FIX community, we do see two network topologies being used generally. One is the ‘Star’ topology and the other is the ‘Peer-to-peer’ (P2P) topology. In the Star topology, also known as the centralised or many-to-one network topology, many nodes connect to a central node. This may be, for example, a number of buy-side organisations connecting to a sell-side organisation or many participants connecting to a hub. The hub may be, for example, an exchange, order routing network, trade reporting facility or a FIX messaging hub. The P2P topology on the other hand is the point-to-point or one-to-one network topology where matters are agreed bi-laterally. An example of P2P may be two organisations connecting to one another for trading, or it could be different applications connecting to one another within an organisation.

We believe that the central organisation in the Star topology generally dictates the FIX connection properties and that participants connect to it rather than this being the other way around.

We conclude that a buy-side typically participates in a limited number of Star FIX networks. This will result in the buy-side having to manage a limited number of outbound FIX connections, for which it does not dictate the connection properties.

A sell-side or service bureau organisation on the other hand is likely to provide a Star FIX network, and as a result it will have to manage potentially a significant number of inbound client connections. Additionally, the same organisation is likely to have to manage a limited number of other connections, either on a P2P basis or as a client to other Star FIX networks in order for it to participate in them.

A point worth noting is that the Star topology introduces an indirect messaging path or network between participants. This should be taken into consideration for information security, and additional security controls, such as end-to-end message security between participants, may be necessary.

1.4 When and where to use FIXS

FIXS can be used to add security to any FIX connection. It is recommended to use FIXS for any FIX session running over the public Internet which is not otherwise satisfactorily protected. It may also be prudent to use FIXS for FIX sessions running over a secure extranet between companies or within a company's own network. In certain circumstances, it may be necessary to balance the use of FIXS with performance considerations.

It is often the case that TLS or SSL is also present as part of underlying network communications. For example, TLS may be used for remote access or site-to-site VPNs. This should not detract from using FIXS where it provides a security advantage.

Ideally, TLS should be used directly from the client FIX engine all the way to the server FIX engine. In practice, though, this may not be feasible. For example, it may be necessary to use a TLS proxy on the server side within a DMZ decoupled from the server FIX engine. Also, it may be necessary to use a Man-in-The-Middle (MITM) proxy or use an out-of-band monitoring solution for monitoring and compliance. A MITM solution would terminate and reestablish TLS, enabling an organisation to have access to the session data and timings. Out-of-band monitoring, also known as passive monitoring, involves decrypting TLS sessions out-of-band, also giving the session data and timings but without interfering with the in-band communications.

1.5 References

We have collected a number of references to articles and work on the Internet, which provide useful reading for a better understanding of TLS and current issues. Please see the references at the end of this document.

2 Authentication Methods

TLS and to some extent FIX provide a variety of ways for the two parties connecting to one another to authenticate each other. For the purposes of this guide, however, we have narrowed down the possible options for authentication, and provided a choice between a limited set of recommended methods.

2.1 Recommended authentication (and key exchange) methods

This table shows the minimum recommended authentication methods against which participants in the Star and P2P topologies must choose. The Star and P2P topologies are described in the introduction to this document.

Id	Method	TLS protocol	Authentication of server	Authentication of client	Star topology	P2P topology
Using certificates (giving authentication of both ends at TLS level)						
1a)	Mutual TLS with Leaf Certificate Pinning both ends	Mutual	Leaf Certificate Pinning	Leaf Certificate Pinning	✓	✓
1b)	Mutual TLS with Leaf Certificate Pinning of client and Certificate Validation of server with CA Pinning	Mutual	Certificate Validation with CA Pinning	Leaf Certificate Pinning	✓	✗ Do not use
Using PSKs (giving authentication of both ends at TLS level)						
2	Mutual TLS with pre-shared keys (PSKs) both ends	Mutual	PSK	PSK	✗ Do not use	✓
Using FIX User Authentication (FIXA) and server certificates (giving authentication of server at TLS level and authentication of client at FIX session level)						
3a)	Simple TLS with Leaf Certificate Pinning of server, and FIXA for client	Simple	Leaf Certificate Pinning	FIXA	✓	✗ Do not use
3b)	Simple TLS with Certificate Validation of server with CA Pinning, and FIXA for client	Simple	Certificate Validation with CA Pinning	FIXA	✓	✗ Do not use

Please note the terminology used in the table above may be difficult to understand at first sight. Please bear with us. What the terminology means and what is involved are explained in more detail in the subsequent parts of this chapter.

It is required to use one of the methods above to authenticate both the FIX acceptor and the FIX initiator.

Users may choose to have additional controls beyond the options in the table. For example, two-factor or multi-factor authentication may be better. Multiple levels of authentication can also be derived by combining options from the table. For example, FIXA from option (3) could be used in conjunction with options (1a), (1b) or (2). This would provide authentication of the client at both the TLS transport level and at the FIX session level, each with its own method of authentication. We have not included this possibility though as it is not a minimum requirement. It is also possible to incorporate end-to-end message security, for example, using a signature and possible sequencing within the FIX message trailer, but again this is beyond the scope of our minimum set of options.

We have allowed each of these options to cater for the different needs between the Star and the P2P topology. For example, using pre-shared keys (PSKs) is useful in the P2P topology. The use of FIXA in options (3a) and (3b) also allows for better end-to-end authentication of the client user in the Star topology.

2.2 Mutual and Simple TLS protocol options

The TLS protocol allows the server and the client to authenticate each other. When both the client and the server authenticate each other, it is called 'Mutual TLS'. When only the client authenticates the server, it is called 'Simple TLS'. Thus, these two terms are commonly used:

- Mutual TLS – both server and client authenticated
- Simple TLS – only server authenticated

There can also be no authentication of either side (total anonymity), but this is not recommended. Authentication of just the client is also possible (by the client not bothering to actually authenticate the server in Mutual TLS), but again this is not recommended.

When Simple TLS is used, it is typically combined with authentication of the client by some other means. For example, this may be at the application protocol level using a username and password within the session. This is possible as a secure method, because: a) the client will have already authenticated the server; and b) the session will be encrypted as part of TLS. Thus, this method of authentication is used commonly across the Internet to gain access to web applications.

Our minimum recommendation for authentication is to use either Mutual TLS or to use Single TLS together with authentication of the client at the FIX session layer. We have termed this 'FIX User Authentication' (FIXA).

2.3 Leaf Certificate Pinning

One method by which a certificate can be checked that it is valid, and that the other party is genuine, is to match the certificate to an expected certificate. We call this '*Leaf Certificate Pinning*' in FIXS.

Initially, the party to be authenticated must have provided a copy of the certificate to the party performing the authentication. This should have been done securely by other means so that the authenticator knows it is a genuine copy of the certificate.

Then, at the start of communication, the party being authenticated provides its certificate (the leaf or the only certificate in its certificate chain) and signs something to prove that it has the private key for the certificate. The authenticator verifies the signature against the public key from the certificate and additionally checks that the certificate is in date. Then, if these are correct, the authenticator must check that the certificate actually matches what is expected, in order to validate the identity of the party and to be sure that it hasn't just received any certificate. This is done by comparing the newly provided certificate against the expected certificate which was provided previously and which is therefore trusted. The trusted certificate is also known as a trust anchor. Thus, the authenticator validates the certificate and that the other party is genuine by 'pinning' the leaf certificate.

Pinning can be done using a thumbprint i.e. a hashed id of the certificate rather than having a copy of the whole certificate. This is how browsers commonly pin root certificates in a public CA chain. Further, pinning can be done using just the public key itself from within the certificate or using the public key and its parameters from the certificate. There are advantages and disadvantages to each method (see [OWASP-CPKP]). However, we have standardised on pinning the certificate as a whole for FIXS Leaf Certificate Pinning. This is because it is currently the only method supported by Stunnel.

Thus, pin to certificates as a whole for FIXS Leaf Certificate Pinning.

A further point is that it is possible to pin to more than one certificate, known as a pinset. This allows one or more alternative certificates to be included for disaster recovery or multiple servers; and it supports migration to a new certificate before an existing certificate expires.

Thus, support and expect to use certificate pinsets for FIXS Leaf Certificate Pinning. However, do keep pinsets up-to-date, in order to ensure revoked certificates are not acceptable.

It is good practice to include all of the certificates in a certificate chain when providing an expected certificate.

Out-of-date certificates should not be used or accepted.

Do not use a pinset for different entities or functions i.e. for different purposes. For example, if Party A connects to parties B and C, try to have different pinsets for the parties. If the certificates for Party C are mixed with the certificates for Party B, it may be possible for Party B to act on behalf of Party C (and likewise for Party C to act on behalf of Party B), which could be a serious breach in security.

It is important to ensure FIX messages and their content are authorised for the party authenticated at the TLS level or using FIXA. It should not, for example, be possible for Party B to send messages on behalf of Party C unless it is authorised to do so.

2.4 Certificate Validation with CA Pinning

Rather than pin every leaf certificate, a certificate can be validated through its certificate chain. A CA certificate from the chain is pinned instead, establishing a trust anchor to that point, and, from this, it is possible to validate and trust potentially numerous leaf certificates. This is what we term '*Certificate*

Validation with CA Pinning' in FIXS. It is essentially the same as how public certificate validation works in web browsers, except CA certificates are not generally pre-pinned as in web browser or operating system software.

The leaf certificate has to be signed by a CA rather than being self-signed. As a result, the certificate can be verified against a certificate for the CA itself, proving that the CA issued the certificate and that the certificate is genuine.

It is not unusual though for the CA certificate itself to be signed by another higher-level CA or by another certificate of the CA itself. Thus, a chain of trust is commonly created and validation is often a recursive process traversing up the certificate chain.

Ultimately, one of the CA certificates within the certificate chain, up to the root CA certificate, will need to be pinned, in order to determine that the user certificate is actually valid.

CAs may revoke certificates. Thus, it is also necessary to check a certificate revocation list (CRL) from the CA which is up-to-date as part of verification. One method of doing this is using the Online Certificate Status Protocol (OCSP) if it is available for the CA. Alternatively, one must maintain an up-to-date (e.g. daily) copy of a CA's CRL for checking.

Certificate validation may appear to be straightforward in modern browsers. However, it is complicated and can be burdensome to implement between FIX engines. Thus, we do not recommend certificate validation for P2P, as Leaf Certificate Pinning or PSKs are likely to easier. That is unless you already have the facilities in place. We do still feel though that it is worthwhile to use Certificate Validation with CA Pinning for clients in a large-scale Star topology.

It is important to check that the certificate and the certificates in the certificate chain have not expired as part of certificate verification.

2.4.1 Certificate subject checking

For Certificate Validation with CA Pinning, often it is not enough to just verify a certificate generally i.e. that it has been issued, it has not been revoked, its signature is valid and that it is current. It is generally necessary to also check that the certificate has been issued to the correct organization, URL/server and or for a certain purpose.

In this case, the certificate subject must be matched against what is expected.

There may also be other certificate properties which need to be matched.

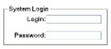
2.5 Pre-shared keys (PSKs)

As an alternative to using certificates, TLS allows pre-shared keys (PSKs) to be used for authentication. These are symmetric keys, also known as passphrases or passwords.

The advantage of PSKs, especially for P2P connections, is that they are convenient and may be simpler to manage than certificates. Provided the keys themselves can be exchanged securely, PSKs can be extremely effective in P2P situations so we have included using them as a possible method for P2P.

The downside of PSKs though is that their secret values need to be exchanged so there is a higher risk that they may become compromised. This is not the case with public and private keys being used in asymmetric cryptography.

2.6 FIX User Authentication (FIXA)



Another alternative is to authenticate the user at the FIX session layer, rather than at the TLS level. This is what we call FIX User Authentication (FIXA), and it is useful as it can provide better end-to-end security in a Star topology from the perspective of the central organisation.

FIXA involves the identification of a user and something to authenticate it, with all of these details being passed as part of the FIX session layer. Credentials may come from a username and password, or authentication could be achieved, for example, from public private key cryptography or from a passphrase alone.

We are only concerned about FIXA in relation to the FIX initiator. The server should always be authenticated at the TLS level. As a result, there is not much of a need to authenticate the FIX acceptor at the FIX session level, and we have not considered it as part of FIXS.

2.6.1 Recommended FIXA methods

The recommended methods will be provided separately as part of a FIXA specification.

Please note plaintext passwords (even within Base64 encoding) should not be captured within log files. FIX Logon(35=A) and FIXP Negotiate messages are typically logged today so behavior needs to be changed in order to redact the message data when needed.

2.6.2 How and where to specify FIXA details

Full details of how and where to specify FIXA details will be provided separately as part of a FIXA specification. This section provides our recommendations to date.

FIXA details if applicable should be added by the client FIX engine and validated by the server FIX engine. They are introduced in the FIX Logon(35=A) message or the FIXP Negotiate message as part of the FIX session protocol or FIXP respectively. There isn't therefore an opportunity for them to be supported between, for example, Order Management Systems (OMSs) unless the FIX engine is embedded within the OMS or an alternative protocol is used between the OMS and the FIX engine.

2.6.2.1 FIXT and earlier FIX session protocol versions

Different fields have been introduced in the FIX Logon message for authentication purposes across the progressive versions of FIX.

We initially had in FIX 4.0 the RawData (96) and RawDataLength(95) data fields together with the EncryptMethod(98) field. This provides an opaque data field which is supported across all FIX versions (excluding FIXP) today. The EncryptMethod(98) field should, however, always have a value of '0', meaning it is irrelevant.

The Username(553) and Password(554) fields were then introduced in FIX 4.3 allowing for plaintext passwords to be specified.

Finally, a number of additional fields were added in FIXT 1.1. These included the EncryptedPassword(1402) and EncryptedPasswordLen(1401) data field and the EncryptedPasswordMethod(1400) field for encrypted passwords. In addition, the NewPassword(925) field and the EncryptedNewPassword(1404) and EncryptedNewPasswordLen(1403) data fields were provided to allow users to change their passwords within the FIX session.

For FIXT and earlier versions of the FIX Session Protocol, the RawData(96) and RawDataLength(95) data fields should be used to carry FIXA details. The EncryptMethod(98) field should only be populated with a '0', and should be ignored. The other username and password fields should not be used, and the fields to change passwords should not be used.

FIXA details should only be specified in the Logon message from the FIX initiator to the FIX acceptor. Information may additionally be returned from the FIX acceptor to the FIX initiator by means of the Logout message and its Text(58) field.

There is no requirement to include FIXA details in the Logon message from the FIX acceptor.

The reason why we recommend using the RawData (96) and RawDataLength(95) data fields alone is that it is available in all the versions of the FIX session protocol and it is opaque, meaning it can hold any value. Credentials do not have to, for example, be limited to only a username and password.

The reason why we do not recommend using the change password fields is that: a) they are not opaque as they are in the context of a username; b) they add unnecessary complexity to the protocol; and c) we are dealing with machine-to-machine communication rather than human-to-machine communication. Thus, changes to credentials do not need to be interactive and are likely be carried out in practice by other means. Another reason also is: d) plaintext passwords should not be used as message text is normally captured within FIX engine log files.

2.6.2.2 FIXP

For FIXP, the FIXP Credentials field should be used to carry FIXA details.

FIXA details should only be specified in the Negotiate message from client to server. Information may additionally be returned from the server to the client by means of the FIXP NegotiationReject message and its Code and Reason fields.

There is no requirement to include FIXA details in the FIXP NegotiationResponse message from the server.

3 Protocol Parameters

This chapter provides our minimum recommended parameters for TLS. This includes which protocol versions and cipher suites to use; which features to support; as well as which parameters to use for certificates, PSKs and FIXA.

3.1 Protocol version

It is recommended to use the latest official version of TLS (currently 1.3 in February 2021), and to support both that version and the previous version if required to do so during transitional phases.

3.2 Protocol features

Session caching – Do support TLS session resumption, aka Session Caching. This will speed up the time it takes to reestablish connections.

No compression – Do not support TLS compression.

No re-negotiation – Do not support TLS re-negotiation.

3.3 Cipher suites

3.3.1 Recommended suites for TLS v1.2 certificate authentication

It is recommended to use the following cipher suite list when using certificates for authentication. This includes using certificates in Simple TLS in conjunction with FIXA. The list ensures Forward Secrecy, avoids deprecated ciphers and should achieve good performance. The cipher suites are specified in our order of preference, starting with the most preferred cipher suite.

```
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

TLS_DHE_RSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

TLS_DHE_RSA_WITH_AES_256_CBC_SHA
```

`TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`

`TLS_DHE_RSA_WITH_AES_128_CBC_SHA256`

`TLS_DHE_RSA_WITH_AES_256_CBC_SHA256`

`TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`

This list matches what is recommended as best practice from SSL Labs currently (November 2016), except we have given preference to performance of the TLS Record Protocol rather than to the performance of the TLS Handshake Protocol. The list should be used in conjunction with Session Caching.

The list details ECDSA certificate cipher suites and, then, RSA certificate ones. It is possible to support both an ECDSA certificate and a RSA certificate at an end point, but only one certificate can be used at a time. In practice, only one kind of certificate is needed. Thus, if you are using an RSA certificate, the ECDSA cipher suites will be ignored so they can be omitted. Likewise, the RSA cipher suites will be ignored and can be omitted for an ECDSA certificate.

3.3.2 Recommended suites for TLS PSK authentication

It is recommended to use the following cipher suite list when using PSKs for authentication. This is because the list ensures Forward Secrecy, avoids deprecated ciphers and should achieve good performance. The cipher suites are specified in our order of preference, starting with the most preferred cipher suite.

`TLS_DHE_PSK_WITH_AES_128_GCM_SHA256`

`TLS_DHE_PSK_WITH_AES_256_GCM_SHA384`

`TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA`

`TLS_DHE_PSK_WITH_AES_128_CBC_SHA`

`TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA`

`TLS_DHE_PSK_WITH_AES_256_CBC_SHA`

`TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256`

`TLS_DHE_PSK_WITH_AES_128_CBC_SHA256`

`TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384`

`TLS_DHE_PSK_WITH_AES_256_CBC_SHA384`

This list follows the same logic as in the list above for certificate authentication, except it uses the cipher suites available for PSK authentication.

3.4 Certificate parameters

Be prepared to accept X.509 version 1, version 2 or version 3 certificates; and only use version 3 or version 2 certificates when field usage requires that X.509 version to be used.

3.4.1 ECDSA certificates

More information will be provided hopefully in the future.

3.4.2 RSA certificates

Use 2,048-bit RSA keys. If stronger security is required, use ECDSA certificates.

More information will be provided hopefully in the future.

3.5 PSK properties

More information will be provided hopefully in the future.

3.6 Application specific TLS

Depending upon the application, TLS is commonly used on a separate TCP port to the default port used for unencrypted communication. For example, HTTPS uses TCP port 443 whilst HTTP uses port 80.

A number of application level protocols do however allow for TLS to be introduced as part of the application protocol on the same TCP port as unencrypted communication. This has a couple of advantages in that a separate port is not required and it can work with or without encryption. Examples of this are STARTTLS for LDAP and RFC 2817 for HTTP.

None of the FIX session protocols support this capability currently, and we do not plan to update them to do so.

4 Policies and Management

4.1 Sharing secrets

4.1.1 Secure distribution

The use of PSKs will require one or more PSKs to be distributed between organisations.

Similarly, a central organisation issuing certificates on behalf of its members (or another party on its behalf) will need to distribute private keys alongside certificates to its members.

Further, organisations will need to distribute certificates for pinning purposes. This may be for Leaf Certificate Pinning or for a CA certificate being pinned as part of Certificate Validation with CA Pinning. Additionally, if a central organisation is providing private keys for its members, it will be providing certificates alongside the private keys. Whilst certificates are not as sensitive as PSKs or private keys, the distribution of them for pinning purposes also needs to be kept secure to ensure that they are genuine.

In all of these cases, one is establishing trust, a trust anchor.

Thus, secrets needed for secure TLS communication (PSKs, private keys and certificates when provided for pinning purposes) should only be distributed in a secure manner between organisations.

Details about how to transfer secrets securely is non-normative. However, examples include:

- Secure web portal (itself using TLS) and possibly using additional file encryption
- Email with encrypted files or via services like DropBox, but using file encryption and not relying upon DropBox alone
- Postal services with possible signing for receipt
    - as encrypted media on CD/DVDs
    - as a secure token or HSM
    - e.g. in two (or more) protected envelopes, with the key value split between the envelopes, with each envelope addressed to a different person (and or location), and with each postage sent on a different day, etc.
- In person, by voice, and or using a mixture of different methods, etc.

In most of cases, further secrets are needed to encrypt/decrypt and to sign/verify files, or to gain access to web portals or access to tokens or HSMs. These secrets must also be exchanged in a secure way, and it is likely that the process will go back to the initial onboarding of firms or people, together with verification of their identity.

A number of file encryption solutions are available. A practical solution may be to use GnuPG, which supports OpenPGP and other methods of encryption. PKCS#12 files (.p12 files) (or former PFX, .pfx, files) are also useful and could be used in conjunction with other controls e.g. secure web portal.

It may be worth considering using a one-time secret for each transfer.

It is important to ensure that the receiving organisation is under an obligation to keep the transferred secrets secret, and that it also has an obligation to keep the service which you are providing as well as your data secure. It is sensible to provide guidance on a minimum set of controls which you expect the receiving organisation to implement. You will additionally want to have provisions for auditing and sampling within your legal agreements.

4.1.2 Trust on first use

A server certificate (and perhaps a CA certificate within the server certificate chain) (but not a client certificate) can be retrieved directly from the server itself. For example, one can use the OpenSSL ‘s_client’ command line tool to retrieve the server certificate.

Whilst the use of a certificate downloaded in this way is not secure for certificate pinning generally, it does provide a practical method for obtaining a copy of the server certificate. This may be useful for testing purposes. It may also provide a satisfactory basis for some firms that are prepared to trust the certificate on first use. This is what is known as Trust on First Use (TOFU).

One can also ensure that the downloaded certificate is genuine though by additionally verifying the properties of the certificate directly with the certificate party by other means. For example, one could telephone the organisation to check that the fingerprint and expiry date are correct.

If certificates have been downloaded in this way, one must take care not to just download a certificate again when the existing certificate no longer works. One should instead obtain a future certificate securely by other means in advance of the existing certificate expiring. If the certificate has already changed though and it is downloaded from the server again, it is important to additionally validate the reasons for the certificate change and that the revised certificate is genuine. This may again be by telephone, provided there is no possibility of the organisation being impersonated.

4.2 Renewing secrets

Secrets should be renewed and checked that they are still required on a regular basis. Additionally, secrets need to be checked for expiry and renewed with sufficient time prior to expiry.

Care should be taken, especially with PSKs, to ensure that secrets are changed at a time which is convenient for both parties, outside of when the secrets are needed. Alternatively, supporting multiple simultaneous secrets, like using a pinset for certificate pinning, may be best.

4.3 Authorisation linked to authentication

Authorisation must be linked back to authentication.

Care should be taken to ensure that the party authorised to exchange messages is the same party which has been authenticated. For example, if Party A is connected to parties B and C, it should only be possible for Party B to send and receive messages for the parties which it has been authorised to do so. It is unlikely that Party B would have been authorised to send and receive messages on behalf of Party C so Party B should not be able to do this.

It is relatively straightforward to ensure coupling of the authenticated and authorized parties for clients authenticated with FIXA, but it is harder to enforce it for parties authenticated at the TLS level. For example, if Stunnel is being used, there is no method to pass identification of the authenticated party to the application.

One must either have one or more channels, or Stunnel services, dedicated to each party, in which each channel would indicate the authenticated party; or have closer integration between the TLS and application layers, in order to allow identification of the authenticated party to be passed back. For example, the TLS layer could provide a callback function which could be used to authenticate the party and capture the party identification for the application layer. However, this would involve closer integration and possibly, in the case of the server, it would require part of the application to be a proxy within a DMZ. A client application though, which is using Stunnel, would typically dedicate one or more Stunnel service instances (channels) to each party which would in turn infer the authenticated party.

5 Appendix A – Cipher Suites

The acronyms used in this appendix match those in the TLS Cipher Suite Registry which is authoritative and held at IANA for IETF. We have chosen to use these acronyms as opposed to, for example, using the ones from the OpenSSL cipher suite names. See

http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml or
https://www.ietf.org/assignments/tls-parameters/tls-parameters.txt

5.1 Ciphers

We have ciphers of type stream, block and Authenticated Encryption with Associated Data (AEAD). AEAD is a block cipher mode of operation. Block ciphers include the CBC mode of operation. AEAD ciphers include the GCM and CCM modes of operation.

Stream ciphers are more efficient than block ciphers which is important for small devices (and maybe for throughput and latency), but RC4 has a number of vulnerabilities and is considered insecure. Thus, work is going on to include ChaCha20-Poly1305 in TLS as it does not have any known vulnerabilities to date.

Hash (aka message authentication code (MAC)) algorithms:

Acronym	Status	Description
	Not yet available in TLS	Secure Hash Algorithm (SHA) 3 SHA-3 includes SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 and SHAKE256 NIST released SHA-3 in late 2015 as contingency against a compromise in SHA-2, but SHA-2 remains robust
SHA384 SHA256	OK	Secure Hash Algorithm (SHA) 2 SHA-2 can produce 224, 256, 384 or 512 bit digests (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256) Sha512/384/256/224 together with SHA-1 are known as SHS – NIST FIPS PUB 180-2, "Secure Hash Standard", National Institute of Standards and Technology, U.S. Department of Commerce, August 2002 Only 384 and 256 bit digests available in TLS cipher suite registry.
SHA	Deprecated (but still acceptable for TLS)	Secure Hash Algorithm (SHA) 1 SHA-1 produces a 160 bit digest (20 bytes or 40 hexadecimal digits), no longer considered secure
MD5	Prohibited	MD5 Message-Digest Algorithm MD5 produces a 128 bit digest, various vulnerabilities but still useful for checksum for unintentional corruption, replaces MD4

Modes of operation:

Acronym	Status	Description
Modes for AEAD block ciphers		
GCM	OK	Galois/Counter Mode (GCM) Good parallel processing gives efficiency and performance. Defined for block ciphers with a block size of 128 bits. Galois Message Authentication Code (GMAC) is an authentication-only variant of the GCM which can be used as an incremental message authentication code.
CCM	OK	The CCM Mode for Authentication and Confidentiality http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf CCM is not as fast as and, therefore, not considered as good as GCM.
Modes for non-AEAD block ciphers		
CBC	OK	Cipher Block Chaining (CBC) Every block is XORed with the previous block before encryption and XORed again after decryption.

No modes of operation are relevant for stream ciphers in TLS.

Ciphers:

Acronym		Status	Description
AEAD block ciphers			
AES/GCM	AES_256_GCM_SHA384 AES_128_GCM_SHA256	OK	
CAMELLIA/GCM	CAMELLIA_256_GCM_SHA384 CAMELLIA_128_GCM_SHA256	OK	
AES/CCM	AES_256_CCM AES_256_CCM_8 AES_128_CCM AES_128_CCM_8	OK (but not as fast as GCM)	Less efficient than GCM, but maybe better than non-AEAD modes
Non-AEAD block ciphers			
AES	AES_256_CBC_SHA384 AES_256_CBC_SHA256	OK (except SHA-1 algorithms)	Advanced Encryption Standard (AES) (aka Rijndael, a subset of

Acronym		Status	Description
	AES_256_CBC_SHA AES_128_CBC_SHA256 AES_128_CBC_SHA		the Rijndael family), by NIST, supersedes DES Block cipher: 128-bit block sizes with 128-, 192- and 256-bit key sizes (but only 128- and 256-bit key sizes supported by TLS)
CAMELLIA	CAMELLIA_256_CBC_SHA384 CAMELLIA_256_CBC_SHA256 CAMELLIA_256_CBC_SHA CAMELLIA_128_CBC_SHA256 CAMELLIA_128_CBC_SHA	OK (except SHA-1 algorithms)	Comparable security to AES, developed by Mitsubishi Electric and NTT of Japan Block cipher: 128-bit block size with 128-, 192- and 256-bit key sizes RFC 4132: Addition of Camellia Cipher Suites to Transport Layer Security (TLS) RFC 5932: Camellia Cipher Suites for TLS RFC 6367: Addition of the Camellia Cipher Suites to Transport Layer Security (TLS)
DES	3DES_EDE_CBC_SHA 3DES_EDE_CBC_MD5 DES_CBC_SHA DES_CBC_MD5 DES40_CBC_SHA DES_CBC_40_SHA DES_CBC_40_MD5	Deprecated	Data Encryption Standard (DES), superseded by AES. Block cipher: 64-bit block size with a 56-bit key. The Triple-DES (3DES) mode of operation uses 3 keys and 3 encryptions = 3*56 = 168-bit key (equivalent to 112 bits of security)
SEED		Legacy/non-standard	Block cipher using a 128-bit key on a 128-bit block size, developed by Korea Information Security Agency (KISA) to overcome 40-bit encryption limitations, seldom used outside Korea and limited support in implementations. Defined for use in TLS by RFC 4162.
Stream ciphers			

Acronym		Status	Description
CHACHA20-POLY1305	CHACHA20_POLY1305_SHA256	Proposed	ChaCha20-Poly1305, replacement for RC4 Stream cipher See RFC 7905 (June 2016)
RC4	RC4_128_SHA RC4_128_MD5 RC4_40_SHA RC4_40_MD5	Prohibited	Rivest Cipher (RC) 4 Stream cipher Simple and fast but a number of vulnerabilities so considered insecure; RFC 7465 now prohibits RC4 in TLS; RC2 similar
RC2	RC2_CBC_40_SHA RC2_CBC_40_MD5	Prohibited	

IDEA – International Data Encryption Algorithm (IDEA) is a block cipher using a 128-bit key on a 64-bit block size. It is not widely used so it is not supported by TLS.

5.2 Key exchange and authentication methods

Key exchange methods:

Acronym	Status	Description
ECDHE	OK	Elliptic Curve Diffie-Hellman exchange with Ephemeral keys (ECDHE); Faster than DHE; Provides Perfect Forward Secrecy (PFS)
ECDH	OK	Elliptic Curve Diffie-Hellman (ECDH) exchange; Faster than DH
DHE	OK	Diffie-Hellman exchange with Ephemeral keys (DHE). Also referred to as EDH. Provides Perfect Forward Secrecy (PFS)
DH	OK	Diffie-Hellman (DH) exchange
RSA	OK	RSA method
PSK	OK	Methods for PSKs. Note there are 3 methods: (a) standard method; (b) combining DHE with PSK; and (c) RSA authentication of server with PSK authentication of client
SRP SRP_SHA_RSA		Secure Remote Password (SRP)

Acronym	Status	Description
SRP_SHA_DSS		
KRB5 KRB5_EXPORT		Kerberos

A cryptographic key is called ephemeral if it is generated for each execution of a key establishment process.

Authentication methods:

Acronym	Status	Description
ECDSA	OK	Elliptic Curve Digital Signature Algorithm (ECDSA) Uses Elliptic Curve Cryptography (ECC) certificates
DSS (aka DSA)	OK	Digital Signature Standard (DSS) employing the Digital Signature Algorithm (DSA) by NSA/NIST
RSA	OK	Algorithm developed by Ron Rivest, Adi Shamir and Leonard Adleman (RSA)
PSK	OK	Pre-shared key (PSK)
SRP SRP_SHA_RSA SRP_SHA_DSS		Secure Remote Password (SRP) defined in RFC 5054, November 2007 (and RFC 2945) e.g. for IMAP 1st class only SRP 2nd class SRP and certificates e.g. server by certificate, client by SRP Cipher suites SRP available for AES256, AES128 and 3DES.
KRB5 KRB5_EXPORT	Deprecated	Kerberos RFC 6251, May 2011, and RFC 2712, October 1999 Symmetric key authentication system RFC 1510 KRB5 KRB5_EXPORT 40-bit cipher suites legacy US export Cipher suites KRB5 and KRB5_EXPORT only available for IDEA, 3DES, DES, RC4 and RC2 so deprecated in this list.

Available key exchange and authentication methods:

Acronym	Reference	Status	Description
Using RSA certificates:			
ECDHE_RSA	RFC 4492	OK	

Acronym	Reference	Status	Description
DHE_RSA	Base 1.2	OK	Diffie-Hellman exchange with Ephemeral keys (DHE) and RSA authentication
ECDH_RSA	RFC 4492	OK	
DH_RSA	Base 1.2	OK	
RSA	Base 1.2	OK	
Using PSKs:			
ECDHE_PSK	RFC 5489	OK (*)	
DHE_PSK	RFC 4279	OK (*)	Diffie-Hellman exchange with Ephemeral keys (DHE) and PSK authentication
PSK	RFC 4279	OK (*)	Pre-shared Key (PSK); (*) symmetric keys need to be exchanged securely by other means
Using ECC certificates:			
ECDHE_ECDSA	RFC 4492	OK	
ECDH_ECDSA	RFC 4492	OK	
Using DSA:			
DHE_DSS	Base 1.2	OK	Diffie-Hellman exchange with Ephemeral keys (DHE) and DSS authentication
DH_DSS	Base 1.2	OK	
Other:			
ECDH_anon	RFC 4492	Prohibited	Not to be used (anonymous i.e. no authentication)
DH_anon	Base 1.2	Prohibited	Not to be used (anonymous i.e. no authentication)
RSA_PSK	RFC 4279	N/A	RSA and certificates to authenticate the server; PSK to authenticate the client
SRP_SHA			
SRP_SHA_RSA			
SRP_SHA_DSS			
KRB5			
KRB5_EXPORT			

5.3 List of recommended cipher suites

	AES_128_GCM_SHA256	AES_256_GCM_SHA384	AES_128_CBC_SHA	AES_256_CBC_SHA	AES_128_CBC_SHA256	AES_256_CBC_SHA256	AES_256_CBC_SHA384
ECDSA							
ECDHE_ECDSA	✓	✓	✓	✓	✓	✓	
ECDH_ECDSA							
RSA							
ECDHE_RSA	✓	✓	✓	✓	✓		✓
DHE_RSA	✓	✓	✓	✓	✓	✓	
ECDH_RSA							
DH_RSA							
RSA							
PSK							
ECDHE_PSK			✓	✓	✓		✓
DHE_PSK	✓	✓	✓	✓	✓		✓
PSK							
DSA							
DHE_DSS							
DH_DSS							

6 Appendix B – Relevant RFCs

Relevant IETF Request for Comments (RFCs) for TLS and TLS cipher suites:

RFC	Date	Title
2246	1999-01	The TLS Protocol, Version 1.0
…		
2712	1999-10	Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)
4162	2005-08	Addition of SEED Cipher Suites to Transport Layer Security (TLS)
4279	2005-12	Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)
4346	2006-04	The Transport Layer Security (TLS) Protocol Version 1.1
4492	2006-05	Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)
4785	2007-01	Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)
5054	2007-11	Using the Secure Remote Password (SRP) Protocol for TLS Authentication
5246	2008-08	The Transport Layer Security (TLS) Protocol Version 1.2
5288	2008-08	AES Galois Counter Mode (GCM) Cipher Suites for TLS
5289	2008-08	TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)
5469	2009-02	DES and IDEA Cipher Suites for Transport Layer Security (TLS)
5487	2009-03	Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode
5489	2009-03	ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)
5746	2010-02	Transport Layer Security (TLS) Renegotiation Indication Extension
5932	2010-06	Camellia Cipher Suites for TLS
6209	2011-04	Addition of the ARIA Cipher Suites to Transport Layer Security (TLS)
6251	2011-05	Using Kerberos Version 5 over the Transport Layer Security (TLS) Protocol
6367	2011-09	Addition of the Camellia Cipher Suites to Transport Layer Security (TLS)
6655	2012-07	AES-CCM Cipher Suites for Transport Layer Security (TLS)
7251	2014-06	AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS
7507	2015-04	TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks
7905	2016-06	ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)

Other RFCs:

7469 - Public Key Pinning Extension for HTTP

5280 - X.509 PKI Certificate and CRL Profile

6797 - HTTP Strict Transport Security (HSTS)

7 Appendix C – Known Vulnerabilities

Known vulnerabilities relevant to FIXS may be provided in a future version of this document.

8 Appendix D – TLS Implementations

See https://en.wikipedia.org/wiki/Comparison_of_TLS_implementations

OpenSSL and Stunnel

http://www.openssl.org

http://www.stunnel.org

8.1 Microsoft Secure Channel (Schannel)

.NET SslStream

Schannel SSP Windows (8.1 / 2012R2, 10) (8 / 2012) (7 / 2008R2) (Vista / 2008) (XP / 2003)

https://technet.microsoft.com/en-us/library/dn786419(v=ws.11).aspx

https://msdn.microsoft.com/en-gb/library/windows/desktop/aa374757(v=vs.85).aspx

https://technet.microsoft.com/en-us/library/hh831771(v=ws.11).aspx

https://technet.microsoft.com/en-us/library/dn786429(v=ws.11).aspx

Schannel uses CryptoAPI

https://msdn.microsoft.com/en-us/library/windows/desktop/aa379809(v=vs.85).aspx

Web API

http://www.asp.net/web-api/overview/hosting-aspnet-web-api/use-owin-to-self-host-web-api

http://chimera.labs.oreilly.com/books/1234000001708/ch15.html

http://www.codeproject.com/Articles/838276/Web-API-Thoughts-of-Working-with-HTTPS

http://vineetyadav.com/development/net/configure-self-hosted-webapi-in-windows-service-to-use-ssl.html

8.2 BouncyCastle

Bouncycastle.org for Java

8.3 RSA BSAFE

8.4 GnuTLS

GnuTLS (http://www.gnutls.org) – being maintained currently (September 2016); supports TLS 1.2; supports OCSP; supports certificate verification using DANE (DNSSEC …) and trust on first use (TOFU); supports SRP and PSK

8.5 Other

JSSE

Apple Inc. Secure Transport (OS X)

SharkSSL

TLSe

wolfSSL

Eldos SecureBlackbox (for Windows, .NET, Linux, MacOS / iOS, Java / Android).

eldos.com

9 References

SSL Labs

See https://www.ssllabs.com/. Publications include:

[SSLLABS-BSSL] Bulletproof SSL and TLS, August 2015, Ivan Ristić, ISBN 978-1907117046. See https://www.feistyduck.com/books/bulletproof-ssl-and-tls/

[SSLLABS-DBP] SSL and TLS Deployment Best Practices, SSL Labs, Version 1.5, 8 June 2016. See https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices

[SSLLABS-SRG] SSL Server Rating Guide, SSL Labs, version 2009k, 14 October 2015. See https://www.ssllabs.com/downloads/SSL_Server_Rating_Guide.pdf

[SSLLABS-OSSL-CB] OpenSSL Cookbook. See https://www.feistyduck.com/books/openssl-cookbook/

bettercrypto.org

See https://bettercrypto.org/. Publications include:

[BC-ACH] Applied Crypto Hardening, Better Crypto, August 7, 2016. See https://bettercrypto.org/static/applied-crypto-hardening.pdf.

Cipherli.st

See https://cipherli.st/.

OWASP

See the following web pages:

[OWASP-TLPCS] https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

[OWASP-CPKP] OWASP Certificate and Public Key Pinning, at 2-Nov-2016 (see https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning)

[OWASP-PCS] OWASP Pinning Cheat Sheet, dated 6-Jul-2016 (see https://www.owasp.org/index.php/Pinning_Cheat_Sheet).

NIST

See the following:

[NIST-GSCUI] NIST Special Publication 800-52 Revision 1, Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations. See http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf.

