MODBUS/TCP Security

Protocol Specification

Table of Contents

List of Figures

List of Tables

1 Conformance Levels

Table 1 Conformance Levels

Latest conventions available up-to-now	In a standard document, specific notations shall be used to define the significance of each particular requirement. These notations (words) are highlighted by **capitalization**. As Consistency Rules may have the target to be presented to a standards body in order to become an international standard, the selection of the words "**SHALL**" and "**MUST**" should be made according to the rules of the organization that covers the standardization in the affected area of the Specification.
Compliance	An implementation that satisfies all the **MUST / SHALL** requirements is said to be "**unconditionally compliant**". One that satisfies all the **MUST** requirements but not all the **SHOULD** recommendations is said to be "**conditionally compliant**". An implementation is not compliant if it fails to satisfy one or more of the **MUST / SHALL** requirements that it implements
MUST **SHALL** **REQUIRED**	All requirements containing the word "**MUST / SHALL**" are mandatory. The word "**MUST / SHALL**", or the adjective "**REQUIRED**", means that the item is an absolute requirement of the implementation.
MUST NOT **SHALL NOT**	All requirements containing the word "**MUST NOT/ SHALL NOT**" are mandatory. The phrase "**MUST NOT" or the phrase "SHALL NOT**" mean that the item is an absolute prohibition of the specification.
SHOULD **RECOMMENDED**	All recommendations containing the word "**SHOULD**", or the adjective "**RECOMMENDED**" are considered desired behaviour. These recommendations should be used as a guideline when choosing between different options to implement functionality. In uncommon circumstances, valid reasons may exist to ignore this item, but the full implication should be understood and the case carefully weighed before choosing a different course.
MAY **OPTIONAL**	The word "**MAY**", or the adjective "**OPTIONAL**", means that this item is truly optional. One implementer may choose to include the item because a particular marketplace requires it or because it enhances the product; another implementer may omit the same item.

2 Normative Statements

Normative statements in this technical specification are called out explicitly as follows:

R-n.m: Normative statement text goes here.

where "n.m" is replaced by the requirement statement tag number which can be a hierarchical number, e.g. R-1.2.3 or a simple integer, e.g. R-1.

Each statement contains exactly one requirement level keyword (e.g., "**MUST**") and one conformance target keyword (e.g., "Message"). Example: “The Message **MUST** be encoded using BER”.

3 References

Table 2 References

Reference	Description
[62443-3-3]	IEC 62443-3-3: System security requirements and security levels
[62443-4-2]	IEC 62443-4-2: Technical security requirements for IACS components
[MB]	Modbus Application Protocol Specification, V1.1b3, 2012-04-26, https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
[MBTCP]	Modbus Messaging on TCP/IP Implementation Guide, V1.0b, 2006-10-24, https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf
[RFC4492]	IETF RFC 4492, Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)
[RFC5246]	IETF RFC 5246, The Transport Layer Security (TLS) Protocol, v1.2, Aug 2008
[RFC5280]	IETF RFC 5280, Internet x.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, May 2008
[RFC5746]	IETF RFC 5746, TLS Renegotiation Indication Extension, Feb 2010
[RFC6066]	IETF RFC 6066, TLS Extensions: Extension Definitions, Jan 2011
[RFC6176]	IETF RFC 6176, Prohibiting Secure Sockets Layer (SSL) Version 2.0, Mar 2011
[TLS-PARAMS]	IANA’s Transport Layer parameter type registry. https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml

4 Glossary of Acronyms & Abbreviations

Table 3 Glossary of Acronyms & Abbreviations

Reference	Description
ADU	Application Data Unit
AuthZ	Authorization
CA	Certificate Authority
CDP	CRL Distribution Point
CRL	Certificate Revocation List
HMAC	Keyed-hash Message Authentication Code
IANA	Internet Assigned Numbers Authority
ICS	Industrial Control System
IEC	International Electrotechnical Commission
MAC	Message Authentication Code
mbap	Modbus Application Protocol
mbaps	Modbus Security Application Protocol
OID	Object Idenitifier standardized by the International Telecommunications Union
PEN	Private Enterprise Number
PDU	Protocol Data Unit
PKI	Public Key Infrastructure
PRF	Psuedorandom Function Family
RA	Registration Authority
SSL	Secure Socket Layer
TCP	Transport Control Protocol
TLS	Transport Layer Security

5 Introduction

The Modbus/TCP protocol is widely deployed in Industrial Control Systems (ICS). The specifications for Modbus/TCP are found at the modbus.org web site. The Modbus/TCP specification defines an Application Data Unit (ADU). This ADU is defined as shown in Figure 1 Modbus/TCP ADU:

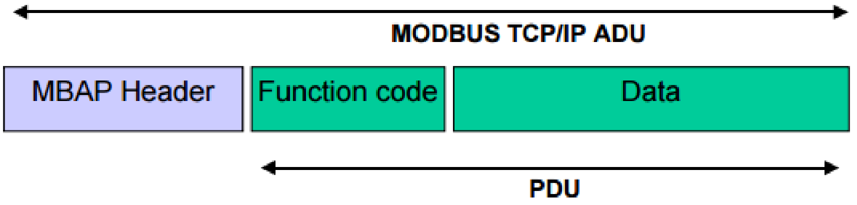

Figure 1 Modbus/TCP ADU

The difference between a traditional Modbus Protocol Data Unit (PDU) and the Modbus/TCP ADU is the addition of the Modbus Application Protocol (mbap) header at the front of the frame.

Modbus/TCP Security Principles

- Modbus/TCP Security @ port 802
- x.509v3 certificate based identity and authentication with TLS
- Mutual client/server TLS authentication
- Authorization using roles transferred via certificates
- Authorization rules are product specific
- No changes to mbap

In 1996 the Modbus/TCP protocol, was registered with IANA (Internet Assigned Number Authority) and assigned the system port number 502. In the course of this registration process with IANA the Modbus/TCP protocol came to be called the mbap protocol because of the mbap header in the Modbus/TCP ADU. This name, the mbap protocol, persisted and is still used for the port 502 registration with the IANA as mbap/TCP

The Modbus/TCP Security protocol is a security focused variant of the Mobdbus/TCP protocol utilizing Transport Layer Security (TLS). IANA has assigned the Modbus/TCP Security protocol the system port number 802. Modbus.org has registered the name Modbus Security Application Protocol to the protocol registered at port 802 with IANA as mbap/TLS/TCP

The selection of TLS as the secure transport protocols is the result of analyzing representative data flows from industry domains in the context of [62443-3-3] and [62443-4-2].

Table 4 Context Specific Terminology lists the names used for the mbap communication profiles in different contexts, e.g. Communication Profile, Modbus.org, the IANA Registry, and this specification. For reasons of brevity, the remainder of this specification will use mbap and mbaps to refer to Modbus/TCP and Modbus/TCP Security respectively.

Table 4 Context Specific Terminology

Communication Profile	Modbus.org	IANA Registry	This specification (for brevity)
mbap/TCP	Modbus/TCP	Modbus Application Protocol at System Port 502	mbap
mbap/TLS/TCP	Modbus/TCP Security	Modbus Security Application Protocol at System Port 802	mbaps

6 Protocol Overview

6.1 General

In the tradition of Modbus, the mbaps requirements are kept simple allowing vendors to develop additional infrastructure around the protocol and allowing backwards compatibility with legacy devices and fieldbuses. Mbaps extends the original mbap protocol as defined in [MBTCP] and [MB]. Mbaps defines a client-server protocol that is a part of a complete security system architecture. As illustrated in Figure 3 mbap ADU Encapsulated in TLS, the mbap ADU is encapsulated by TLS. TLS provides a security focused protocol alternative to mbap by adding confidential transport of the data, data integrity, anti-replay protection, endpoint authentication via certificates, and authorization via information embedded in the certificate such as user and device roles.

The protocols mbap and mbaps are similar to http and its secure variant https respectively. In mbaps, the mbap protocol is transported via TLS. TLS provides an authentication capability via x.509v3 certificates. The mbaps clients and servers must be provisioned with the these certificates to participate in the TLS Authentication function.

An important difference between mbap and mbaps is that mbaps provides the capability of the server invoking an authorization function whose rules are driven by the vendor or customer, utilizing role data that is provided via an extension field in the x.509v3 certificate. The extension is registered with Modbus.org's IANA OID. TLS provides for the use of pre-shared keys to establish a secure connection, but the use is not considered for this specification as it does not allow for the trasfer of role information to provide an authorization function.

6.2 Transport Layer Security Introduction

The mbaps/TLS/TCP profile uses the secure TLS transport protocol defined in IETF RFC 5246. [RFC5246] defines TLS v1.2 and provides countermeasures and mitigations for known vulnerabilities in earlier versions. While this specification is based on TLS v1.2, newer versions will be considered as their implementations become widely adopted.

TLS is composed of a set of protocols as illustrated in Figure 2 TLS Communications Protocol Stack. The main protocol in the set is the TLS Record Protocol. The remaining protocols are sub-protocols which are carried by the TLS Record Protocol. These are managed by a TLS middleware.

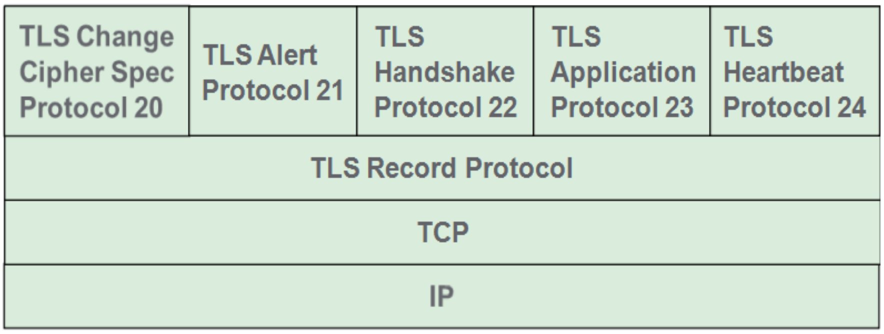

Figure 2 TLS Communications Protocol Stack

The mbap ADU which is unchanged in the mbaps profile is encapsulated in a TLS Application Protocol message as illustrated in Figure 3 mbap ADU Encapsulated in TLS.

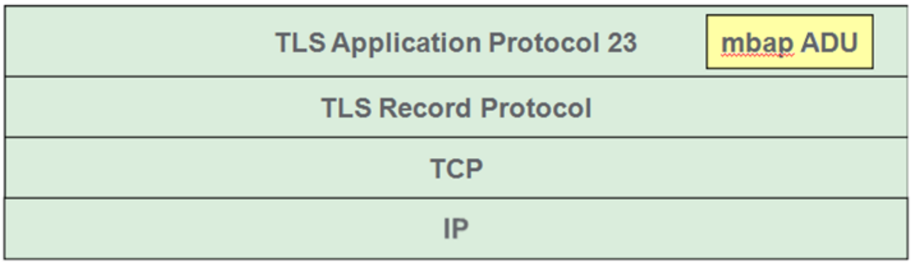

Figure 3 mbap ADU Encapsulated in TLS

Modbus/TCP Security

- Mutual client/server TLS Authentication.
- Certificate based Identity and Authentication with TLS.
- Certificate based Authorization using role information transferred via certificate extensions.
- Authorization is product specific and invoked by mbap function code handler.
- Authorization roles to rights rules are product specific and configured in the Authorization function.

The TLS Handshake Protocol shown in Figure 4 Modbus/TCP Security Concept View:

- Negotiates cryptography for secure channel including algorithms, keys, etc. between end points.
- Provides mutual client/server authentication based on x.509v3 certificates
- Extracts the client role OID from the certificate
- Establishes the TLS session.

After the TLS session is established normal modbus request and response sequences are transmitted in the secured TLS Application Protocol channel. During the procesing of the request, the mbaps protocol handler invokes a vendor specific authorization function. This authorization function evaluates a roles-to-rights algorithm using inputs from the mbap ADU and the role extracted from the x.509 client certificate of the connection. The algorithm determines if the ADU can be processed based on role of the peer. If the authorization function determines that the mbap ADU code cannot be processed, the mbap handler returns a 01 – Illegal Function Modbus exception code. This authorization process occurs on every request, ensuring complete validation of the request stream.

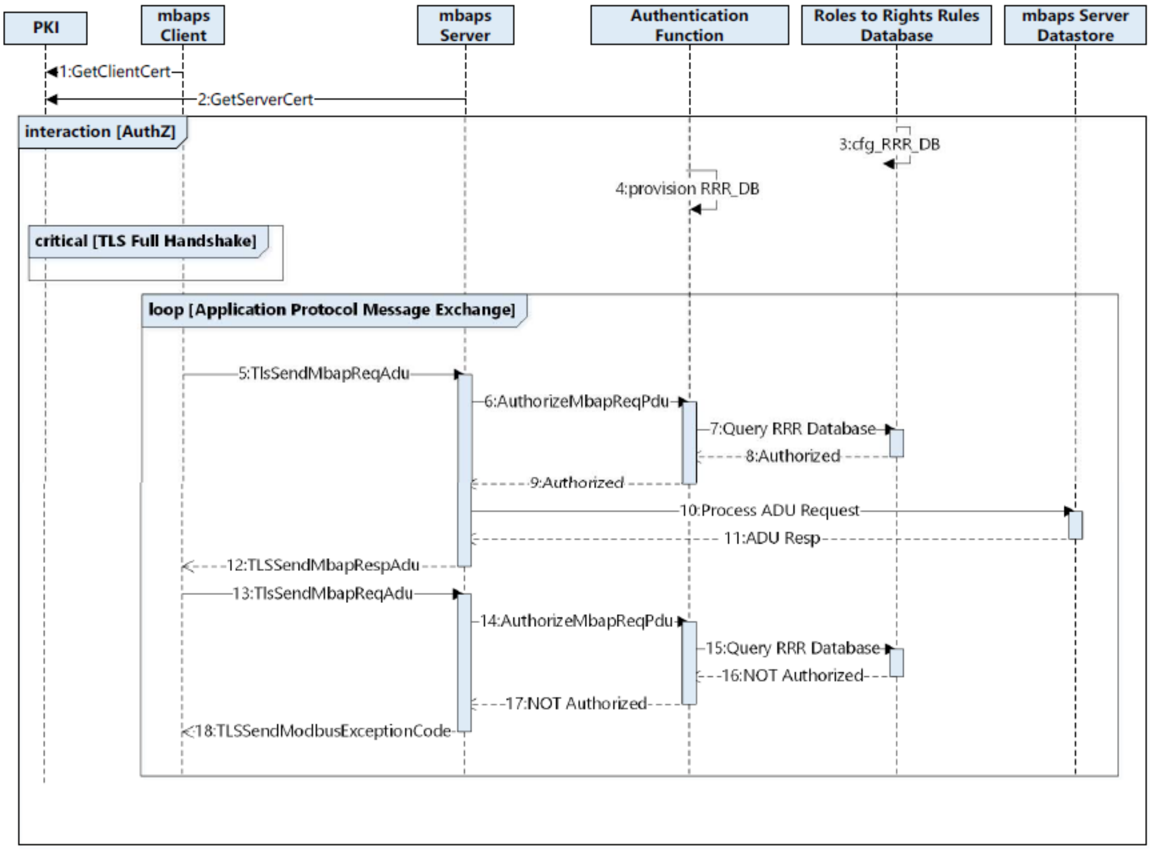

Figure 4 Modbus/TCP Security Concept View

```
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number: 4135 (0x1027)
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=US, ST=STATE, L=LOCAL, O=ORG, OU=SUBORG, CN=INTER-CA
    Validity
        Not Before: Oct 27 12:58:27 2017 GMT
        Not After : Oct 27 12:58:27 2018 GMT
    Subject: C=US, ST=STATE, L=LOCAL, O=ORG, OU=SUBORG, CN=ModbusSecurityClient
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:be:3d:4d:9e:8c:fe:1e:06:e6:19:cd:52:68:07:
                54:c6:d3:b3:cd:bb:da:dd:29:29:b5:2d:2f:3b:bf:
                b9:3c:c7:c2:f4:a9:98:ce:6e:47:f5:64:7d:6d:e8:
                a3:6b:02:da:4c:e9:05:b8:aa:30:d9:95:13:1f:14:
                58:3e:c1:dc:a7:21:ca:c0:90:c9:e5:80:70:2b:8d:
                4d:0a:78:96:c0:9e:1f:f1:1d:e7:e8:24:be:06:a1:
                b8:6a:67:d3:7f:1c:d4:cb:c3:85:5a:f8:a7:ef:d1:
                e0:df:30:60:44:29:a3:4d:63:24:d2:7f:e9:45:29:
                2d:e9:fa:53:3d:be:f8:cd:72:64:08:dc:7e:b0:e9:
                d1:c2:e7:52:de:eb:9d:b0:60:b1:73:62:24:ac:ba:
                08:5f:65:23:9a:38:b5:48:53:08:bc:79:ae:b1:55:
                fd:b1:f3:6f:c9:fa:ac:aa:89:aa:f9:59:ca:bf:fe:
                7a:12:cf:88:20:5b:5e:8b:b5:b1:58:04:41:19:2c:
                26:91:0d:ce:86:38:93:32:a0:ab:57:01:38:5a:41:
                36:77:ae:2b:89:28:8e:22:48:84:b6:18:b9:31:aa:
                52:c3:72:3a:19:41:65:21:87:32:4b:c0:53:3e:aa:
                36:dd:d6:40:09:55:e3:65:2c:f9:d4:61:24:6d:60:
                64:87
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        X509v3 Subject Key Identifier:
            B3:09:92:E3:60:44:DE:F5:5B:30:8B:3B:D3:EA:78:FF:CE:DA:E3:48
        X509v3 Key Usage: critical
            Digital Signature, Non Repudiation, Key Encipherment
          RoleOID:1.3.6.1.4.1.50316.802.1:
              Operator
        X509v3 Subject Alternative Name:
            IP Address:192.168.2.12, IP Address:192.168.2.22
Signature Algorithm: sha256WithRSAEncryption
     4f:a2:ca:1f:ea:11:b8:55:89:97:6a:b8:f2:bc:a6:30:e4:6a:
     d7:1e:25:8e:db:cb:f1:54:23:9a:ce:39:e4:dd:96:5f:ce:2a:
     0c:73:43:23:06:7d:a4:fa:33:48:2c:86:42:a7:eb:d8:d4:fa:
     d1:08:07:e9:b1:9c:51:b6:78:9c:e7:2e:fb:22:cc:89:28:ef:
     8f:7a:30:a9:73:e8:28:9a:ab:a4:f2:d5:ec:29:e8:dc:77:a7:
     f5:e1:71:8a:0f:76:4c:78:a5:5c:b7:ea:4e:86:c7:fe:01:17:
     8c:4a:b1:7c:11:d7:f7:a6:81:d4:1c:bb:86:af:d5:20:fe:05:
     ec:0f:de:8d:d1:c0:76:40:31:0f:15:23:65:4d:5c:7c:52:d3:
     cd:c7:81:a5:8a:4f:51:e1:2b:07:9a:8b:83:0d:95:91:97:37:
     6d:59:c5:ca:2e:5d:82:a8:ac:1c:f8:0a:56:06:dc:47:93:db:
     bc:c6:21:94:dd:55:ee:90:3f:ad:f8:15:22:16:99:cf:3f:bc:
     2f:af:aa:04:16:0d:e6:89:c2:f4:af:cb:0e:27:fc:5c:d9:3f:
     5c:5a:b7:4b:aa:d9:a5:eb:0a:3e:53:16:1a:3f:10:20:7b:52:
     ea:93:ed:b8:21:43:b3:dd:cb:38:1f:d9:38:d1:10:09:c0:25:
     df:bf:6a:b7
```

Example x.509v3 Certificate with Role Encoded as a Certificate Extension

- Example Role is Operator
- The OID for the Role is defined in the Modbus.org Private MIB whose PEN (Private Enterprise Number) is 50316.

Figure 5 Example x.509v3 Certificate with Role Extension

The development of mbaps and its deployment in a device were guided by a set of principles including:

- R-01: The TLS Protocol v1.2 as defined in [RFC5246] or newer **MUST** be used as the secure transport protocol for an mbaps Device.
- R-02: Secure communications to an mbaps Device **MUST** use Mutual client/server authentication as provided by the TLS Handshake Protocol.
- R-03: x.509v3 Certificates as defined in [RFC5280] **MUST** be used as mbaps device credentials for Identity/Authentication by the TLS protocol.
- R-04: If the Authorization function is enforced it **MUST** use the role transferred via x.509v3 certificate extensions.
- R-05: There **MUST** be no change to the mbap protocol as a consequence of it being encapsulated by the secure transport.

7 Service Definition

Standard function codes used on Modbus Application layer protocol are described in details in the [MB] specification. There is no modification to the standard function codes in this specification.

8 Protocol Specification

8.1 General

The communication of an mbap ADU is secured using the Transport Layer Security protocol, TLS, defined in [RFC5246]. Figure 3 mbap ADU Encapsulated in TLS illustrates how an mbap ADU is transmitted via the TLS Application Protocol.

TLS provides Transport Layer Security between two end points. To do this, the TLS end points execute the TLS Handshake protocol to negotiate security parameters and to create a TLS session.

8.2 TLS Handshake

For two mbaps end devices to communicate securely using TLS, a security context between the end points of the TLS connection must be established. The TLS Handshake protocol establishes the secure context, i.e. the TLS session. The TLS session has a session identifier and the security context is described by a set of security parameters as defined in [RFC5246] section A.6.

Mutual Authentication requires that each end point will send its domain certificate chain to the remote end point. Upon receipt of a certificate chain from the remote peer, the TLS end point will verify the each certificate signature using the next CA certificate in the chain until it can verify the root of the chain.

The TLS Full Handshake Protocol, which is defined in [RFC5246] section 7.3, is illustrated in Figure 6 TLS Full Handshake Protocol.

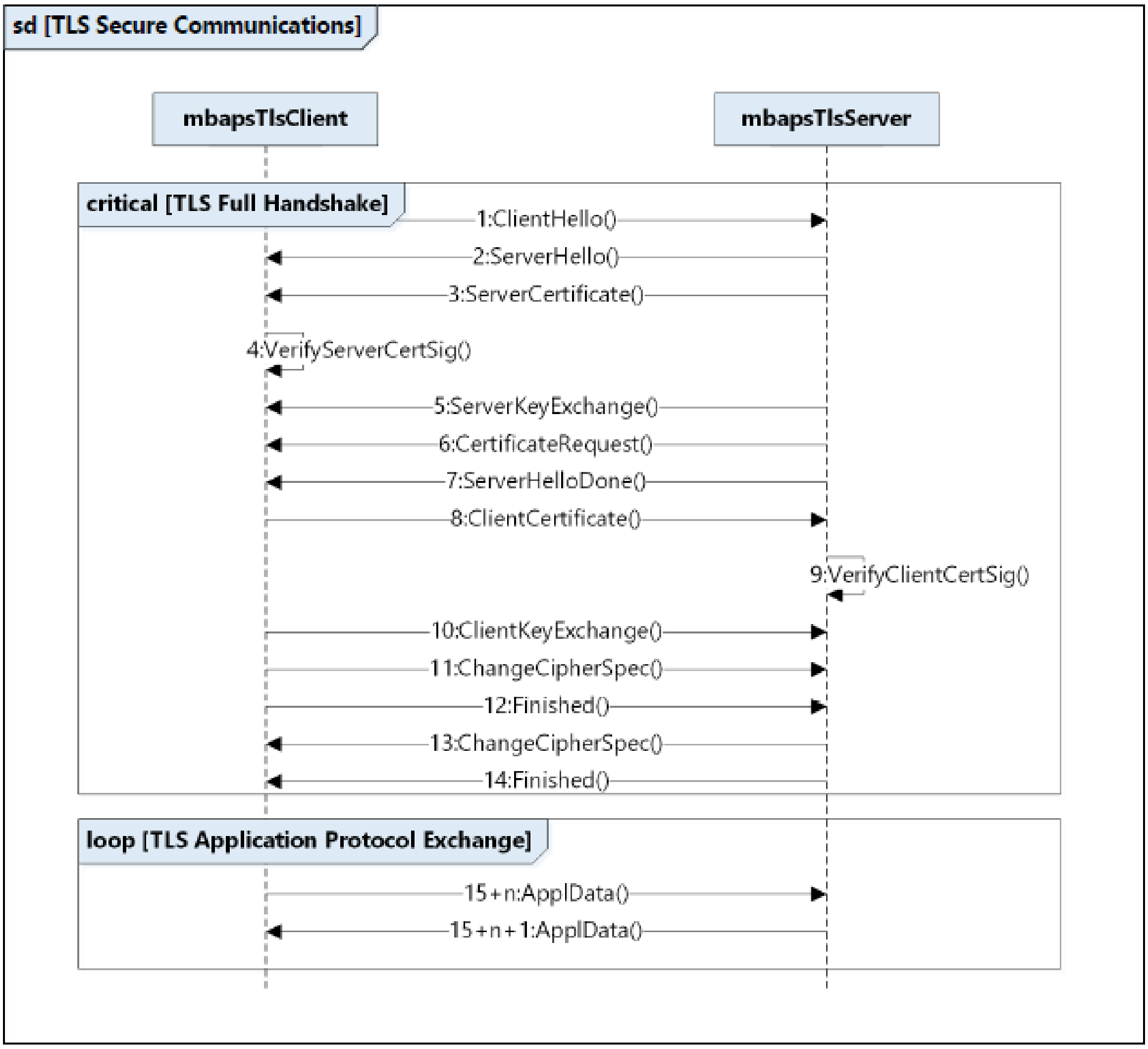

Figure 6 TLS Full Handshake Protocol

Table 5 TLS Full Handshake Protocol

Message	Description
1:ClientHello	The TlsClient sends a ClientHello message to the TlsServer to begin negotiation process. The TlsClient offers a cipher suite list in the message. The cipher suite list is ordered by the the client's preference.
2:ServerHello	TlsServer sends a ServerHello message in response to ClientHello. The message identifies an acceptable set of cryptographic algorithms and returns a new sessionID.
3:ServerCertificate	The TlsServer sends its certificate chain as the payload of a Certificate message. This chain contains the server device's domain certificate, as well as the certificate for each issuing CA down to the root CA. This server's domain certificate may also contain the role of the server; when it happens this role is not used by the client.

Message	Description
4:VerifyServerCertSig	When peer received certificate of remote peer it will check it by • verifying each certificate’s signature in the chain using public key of the issuer CA • validate the certificate path to a trusted root certificate • check the revocation status of each certificate in the chain
5:ServerKeyExchange	The TlsServer sends a ServerKeyExchange message to the TlsClient to provide data for setting the pre-master key.
6:CertificateRequest	The TlsServer sends a Certificate Request message to the TlsClient to obtain the Client Certificate.
7:ServerHelloDone	The TlsServer sends a ServerHelloDone message to the TlsClient to indicate the end of the ServerHello and associated messages.
8:ClientCertificate	The TlsClient sends its certificate chain as the payload of a Certificate message. This chain contains the client device’s domain certificate, as well as the certificate for each issuing CA down to the root CA. This client’s end certificate also contains the role of the client. This is used by the server to authorize a later application level request.
9:VerifyClientCertSig	When peer received certificate of remote peer it will check it by • verifying each certificate’s signature in the chain using public key of the issuer CA • validate the certificate path to a trusted root certificate • check the revocation status of each certificate in the chain
10:ClientKeyExchange	The TlsClient sends a ClientKeyExchange message to the TlsServer. With this message the pre-master secret is set.
11:ChangeCipherSpec	The TlsClient sends a ChangeCipherSpec message to the TlsServer to indicate that subsequent messages sent by the Client will be sent using newly negotiated cipher spec and keys.
12:Finished	The TlsClient sends a Finished message to the TlsServer. This message is the first message protected with the just negotiated algorithms, keys, and secrets.
13:ChangeCipherSpec	The TlsServer sends a ChangeCipherSpec message to the TlsClient to indicate that subsequent messages sent by the Server will be sent using newly negotiated cipher spec and keys.
14:Finished	The TlsServer sends a Finished message to the TlsClient. This message is protected with the just negotiated algorithms, keys, and secrets.
15+n:ApplData()	n ::= { 0 .. m}
15+n+1:ApplData()	n ::= { 0 .. m}

TLS [RFC5246] also provides for session resumption. The server side partner caches the last security state known, and pairs it the session ID used in the client and server hello. If the client caches the security context and sessionId it can present this sessionID to the server on the next ClientHello. If this sessionID matches with a cached sessionID on the server, the server will immediately change the cipher spec as shown in Figure 7 TLS Resumption and the connection will resume. This reduces the TLS negotiation time to 1 application round trip time, and removes the public/private key cryptographic function needed to authorize a new peer. This resumption will require the server to cache the role associated with the connection’s client certificate and associate it with the sessionID.

If the sessionID presented by the ClientHello does not match a known server session, a new sessionID is returned in the serverHello message and a full TLS handshake is performed as in Figure 6 TLS Full Handshake Protocol.

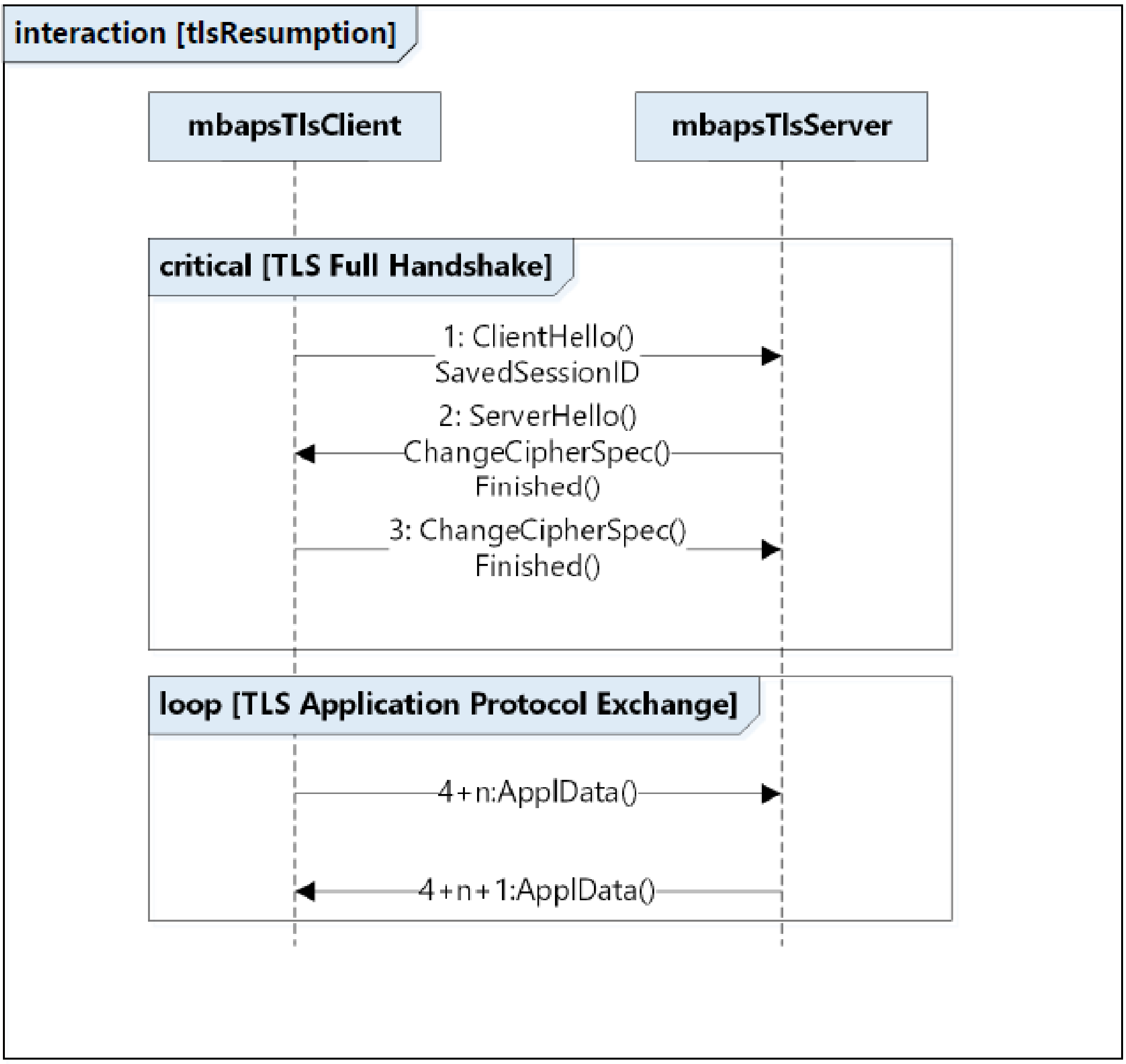

Figure 7 TLS Resumption

Table 6 TLS Resumption handshake

Message	Description
1:ClientHello	The TlsClient sends a ClientHello message to the TlsServer to begin negotiation process. The TlsClient offers a cipher suite list in the message. It also offers a cached non-zero sessionID
2:ServerHello	TlsServer sends a ServerHello message in response to ClientHello. The message identifies an acceptable cipher suite, returns the same sessionID, and includes a ChangeCipherSpec record
2:ChangeCipherSpec	The TlsServer sends a ChangeCipherSpec message to the TlsClient to indicate that subsequent messages sent by the Server will be sent using newly negotiated cipher spec and keys.
2:Finished	The TlsServer sends a Finished message to the TlsClient. This message is the first message protected with the just negotiated algorithms, keys, and secrets.
3:ChangeCipherSpec	The TlsClient sends a ChangeCipherSpec message to the TlsServer to indicate that subsequent messages sent by the Client will be sent using newly negotiated cipher spec and keys.

Message	Description
3:Finished	The TlsClient sends a Finished message to the TlsServer. This message is protected with the just negotiated algorithms, keys, and secrets.
4+n:ApplData()	n ::= { 0 .. m}
4+n+1:ApplData()	n ::= { 0 .. m}

R-06: mbaps end devices **MUST** provide mutual authentication when executing the TLS Handshake Protocol to create the TLS session.
R-07: The TlsServer **MUST** send the CertificateRequest message during the TLS handshake.
R-08: The TlsClient **MUST** send a ClientCertificate message upon receiving a request containing the Client Certificate Request.
R-10: If the TlsClient does not send a ClientCertificate message, then the TlsServer **MUST** send a ‘fatal alert’ message to TlsClient and terminate the connection.
R-11: Per RFC5246-7.2.2, the TLS connection **MUST NOT** be resumed after a ‘fatal alert’.

8.3 Cipher suite selection

The security strength of the resulting TLS session is dependent on the cipher suite negotiated between the TLS end points. Cipher suites designate what cryptography will be used by the TLS session to provide a certain level of security.

Only cipher suites registered with IANA and not known to have current weaknesses should be used in mbaps.

R-12: Cipher suites used with TLS for mbaps **MUST** be listed at the IANA Registry found @ https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml .

R-13: The cipher allowed for TLS with mbaps **MUST** accommodate the use of x.509v3 certificates.

R-14: mbaps Devices **MUST** provide at minimum the following TLS v1.2 ciper suites when using an RSA private key:

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

R-66: Client devices with bulk transport encryption and NULL bulk encryption **SHOULD** always place NULL bulk transport cipher suites last in cipher suite priority

R-67: Server devices **SHOULD** have the ability to enable use of the authentication only cipher suite TLS_RSA_WITH_NULL_SHA256.

8.4 mbaps Role-Based Client Authorization

Role-based Authorization

- Roles are encoded in x.509v3 Certificate Extension.
- Authorization function is vendor specific.
- Authorization roles to rights rules are vendor specific and configured into the Authorization function.

The mbaps protocol provides the capability to perform role-based client authorization (AuthZ). The client role data is transported in an extension of its x.509v3 domain certificate. An example of a certificate with a Role extension is shown in Figure 5 Example x.509v3 Certificate with Role Extension.

Role-Based Client Authorization for mbaps is illustrated in Figure 8 Role-Base Client AuthZ.

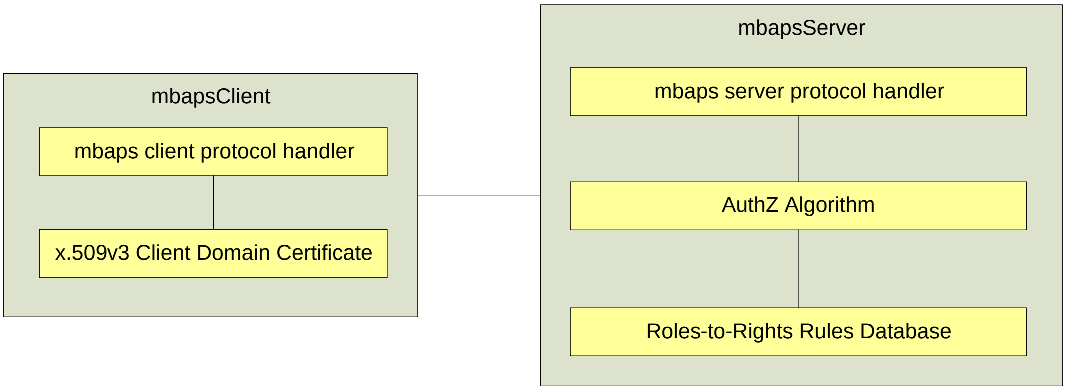

Figure 8 Role-Base Client AuthZ

Once a TLS Session is established between the two TLS end points, the execution of role-based client AuthZ is a two-step process.

During the first step, the mbaps server obtains the x.509v3 client domain certificate. This step occurs when the mbaps server receives message 8 as shown in Figure 6 TLS Full Handshake Protocol. The role is extracted from the x.509v3 certificate and cached. If a session is resumed, this role must be associated with the resumed session.

The role extension is an ASN1 encoded UTF8 string.

...
Role:
1.3.6.1.4.1.50316.802.1:Operator

Figure 9 Example Role Extension

In the example Role extension, shown in Figure 9 Example Role Extension, the Role value is 'Operator'.

The second step of the mbaps role-based client AuthZ capability involves using the extracted client Role and the Modbus request. Both fields are input to the mbaps AuthZ Algorithm. The AuthZ Algorithm determines whether the client is AUTHORIZED or NOT_AUTHORIZED to perform the indicated function on the indicated resource that was specified in the Modbus Function Code received by the mbaps server using the provisioned Roles-to-Rights Rules Database. If the request is NOT_AUTHORIZED, Modbus exception code 01 – Illegal function code will be returned. If the request is AUTHORIZED, it will be processed as normal by the mbap server.

The Authorization Function and Roles-to-Rights Rules Database may exist on the server device or may be remote requiring a separate protocol to determine the authorization status of the request. This is outside the scope of this document.

The two-step process is shown in Figure 4 Modbus/TCP Security Concept View.

R-16: A mbaps Server Device **SHOULD** provide the role-based client AuthZ as described in this section.

R-17: If a mbaps Server Device provides role-based client AuthZ, it **MUST** comply with the requirements identified in this section.

R-18: To provide mbaps role-based client authorization capability the following elements are **REQUIRED**:

x.509v3 client domain certificate ‘Role’ extension,
mbaps server AuthZ algorithm,
mbaps server Roles-to-Rights Rules Database.

R-19: The mbaps client device **MUST** be provisioned with its x.509v3 domain certificate.

R-20: The x.509v3 client domain certificate **SHOULD** include the Role extension.

R-21: The Role in the X.509v3 certificate **MUST** use the Modbus.org PEM OID 1.3.6.1.4.1.50316.802.1

R-22: The Role in the x.509v3 certificate **MUST** use ASN1:UTF8String encoding

R-65: There **MUST** only be one role defined per certificate. The entire string will be treated as one role.

R-23: If no Role is specified in the X.509v3 certificate, the mbaps server **MUST** provide a NULL role to the AuthZ algorithm.

R-24: The mbaps AuthZ Algorithm **MUST** be defined and provided by the device vendor.

R-25: The Roles-to-Rights Rules Database design, both syntax and semantics, **MUST** be defined by the device vendor.

R-26: The Roles-to-Rights Rules Database for a particular application **MUST** be configured according to the device vendor’s design, and provisioned in the mbaps Server by the end user.

R-27: The Roles-to-Rights Rules Database for a particular application **MUST** be configurable by the end user.

R-28: The Roles-to-Rights Rules Database for a particular application **MUST NOT** have hardcoded default roles that are unchangeable.

R-29: The Role values used in the x.509v3 client domain certificates **MUST** be consistent with the device vendor’s design of the Roles-to-Rights Rules Database.

R-30: The mbaps server **MUST** extract the client Role from the received x.509v3 client domain certificate.

R-31: If the mbap protocol handler for authorization rejects a request it **MUST** use the exception code 01 – Illegal function code.

9 System Dependencies

To participate in a solution architecture, mbaps devices are dependent on the certificate management services of a Public Key Infrastructure (PKI). The details are not materially important to the implementation of the mbaps server or client behaviour.

10 TLS Requirements

10.1 TLS Version

R-32: mbaps devices **MUST** provide TLS v1.2 or better.

R-33: mbaps Devices **MUST** conform to the requirements of [RFC5246].

R-34: mbaps devices **MUST NOT** negotiate down to TLS v1.1, TLS v1.0, or SSL V3.0.

R-35: mbaps devices **MUST NOT** negotiate the use SSL v2.0 and SSL v1.0 in conformance with [RFC6176].

10.2 TLS v1.2 Cryptography

10.2.1 General

R-36: mbaps Devices **SHOULD** provide a counter mode cipher suite.

Counter mode cipher suites include

TLS_RSA_WITH_AES_128_GCM_SHA256, {0x00, 0x9C}
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, {0xC0, 0x2B}

R-37: mbaps Devices **MUST NOT** negotiate the cipher suite, TLS_NULL_WITH_NULL_NULL

R-38: Any cipher suite used by mbaps Devices and negotiated in a TLS Handshake Protocol exchange **MUST** be listed at IANA's TLS Cipher Suite Registry in the [TLS-PARAMS].

10.2.2 TLS Key Exchange

R-39: mbaps Devices **MUST** provide TLS Client-Server key exchange based-on RSA technology as specified by the mandatory cipher suite and described in [RFC 5246].

R-40: mbaps Devices **SHOULD** provide TLS Client-Server key exchange based on ECC technology.

R-61: mbaps Devices using ECC technology **MUST** support at least P-256 NIST curve.

R-62: mbaps Devices using ECC technology **MUST** support at least the minimum cipher suite of TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

R-63: mbaps Devices using ECC technology **MUST** specify the curves used in their ClientHello using the Supported Elliptic Curves extension in [RFC4492]

R-64: mbaps Devices using ECC technology **MUST** specify the point format used in their ClientHello using the Supported Point Format extension in [RFC4492]

10.2.3 TLS Authentication

Authentication against trust anchors may be done using self signed device certificates. It is recommended to use certificates signed by a Certificate Authority for authentication.

Session resumption using session tickets or resuming session IDs should be supported to reduce the handshake time of a connection.Session resumption using session IDs is preferred. In session resumption it is the responsibility of the server to cache and maintain session information for later use. It is more well supported and places less demand on clients to manage session information with their peer.

Session tickets place the burden of session information on the client. This information is encrypted by the server and transmitted to the client. On new session, this information is transmitted back to the server and used to re-establish a connection. Less server resources are needed to accomplish this but network resources are wasted and due to the transmission of information it takes longer to re-establish a connection.

R-41: mbaps Devices **MUST** support the TLS Client-Server Mutual Authentication Handshake.

R-42: mbaps Device **SHOULD** support the TLS Resumed Session Handshake on Client and Server.

R-43: mbaps Device **MAY** support the TLS Session Ticket resumption on Client and Server

R-44: mbaps Servers **MUST** reject a TLS Handshake where the Client has not responded to a Client Certificate request with certificate.

R-45: mbaps Devices **SHOULD** provide x.509v3 Certificates signed by a Certificate Authority.

R-46: mbaps Devices **MUST** send the entire certificate chain down to the root CA when sending their certificate

R-47: x.509v3 Certificates provided by mbaps Devices **MUST** conform to the requirements of [RFC5280].

10.2.4 TLS Encryption

R-48: If an mbaps Device is to be used in a scenario where encryption is required, then a cipher suite with the required encryption indicator **MUST** be chosen from the list at IANA's TLS Cipher Suite Registry in the [TLS-PARAMS].

R-49: If an mbaps Device is to be used in a scenario where encryption is not required, then a cipher suite with a NULL bulk encryption indicator **MUST** be chosen from the list at IANA's TLS Cipher Suite Registry in the [TLS-PARAMS].

10.2.5 TLS MAC

R-50: mbaps Devices **MUST NOT** use the HMAC-MD5 hash algorithm.

R-51: mbaps Devices **MUST NOT** use the HMAC-SHA-1 hash algorithm.

R-52: mbaps Devices **MUST** provide the HMAC-SHA-256 hash algorithm.

R-53: mbaps Device **MUST NOT** use a NULL HMAC hash algorithm

10.2.6 TLS PRF

R-54: mbaps Devices **MUST NOT** provide the HMAC-SHA-1 hash algorithm for use in the PRF function to calculate the key block as defined in [RFC5246] sections 5, 6.3 and 8.1.

R-55: mbaps Devices **MUST** provide the HMAC-SHA-256 hash algorithm for use in the PRF function to calculate the key block as defined in [RFC5246] sections 5, 6.3 and 8.1.

10.2.7 TLS Cryptography Import/Export Policy

R-56: As early as possible in their development cycle, mbaps devices **MUST** determine that they comply with the import/export conformance policies of their respective countries for the cryptography they provide.

10.3 TLS Fragmentation

R-57: mbaps devices **MUST** provide the Maximum Fragment Length Negotiation Extension as defined in [RFC6066].

R-58: mbaps devices **MUST** provide the ability to negotiate a Maximum Fragment Length of 2^9 (512) bytes as defined in [RFC6066].

10.4 TLS Compression

R-59: mbaps devices **MUST** set the TLS CompressionMethod field of the ClientHello message to the value of NULL.

10.5 TLS Session Renegotiation

R-60: mbaps devices **MUST** provide the TLS Renegotiation Indication Extension defined in [RFC5746] to provide the secure renegotiation of TLS sessions.

11 APPENDIX A: mbaps Packet Structure

Figure A.1 TLS Transportation of mbap ADU shows the layering of the TLS protocol on TCP. The mbap ADU encapsulated in a TLS Application Protocol Packet. The mbaps protocol which is the mbap protocol transported by TLS is found at TCP port 802.

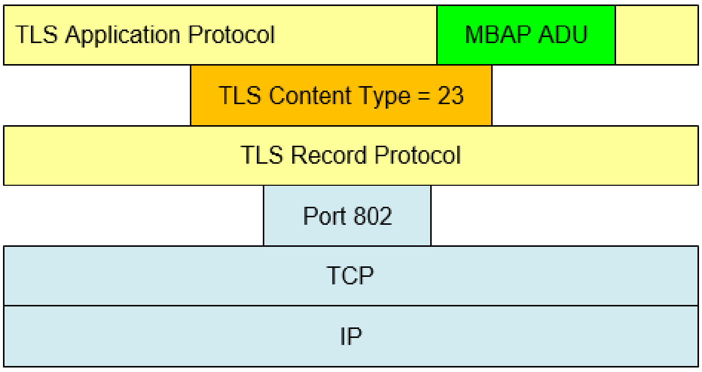

Figure A.1 TLS Transportation of mbap ADU

The structure of the TLS Record Layer used by mbaps is defined in [RFC5246] sec A-1, where:

- ContentType type = 23, Application Protocol
- ProtocolVersion version = {3,3} for TLS v1.2
- uint16 length = number of bytes of the following TLSCiphertext.fragment,
  MUST NOT exceed 16384 + 2048 (18432)
- fragment = The encrypted form of TLSCompressed.Fragment, with the MAC

```
struct {
      ContentType type;
      ProtocolVersion version;
      uint16 length;
      select (SecurityParameters.cipher_type) {
         case stream: GenericStreamCipher;
         case block:  GenericBlockCipher;
         case aead:   GenericAEADCipher;
      } fragment;
   } TLSCiphertext;
```

Figure A.2 TLS Record Layer Structure

For block ciphers such as AES, the fragment type is GenericBlockCipher. As defined in section 10.4 TLS Compression, the CompressionMethod is set to NULL. Consequently, TLSCompressed.length is the same as the uncompressed fragment length.

```
struct {
          opaque IV[SecurityParameters.record_iv_length];
          block-ciphered struct {
          opaque content[TLSCompressed.length];
          opaque MAC[SecurityParameters.mac_length];
          uint8
          padding[GenericBlockCipher.padding_length];
          uint8 padding_length;
          };
} GenericBlockCipher;
```

mbap ADU

Figure A.3 TLS Generic Block Cipher

The content element of the Generic Block Structure is the mbap ADU.