

ANNOUNCEMENT OF MIDDLEBOX SECURITY PROTOCOL (MSP) DRAFT PARTS

ETSI TC CYBER announces the release of two draft parts of an important new cyber security technical specification. These first two parts of a Technical Specification called the Middlebox Security Protocol address one of the most difficult security challenges today: how to enable network operators and end-users to cooperate in managing encryption security for their applications.

An exponential increase in the use of encrypted traffic is occurring at the same time as network cyber security requirements are resulting in massive numbers of intelligent systems in network infrastructures known as “middleboxes.” In order to function for dozens of different essential needs, including cyber security, middleboxes need to understand the traffic being transported through the network to end-users.

The Middlebox Security Protocol enables the existence of a “smart proxy” where end-users can be potentially aware of a middlebox in their traffic stream (visibility) and control what that middlebox sees for different purposes (observability). The result allows for balancing privacy, network operations, and security for different applications. With the Protocol, both users and providers gain the ability to grant or restrict the permissions for visibility and observability.

Part 1 of the Middlebox Security Protocol specification defines the generic capabilities and security requirements. Additional parts define specific implementations in the form of profiles for different use cases that can be mapped to the Part 1 requirements. Part 2 provides a common profile for widespread network use known in the research community as mcTLS. Included with Part 2 are a patch for a known vulnerability as well as an exemplar of use by Mobile Network Operators. Other profiles will be released over the coming months – especially one for data centre access control to meet the critical needs of enterprise network communities.

These initial two draft specifications are relatively complete and stable, and derived from best-of-breed solutions drawn from extensive surveys and evaluation of the considerable published technical literature. However, this standards work is new, complex, and unique. The specifications will remain draft for a period during which widespread industry and public comments and views are sought. In addition, TC CYBER is proactively sending the drafts to other industry standards bodies as well as holding a [Hot Middlebox Workshop](#) (12 June 2018) and [Hackathon](#) (12-13 June 2018), in Sophia-Antipolis France, where the coding community can seek to implement and hack a test implementation of Part 2.

Comments may be sent until 30 June 2018 to: cybersupport@etsi.org using the template for comments:

https://docbox.etsi.org/CYBER/CYBER/Open/Latest_Drafts/Template-for-comments.doc



CYBER; Middlebox Security Protocol; Part 1: Profile Capability Requirements

DRAFT PUBLICLY AVAILABLE UNTIL 30 JUNE 2018

Send comments ONLY to CYBERSupport@etsi.org

Download the template for comments:

<https://docbox.etsi.org/CYBER/CYBER/Open/Latest Drafts/Template-for-comments.doc>

CAUTION: This **DRAFT document** is provided for information and is for future development work within the ETSI Technical Committee CYBER only. ETSI and its Members accept no liability for any further use/implementation of this Specification.

Approved and published specifications and reports shall be obtained exclusively via the ETSI Documentation Service at

<http://www.etsi.org/standards-search>

Reference

DTS/CYBER-0027-1

Keywords

Cyber security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
<https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>.

If you find errors in the present document, please send your comment to one of the following services:
<https://portal.etsi.org/People/CommiteeSupportStaff.aspx>

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2018
All rights reserved.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members.
3GPP™ and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.
oneM2M logo is protected for the benefit of its Members.
GSM® and the GSM logo are trademarks registered and owned by the GSM Association.

Contents

Contents	3
Intellectual Property Rights	5
Foreword.....	5
Modal verbs terminology	5
Executive summary	5
1 Scope	7
2 References	7
2.1 Normative references	7
2.2 Informative references	7
3 Definitions, symbols and abbreviations	7
3.1 Definitions	7
3.2 Symbols	9
3.3 Abbreviations.....	9
4 MSP reference model	10
4.1 Introduction.....	10
4.2 Reference model	10
4.3 Messaging.....	11
5 MSP reference model implementation properties	12
5.1 Granularity controls of middlebox observability and visibility	12
5.2 MSP Security Considerations	14
5.3 MSP Design Abstraction	14
5.3.1 Abstraction Stages Overview	14
5.3.2 Abstraction Stages.....	19
5.3.2.1 Stage 0 - Pre-requisites	19
5.3.2.1.1 Pre-requisite overview	19
5.3.2.1.2 Pre-requisite Data.....	19
5.3.2.1.2 Primitive Operations	20
5.3.2.1.3 Primitive Operation Requirements.....	21
5.3.2.1.4 Pre-requisite Composite Operations	22
5.3.2.1.5 Pre-requisite Composite Operation Requirements	24
5.3.2.2 Stage 1 - Hello stage.....	25
5.3.2.3 Stage 2 – Endpoint Key Exchange	26
5.3.2.4 Stage 3 – Middlebox Key Exchange	27
5.3.2.5 Stage 4 – Multiparty Exchange.....	28
5.3.2.6 Stage 5 – Exchange Finish.....	28
5.3.2.7 Stage 6 – Renegotiation Hello (optional)	29
5.3.2.8 Stage 7 – Renegotiation Endpoint Exchange (optional)	29
5.3.2.9 Stage 8 – Renegotiation Middlebox Exchange (optional)	29
5.3.2.10 Stage 9 – Renegotiation Multiparty Exchange (optional).....	29
5.3.2.11 Stage 10 – Renegotiation Finish (optional)	29
5.4 Conformance and compatibility.....	29
Annex A (informative): Use Cases of MSP	31
A.1 Introduction and history	31
A.2 New infrastructure, services, and innovation use cases	32
A.3 System and user security use cases	32
A.4 Performance use cases	32
A.5 Operational use cases	33
A.6 Compliance obligation use cases	33
A.7 Enterprise Network and Data Centre Use Cases [i.2]	36
Annex B (informative): Exemplar of Mapping a Protocol to MSP Profile	37

Annex C (informative): Exemplar of satisfying MSP Security Characteristics	38
Annex D: Bibliography (Informative)	40
D.1 Published references relating to mcTLS and mbTLS	40
History	43

Draft

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: "*Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards*", which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<https://ipr.etsi.org>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Cyber Security (CYBER).

The present document is the first part of a multi-part deliverable covering the Middlebox Security Protocol.

- Part 1:** **“Profile Capability Requirements”;**
- Part 2: “Transport layer MSP, profile for fine grained access control”;
- Part 3: “Transport layer MSP, Profile for data centre access control”;
- Part 4: “Network layer MSP, Profile for fine grained access control”.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the [ETSI Drafting Rules](#) (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

Traditional Man-In-The-Middle encryption proxies are limited in their applicability because they break both authentication and end-to-end encryption; this is unsatisfactory for a lot of use-cases. The next generation of “smart proxies” will allow greater applicability by only altering the security model or infringing privacy to the minimum extent necessary to fulfil the use-case. They allow proxies access to the parts of the data that they need, control whether the data can be modified or not, give the client and server visibility of what they are doing, and protect proxies from malicious clients or servers if appropriate. Smart proxies are beneficial at the server and content-edge for compression proxies, content delivery, load balancing, compliance, troubleshooting and more. They are beneficial at the client in the enterprise or domestic network, for cyber security monitoring, personal data protection, IP filtering, compliance and more. Smart proxies enable overall system security to improve compared to usage of a traditional MITM proxy, and enable new scenarios to be supported that were traditionally unavailable.

The present document, Part 1, defines the generic capabilities and security properties of a Middlebox Security Protocol. Additional technical specification parts define specific protocol implementations in the form of profiles for different use

cases which can be mapped to the MSP protocol abstraction defined in this part, thereby demonstrating the security properties that these protocols provide.

Included in this part of the specification are the MSP reference model and the necessary properties for implementations, including security considerations, and conformance and compatibility considerations. Exemplars are provided as guidance. See Annexes B and C. Any profile meeting the MSP properties can be regarded as an instantiation of MSP. This approach usefully allows multiple instantiations of MSP. They include an abstraction of the cryptographic functionality that allows a proxy to be end-controlled, fine-grained, read or write. They include an interface between client devices/server devices and a smart proxy to allow for the addition and identification of such proxies and for those proxies to provide assurance of the level of service of the overall end-to-end communications pathway to both endpoints.

This Middlebox Security Specification enables two important capabilities essential for a broad array of use cases. One of the capabilities is “visibility,” that is, the ability to “see” middleboxes. The second capability is “observability,” that is the ability of middleboxes to “see” traffic and/or metadata. What is unique and important about MSP is the manner in which those capabilities are implemented – which gives both users and providers the ability to grant or restrict the permissions for visibility and observability. The controls allow both for an array of different granularities, as well as different controlling party arrangements. This compelling value proposition allows a user to be aware of service provider or enterprise middleboxes for defence, or lack thereof. Where appropriate, it is also important for privacy, as it will show whether the provider actually enhances privacy using the MSP capabilities. For example, if a web server asks to fill in personal information and it is not protected by a MSP enabled middlebox for data loss prevention and intrusion detection, the session is untrusted for that personal data. At present, there is no way for a user, host server, or even another middlebox to determine middlebox cyber-defence posture.

While it is common for middleboxes to act directly on passing traffic, it is equally valid for a middlebox to store traffic and act upon it later, and MSP supports both cases. This allows for diagnostics and cyber incident response to identify affected machines for remedial action.

This protocol is suitable for use in the use cases identified in [i.4] and includes those relevant to “critical” organisations implementing the EU NIS Directive [i.3], and thus acts as a guide as to how components of the NIS Directive, in which privacy may be infringed, can be most appropriately met. The informative Annex contains an extensive list of the use cases in order to improve applicability and adoption.

1 Scope

The present document specifies capability requirements for protocols to enable secure communication sessions between network endpoints and one or more middleboxes between them using encryption and authentication of the identity of the middleboxes so that a level of trust may be assigned by the endpoints. The present document is intended to facilitate implementation profiles for a wide array of implementations and applications.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at <https://docbox.etsi.org/Reference>.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

None

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

- [i.1] SIGCOMM '15, Naylor et al., "Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS", August 17 - 21, 2015, London, United Kingdom.
<http://conferences.sigcomm.org/sigcomm/2015/pdf/papers/p199.pdf>
- [i.2] S. Fenter, "Why Enterprises Need Out-of-Band TLS Decryption, draft-fenter-tls-decryption-00", IETF, 2018.
- [i.3] ETSI TR 103 456: "Implementation of the Network and Information Security (NIS) Directive".
- [i.4] ETSI TR 103 421: "Network Gateway Cyber Defence".
- [i.5] CIS, CIS Controls, Version 7.0, 2018. <https://www.cisecurity.org/controls/>. Ref. ETSI, TR 103305-1, "Critical Security Controls for Effective Cyber Defence; Part 1: The Critical Security Controls."

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

1-sided control: a MSP granularity control enabled unilaterally by one end-point, e.g., a client, server, or other entity.

2-sided control: a MSP granularity control enabled by agreement by two end-points, e.g., both client and server.

Black Key: A key, or a value from which a key can be derived, that can be transmitted in plain form (where it may be subject to manipulation) without compromising security of the protocol.

NOTE: Public keys, encrypted private keys, and encrypted secret keys are examples, when combined with an integrity mechanism.

Fine-grained: The ability to grant or restrict an array of permissions for middlebox observability capabilities. Data is separated into different contexts to allow different access permissions to be granted. Compare with Single-context.

Granularity: Ability to grant or restrict MSP permissions for minimum necessary for the desired middlebox operation.

Handle: A non-sensitive value that has no meaning to the function holding it but that can be used as an input by another process to look up, recover or derive a value including sensitive values. A handle shall be assumed not to have any meaning outside the current session.

In-band: A MSP control capability occurring via the network traffic stream (i.e., the same protocol layer and connection). Compare with Out-of-band. Ref. MB-1 interface in Fig. 4.1.

Identifier (key): A non-sensitive value that has no meaning to the function holding it but that can be used as an input by another process to look up, recover or derive a value including sensitive values. An identifier may have a meaning to other parties and may be used over multiple sessions.

Middlebox: Any device or process, physical or virtual, in the transport path of communication traffic between network end-points other than a transparent switch.

NOTE: For the purposes of the present document, a middlebox is one which is capable of implementing Middlebox Security Protocol profiles in accordance with the present document. Middleboxes can act directly on passing communication traffic or can store traffic and act later.

Middlebox Audit: Ability for a middlebox to be audited using MSP.

Middlebox Access: Availability for a middlebox to be accessed using MSP.

Middlebox Visibility: Ability for a middlebox to be discovered using MSP.

Nonce (Public): A non-secret value used once only (once meaning one exchange, which involve multiple related steps).

Observability: The degree of observation of network traffic and patterns at a Middlebox using MSP, i.e., what, by whom, what granularity, or concealed.

Out-of-band: A MSP control capability occurring separate from the network traffic stream. Compare with In-Band. Ref. MB-2 interface in Fig. 4.1.

Public Key: A key for an asymmetric algorithm, knowledge of which by an adversary does not compromise the security properties of the asymmetric algorithm.

Private Key: A key for an asymmetric algorithm, compromise of which would compromise the security properties of the algorithm

Red Key: A key, or value from which a key can be derived, that is not transmitted in plain form, as doing so would compromise the security of the protocol.

NOTE: In addition to private and secret keys, public keys without any integrity protection are considered red keys if their substitution would undermine security.

Secret Key: A key for a symmetric algorithm that remains known only by the parties required to use it.

Secret Nonce: A sensitive value that is only used once.

Session: A sequence of exchanges of data that are triggered from a higher level protocol opening the communication up until the higher level protocol terminates the communication or receives notification of termination (e.g. communication failure).

Single-context: The ability to grant or restrict permissions for middlebox observability capability, data is not separated into different contexts where different permissions can be granted. Compare with Fine-grained.

Visibility: The discoverability of the existence and attributes of a middlebox by the client, server, or other middleboxes.

3.2 Symbols

For the purposes of the present document, the following symbols apply:

None

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

5G	5th Generation
AES-GCM	Advanced Encryption Standard - Galois Counter Mode
ASIC	Application-Specific Integrated Circuit
BYOD	Bring Your Own Device
CIS	Center for Internet Security
CMAC	Cipher-based message authentication code
DDoS	Distributed Denial of Service
DH	Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
HMAC	Hash-Based Message Authentication Code
HTTP2	Hypertext Transfer Protocol Version 2
IDM	Middlebox Identity
IKC	Client Identity Key
IKEv2	Internet Key Exchange Version 2
IKM	Middlebox Identity Key
IKS	Server Identity Key
IP	Internet Protocol
IPSec	Internet Protocol security
ISG	Industry Specification Group
IoT	Internet of Things
IvKC	Identity Verification Key - client
IvKM	Identity Verification Key - middlebox
IvKS	Identity Verification Key - server
KDF	Master Secret Key
KEK	Key Encryption Key
KHASH	Key Hash
LFSR	Linear-feedback shift register
MAC	Message Authentication Code
MITM	Man in the Middle
NAT	Network Address Translation
NFV	Network Functions Virtualisation
NIS	Network and Information Security
OTT	Over the Top
PRF	Pseudorandom Function
RNG	Random Number Generator
RSA	Rivest-Shamir-Adleman
SDN	Software Defined Network
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
TLMSP	Transport Layer Middlebox Security Protocol
TLS	Transport Layer Security
VNF	Virtualised Network Functions
WAN	Wide Area Network

4 MSP reference model

4.1 Introduction

Middleboxes encompass an enormous number of functional physical and virtual equipment components that exist in the complex paths typically found between communication endpoints. Middleboxes are essential to the operation of all telecommunication and ICT networks today, and large infrastructures will typically have thousands of ubiquitously deployed middleboxes. See Annex A. By almost any metric, middleboxes also represent perhaps the most active and innovative sector of network technology, research, and product development today. See Annex D.

Middleboxes are especially important in managing bandwidth-limited transport paths such as many access networks as well as the delivery of many content delivery services such as streaming High Definition video – by prefetching and re-encoding by the local access operator. Middleboxes are also essential in enabling meaningful cyber security capabilities, and other compliance obligations of all kinds faced by all network operators. See Annex A.6. They are also essential for regulated industries such as the financial sector in meeting their transparency and audit obligations. See Annex A.7. Because of these requirements, the use of Middlebox Security Protocols contained in the present Technical Specifications, were recommended to the European Union for implementation of the Network and Information Services (NIS) Directive. [i.3]

In order to support the many use-cases set forth in Annex A, because increasing amounts of network traffic are encrypted, middleboxes today are used to achieve some level of “observability” of encrypted traffic and metadata at their location in the traffic transport path or a data centre, and modify the traffic in a manner that may be known or unknown to the end users – with or without their approval. [i.4] For example, the most common example is known as Network Address Translation (NAT) – which for many reasons occurs at boundaries between networks. Middleboxes are also essential in enabling meaningful cyber security capabilities, and other compliance obligations of all kinds faced by all network operators. See Annex A.6, [i.5].

The many different middlebox use cases give rise to very different levels of visibility and observability requirements that are enabled using the granularity controls contained in the various parts of this Technical Specification. The MSP reference models provide for mechanisms for a wide range of context-dependent controls. In most contexts, both providers and users should find MSP implementations a more secure and privacy enhancing alternative than proprietary solutions that enable a wide variety of Man-in-the-Middle (MITM) capabilities that are unknown and with no granularity options.

4.2 Reference model

Achieving the desired granularity for visibility and observability using middlebox controls is a challenge that need to meet many complex, interacting requirements. The general requirements include:

- Secure and controlled exposure of traffic observables
- Sufficient observable information for acquisition and analysis for defence measures
- Ability to institute desired defence measures
- Suitability for architecture location (transport path versus data centre)
- Ability to scale
- Ease of deployment
- Minimal impact on existing client and server implementations
- Minimal threat exposure
- Sufficient performance metrics

The Middlebox Security Protocol Reference Model and requirements are described in this Technical Specification Part. Subsequent parts of the specification then implement this Reference Model and requirements as Profiles contained in those Parts.

There are two distinct MSP Reference Model implementation profile types as shown in Fig. 4.1, below. One profile type is intended for use at any arbitrary point in a network transport path, the other for use at the gateway to a data centre or enterprise network. The MSP controls for the network-based model can occur using interfaces and means either in-band or out-of-band. The MSP observability and granularity controls for the data centre-based model can also occur either in-band or out-of-band.

Both models are messaging, media, service, and encryption protocol agnostic – allowing for a broad array of different profiles for visibility and observability to be specified for transport, network, or application layers. The control

messaging for in-band implementations will have dependencies on the traffic transport and network protocols being used. Out-of-band control messaging have no transport or network protocol dependencies.

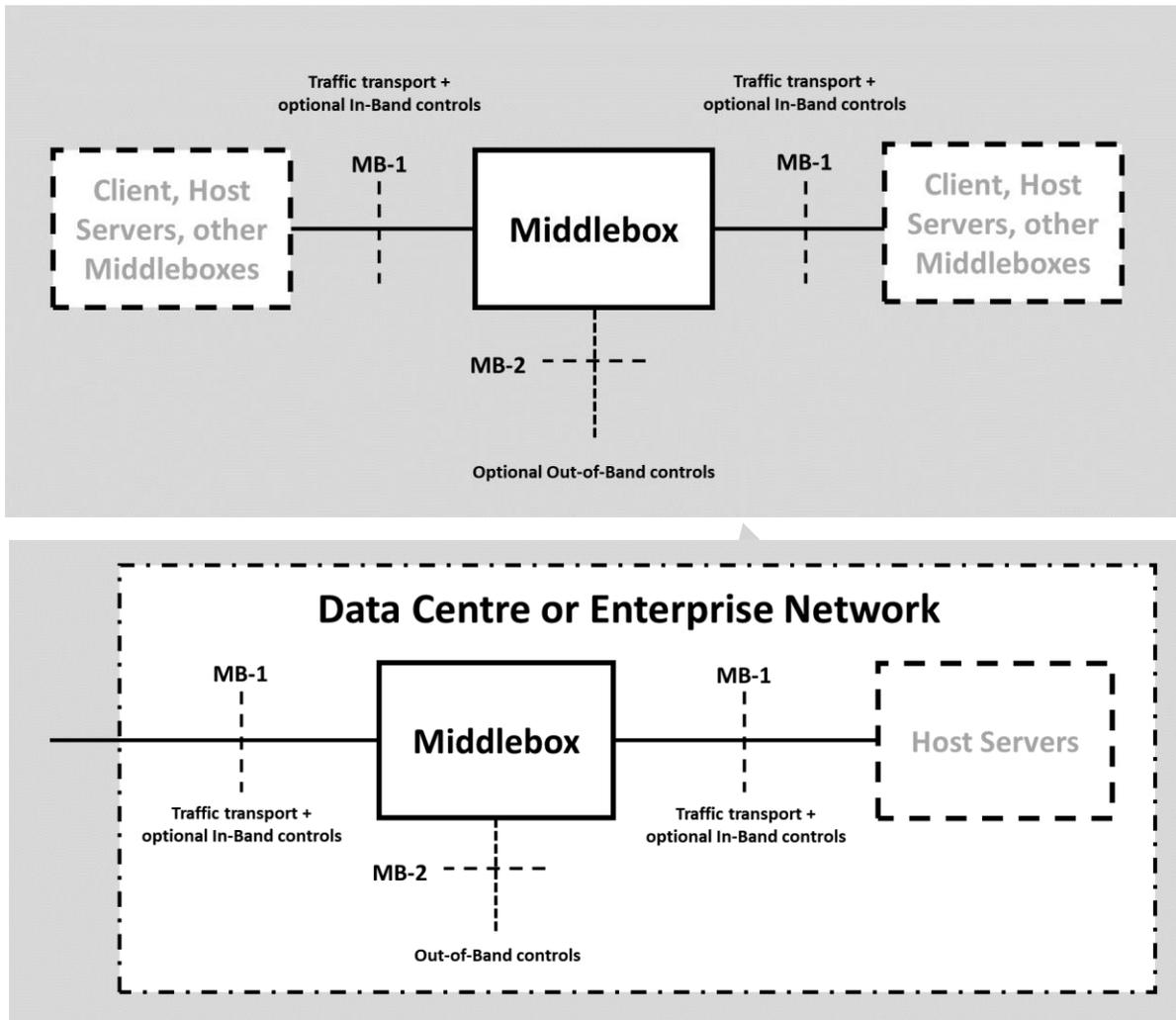


Figure 4.1: Network based and Data Centre based MSP Reference Model profile types

The Middlebox reference interfaces MB-1 and MB-2 are depicted in the model to differentiate between the use of in-band, and out-of-band controls. An in-band MSP control capability occurs via the network traffic stream (i.e., the same protocol layer and connection). An out-of-band MSP control capability occurs separate from the network traffic stream.

The reference model implementations, as described in Clause 5, below, allows for two different control classes that distinguish how the control negotiation occurs: 1-sided (where either a client or server can agree on the existence and control of the middlebox), and 2-sided (where both the client and server agree).

4.3 Messaging

No specific messaging protocol is required to implement MSP, and it is left to the implementing profiles to specify. However, the messaging protocol should meet security and functionality requirements for the implementation.

5 MSP reference model implementation properties

5.1 Granularity controls of middlebox observability and visibility

This clause specifies the granularity controls necessary for MSP implementations. [i.4] Annexes B and C are intended to help protocol designers by abstracting out an exemplar MSP design, showing how it meets the profile, and mapping the security considerations on to the design components.

The control granularity of middlebox visibility and observability is divided into four classes:

- 1-sided, single context
- 2-sided, single context
- 1-sided, fine grained
- 2-sided, fine grained

The 1-sided class only requires that either the client or the server invites a middlebox, whereas the 2-sided class requires that both client and server agree to the presence of the middlebox. MSP enables division of the information being transferred into **fine grained** context parts, where a middlebox will only be able to access the context necessary for the middlebox to perform its intended function, and the rest of the information is withheld from the middlebox. However, a special case is support for a **single context**.

The following three tables lists the requirements and how the requirements apply to the four classes described above.

Note: Throughout the tables below, 'unauthorised change' means any change of greater privilege than has been granted, so: in the case of network adversaries with no privilege then unauthorised change is any change; in the case of a middlebox with read-only permission to a portion of the content, then an unauthorised change is any change to a read-only portion of the content or a change to the audit information inferred by the destination endpoint or a subsequent middlebox if there is any audit information; in the case of a middlebox with write permission to a portion of content, then an unauthorised change to that portion of content is a change to the audit information inferred by the destination endpoint or a subsequent middlebox if there is any audit information

Key: R = Required, O = Optional, NS = Not Supported

Table 5.1: Audit requirements

Ref	Audit Requirement	1-sided single context	2-sided single context	1-sided fine grained	2-sided fine grained
A1	Destination endpoint shall be able to detect if an unauthorised change to the data has occurred	R	R	R	R
A2	Middleboxes with read-only access shall be able to detect if a network adversary has made a change to the data	R	R	R	R
A3	Middleboxes with read-only access shall be able to detect if a middlebox or endpoint has made a change to the data in excess of privileges	O	O	O	O
A4	Middleboxes with permission to modify content shall be able to detect if an unauthorised change to the data has occurred	R	R	R	R
A5	Middleboxes modifying content shall validate that no unauthorised changes have occurred prior to receipt.	R	R	R	R
A6	Destination endpoints shall be able to determine the middlebox or endpoint responsible for the most recent authorised change	O	O	O	O
A7	Middleboxes shall be able to determine the middlebox or endpoint responsible for the most recent authorised change	O	O	O	O
A8	Destination endpoint shall be able to determine all middleboxes that have performed authorised changes	O	O	O	O
A9	Destination endpoint shall be able to determine all authorised changes that have occurred and the parties responsible	O	O	O	O
A10	Destination endpoint shall be able to determine which middleboxes have inspected content	O	O	O	O

A11	Middleboxes shall be able to determine all middleboxes that have performed authorised changes	O	O	O	O
A12	Middleboxes shall be able to determine all authorised changes that have occurred and the parties responsible	O	O	O	O
A13	Middleboxes shall be able to determine all middleboxes that have inspected content	O	O	O	O
A14	Where a middlebox can detect unauthorised changes, the protocol shall specify action to be taken by middleboxes on receipt of content with unauthorised changes.	R	R	R	R

[Editorial notes on Audit table:

A3 - This needs to remain optional. If a middlebox with read only access modifies the data, it can still produce a valid read mac. Therefore, another middlebox with read access cannot detect this change.

A7, A11, A12, A13 - Part 2 does not achieve this, it would need verifiable middlebox signatures or a unique key exchange between every pair of middleboxes, without such a key or signature then other middleboxes can 'trick' this middlebox. We are not aware of practical protocols that would achieve this hence these requirements are optional. The goal of Part 1 is to cover all possible MSPs, not just Part 2. It may be that there is another protocol and use case where this is a requirement and this would then go in the compliance matrix. Another reason to have these requirements is to be clear on what each protocol does not give you.]

Table 5.2: Access requirements

Ref	Access Requirement	1-sided single context	2-sided single context	1-sided fine grained	2-sided fine grained
P1	Client and server shall mutually agree to grant middlebox access	O	R	O	R
P2	Client or server, alone, shall be able to grant access	R	NS	R	NS
P3	The protocol shall provide mechanisms for fine grained control of access to portions content	NS	NS	R	R
P4	The protocol shall provide mechanisms different permissions for access to content	O	O	O	O

[Editorial notes on Access table:

The application of the access and visibility requirements in Tables 5.2 and 5.3 necessitates consideration of two leading candidate profiles for Part 3, "Transport layer MSP, Profile for data centre access control" – which are referred to as "Green" (IETF, Data Center use of Static Diffie-Hellman in TLS 1.3, draft-green-tls-static-dh-in-tls13-01) and "RHRD" (IETF, TLS 1.3 Option for Negotiation of Visibility in the Datacenter, draft-rhrd-tls13-visibility-01)

P1 - In 1-sided case, can we require that the other endpoint know of each middlebox? In Green, they do not. In RHRD, they only do up to the fingerprint (see Visibility comment below). In general, or for a protocol claiming to be a MSP compliant protocol? Should a MSP label be given to protocols that do not support this?]

Table 5.3: Visibility requirements

Ref	Visibility Requirement	1-sided single context	2-sided single context	1-sided fine grained	2-sided fine grained
V1	Client shall learn the identity of all middleboxes	R	R	R	R
V2	Server shall learn the identity of all middleboxes	R	R	R	R
V3	The client and server shall, if requested, receive validation of identity by each middlebox	R	R	R	R
V4	Client shall learn the identity of the server	R	R	R	R
V5	Server shall learn the identity of the client	O	O	O	O
V6	The protocol shall allow for Middleboxes to offer anonymity services to protect clients' identity	O	O	O	O
V7	The protocol shall allow for mutual authentication of client and server	O	O	O	O
V8	If the protocol supports anonymous clients, the protocol shall provide a means for the server to choose to reject anonymous clients	R	R	R	R

[Editorial notes on Visibility table:

V1 - In RHRD the client learns the fingerprint (first 20 bytes of the sha-256 hash) of the middlebox public key SSWrapDH1. That's the only 'identifying' information there is so the fingerprint is the identity. Identity != protocol address. In Green, the client cannot distinguish it's use because it cannot distinguish between the server offering static DH or ephemeral DH except by comparing the public key share across multiple connections, if the client assumes it is static DH then all they know is that decryption is possible by any middleboxes privileged with access to the static DH private key.

V2 – In RHRD the fingerprint is also the only identifying information which the server has.

V3 - In RHRD the fingerprint as an identity is cryptographically bound to the SSWrapDH1 public key and hence private key and therefore the fingerprint self-validates to those middleboxes authorised to access this private key.]

5.2 MSP Security Considerations

MSP security will be context dependent and vary among the various implementation protocols. In general, the MSP controls should be implemented in a manner to prevent client, server and middlebox eavesdropping, and mitigate any known vulnerabilities to the middlebox itself, such as validation of certificates (particularly of middleboxes), attempting to force to weaker protocol. The security capabilities should cover which keys can be shared/omitted when particular requirements are deemed unnecessary. For example, one MAC key is satisfactory if all middleboxes are given write permission and it is unnecessary to know who, if anyone, has performed an authorised modification.

5.3 MSP Design Abstraction

5.3.1 Abstraction Stages Overview

In order to map any instantiation of a protocol to the MSP profile, therefore demonstrating that it meets the security characteristics specified here, the protocol may be considered as consisting of the following stages:

- Hello Stage
- Endpoint Key Exchange Stage
- Middlebox Key exchange stage
- Multiparty Key Exchange Stage
- Exchange Finish

For some instantiations, multiple stages may be combined into a single message (e.g. the Hello and Endpoint Key Exchange may be the same message) and some stages may be absent or unsupported (e.g. re-negotiation).

A similar set of stages may be supported to allow the resumption of a session or the re-negotiation of permissions for a session; these stages assume that shared secrets already exists. Dependent on which shared secrets are stored, and which permissions are being changed, it is possible that only a subset of these stages will be required for renegotiation (for example, if all shared secrets are stored, a single hello/finish combination message may be all that is required).

- Renegotiation Hello Stage
- Renegotiation Endpoint Key Exchange Stage
- Renegotiation Middlebox Key Exchange stage
- Renegotiation Multiparty Key Exchange Stage
- Renegotiation Exchange Finish

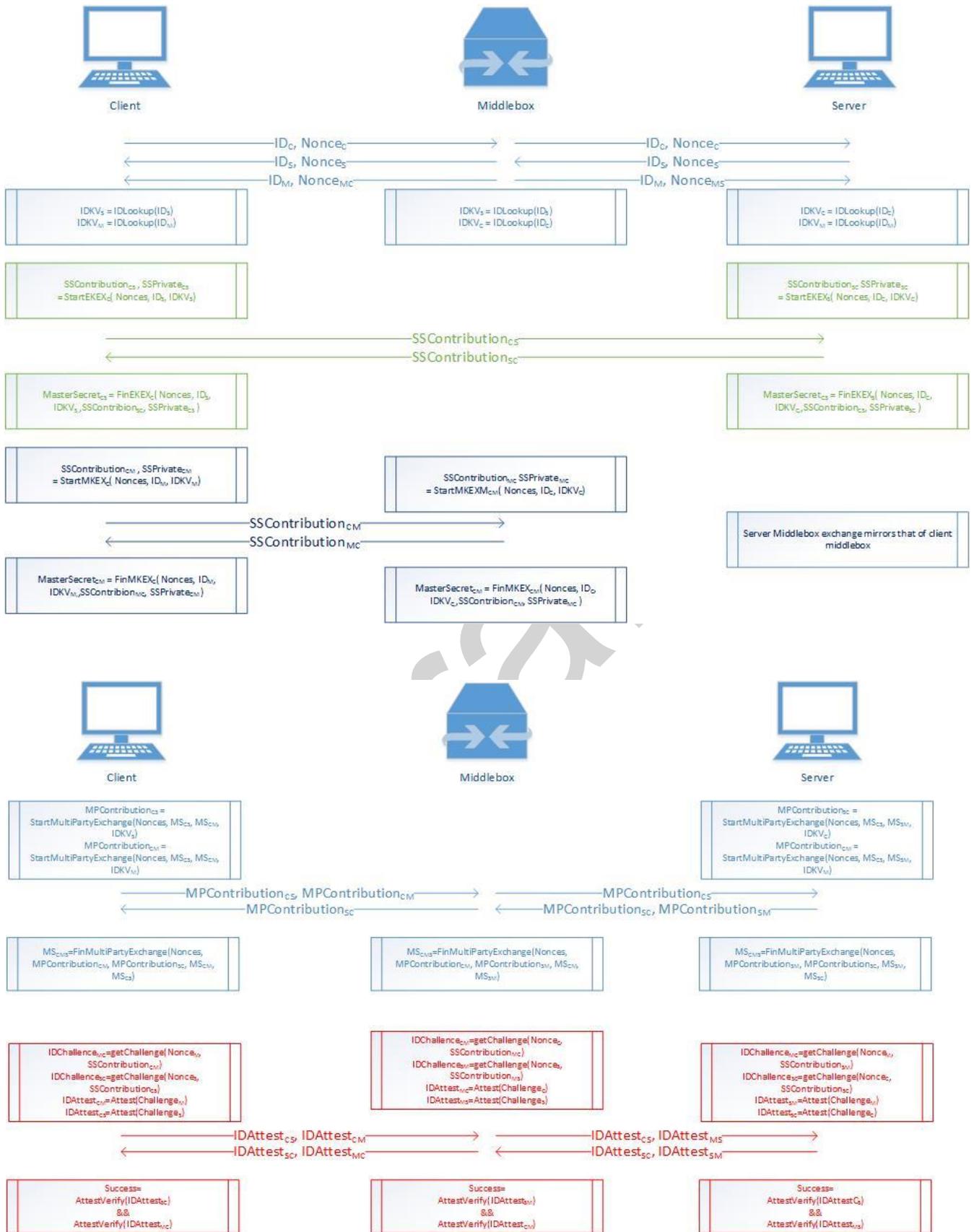


Figure 5.1: Client, middlebox and server negotiation steps

The hello stage is where the endpoints and middleboxes identify themselves to each other and the lookup of the necessary keys to assure each other's identities. The endpoint and middlebox key exchange stages establish shared secrets between the two endpoints and between each endpoint and each middlebox. The multiparty exchange stage, using these shared secrets, establishes the necessary keys for the traffic encryption and access by middleboxes. The finish stage is where the secrets derived are cryptographically tied to the identities of the parties involved, thereby assuring that there is no tampering with the protocol. Some of these stages may happen concurrently in the same message or even be reordered.

The resulting abstract key hierarchy is shown in Figures 5.2 and 5.3 below, using an exemplar mapping to the profile found in Part 2, "Profile for network fine grained TLMSP." The various operations have security requirements that are detailed in the following sections. In some cases, what is described as a single operation in the hierarchy may, in the concrete protocol specification, be satisfied by two different operations operating at different times, one of which may be a null operation. This is permitted as long as the combination satisfies the requirements detailed later in this specification. Equally, it is possible that a single operation may satisfy the security properties required for multiple different operations as described in this abstract specification and therefore may be considered to fulfil multiple operations. An example of this can be seen in the exemplar mapping; the ephemeral public Diffie-Hellman keys generated by the endpoints for the agreement of a shared secret between client and server may be re-used with different nonce values and a different middlebox public/private keypair in the mechanism to generate a shared secret between the middlebox and endpoints. Another example of where mechanisms may be combined lies in the IKEv2 key exchange; the signing of the hash of the shared secret, along with other data, by an endpoint satisfies both the identity attestation, i.e., that the endpoint is the owner of the private key that corresponds to their certificate, and also that the proof of that the endpoint has derived the same shared secret and that the key exchange has not been tampered with.

Draft

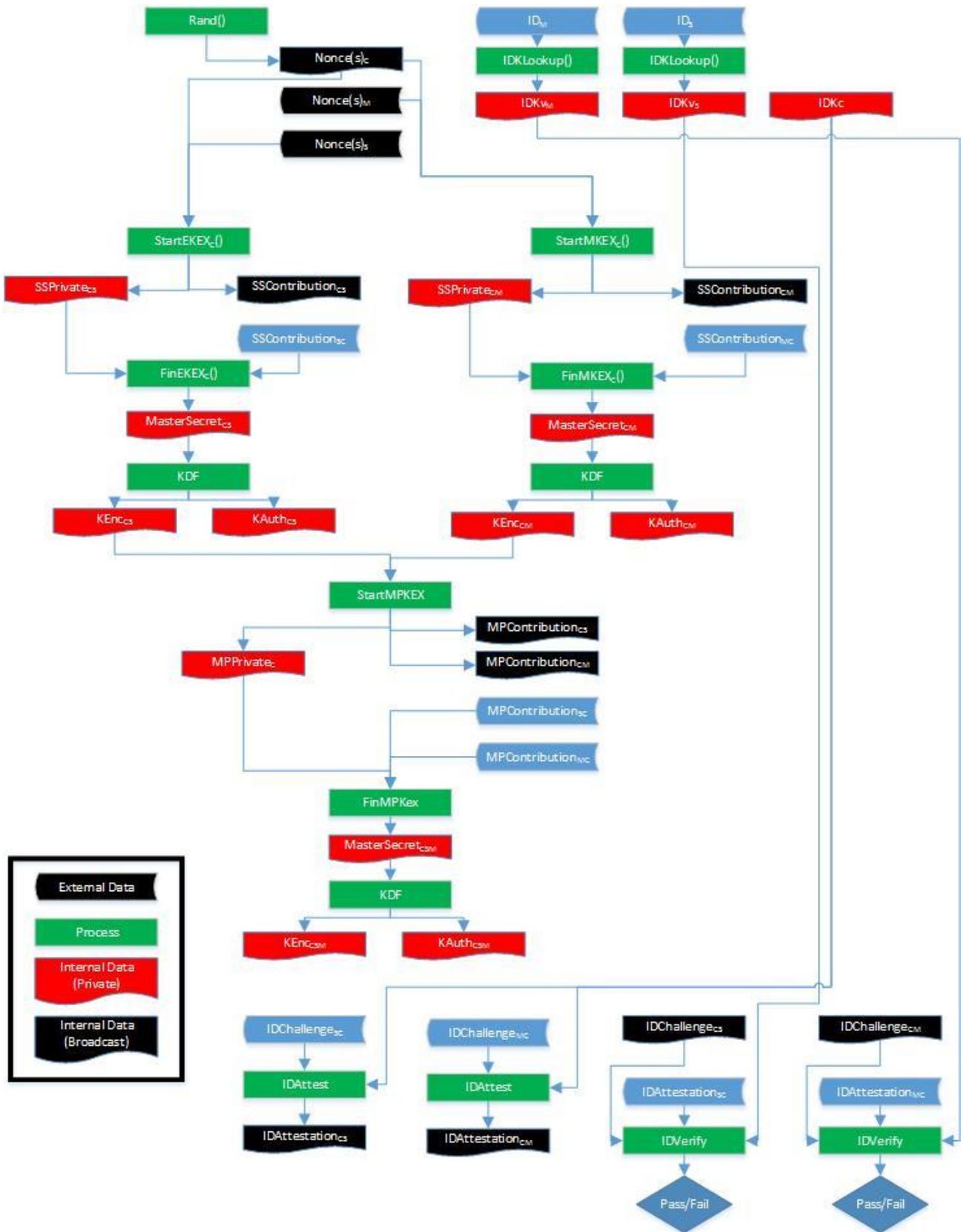


Figure 5.2: Key hierarchy 1

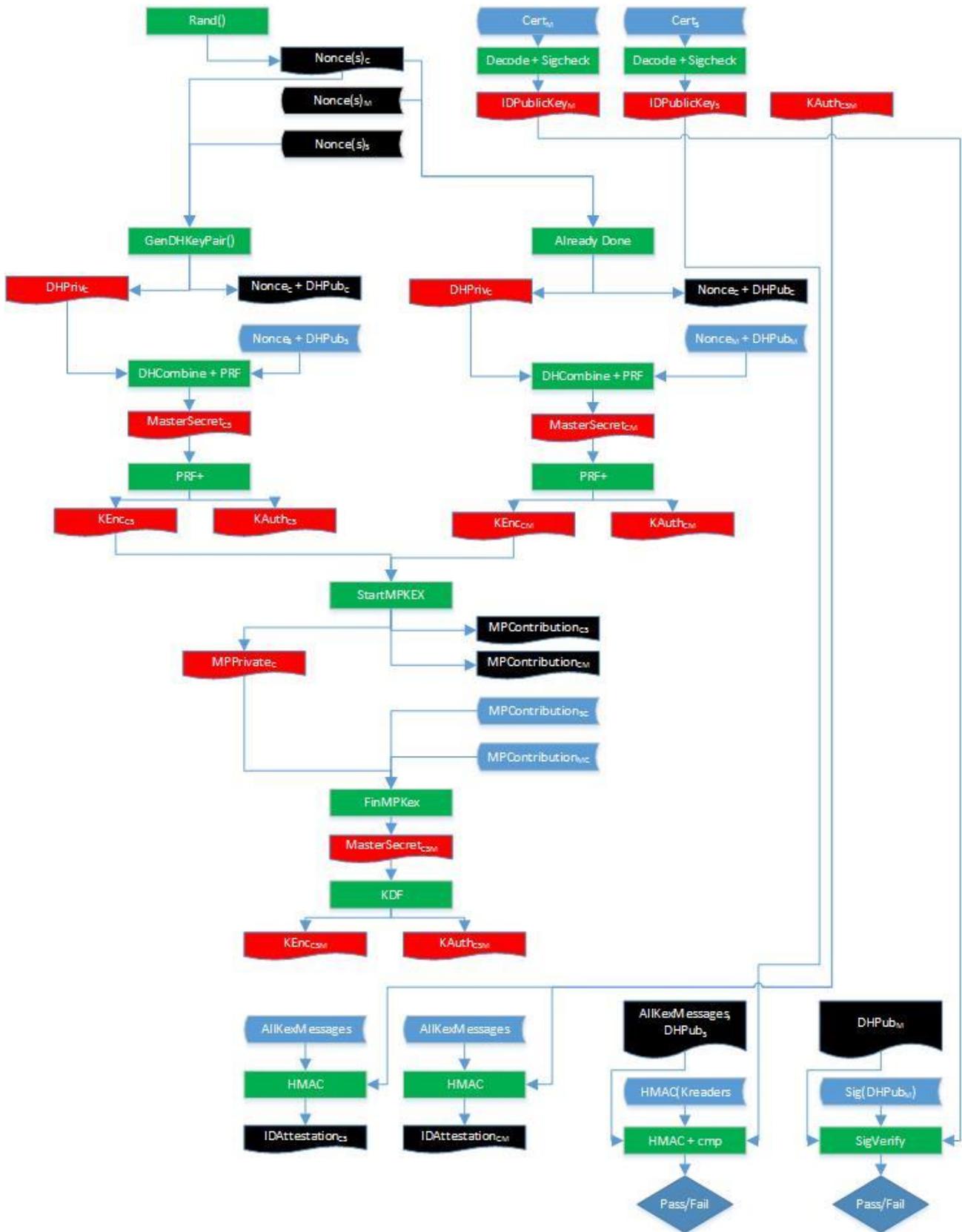


Figure 5.3: Key hierarchy 2

5.3.2 Abstraction Stages

5.3.2.1 Stage 0 - Pre-requisites

5.3.2.1.1 Pre-requisite overview

Any protocol makes assumptions about the pre-placement of certain primitives (be those keys, identifiers, cryptographic operations, etc) that are a pre-requisite to the protocol functioning properly. The list of operations that are required are given here. Some of these may be identity operations (e.g., the conversion of a secret nonce into a secret key may be “just use it as is”) and some may involve third parties (e.g., looking up revocation certificates).

For cryptographic mechanisms, MSP does not attempt to define which operations offer “suitable security” and which do not. For example, for a hash algorithm, it is expected to be “difficult” to find a hash collision. MSP does not try to define which hash algorithms have this property and which do not, other bodies should be consulted for the security level of the cryptographic algorithms.

In order to assist with mapping protocols to cryptographic primitives and assuring security, MSP provides a list of “primitive” operations with requirements that will align to well understood cryptographic operations (and potentially more than one such operation). MSP then provides for “composite operations” where the operation that MSP requires is described and the requirements for it and some methods of assembling such a composite operation from primitive operations, maybe with some added constraints. The requirements of the composite operation are also given.

Any protocol may show compliance with MSP in one of two ways:

- 1) Demonstrating that they are using primitives that meet the requirements listed in the primitive operations section (for which there may be existing acceptance) and then using the constructions listed in the composite section
- 2) Demonstrating that for each stage mapped across that does not meet (1), the composite requirements are otherwise satisfied.

5.3.2.1.2 Pre-requisite Data

Table 5.4: Pre-requisite Data

ID/Symbol	Data Description	Sensitivity	Known By
ID _C	The identity of the client	Black	Client
ID _S	The identity of the server	Black	Server
ID _{M[x]}	The identity of Middlebox X	Black	Middlebox[x]
ADDR _C	The underlying protocol address of the client.	Black	Client
ADDR _S	The underlying protocol address of the server	Black	Server
ADDR _{M[x]}	The underlying protocol address of Middlebox X	Black	Middlebox[x]
IK _C	The Client Identity Key, used by the client to attest his/her identity (this may need to be different to different parties to preserve secrecy).	Red	Client
IvK _C	The Client Identity Verification key. If a symmetric algorithm is used, this is a red pre-shared key and the same as IK _C . If an asymmetric algorithm is used, this is a public key corresponding to IK _C . It is still considered “red” as the use of an incorrect public key breaks security.	Red	Client, Server (possibly middleboxes)
ID(IvK _C)	The identifier by which the client identity verification key can be securely recovered. An example could be a certificate ID or the entire signed certificate.	Black	Client, Server (possibly middleboxes)
IK _S	The Server Secret Identity Key, used by the server to attest his/her identity (this may need to be different to different parties to preserve secrecy).	Red	Server
IvK _S	The Server Identity Verification key. If a symmetric algorithm is used, this is a red pre-shared key and the same as IK _S . If an asymmetric algorithm is used, this is a public key corresponding to IK _S . It is still considered “red” as the use of an incorrect public key breaks security.	Red	Client, Server (possibly middleboxes)

ID{IvK _s }	The identifier by which the server identity verification key can be securely recovered. An example could be a certificate ID or the entire signed certificate.	Black	Client, Server (possibly middleboxes)
IK _{M[X]}	Middlebox X's Secret Identity Key, used by the middlebox to attest his/her identity (this may need to be different to different parties to preserve secrecy).	Red	Middlebox[X]
IvK _{M[X]}	Middlebox X's Identity Verification key. If a symmetric algorithm is used, this is a red pre-shared key and the same as IK _{M[X]} . If an asymmetric algorithm is used, this is a public key corresponding to IK _{M[X]} . It is still considered "red" as the use of an incorrect public key breaks security.	Red	Client, Server
ID{IvK _{M[X]} }	The identifier by which the middlebox X's identity verification key can be securely recovered. An example could be a certificate ID or the entire signed certificate.	Black	Client, Server

5.3.2.1.2 Primitive Operations

Table 5.5: Primitive Operations

ID/Symbol	Data Description and requirements	Input(s)	Output(s)
RNG(bits)	A cryptographically secure source of random bits. May be a pseudo-random number generated appropriately seeded or a hardware noise source with appropriate bias removal.	Size of random block required	Random Block
FastRNG(bits)	A source of a infrequently-repeating values that doesn't have to be cryptographically secure but are infeasible to guess in advance. A RNG() can always be used here if no separate mechanism available.	Size of random block required	Random Block
CTR(S)	Produces a non-repeating value for the duration of the key exchange and session. These may be guessable but have a cycle long enough to guarantee non-repetition (e.g. a suitably large counter, an LFSR with suitably large primitive polynomial).	The current state for this operations	A new value and an updated state.
HASH(m)	A deterministic algorithm that, given an arbitrary input message, produces a fixed-sized block output.	A message – a sequence of bits.	A fixed sized block of bytes determined by m
PKSign(K,D)	An operation using an asymmetric private key that, given input data and the key, produces a non-sensitive proof that the party signing is in possession of the signing key and that the data is the original data signed	A key to "sign" with and the data to be signed	A "signature"
PKVerify(K,D,S)	An operation using an asymmetric public key that, given input data D, signature S and the verification key K, determines whether the data received was data signed by someone in possession of the corresponding signing key.	The verification key, the data and the signature of that data	True or False, depending on whether or not the signature is valid for that key and data.
SKSign(K,D)	An operation using a symmetric key that, given input data and the key, produces a non-sensitive proof that the party signing is in possession of the signing key and that the data is the original data signed	A key to "sign" with and the data to be signed	A "signature"

SKVerify(K,D,S)	An operation using an symmetric public key that, given input data D, signature S and the verification key K, determines whether the data received was data signed by someone in possession of the corresponding signing key.	The verification key, the data and the signature of that data	True or False, depending on whether or not the signature is valid for that key and data.
PKEncrypt(K,P,[IV])	An operation using an asymmetric public key that, given input plaintext data and the key, converts the data into a form, ciphertext, that prevents any knowledge of the plaintext being derived without the key. An IV may be required.	A key to “encrypt” with and the data encrypted	The cipher text
PKDecrypt(K,C,[IV])	An operation using an asymmetric private key that, given input ciphertext C, recovers the original plaintext as long as the correct key K is provided. An IV may be required.	The key to “decrypt” with and the ciphertext to decrypt.	The plain text originally encrypted if the correct key is provided.
SKEncrypt(K,P,[IV])	An operation using a symmetric key that, given input plaintext data and the key, converts the data into a form, ciphertext, that prevents any knowledge of the plaintext being derived without the key. An IV may be required.	A key to “encrypt” with and the data encrypted	The cipher text
SKDecrypt(K,C,[IV])	An operation using a symmetric key that, given input ciphertext C, recovers the original plaintext as long as the correct key K is provided. An IV may be required.	The key to “decrypt” with and the ciphertext to decrypt.	The plain text originally encrypted if the correct key is provided.

For some protocols, it may be that not all of these primitives are required. Some of these primitive operations may be used in multiple places. For example, a hash function may be used in the signing process and also in the PRF process.

Where an operation, primitive or composite, is required in different places there is no requirement that the same actual operation is used. For example, an MSP instantiation may, as the SKEncrypt algorithm, wish to use AES-KeyWrap to encrypt and send keys but AES-GCM to encrypt and send data. This would still satisfy the MSP requirements, at the expense of adding a potentially unnecessary layer of complexity to the protocol.

5.3.2.1.3 Primitive Operation Requirements

These describe the requirements of the primitive operations in MSP. These are not intended to mandate a particular type of operation and some cryptographic operations may satisfy more than one of these requirements.

Table 5.6: Primitive Operation Requirements

Operation	Requirements
RNG(bits)	<ul style="list-style-type: none"> Given bits $b[0] \rightarrow b[n]$ is shall not be possible to derive any information about bit $b[n+k]$ or bit $[-k]$ (for $k > 0$). All possible n-bit sequences shall occur for $b[0] \rightarrow b[n]$ with equal probability <p>Note: This is a cryptographically secure RNG, no matter how much output is observed it shall not be possible to derive a key produced by this RNG any faster than exhausting over valid keys that may be output. It may not be possible in all scenarios for a single mechanism to satisfy this. In this case, RNG may be a mechanism composed of multiple mechanisms to achieve this goal (e.g. the RNG mechanism may produce a single block by taking multiple blocks from a biased noise source that are hashed to produce a single output block).</p>
FastRNG(bits)	<ul style="list-style-type: none"> Infrequent repetition.

	Note: This is intended to produce values that an adversary cannot predict or choose in advance but that need not remain secret from an adversary (e.g. IV values that are transmitted in the clear, Miller-Rabin witnesses for primality testing)
CTR(S)	<ul style="list-style-type: none"> The output block shall not repeat itself within the duration of any exchange and session, e.g. a counter, an LFSR with primitive polynomial.
HASH(m)	<ul style="list-style-type: none"> It shall be infeasible to find two messages, m_1 and m_2 such that $\text{HASH}(m_1)=\text{HASH}(m_2)$ It shall be infeasible to determine m given $\text{HASH}(m)$ (unless m is from a small set of values where the fastest method shall be exhaustive searching) It shall be infeasible to modify a message m without changing the value of $\text{HASH}(m)$
PKSign(Kpriv,D) PKVerify(Kpub,D,S)	<ul style="list-style-type: none"> Given a set of documents DS and valid signatures SS, it shall be infeasible to calculate the signature of a document D not in DS without K_{priv} Given a document D and signature S, it shall be infeasible to modify D without making S invalid. It shall be infeasible to calculate K_{priv} in whole or in part from a set of documents and valid signatures or from K_{pub}. Verify shall always return true if the document has been signed by the private key corresponding to K The probability of a an arbitrary bitstring S being a valid signature for document D shall be negligibly small
SKSign(K,D) SKVerify(K,D,S)	<ul style="list-style-type: none"> Given a set of documents DS and valid signatures SS, it shall be infeasible to calculate the signature of a document D not in DS without K Given a document D and signature S, it shall be infeasible to modify D without making S invalid. It shall be infeasible to calculate K in whole or in part from a set of documents and valid signatures. Verify shall always return true if the document has been signed by K The probability of a an arbitrary bitstring S being a valid signature for document D shall be negligibly small
PKEncrypt(Kpub,P,[IV]) PKDecrypt(Kpriv,C,[IV])	<ul style="list-style-type: none"> Given two plaintext messages for the same length, M_1 and M_2, and ciphertext C which is a valid encryption of either M_1 or M_2, it shall not be possible to determine which of M_1 or M_2 was encrypted without K. It shall not be possible to calculate K_{priv} in whole or in part from a set of matched ciphertext and plaintexts and/or K_{pub} Correlation immunity <p>Note: No assumption is made of non-malleability. Integrity mechanisms are used instead to detect unauthorised changes.</p>
SEncrypt(Kpub,P,[IV]) SKDecrypt(Kpriv,C,[IV])	<ul style="list-style-type: none"> Given two plaintext messages for the same length, M_1 and M_2, and ciphertext C which is a valid encryption of either M_1 or M_2, it shall not be possible to determine which of M_1 or M_2 was encrypted without K_{priv}. It shall not be possible to calculate K in whole or in part from a set of matched ciphertext and plaintexts Correlation immunity <p>Note: No assumption is made of non-malleability. Integrity mechanisms are used instead to detect unauthorised changes.</p>

5.3.2.1.4 Pre-requisite Composite Operations

For many MSP operations, the requirements of that operation may not align up exactly with a single primitive operation. There may be multiple ways to achieve the same result, or a primitive operation may need some additional constraints. For that reason, many of the MSP stages are given as composite operations with their requirements. For these composite operations, examples of combinations of primitive operations that provide the mechanism are also provided but these are not intended to be an exhaustive list.

As an example to demonstrate why this is done, consider the case that Alice wants to know if Bob was the author of a message she received and is requiring Bob to provide proof. Two different ways of achieving this result:

- 1) Bob uses a asymmetric signature algorithm (e.g. ECDSA) with his private key.
- 2) Bob uses a symmetric signature algorithm (e.g. HMAC-SHA256) with a symmetric key known only to him and Alice.

Both achieve the desired result. One is the primitive operation PKSign, the other is the primitive operation SKSign with the added constraint that the key is unique to Alice and Bob.

Some of these operations may take additional inputs to the ones listed. For example, in TLS the mechanism by which a challenge is generated is the take the hash over the entire key exchange. This includes the random Nonce that the challenger generated at the beginning of the key exchange (as well as a lot more) an would therefore satisfy the Challenge() function.

Table 5.7: Pre-requisite Composite Operations

ID/Symbol	Description	Input(s)	Output(s)
IDVKLookup(I)	An operation that given the ID of the identity key of a remote party can recover the identity verification key itself.	Identity Key ID	Identity Verification Key
GetChallenge(Random)	An operation that generates a value that can be used by a remote party to attest their identity. [editors note: this requires either local random input or tie to shared secret, derived from local random input, to prevent replay]	A random value, derived either from local random source or from the shared secret	A "Challenge" value
Attest(C,sK)	An operation to produce the response value for a given challenge and key. The challenge should, where possible, include input from the remote party. Where this is not possible, the challenge shall be cryptographically tied to a shared secret that is dependent on input from both parties.	The "Challenge" and the key used to "sign" the response	The response value, the result of a cryptographic operation (e.g. a signature) on the challenge the only the holder of the secret key can perform.
AttestVerify(C,R,vK)	An operation to verify the response was as expected.	The "Challenge", the response received and the verification key	True or false, whether the challenge was correctly "signed"
StartKEX(Random)	An operation to create the local secret value and the public value for transmission at the start of a key exchange.	A random value or shared secret	A local secret value and a public value that can be sent to the remote party.
FinKEX(Private,Public)	An operation to produce a shared secret given the local secret value and the corresponding public value from the remote party.	The local private value, the remote public value	A shared secret that can only be calculated by somebody that holds a private value that corresponds to one of the public values.
StartMPKEX(Random, SharedSecrets)	An operation to create the local secret value and the public value for transmission as part of a multiparty key exchange.	A random value and the shared secret	A local secret value and public values (which may be an encrypted versions of the local secret) that multiple parties can use to derive a common shared secret.

FinMPKex(localSecret, RemotePublicValues)	An operation that, given the local private value and the public values from all other parties, can derive a common shared secret	A local secret value and the public values from every other party	The common shared secret
KDF(MasterSecret, key lengths required)	An operation that, given a master secret and the lengths of the keys that are required can derive key material from the master secret.	The MasterSecret, usually a shared secret from a key exchange	Derived keys of the length required.

Some of these operations may be used in multiple places. For example, a hash function may be used in the signing process and also in the PRF process. These need not be the same hash, although often they would be expected to be. It is also possible to use other operations as long as they meet the same security requirements that follow. For example, KHASH could be replaced by a CMAC algorithm. To demonstrate compliance with the present document, all operations shall be shown to meet the requirements herein, there is no requirement for a specific type of operation to be used (any similarity in naming convention to existing well-known cryptographic operations is purely for clarity).

5.3.2.1.5 Pre-requisite Composite Operation Requirements

These describe the requirements of the primitive operations in MSP. These are not intended to mandate a particular type of operation and some cryptographic operations may satisfy more than one of these requirements. For example, a signature operation will satisfy the requirements of a MAC operation. Therefore, a signature algorithm (e.g. ECDSA) could be used for both signature operations and MAC operations (although for most known applications this would add an unnecessary overhead and is therefore not an expected MSP scenario).

Table 5.8: Pre-requisite Composite Operation Requirements

Operation	Requirements
IDVKLookup(I)	<ul style="list-style-type: none"> The Identity Verification Key returned, symmetric or asymmetric, can be considered trusted to authenticate identity I. It shall not be possible for an adversary to produce an identity I for which he knows or can derive the Identity Key It shall not be possible for an adversary to get cryptographic operations performed on data using the Identity Key corresponding to I. The Identity Key should not be known to other parties. If it is, these parties shall be trusted parties only; abuse of the key by these parties shall not be considered a threat.
Attest(C,sK) AttestVerify(C,vK)	<ul style="list-style-type: none"> It shall not be possible to recover either the signing key or the challenge from the output value of the Attest() function. It shall not be possible to create the value that passes the AttestVerify without the signing key
GenChallenge(Random)	<ul style="list-style-type: none"> It shall not be possible to recover the random input from the challenge output without the use of a secret key and decryption operation. The input shall include a random element chosen by the challenger – to prevent replay. This may be indirectly (e.g. the shared secret could be considered to the challenge if, and only if, the challenger contributed a random component into its derivation)
StartKEX(Random) FinKEX(public,private)	<ul style="list-style-type: none"> To generate a local secret, the random input to StartKEX shall include a locally generated random component or a non-random component from a trusted and non-repeating source (e.g. an assured time, a counter) together with at least one secret value. It shall not be possible to derive the private output of StartKEX from the public value(s). FinKEX shall produce the same output given the local secret values and remote public values as from the local public values and remote secret values. <p>Note: It is possible to have an endpoint that only generates a public value and does not need a local secret value.</p>

These describe the requirements of the operation, these are not intended to mandate a particular type of operation. Some cryptographic operations may satisfy the requirements of more than one operation. For example, a signature operation will satisfy the requirements of a MAC operation. Therefore, a signature algorithm (e.g. ECDSA) could be used for both these operations. Equally, some protocols may make use of multiple primitives that, in conjunction, perform an operation satisfying these requirements. For example, a hardware-based noise source may be biased and not satisfy the RNG requirement that all outputs occur with equal probability. In this case, using the hardware based noise in conjunction with a HASH function may satisfy these requirements and together may constitute the RNG function.

These describe the requirements of operations that are fundamental to MSP and that may be satisfied by a combination, or even multiple different combinations, of the primitive operations. In all cases, an example is provided that meets the MSP requirements. Any other mechanism may be used as long as it also meets the same requirements.

5.3.2.2 Stage 1 - Hello stage

This is where the client, server, and any middleboxes first establish connection to each other. In this stage, participants are expected to exchange and learn the following:

- Client, Server and Middlebox addresses – the address used by the underlying protocol to route the exchanged data. These may be expected to change between sessions.
- Client, Server and Middlebox identities – the client, server and middlebox may have additional identities that need to be exchanged as part of the protocol. Identities should not be expected to change between sessions. These are deemed not to be sensitive as MSP is for endpoint security not anonymity.
- Client, Server and Middlebox capabilities – where a protocol has multiple possible modes of operation, some of which are optional, the ones supported by the parties and selected for the session.
- Session identifier.

Some of these may be the same item. For example, a protocol may define that a session can be uniquely identified by the addresses of the endpoints; in this case the addresses and the session identifier may be one and the same.

Table 5.9: Hello Requirements

ID	Requirement	Mandatory/Optional
H.R1	The client shall be given the network address of the endpoint server, whether connecting to it directly or through a middlebox.	Mandatory
H.R2	The server shall learn the network address of the client	Mandatory
H.R3	The client shall be given an identity that it expects the server to be able to attest to being (which may be the same as the network address)	Mandatory
H.R4	The server may require an identity for the client (which may be the same as the network address). The client may be required to attest this identity by the server or by a middlebox. Support for client identity attestation requiring a persistent identity shall not be included for any protocol that supports client anonymity. Support for an ephemeral identity may be included as a requirement in such a protocol. For example, the client may generate a random ID and the server then uses this ID in the shared secret derivation and then attests this shared secret, to prove that there isn't another party acting between them that they are not aware of.	Optional
H.R5	All middleboxes shall have an identity and a means to attest their identity.	Mandatory
H.R6	The client shall authenticate the identity of any middlebox any middlebox it wishes to participate in the session	Mandatory
H.R7	The server shall authenticate the identity of any middlebox it wishes to participate in the session	Mandatory
H.R8	The client and the server may require authentication of middleboxes that their counterpart had introduced to the session.	Optional
H.R9	No middlebox shall be able to add itself to the session without either the server or client authenticating it.	Mandatory
H.R10	The client shall require the server to attest its identity.	Mandatory
H.R11	The server may require the client to attest its identity.	Optional
H.R12	The level of access of a middlebox shall be determined by the endpoint(s) that are required to authorise its introduction to the session.	Mandatory

H.R13	All parties shall learn a session identifier unique for that party. This may be explicit (e.g. a value to be stored for session resumption) or implicit (where it could be the addresses of the endpoints) or a combination.	Mandatory
H.R14	Client and server shall exchange information that allows both parties to determine the optional components of cryptographic suite that will be used for the remainder of the operations. If the protocol has no options, this is a null operation	Mandatory
H.R15	Client and/or server shall communicate the suite determined in H14 to the middleboxes.	Mandatory
H.R16	Client and server shall be able to verify that the information exchanged for H14 had not been changed by any unauthorised party.	Mandatory

Table 5.10: Hello Data Exchanged

ID/Symbol	Data Description
ID _C	The identity of the client
ID _S	The identity of the server
ID _{M[x]}	The identity of Middlebox X
ADDR _C	The underlying protocol address of the client.
ADDR _S	The underlying protocol address of the server
ADDR _{M[x]}	The underlying protocol address of Middlebox X

Table 5.11: Hello Data Derived

ID/Symbol	Location	Data Description	Derivation Method
IvK _C	Server, Middlebox	The Identity Verification Key of the client	IDVKLookup(ID _C)
IvK _S	Client, Middlebox	The Identity Verification Key of the server	IDVKLookup(ID _S)
IvK _{M[x]}	Client, Server	The Identity Verification Key for Middlebox X	IDVKLookup(ID _{M[x]})

5.3.2.3 Stage 2 – Endpoint Key Exchange

This is the stage where the necessary key material, random values and challenges and responses are exchanged between the two endpoints in order to initialise the session. At the end of this stage, the client and server should have agreed a master secret from which to generate end to end session keys. The master secret shall be derived from the shared secret; the shared secret derivation may not include replay protection; the master secret shall be derived from the shared secret by a means that adds replay protection if required.

There may not be a specific key exchange message. For example, the shared secret may be a pre-placed keys associated with the identifiers exchanged in the hello messages and the nonce may be a counter that both parties have maintained. In this case there may be no need to have an endpoint exchange, the endpoint exchange will be deemed to consist of zero messages and to have happened at the same time as the hello.

There may also not be a specific challenge and response to for identity attestation or the attestation of exchanged values. For example, the identity keys may be used as in input to the shared secret derivation, in combination with server and client ephemeral keys, in which case proof that the same shared secret has been derived may be sufficient to prove that only the expected endpoint could have derived the shared secret.

Table 5.12: Endpoint Key Exchange Requirements

ID	Requirement	Mandatory/Optional
E.R1	If the client receives values that are required in the shared secret derivation from the server, the server shall attest their authenticity. This may be done directly, by the server attesting the authenticity of the input values sent (e.g. by signing them) or indirectly by a mechanism to verify that both parties have derived the same output shared secret (thereby guaranteeing same input).	

E.R2	If the server receives values that are required in the shared secret derivation from the client, the client may be required to attest their authenticity (directly or indirectly)	
E.R3	The derivation of the master secret shall include a nonce value to prevent the same master secret being used across multiple sessions.	
E.R4	The server shall attest its identity to the client	
E.R5		
E.R6		

Table 5.13: Endpoint Key Exchange Data Exchanged

ID/Symbol	Data Description
EESInput _s	A server generated Nonce or Public Key (possibly more than one of these)
EESInput _c	A client generated Nonce or Public Key (possibly more than one of these)
ChallengeCS	A challenge value with client chosen random input for the server to use to attest its identity
ChallengeSC	A challenge value with server chosen random input for the client to use to attest its identity
AttestationSC	A server generated response to the client's attestation challenge
AttestationCS	A client generated response to the server's attestation challenge

Table 5.14: Endpoint Key Exchange Data Derived

ID/Symbol	Location	Data Description	Derivation Method
EESInputPriv _s	Server	A private value corresponding to EESInput _s	
EESInputPriv _c	Client	A private value corresponding to EESInput _c	
MasterSecret _{CS}	Client + Server	A shared secret between the client and server endpoints	

For the purposes of mapping the protocol to MSP, the parameters exchanged in this stage are:

- Endpoint Identity Key Identifier – a non-sensitive value that allows the remote endpoint to identify a key suitable for assuring our identity. Examples could include an identifier for a pre-shared key the remote endpoint already has, and the certificate ID for a certificate with our public key, ...
- Endpoint Identity Attestation Input – a non-sensitive value sent by one endpoint for the other to use as an input into a challenge response mechanism. A shared secret value may be used but, where exchanged, this value shall not be sensitive.
- Endpoint Identity Attestation Response – The proof-of-identity computed as a result. This could be a cryptographic operation on a challenge, the signing of a shared secret, ...
- Endpoint-Endpoint Session Key Input – a non-sensitive value that allows the remote party to generate, identify or derive, a shared secret value. "Non-sensitive value" may include a sensitive value that is encrypted, e.g. a session key derived by one party and sent to the other using public key encryption.

In addition to the parameters exchanged, the following parameters may be produced at the endpoints that are required to be kept secret by the endpoints:

- Endpoint-Endpoint Session Key Private – a sensitive value that is not transmitted and corresponding to the Endpoint-Endpoint Session Key Input that allows the two parties to derive the same shared secret in a manner that someone observing the shared input values alone could not replicate. An example could be an ephemeral Diffie-Hellman key, the public part would be the Endpoint-Endpoint Session Key Input that is exchanged, and the private value would be the Endpoint-Endpoint Session Key Private.

Each one of these parameters may consist of multiple items or the two might be the same as each other.

5.3.2.4 Stage 3 – Middlebox Key Exchange

This is the stage where the necessary key material, random values and challenges and responses are exchanged in order to initialise a session between an endpoint and a middlebox. The keys derived here will be used to allow the middleboxes to obtain the keys necessary to access the end-to-end session in a secure manner (e.g. by using these keys as a KEK to send the session key wrapped under). Again, if keys are pre-placed, there not be a specific middlebox exchange message. Equally, the middlebox exchange may be combined into the same messages (end even using some of the same data) as the endpoint exchange messages.

For the purposes of mapping the protocol, the parameters exchanged here are:

- Client-Middlebox Session Key Input
- Middlebox-Client Session Key Input
- Server-Middlebox Session Key Input
- Middlebox-Server Session Key Input
- Client-Middlebox Session Key Input
- Server-Middlebox Session Key Input
- Client-Middlebox Identity Attestation Input
- Middlebox-Client Identity Attestation Response
- Server-Middlebox Identity Attestation Input
- Middlebox-Server Identity Attestation Response
- Session Identifier

In addition to the parameters exchanged, the following parameters may be produced at the endpoints that are required to be kept secret by the endpoints:

- Endpoint-Middlebox Session Key Private - a sensitive value held by the endpoint that is not transmitted and corresponding to the Endpoint-Middlebox Session Key Input, allowing the two parties to derive the same shared secret in a manner that someone observing the shared input values alone could not replicate.
- Middlebox-Endpoint Session Key Private - a sensitive value held by the Middlebox that is not transmitted and corresponding to the Endpoint-Middlebox Session Key Input, allowing the two parties to derive the same shared secret in a manner that someone observing the shared input values alone could not replicate.

The endpoint-middlebox and middlebox-endpoint Session Key Inputs are to allow for the derivation of session secret between the endpoint and the middlebox.

5.3.2.5 Stage 4 – Multiparty Exchange

With carefully chosen cryptography, the above exchanges may already have already shared the necessary secrets in a secure manner between the parties that require them. If this is not the case, the multiparty exchange step allows all parties involved to derive the same resulting session key(s). The inputs are the session key(s) from the endpoint key exchange and the middlebox key exchange(s), the outputs will be multiparty session keys. It is possible for the keys agreed at the end of the endpoint exchange (i.e. derived from the endpoint master secret) to become the multiparty session keys. In this case, this stage will be a key distribution stage rather than a key exchange or agreement mechanism. It shall be subject to the same security requirements.

For the purposes of mapping the protocol, the parameters exchanged here are:

- Client-Middlebox Multiparty Key Input
- Client-Server Multiparty Key Input
- Server-Middlebox Multiparty Key Input
- Server-Client Multiparty Key Input

The outputs shall be the Multiparty Session Keys. These shall include:

- Multiparty Reader Key(s) – a key or keys that allows decryption of the portion of the traffic someone with read access is authorised to view. This shall not allow for traffic to be modified in a manner that the endpoint cannot detect. A middlebox with read access shall be able to confirm that the traffic has not been modified by an adversary who does not have read access. A middlebox with read access may not be able to detect that another middlebox with read access has made an unauthorised modification, the endpoint shall be able to detect this.
- Multiparty Writer Key(s) – a key or keys that allows a middlebox to modify the portion of the traffic that they have been granted write permission to. A middlebox with write access shall be able to verify that, prior to receipt, the traffic has not been modified by anybody without write access. If all parties are to be granted write access, there is no read-only access, this may be assumed to be the same as the Multiparty reader key.
- Multiparty Identity key(s) – an optional key that may be used to allow endpoints, or endpoints and middleboxes, to identify who was responsible for the last write to the data.

5.3.2.6 Stage 5 – Exchange Finish

If the key exchanges above did not contain protection from an active adversary modifying the content and therefore undermining the security of the protocol, this stage shall be present to authenticate that the key exchanges have not been tampered with. This should where appropriate include authentication of key exchanges with middleboxes, in order to

ensure that middleboxes are not deceived by an active adversary. An example of such a mechanism is present in TLS where a hash of the key exchange messages is exchanged. No such message is present in IPsec, the protocol has protection for the key exchange built into it.

Where the underlying protocol does not have any requirement to exchange data in order to validate that the key exchanges are free from tampering, for the purposes of mapping a protocol to this MSP profile, this stage may be considered to be the part of the last key exchange message occupying zero bytes.

No application data shall be transmitted using an MSP protocol until after this stage has completed.

5.3.2.7 Stage 6 – Renegotiation Hello (optional)

Renegotiation of the security parameters is not a requirement of MSP. However, where it is supported, the renegotiation exchange is mapped into the same stages as the initial setup. Therefore, the “Renegotiation Hello” is the message that triggers the start of renegotiation.

Renegotiation may also proceed without any hello message. For example, the underlying may define a key expiry mechanism (possibly based on time, possibly based on counts of packets, etc). Upon expiry, both sides initiate exchange, there is no hello message to trigger the exchange.

Renegotiation (if supported) need not permit additional middleboxes to be added, nor allow for the escalation/increasing of a middlebox's access. Where a protocol supports the addition of middleboxes during renegotiation, or the changing of a middlebox's permissions, any change in the permissions of a middlebox shall be authorised by the same endpoint that authorised the presence of the middlebox (both endpoints if mutually authorised). Any addition of a middlebox shall be authorised by at least one endpoint. In all cases, where additional access is requested, new multiparty session keys shall be derived and exchanged. The mechanism for this shall not permit the previous session keys to be derived by any party that did not have access to them.

5.3.2.8 Stage 7 – Renegotiation Endpoint Exchange (optional)

This may or may not be the same mechanism as the initial endpoint exchange and may also not require the transmission of any new data. For example, if the initial key exchange provided a mutually agreed secret derivation key, the renegotiation may be to use keys generated from that secret. Therefore, no exchange messages would be required.

5.3.2.9 Stage 8 – Renegotiation Middlebox Exchange (optional)

5.3.2.10 Stage 9 – Renegotiation Multiparty Exchange (optional)

5.3.2.11 Stage 10 – Renegotiation Finish (optional)

5.4 Conformance and compatibility

Protocols satisfying the MSP profile may allow for situations where an end unit does not support an MSP protocol. In this scenario, the final MSP middlebox prior to the non-MSP endpoint may terminate the MSP tunnel and re-encrypt the data to forward under a new non-MSP tunnel negotiated between the middlebox and the non-MSP endpoint.

Alternatively, the key negotiated between the final endpoint and the final MSP-middlebox may be securely distributed by the MSP protocol to the MSP middleboxes and endpoint. Where this happens, the MSP-compliant endpoint shall be able to determine that this break in the end-to-end encryption or compatibility mode operation is being performed, which middlebox is performing it and the identity and the cryptographic security of the forwarding tunnel that the middlebox has negotiated.

For example, this situation would arise if both the client web browser and the client's enterprise firewall gateway supported an MSP protocol (e.g. mcTLS) but the webserver on the website being visited had not been upgraded to a version that supported MSP protocols. Under a traditional MITM proxy, the proxy would negotiate a connection with the server and the client. However, the client would receive no security assurance of the onward tunnel negotiated by the proxy or the identity of the endpoint.

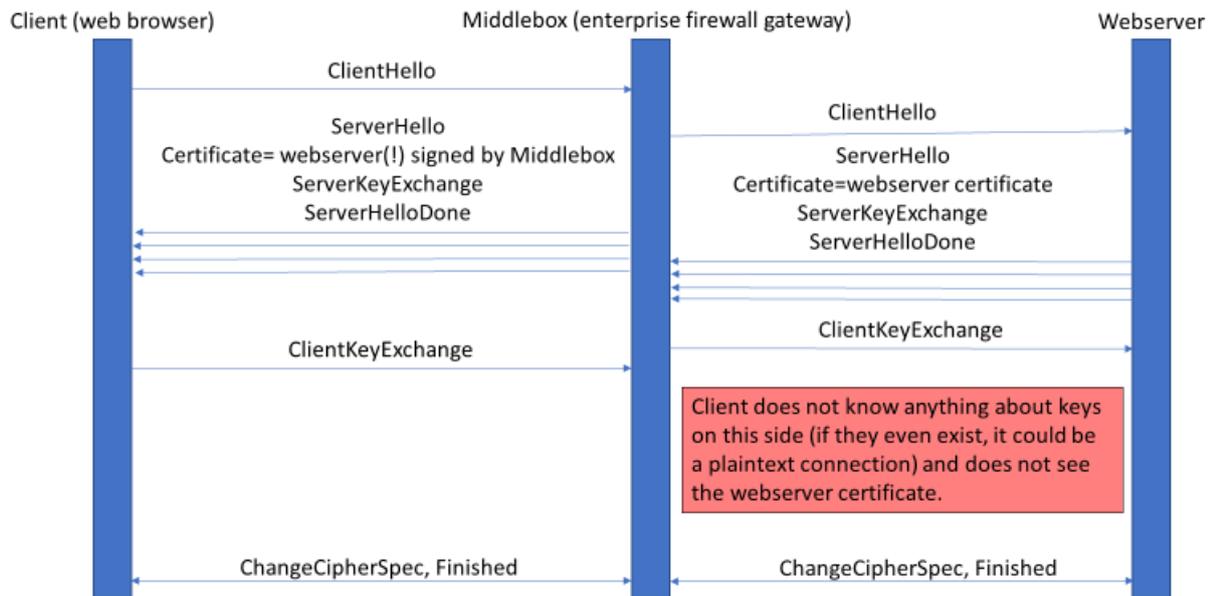


Figure 5.4: Split proxy design, no visibility of end-to-end security

Under MSP, the re-encryption at the middlebox would still occur but the client would receive the server endpoint's certificate and the details of the cryptographic suite negotiated between the middlebox and the non-MSP server, thereby allowing security mechanisms such as certificate pinning and extended-validation certificate to be used in conjunction with middleboxes.

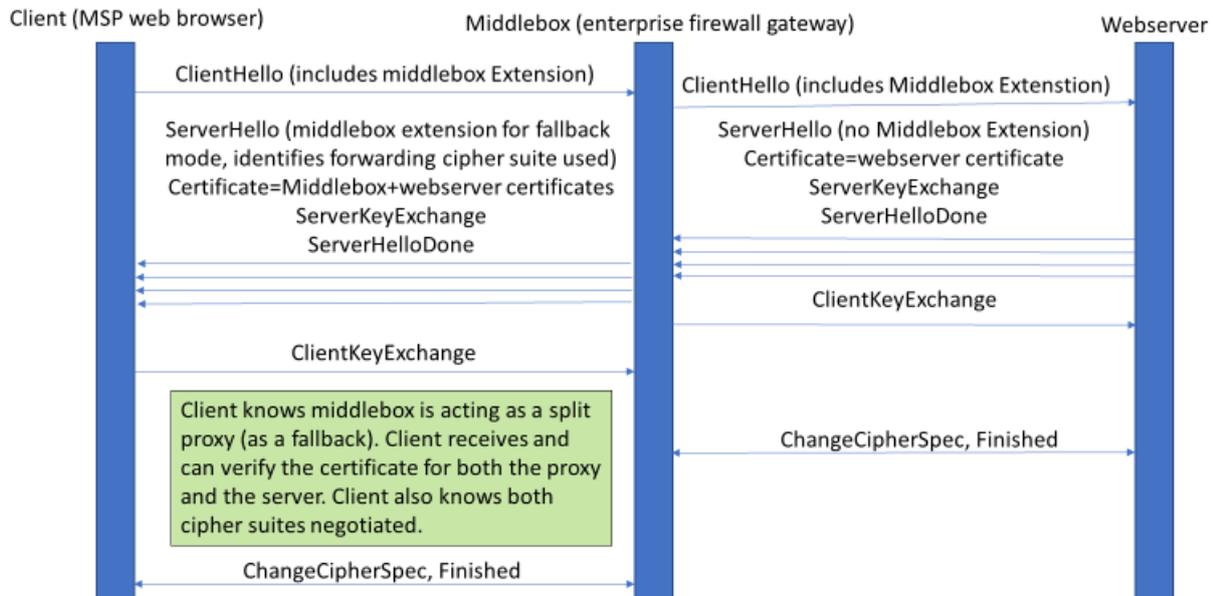


Figure 5.5: MSP re-encryption, client has visibility of end-to-end properties

Annex A (informative): Use Cases of MSP

A.1 Introduction and history

This annex was prepared from an exhaustive review and analysis of the rather large array of published studies of actual uses, industry specifications, conference presentations, scholarly reports and papers in publicly available materials worldwide. See Annex D: Bibliography.

One of the difficulties in treating the subject “middlebox” is the varied use of the term itself in different professional communities – ranging from unknown to pervasive. Because the term “middlebox” includes any device between user end-points other than a transparent switch – it embraces an enormous number of functional physical and virtual equipment components that exist in the complex paths typically found between communication endpoints. Middleboxes are essential to the operation of all telecommunication and ICT networks today, and large infrastructures will typically have thousands of ubiquitously deployed middleboxes. By almost any metric, middleboxes also represent perhaps the most active and innovative sector of network technology, research, and product development today with scholar research search engines displaying more than 10,000 published papers over the past decade and hundreds of new ones appearing every month. The published papers section of the Bibliography contains a distillation of the 52 different papers specifically relating to the platforms mcTLS and mbTLS.

Although middlebox devices have essentially always existed in telecommunication and ICT networks, the term itself did not come into use until about twenty years ago in conjunction with IP networks. The word “middlebox” is generally attributed to Berkeley professor Lixia Zhang in 1999 and began to appear as an expression of long-standing, fundamentally divergent views on network design objectives. Since the emergence of datagram networks (i.e., connectionless, host-to-host, CYCLADES, internets) in the 1970s, a technical and operational community dichotomy has existed concerning the need for and role of middleboxes. On one side, altruistic factions have sought transparent, unfettered transmission bandwidth directly between and controlled by end-points – essentially free from middleboxes. On the other side are operators of wireline and radio based networks, innovators, service providers, and regulatory communities who use middleboxes to meet real-world physical, economic, technology, business obligation, and societal needs. Some new enterprises have also emerged to leverage this divergence for business purposes that have become known as Over the Top (OTT) providers. The rapid emergence of fully virtualised network architectures using Network Functions Virtualisation (NFV) in conjunction with Software Defined Networks (SDN), enables middleboxes to be integrated into those architectures. Enabling the orchestration of NFV middleboxes is occurring on a significant scale in ETSI’s NFV Industry Standards Group (NFV ISG) - that is found in the adopted enabling technical specifications and more than 8,000 published materials.

Over the past several years, as increasingly powerful computer processing capabilities became available at end-points, widespread deployment of strong encryption capabilities were enabled, and a new phase in the middlebox controversies emerged. End-to-end encryption significantly impaired the ability of middleboxes to function, and networks increasingly became poorly performing with many essential operational capabilities non-functional. Proprietary middlebox technology and deployments emerged and evolved in many different ways to compensate – represented in part by the thousands of research papers and hundreds of middlebox patents filed by scores of vendors and providers.

The efforts to devise taxonomies, use case organizations and common trusted means to discover and effect communications with and among middleboxes have been ongoing for the past decade. This ETSI MSP Technical Specification draws upon work within the industry to create common interfaces and protocols.

Articulating a coherent structure for use cases from the outset have remained a challenge because middleboxes literally encompass almost anything and everything within communication and ICT networks. In one of the initial efforts in the IETF in 2002, the authors admitted “there is no obvious way of classifying them to form a hierarchy or other simple form of taxonomy. Middleboxes have a number of facets that might be used to classify them in a multidimensional taxonomy.” See RFC3234, Middleboxes: Taxonomy and Issues, Feb 2002. That initial taxonomy soon became irrelevant, and a variety of others are found in the literature, specifications, and patent declarations. One recent use case structure based on technical functions consists of 1) measurements (Packet Loss, Round Trip Times, Measuring Packet Reordering, Throughput and Bottleneck Identification, Congestion Responsiveness, Attack Detection, Packet Corruption, Application-Layer Measurements) and 2) other functions (NAT, Firewall, DDoS Scrubbing, Implicit Identification, Performance-Enhancing Proxies, Network Coding, Network-Assisted Bandwidth Aggregation, Prioritization and Differentiated Services, Measurement-Based Shaping, Fairness to End-User Quota). See Dolson, IETF Internet Draft, “Beneficial Functions of Middleboxes.”

The use cases below describe the many benefits enabled by middlebox physical and virtual instantiations in telecommunication and ICT networks and services. If the traffic is encrypted, the middlebox needs to have some manner of controlled awareness/exposure to that traffic to remain functional. The ETSI Middlebox Security Protocol

profiles enable a secure means to proxy TLS, IPsec, or other security protocol traffic: off-line buffering, fine-grained read-only, users retain sight of authentication and algorithm selection. When implemented, the MSP profiles allow secure discovery and control of those middleboxes by users across all of the use cases described below.

A.2 New infrastructure, services, and innovation use cases

Middleboxes are referred to as “innovation engines” because of their ability to provide tailored capabilities within transport paths and at services data centres. As a result, middleboxes are core components of new infrastructure such as Network Functions Virtualisation (NFV) and Software Defined Networks (SDNs) and their implementations as 5G. More patents, scholarly papers, and standards activities deal with these new developments than any others and have significantly increased from 2015 onwards. Computer Science curricula are devoted to middlebox innovation at schools worldwide. Middleboxes are explicitly included in basic NFV specifications in the form of VNFs (Virtualised Network Functions) for their creation.

These benefits include much more than just emerging new infrastructure. Innovative middlebox techniques have been applied to satellite, mobile, IoT, industrial control systems, automobile communications, and WiFi-clustered installations. Even new reductions in power consumption are dependent on middleboxes. All of these use cases have enabled new uses that were not previously available – in many cases simply by developing innovative new software.

Thus, one of the more general but most significant sets of middlebox use cases go to the fundamental enablement of new infrastructure, services, and innovations. Specific middlebox taxonomy groupings are set forth below under categories such as security, performance, operational, and compliance benefits.

A.3 System and user security use cases

Middleboxes have long been the security workhorses of networks and are essential for all of the basic capabilities in networks. They characteristically fall into four categories:

- **Network firewalls.** Firewalls are security barriers found in telecommunication and ICT networks usually implemented with some form of middlebox using rules that prevent undesired or harmful traffic reaching an end point. Network firewalls consist of a middlebox during the path between end-points – often at gateways between networks. They are often described as the first line of defence against unwanted and malicious traffic targeting network users and one of the earliest kinds of middlebox – literally to the beginning of communication networks.
- **Application firewalls.** Because many successful attacks today are not on the network level, but on application level applications are not protected by specialised middleboxes that are used to detect and block attacks against vulnerable applications. They are often included as part of defence-in-depth designs.
- **Intrusion detection systems.** This is background process continuously monitoring the network traffic and detecting and preventing the attacks. that monitors a network in real-time for activity indicative of attempted or actual access by unauthorized persons or computers. The system detects unauthorized users attempting to enter into network end-point systems or devices.
- **Intrusion prevention systems.** Unlike intrusion detection systems, intrusion prevention systems consist of middleboxes with additional features to secure networks by focusing on attacks before they have achieved access and done damage. It does this by traffic inspection to detect new types of attacks. A middlebox performs TCP segment reassembly, traffic analysis, application protocol validation, and signature matching to identify the attack.

A.4 Performance use cases

Middleboxes have long been the network performance workhorses of networks and are essential for all of the basic capabilities in networks where they provide for edge delivery of content, caching, transcoding, compression, forward-deployment insertion of cached content, and protocol improvements. Performance enhancing middleboxes are especially important for mobile networks where many users are supported on a shared local radio access network bandwidth. Entirely new industry 5G initiatives such as Mobile Edge Computing (MEC) are implemented through cloud-based middlebox arrays. Performance use cases characteristically fall into three categories:

- **Proxy/caches.** Proxy caching has been used since the early 1990s to significantly speed up traffic flows and reduce costs. Proxy caching middleboxes detect repetitive requests for the same information – generally multimedia content streaming or large software files – and store the information locally combined with transparently redirecting the user to the local store site. Modern networks could not exist on any scale without such middleboxes, and considerable innovation has gone into techniques to optimize proxy caches.

- **WAN optimizers.** A WAN optimization middlebox analyses network traffic from application clients within a private network and the edge gateway of a larger public network. The computers at these edge locations may prefetch and store copies of static content near potential destinations to decrease transit time and latency. A WAN optimization device typically monitors clients' network traffic to attempt to predict data likely to be requested by clients in the near future. This predicted data is prefetched over the WAN and stored by the WAN optimization devices at the clients' respective network locations, so that this data can be quickly accessed by users if requested.
- **Protocol accelerators.** The protocols used for network communication are both highly layered with individual frames or packets and their headers containing considerable “wasted” capacity for structured bits that are not used. A wide array of middleboxes in networks and on their edges leverage these characteristics of communication protocols – especially with extremely high-performance hardware Application Specific Integrated Circuit (ASIC) chips coupled with segregation of traffic types – to considerably optimize the throughput and reduce latencies.

A.5 Operational use cases

Middleboxes have long been the operational workhorses of networks and services and are essential for all of the basic capabilities. They characteristically fall into the following categories:

- **Access control.** Access control using middleboxes constrains the attachment of a device to a network to establish a communication end-point, as well as what that device or its user or agent can do directly. It includes what programs executing anywhere to which network connectivity is provided. Access control seeks to prevent activity that could lead to a breach of security. With the emergence of IoT and other autonomous devices, including highly mobile vehicle communication requiring constant handoffs, access control middleboxes are critical preventing large-scale network attacks.
- **Billing and usage monitoring.** Essentially all communication networks and service support systems monitor use for a wide variety of purposes, including security, troubleshooting and billing. Most of these monitoring functions employ logging or other forms of auditing mechanisms, implemented with middleboxes.
- **Asset tracking.** The discovery, identification, inventory, and management of physical and software assets at network end-points is critical for a wide array of network operational and security needs that are met through the deployment of specialised middleboxes. The activity is the first of the ETSI Critical Security Controls technical report, and necessary for the exchange of threat information required, for example, by the NIS Directive.
- **Network Address and Port Translation.** A considerable array of different traffic transport addresses and ports are used in contemporary communication network protocols. At network boundaries or gateways, a wide variety of translations of those addresses and ports are required – often sharing them at network end-user edges. These functions are accomplished using specialised middleboxes optimized for these purposes.
- **Protocol conversion.** Similar to address and port translations described above, the layering or encapsulations required by communication protocols require conversions among those protocols. These protocol conversions are accomplished using specialised middleboxes optimised for this purpose.
- **Name or tag resolution.** Both telecommunication networks in the form of caller identification or IP networks in the form for Domain Names, require high-speed lookups of addresses or tags accompanying the communication traffic. Increasingly, these lookups are accompanied by security verifications, and cached at different points in network paths and infrastructures. These specialised tasks are accomplished by middleboxes.
- **Operations control.** Communication networks and services – including those undertaken at cloud data centres – are highly complex and require an array of management capabilities that constitute an overlay for monitoring and allocating resources, and defending against threats. These essential operations control mechanisms are implemented using specialised middleboxes. These capabilities may include content delivery capabilities, load balancing, and collation of multi-sources for end users. An especially significant need and challenge today is global roaming by users or devices visiting many different networks.

A.6 Compliance obligation use cases

Middleboxes have long been the principal means for implementing a significant array of network and service compliance obligations required through a variety of legal and regulatory mechanisms, including Service Level Agreements among providers and with enterprise users. They characteristically fall into nine categories. Some of these categories overlap the security, performance, and operational use cases described above.

- **Availability/resilience use cases.** Perhaps the most basic of all compliance obligation requirements is that of availability. Users of all kinds that make use of a device, network or service expect it to be functioning and available to meet their desired needs. That desired level of availability is often effected through various diverse legal mechanisms such as combinations of service level agreements, implied warranties, or regulatory requirements. Middleboxes are essential implementing this capability.

Included in this benefit category are public services where some network infrastructures and services are either owned by governmental bodies and support national services. Such networks and services may also be regarded as essential or critical and include a wide array of financial system, public utility, or industrial control uses.

An additional subset of public or private networks and services may also be subject to specific resilience and survivability design requirements - especially during national or local emergencies. These kinds of requirements often require an array of device level designs (e.g., special aerospace or military grade components and testing), multiple redundant systems, backup, failure isolation capabilities, elimination of any single-point failure components for failsafe purposes. Such requirements may include access and prioritization for authenticated users identified in conjunction with emergency and public safety communication implemented by middleboxes.

Certain infrastructures and services may be subject to outage auditing and reporting requirements that require additional capabilities supported by middleboxes. These services typically include public networks and services, or private ones that are subject to additional contractual requirements concerning availability.

- **Emergency and public safety communication use cases.** Emergency and public safety communication requirements typically enjoy the highest of compliance obligation priorities. These capabilities support an array of critical national police, fire protection, and emergency management needs at all levels from local to global and implemented using middlebox technologies. There are several.

During public or even individual emergencies, at local, national, or international levels, a highly important need exists to reach different arrays of individuals through any available electronic communications. Tsunami warning capabilities have become prominent several years ago as global disaster conditions unfolded. Such needs vary from that breadth of users to national and city levels, down to local roadways for a wide array of circumstances that may include an impending natural disaster, a major accident, or even an abducted child or a disoriented elderly person. The compliance capabilities include diverse structured information formats, interworking, authentication, and delivery methods including effective human interface requirements facilitated by middleboxes.

The inverse of the authority-to-many compliance obligation occurs when an individual much reach local emergency departments for medical, police, or fire purposes. The capabilities are known by their telephony short number designations - e.g., 112 or 911. Increasingly these requirements include authentication and geolocation capabilities, as well as alternative telecommunication transport and applications, which include text, images, and even video.

During the declaration of a serious emergency - especially a national one - the telecommunication networks and designated public services are typically subject to limited access and prioritization of traffic to serve government authorities. Private organizations may also enforce such a condition within their private networks for similar purposes. Such requirements necessitate significant compliance obligation capabilities that include substantial authentication requirements and control of switches and routers through middleboxes.

Large scale theft of mobile devices has produced requirements instituted by both regulatory authorities and consumer demand for the ability to remotely discover those devices and disable them. Such requirements impose an array of special design and authentication requirements both within the supporting networks and the devices themselves.

- **Lawful interception use cases.** Essentially every nation as well as many private network owners require the electronic communication network and services within their jurisdiction or under their control to provide available forensics upon lawful authority. For many designated networks and services, including those available to the public, the requirement to support the interception provisioning capabilities is a condition of licensing or required by law through regulation or contractual agreement. Lawful interception as described here consists of the real-time "handover" of network forensics described in a legal instrument - usually pursuant to technical specifications and divided into three subtypes described below. Acquisition can occur

anywhere in the communications path - from the end user device to the transport paths to the centres supporting the services. To the extent the provider or vendor in an investigation is able to handover the information unencrypted or to provide the associated keys or otherwise decrypt the information, they are obligated to do so. The virtualization of these capabilities is also producing new design challenges to meet requirements. These capabilities are met using middleboxes.

- **Retained data use cases.** All electronic communication and IT networks and services create significant amounts of information that is retained in temporary caches, log files, or auditing and accounting systems using middleboxes. This information is retained for diverse operational, business, or legal compliance purposes.

There are several use case variants. Data is retained and accessed for criminal evidentiary and investigative purposes either through law instituted in different jurisdictions (often referred to as Data Retention), or through preservation orders of different kinds and durations. Data is also retained and accessed for civil evidentiary and investigative purposes through law or judicial rules instituted in different jurisdictions (referred to as eDiscovery). Data is also retained and accessed to meet a broad array of compliance (especially in finance and banking sectors) or to meet contractual requirement, or business auditing especially when there is a dispute among the parties.

- **Identity management use cases.** Identity management encompasses an array of compliance obligations that enable any object, including a human user, to manifest an identifier at varying levels of trust and uniqueness in the context of the use or operation of an electronic communications or IT device, network, or service. These capabilities are implemented using middleboxes and include several variants. For example, iIdentifiers associated with individuals, service accounts and network end user devices are widely used in conjunction with access to a network infrastructure or service. Object identifiers are used as part of all electronic communications and IT services to transport information between two end points. Where it becomes necessary, for example to enable communicating or process party blocking, object identifiers are used by middleboxes to prevent communication or information transfer from occurring.
- **Cyber Security use cases.** Cyber security generally encompasses two sets of middlebox enabled capabilities - the instantiation of defensive measures and the ability to exchange structured cyber security threat information. Defensive measures include an array of generally adaptive controls that are designed to be part of every lifecycle phase of a device, network, or service from its inception to its removal from service. These controls produce and leverage forensic information, and themselves apply to maintaining the integrity of that information. Defensive measures depend significantly on timely exchange of structured threat intelligence. Structured threat information exchange includes the ability for a device, network or service to continuously acquire, provide, and utilize current threat intelligence relevant to its use – all implemented through middlebox platforms. As of the release of Version 7 of the Critical Security Controls, two MSP related use cases are included as Controls 12.7 (Deploy network-based Intrusion Prevention Systems) and 18.10 (Deploy Web Application Firewalls) and the related diagrams.
- **Control of illicit content and privacy use cases.** requirements that necessitate middleboxes to implement those requirements. The use cases fall into three categories.

Essentially every nation has requirements that provide for the maintenance of intellectual property rights associated with almost electronic communication and IT devices and content. Some of these requirements also arise pursuant to international treaties to which they are a party. Implementation of these rights can invoke a broad array of capabilities. Implementing these capabilities requires an array of identity management, auditing, and filtering capabilities provided by middleboxes.

Every nation and most organizations maintain limits on content arising from societal or organization norms. These may range from harassment or predatory behaviour to speech or depictions that are highly offensive to most people or can cause substantial harm. Implementing these capabilities requires an array of identity management, auditing, and filtering design capabilities.

Some regional and national conceptualizations of privacy and instantiations in law allow for individuals to control the available public information about themselves through, for example, altering search engine results. These provisions have given rise to privacy by design capabilities requiring the use of middleboxes.

- **Support for persons with disabilities use cases.** Many end users have disabilities that include vision, hearing, and physical impairments arising from an array of causes. In many national and local jurisdictions,

including private organization environments, electronic communication and IT networks and services are required to provide the ability for these classes of users to have effective access and use. The requirements may apply to the entire end-to-end infrastructure and implemented via middleboxes.

A.7 Enterprise Network and Data Centre Use Cases [i.2]

Most enterprise networks originally transmitted packet data in the clear inside their internal networks. Many still do today. When certain enterprises started encrypting at the transport layer in their internal networks to protect against insider threat and/or for regulatory compliance reasons, they always had the option of using RSA key exchanges and using static RSA private keys for a small, privileged group to decrypt and inspect their traffic out-of-band. Out-of-band decryption provides ubiquitous packet payload visibility inside the enterprise that cannot be replaced by inline/MITM decryption solutions. Today there are enterprises with extensive packet broker networks who are doing out-of-band TLS decryption to feed network sniffers, intrusion detection devices, fraud detection, malware detection, application performance monitoring tools, customer experience monitoring tools, and other solutions.

The capability to do out-of-band decryption has been available for twenty years. A large body of tools has grown up over the last twenty years that is dependent on out-of-band decryption. These tools are performing mission critical functions for enterprises, and the loss of out-of-band decryption will create major operational problems for TLS encrypted enterprises if new TLS versions are implemented as-is inside the enterprise. Ubiquitous packet capture and decryption are required for enterprise troubleshooting, and without this capability there will be high severity outages that cannot be solved in an acceptable time frame. The outcome will be the same as extended Denial of Service attacks on enterprises worldwide. Without an out-of-band decryption solution, enterprises are left with the unattractive option of inline/MITM decryption at the data centre edge and running traffic with legacy protocols or in the clear throughout the data centre if they need packet payload visibility. This opens certain enterprises up to significant regulatory and insider threat problems. There are reasons why other forms of troubleshooting and monitoring do not functionally replace the visibility lost from losing out-of-band TLS decryption.

Annex B (informative): Exemplar of Mapping a Protocol to MSP Profile

This is an example of mapping the MSP items to the concrete values exchanged in the protocol

Table B.1: Mapping a Protocol to MSP Profile

MSP Stage	MSP Item	Protocol Message	Protocol Value
Hello	Client_Address	ClientHello ServerHello	MiddleboxList[Client] MiddleboxList[Client]
	Server_Address	ClientHello ServerHello	MiddleboxList[Server] MiddleboxList[Server]
	Middlebox_Address[x]	ClientHello ServerHello	MiddleboxList[x] MiddleboxList[x]
	Client_Identity (optional)	Certificate (client originated)	Certificate
	Server_Identity	Certificate (Server originated)	Certificate
	Middlebox_Identity[x]	Certificate (Middlebox[x] originated)	Certificate
	Session_Identifier	ServerHello	SessionID
	Endpoint Key Exchange	Client Identity Key Identifier (optional)	Certificate (client originated)
Server Identity Key Identifier		Certificate (server originated)	Certificate
Client Identity Attestation Input		Handshake messages	
Server Identity Attestation Input		ServerKeyExchange	Client.random, server.random, ServerDHParams
Client Identity Attestation Response		CertificateVerify	CertificateVerify
Server Identity Attestation Response		ServerKeyExchange	Signed_params

Annex C (informative): Exemplar of satisfying MSP Security Characteristics

In the case of mcTLS, fine-grained permissions-based access is controlled by having keys for each context for readers and writers, and an overall key for endpoints. Kreaders allows the detection of authorised/unauthorised changes but not who has made those changes. Kwriters allows detection of a middlebox vs. endpoint change but does not allow identification of which middlebox made the change. Additional keys could be put in MSP spec for detection of which middlebox made the change (if each middlebox had a different writer key, endpoints could detect which middlebox made the change). This capability requirement should be optional in the MSP profile, it is not a requirement of the protocol to support this and a single common writer key is acceptable.

One means of controlling middlebox access is with separate keys to control each fine-grained usage desired. For example, let plaintext be encrypted using a stream cipher or codebook mode, then the symmetric key for this controls read access to the plaintext. Typically, the same key will also be used for a guarantee of the integrity of the plaintext by a keyed Message Authentication Code or equivalent. For the fine-grained usage restrictions, there could be one or more Message Authentication Codes (i.e. keyed integrity checks) or integrity checks by other mechanisms (e.g. signatures). Then the key or key pair for each integrity check mechanism allows rewriting of that integrity check and thus controls write access. By careful distribution of these keys to authorised middleboxes, read or write access can be granted, and unauthorised changes can be detected.

At the lowest level of privilege, middleboxes with read-only permission would need possession of the decryption read key. They should also possess the read integrity check verification key to detect unauthorised changes to the plaintext. The read integrity check key can be a MAC key and can be the same as the read key (and often is). In this case, the reader can change the ciphertext and the read integrity check value but none of the other integrity check values, thus reader changes are detectable by entities with a key for an additional integrity check (i.e. other reader middleboxes could not detect a change by another reader, but endpoints and middleboxes with write access could detect an unauthorised change by a reader).

At the next level of privilege, middleboxes with read-write permission would need possession of all the read keys and at least one further integrity key. These writers can thus change the ciphertext, the read integrity check value, and at least one further write integrity check value. Using MACs for integrity, these changes are not detectable by readers but are by entities with a key for any additional MACs applied (e.g. endpoint MACs). Using signatures, these changes could be detectable by readers who have been given the writer's public key.

At the highest level of privilege, the endpoints can have integrity check keys that no middleboxes have. Thus the endpoint integrity check cannot be updated by any middlebox and thus provides a means to detect if a middlebox has changed the plaintext.

Fine-grained read access control is achieved by each part of the plaintext (i.e. each context) being encrypted with separate encipherment keys. Fine grained write access control is achieved by different contexts being protected from modification using different integrity keys. Thus possession of these read and write keys controls access to each part of the plaintext. An endpoint integrity key can be common across all parts of the plaintext (i.e. all contexts) because the endpoints have access to all parts of the plaintext so do not need to separate their ability to produce these integrity checks as shown in Table C.1

Table C.1: Fine-grained read access control with separate encipherment keys

Key		Controls	Functionality
For each context:	Read key (encryption and integrity, may or may not be the same)	Ciphering and read Integrity check	Change plaintext and detect changes by anyone not possessing the read integrity key
	Write key(s)	Write Integrity check value(s)	Detect changes made by another entity not in possession of the write key. When used with read key, can change plaintext and read integrity check value and

			this write integrity check value.
Endpoint key		Endpoint Integrity key	Can detect changes by any middlebox.

Table C.2, below, shows how this design can meet the MSP profile.

Table C.2: Satisfying MSP Security Characteristics

Middlebox Security Protocol capability profile	Property of design abstraction that meets this
<ul style="list-style-type: none"> • Ability of the client and server at a sufficient level of granularity to <ul style="list-style-type: none"> ○ discover and identify all gateway middlebox devices that are potentially able to read/modify the traffic ○ decide whether each middlebox can read and/or modify content ○ detect any changes to the content ○ detect who has done modifications ○ obtain proof that it can see all devices without depending on the first middlebox ○ configure different access rights for different blocks of traffic, e.g. header/body or different streams in HTTP2 • Ability of the client to control whether the server can see beyond middleboxes (for client anonymity) • Ability of an authorised device to read traffic without the need for re-encryption • Ability of a client to support a post mortem analysis • Transparency for load balancers without requiring out-of-band communication • Acceptable overhead (performance, usability) • A sufficient degree of backward compatibility with different clients 	<ul style="list-style-type: none"> • <ul style="list-style-type: none"> ○ endpoint key being only available to the client and server ○ client and server authorise distribution of context read and write keys to middleboxes in the middlebox exchange stage ○ read key, write key and endpoint key ○ distinct write key to each middlebox, endpoint key allows detection of which write integrity check value has changed ○ endpoint verification of the hello stage ○ read and write keys per context, overall endpoint key • Whether the client public key identifies it, if used • Distribution of read key to intercepting middlebox allows read access without need to change the ciphertext or integrity check value(s) • Authentication and authorisation logs of middlebox exchange • [clarify this characteristic of the profile, not sure what it means, is it that server should be able to delegate to load balancers without the client being aware?] • Depends on the specific instantiation, but overhead is in the key establishment phase (hello, key exchange, middlebox exchange, finalisation) rather than in the ciphering phase, which for read-only can be buffered offline by context • Depends on the specific instantiation. The native encryption protocol (such as TLS or IPsec) could be run as normal, and its derived keys treated as the read key for ciphering and the read MAC key. These read keys, plus write keys, write MACs, endpoint keys and endpoint MACs, could be distributed by an add-on extension to the protocol implementing the hello, middlebox exchange, and finalisation stages. Only those entities supporting the add-on extension would benefit from the ability to detect changes and those entities not supporting the extension would proceed unaware as if operating the native protocol.

Annex D: Bibliography (Informative)

D.1 Published references relating to mcTLS and mbTLS

D Naylor, K Schomp, M Varvello, I Leontiadis, et al., Multi-context TLS (mcTLS): Enabling secure in-network functionality in TLS, ACM SIGCOMM, 2015, 54 citations, 14 versions

J Ren, A Rao, M Lindorfer, A Legout, et al., Recon: Revealing and controlling pii leaks in mobile network traffic, 2016, 44 citations, 17 versions

S Kim, Y Shin, J Ha, T Kim, D Han, A first step towards leveraging commodity trusted execution environments for network applications 14th ACM Workshop on Hot Topics, 2015, 37 citations, 8 versions

C Lan, J Sherry, RA Popa, S Ratnasamy, Z Liu, Embark: Securely Outsourcing Middleboxes to the Cloud, NSDI, 2016, 26 citations, 14 versions

U Goel, M Steiner, MP Wittie, M Flack, et al., Detecting cellular middleboxes using passive measurement techniques 2016, 15 citations, 11 versions

Y Yiakoumis, S Katti, N McKeown, Neutral net neutrality, 2016, 9 citations, 10 versions

Z Li, W Wang, T Xu, X Zhong, XY Li, Y Liu, C Wilson, et al., Exploring Cross-Application Cellular Traffic Optimization with Baidu TrafficGuard. NSDI, 2016, 7 citations, 13 versions

G Tyson, S Huang, F Cuadrado, I Castro, et al., Exploring HTTP Header Manipulation In-The-Wild, 2017, 4 citations, 4 versions

K Bhargavan, IC Boureanu, PA Fouque, et al., Content Delivery over TLS: A Cryptographic Analysis of Keyless SSL, 2017, 3 citations, 2 versions

L Wang, R Pass, A Shelat, et al., Secure Channel Injection and Anonymous Proofs of Account Ownership, IACR Cryptology ePrint, 2016, 3 citations, 3 versions

O Sanders, C Onete, PA Fouque, Pattern Matching on Encrypted Streams: Applications to DPI and searches on genomic data, IACR Cryptology ePrint Archive, 2017, 2 citations, 2 versions

X Liu, Y Ma, X Wang, Y Liu, T Xie, et al., Swarovsky: Optimizing resource loading for mobile web browsing IEEE Transactions, 2017, 2 citations, 2 versions

J Maisonneuve, VK Gurbani, T Fossati, The security pendulum Managing Radio Networks, 2015, 2 citations, 2 versions

S Huang, G Aceto, F Cuadrado, S Uhlig, A Pescapè, Detecting Middlebox Interference on Applications wpage.unina.it, 1 citation, 4 versions

J Kim, H Choi, H Namkung, W Choi, B Choi, H Hong, et al., Enabling Automatic Protocol Behavior Analysis for Android Applications, CoNEXT, 2016, 1 citation, 3 versions

R Bonafiglia, A Sapio, M Baldi, F Risso, PC Pomi, Enforcement of dynamic HTTP policies on resource-constrained residential gateways Computer Networks, 2017, 1 citation

P Bailis, S Peter, J Sherry, Introducing research for practice Communications of the ACM, 2016, 1 citation, 2 versions

A Silvestro, R Bifulco, F Schneider, X Fu, et al., Is today's DNS the right solution for middleboxes selection?, 2017, 1 citation, 2 versions

A Silvestro, R Bifulco, S Sharma, F Schneider, et al., Issues in Supporting Third-Partys In-Network Services in the Internet netsys17.uni-goettingen.de, 1 citation

Y Hayakawa, L Eggert, M Honda, D Santry, Prism: a proxy architecture for datacenter networks, 2017, 1 citation

S Lee, H Shi, K Tan, Y Liu, SK Lee, Y Cui, Smart and Secure: Preserving Privacy in Untrusted Home Routers 7th ACM SIGOPS Asia, 2016, 1 citation

- J Wilson, RS Wahby, H Corrigan-Gibbs, et al., Trust but Verify: Auditing the Secure Internet of Things, 2017, 1 citations, 5 versions
- C Lan, A Framework for Network Function Virtualization, 2016, 2 versions
- P Chaudhary, VK Yadav, A Survey on Security Issues and the Existing Solutions in Big Data, 2017, 2 versions
- D Naylor, R Li, C Gkantsidis, T Karagiannis, et al., And Then There Were More: Secure Communication for More Than Two Parties, 2017, 4 versions
- D Naylor, Architectural Support for Managing Privacy Tradeoffs in the Internet reports-archive.adm.cs.cmu.edu
- B Richard, Authenticated key exchange protocols in three parties, 2017, 50 versions
- A Khanna, concordia.ca Towards usable and fine-grained security for HTTPS with middleboxes, 2017, 2 versions
- Z Li, Y Dai, G Chen, Y Liu, Content Distribution for Mobile Internet: A Cloud-based Approach, 2016, 7 versions
- Z Li, Y Dai, G Chen, Y Liu, Cross-Application Cellular Traffic Optimization Content Distribution for Mobile Internet, 2016
- S Ludin, Detecting Cellular Middleboxes Using Passive Measurement Techniques Passive and Active Measurement: 17th International, 2016
- V Jyothi, SK Addepalli, R Karri, DPFEE: A High Performance Scalable Pre-processor for Network Security Systems, 2017, 2 versions
- P Zave, RA Ferreira, XK Zou, M Morimoto, et al., Dynamic service chaining with dysco Proceedings of the, 2017
- M Varvello, D Perino, FreeLab: A Free Experimentation Platform, 2017
- S Sevilla, JJ Garcia-Luna-Aceves, et al., GroupSec: A new security model for the web (ICC), IEEE, 2017, 3 versions
- S Sevilla, Improving the Internet Architecture through Indirection and Virtualization, 2017, 3 versions
- U Goel, Internet measurements and application layer optimizations for faster web communications, 2017, 3 versions
- H Duan, X Yuan, C Wang, LightBox: SGX-assisted Secure Network Functions at Near-native Speed, 2017, 3 versions
- M Bierma, A Brown, T DeLano, et al., Locally Operated Cooperative Key Sharing (LOCKS) Computing, 2017
- M IDEA, mcTLS Connections, 2 versions
- S Huang, F Cuadrado, S Uhlig, Middleboxes in the Internet: a HTTP perspective Network Traffic Measurement, 2017, 4 versions
- A Silvestro, R Bifulco, F Schneider, X Fu, et al., MISE: Middleboxes SElection for Multi-domain Service Function Chains, 2017
- A Sivakumar, C Jiang, YS Nam, et al., NutShell: Scalable Whittled Proxy Execution for Low-Latency Web over Cellular Networks, 2017, 4 versions
- R Secchi, AC Mohideen, et al., Performance analysis of next generation web access via satellite, 2016
- R Netravali, J Mickens, Remote-Control Caching: Proxy-based URL Rewriting to Decrease Mobile Browsing Bandwidth, mit.edu
- X Liu, F Qian, Z Qian, Selective HTTPS traffic manipulation at middleboxes for BYOD devices, 2017, 3 versions
- J Han, S Kim, J Ha, D Han, SGX-Box: Enabling Visibility on Encrypted Traffic using a Secure Middlebox Module, 2017, 3 versions
- J Fan, C Guan, K Ren, Y Cui, et al., SPABox: Safeguarding Privacy During Deep Packet Inspection at a MiddleBox IEEE/ACM Transactions, 2017, 4 versions
- CA Wood, ME Mosko, System for a secure encryption proxy in a content centric network US Patent 20170331800A1, 2016

Z Durumeric, Z Ma, D Springall, R Barnes, et al., The security impact of HTTPS interception, 2017

C Wang, X Yuan, Y Cui, K Ren, Toward Secure Outsourced Middlebox Services: Practices, Challenges, and Beyond, 2017

Draft

History

Document history		
001	March 2017	Initial draft skeleton to seek contributions
002	April 2017	Draft skeleton to seek contributions
003	May 2017	Draft skeleton to seek contributions
004	May 2017	Draft skeleton to seek contributions
005	May 2017	Draft skeleton to seek contributions
006	June 2017	Moved chapter 6 to a new -2 document
007	Aug 2017	Added more info for a new skeleton and design abstraction stages
008	Jan 2018	Major changes/rework to whole document
009	Jan 2018	Minor editorial changes
010	Feb 2018	New figure in section 5.3.1
011	Feb 2018	Annexes updated
012	Apr 2018	Extensive editing to obtain a stable draft
013	April 2018	Minor editorial updates + few updates to comply with ETSI Drafting Rules. Version made publicly available