maTLS: How to Make TLS middlebox-aware?

Hyunwoo Lee, Zach Smith, Junghwan Lim, Gyeongjae Choi, Selin Chun, Taejoong Chung, Ted “Taekyoung” Kwon
Seoul National University, University of Luxembourg, Rochester Institute of Technology
Middleboxes

Client

Server
Middleboxes

- Web Application Firewalls
- Security Gateways
- Parental Controls
Middleboxes and Transport Layer Security

- Web Application Firewalls
- Security Gateways
- Parental Controls

: Cannot work!
Motivation for SplitTLS

To perform their functions
Middleboxes *split* the TLS session
- Private key sharing
- Custom root certificate
Session and Segment

To perform their functions
Middleboxes **split** the TLS session
- Private key sharing
- Custom root certificate
SplitTLS (1) Private Key Sharing

Client    Middlebox    Server

Certificate
CN: alice.com
Issuer: ca.com
SplitTLS (1) Private Key Sharing

Client

Middlebox

Server

Certificate
CN: alice.com
Issuer: ca.com

Certificate
CN: alice.com
Issuer: ca.com

Server transfers their private key and certificate
SplitTLS (1) Private Key Sharing

Client initiates a TLS handshake

Client

Middlebox

Server

Certificate
CN: alice.com
Issuer: ca.com

Certificate
CN: alice.com
Issuer: ca.com
SplitTLS (1) Private Key Sharing

Middlebox initiates another TLS handshake

Client → Middlebox → Server

Certificate
CN: alice.com
Issuer: ca.com

Certificate
CN: alice.com
Issuer: ca.com
SplitTLS (1) Private Key Sharing

Middlebox **impersonates** Server with the **transferred** key pair.

- **Certificate**
  - CN: alice.com
  - Issuer: ca.com
SplitTLS (1) Private Key Sharing

Client believes they have established a TLS session with Server, not Middlebox!
SplitTLS (2) Custom Root Certificate

Client

Middlebox

Server

Custom Root Certificate
CN: mitm.com
Issuer: mitm.com

Certificate
CN: alice.com
Issuer: ca.com
SplitTLS (2) Custom Root Certificate

Middlebox installs a root certificate in the client
SplitTLS (2) Custom Root Certificate

Client initiates a TLS handshake

Client

Middlebox

Server

Custom Root Certificate
CN: mitm.com
Issuer: mitm.com

Custom Root Certificate
CN: mitm.com
Issuer: mitm.com

Certificate
CN: alice.com
Issuer: ca.com
SplitTLS (2) Custom Root Certificate

Middlebox initiates another TLS handshake

Client

Middlebox

Server

Custom Root Certificate
CN: mitm.com
Issuer: mitm.com

Custom Root Certificate
CN: mitm.com
Issuer: mitm.com

Certificate
CN: alice.com
Issuer: ca.com
SplitTLS (2) Custom Root Certificate

Middlebox *impersonates* Server with the **forged** key pair

**Custom Root Certificate**
CN: mitm.com
Issuer: mitm.com

**Custom Root Certificate**
CN: mitm.com
Issuer: mitm.com

**Certificate**
CN: alice.com
Issuer: ca.com

**Forged Certificate**
CN: alice.com
Issuer: mitm.com

Generate a forged certificate with the name, alice.com
SplitTLS (2) Custom Root Certificate

Client

Middlebox

Server

Custom Root Certificate
CN: mitm.com
Issuer: mitm.com

Custom Root Certificate
CN: mitm.com
Issuer: mitm.com

Certificate
CN: alice.com
Issuer: ca.com

Forged Certificate
CN: alice.com
Issuer: mitm.com
SplitTLS (2) Custom Root Certificate

Client believes they have established a TLS session with Server, not Middlebox!
Problems in SplitTLS

No information for Client

Client ➔ Middlebox ➔ Server

Client

Middlebox

Server
Problems in SplitTLS - Authentication

Authentication

Client does not authenticate Server
Problems in SplitTLS - Authentication

Authentication

Client does not authenticate Server
Problems in SplitTLS - Authentication

Client 

Middlebox 

Server 

Authentication Client does not authenticate Server 

Not Expired Forged Certificate CN: alice.com Issuer: mitm.com 

Expired Certificate CN: alice.com Issuer: ca.com
Problems in SplitTLS - Confidentiality

- **Authentication**: Client does not authenticate Server
- **Confidentiality**: Client does not know whether or not the segment is encrypted with a strong ciphersuite
Problems in SplitTLS - Confidentiality

- **Authentication**: Client does not authenticate Server
- **Confidentiality**: Client does not know whether or not the segment is encrypted with a strong ciphersuite

RC4 or SHA-1?
Problems in SplitTLS - Confidentiality

- **Authentication**: Client does not authenticate Server
- **Confidentiality**: Client does not know whether or not the segment is encrypted with a strong ciphersuite
Problems in SplitTLS - Integrity

- **Authentication**: Client does not authenticate Server
- **Confidentiality**: Client does not know whether or not the segment is encrypted with a strong ciphersuite
- **Integrity**: Client cannot confirm that Server sent the message, or which middleboxes have modified it
Problems in SplitTLS - Integrity

Middlebox inserts the unwanted script!

- **Authentication**: Client does not authenticate Server
- **Confidentiality**: Client does not know whether or not the segment is encrypted with a strong ciphersuite
- **Integrity**: Client cannot confirm that Server sent the message, or which middleboxes have modified it
Problems in SplitTLS - Integrity

- **Authentication**: Client does not authenticate Server
- **Confidentiality**: Client does not know whether or not the segment is encrypted with a strong ciphersuite
- **Integrity**: Client cannot confirm that Server sent the message, or which middleboxes have modified it
Goal: Middlebox-aware TLS (maTLS)

Establish a secure session with middleboxes as well as overcoming the challenges in SplitTLS
Goal: Middlebox-aware TLS (maTLS)

Establish a secure session *with middleboxes* as well as overcoming the challenges in SplitTLS

<table>
<thead>
<tr>
<th>Authentication</th>
<th>Problems in SplitTLS</th>
<th>Solution in maTLS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Client can’t authenticate Server</td>
<td>Explicit Authentication</td>
</tr>
</tbody>
</table>
Goal: Middlebox-aware TLS (maTLS)

Establish a secure session *with middleboxes* as well as overcoming the challenges in SplitTLS

<table>
<thead>
<tr>
<th></th>
<th>Problems in SplitTLS</th>
<th>Solution in maTLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Client can’t authenticate Server</td>
<td>Explicit Authentication</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Client can’t know if each of the segments has been encrypted with strong ciphersuites</td>
<td>Security Parameter Verification</td>
</tr>
</tbody>
</table>
Goal: Middlebox-aware TLS (maTLS)

Establish a secure session *with middleboxes* as well as overcoming the challenges in SplitTLS

<table>
<thead>
<tr>
<th></th>
<th>Problems in SplitTLS</th>
<th>Solution in maTLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Client can’t authenticate Server</td>
<td>Explicit Authentication</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Client can’t know if each of the segments has been encrypted with strong ciphersuites</td>
<td>Security Parameter Verification</td>
</tr>
<tr>
<td>Integrity</td>
<td>Client can’t confirm (1) who actually sent the message (2) if it has been modified</td>
<td>Valid Modification Checks</td>
</tr>
</tbody>
</table>
Auditable Middleboxes

Certificate Authority

Middlebox (mb.com)

Middlebox Transparency Log Server

Middlebox Certificate
CN: mb.com
Issuer: ca.com

Middlebox Certificate
CN: mb.com
Issuer: ca.com

Auditable Middleboxes

Middleboxes that have their own *middlebox certificates* logged in a *middlebox transparency* log server
Auditable Middleboxes

Certificate Authority

Middlebox Certificate
CN: mb.com
Issuer: ca.com

Middlebox Transparency Log Server

Information about Middlebox
- Type of Service
- URL
- Permission

Middlebox (mb.com)
Auditable Middleboxes

Certificate Authority

Middlebox (mb.com)

Middlebox Transparency Log Server

Middlebox Certificate
CN: mb.com
Issuer: ca.com

Middlebox Certificate
CN: mb.com
Issuer: ca.com
Advantages of Auditable Middleboxes

- No impersonation
- Awareness
- Auditability
- Revocability
Advantages of Auditable Middleboxes

- **No impersonation**
  
  Middleboxes now have their *own key pairs* and do not need to impersonate others (in TLS)

- **Awareness**

- **Auditability**

- **Revocability**
Advantages of Auditable Middleboxes

- **No impersonation**  
  Middleboxes now have their *own key pairs* and do not need to impersonate others (in TLS)

- **Awareness**  
  Anyone can know the name and properties of a middlebox from its *middlebox certificate*

- **Auditability**

- **Revocability**
Advantages of Auditable Middleboxes

- **No impersonation**
  Middleboxes now have their *own key pairs* and do not need to impersonate others (in TLS)

- **Awareness**
  Anyone can know the name and properties of a middlebox from its *middlebox certificate*

- **Auditability**
  Any interested parties can check for fraudulent certificates using the *middlebox transparency* system

- **Revocability**
Advantages of Auditable Middleboxes

- **No impersonation**
  Middleboxes now have their *own key pairs* and do not need to impersonate others (in TLS)

- **Awareness**
  Anyone can know the name and properties of a middlebox from its *middlebox certificate*

- **Auditability**
  Any interested parties can check for fraudulent certificates using the *middlebox transparency* system

- **Revocability**
  Any incorrect middleboxes can be blocked following the *certificate revocation mechanisms* (e.g., CRL or OCSP)
Security Goals of maTLS

- Server Authentication
- Middlebox Authentication
- Segment Secrecy
- Individual Secrecy
- Data Source Authentication
- Modification Accountability
- Path Integrity
Security Goals of maTLS - Authentication

- Server Authentication

Certificate:
CN: alice.com
Issuer: ca2.com

Certificate:
CN: mb.com
Issuer: ca1.com
Security Goals of maTLS - Authentication

- Server Authentication
- Middlebox Authentication
Security Goals of maTLS - Authentication

- **Server Authentication**
- **Middlebox Authentication**

Explicit Authentication
Explicit Authentication

Each entity sends its certificate (with its signed certificate timestamp)

No impersonation
Security Goals of maTLS - Confidentiality

- Confidentiality
- High TLS version with strong ciphersuite
- Segment Secrecy

Client → Middlebox → Server

- Middlebox Certificate
  - CN: mb.com
  - Issuer: ca1.com
- Server Certificate
  - CN: alice.com
  - Issuer: ca2.com
Security Goals of maTLS - Confidentiality

- Segment Secrecy
- Individual Secrecy

Certificate
- CN: mb.com
- Issuer: ca1.com

Certificate
- CN: alice.com
- Issuer: ca2.com
Security Goals of maTLS - Confidentiality

- Segment Secrecy
- Individual Secrecy

Client → Middlebox → Server

- Middlebox Certificate:
  - CN: mb.com
  - Issuer: ca1.com

- Server Certificate:
  - CN: alice.com
  - Issuer: ca2.com

Security Parameter Verification
Security Parameter Verification

Each entity describes information about its related segment(s)
Security Parameter Verification

Each entity describes information about its related segment(s)

- **Version**, **Ciphersuite**, ...
- **TLS version**, **Ciphersuite**
- **Transcript of Handshake**, **Hash of Master Secret**

**Segment Secrecy**

**Individual Secrecy**
Security Parameter Verification

Each entity describes information about its related segment(s)

- **Segment Secrecy**
- **Individual Secrecy**

TLS version
Ciphersuite

Transcript of Handshake
Hash of Master Secret

No low TLS versions and weak ciphersuites
Security Parameter Verification

Each entity describes information about its related segment(s)

- TLS version
- Ciphersuite
- Transcript of Handshake
- Hash of Master Secret

Segment Secrecy
Individual Secrecy

No low TLS versions and weak ciphersuites
Confirmation of different segment keys
Security Goals of maTLS - Integrity

Client → Middlebox → Server

The message is from Server

Checkmark: Data Source Authentication

- Middlebox Certificate
  - CN: mb.com
  - Issuer: ca1.com

- Certificate
  - CN: alice.com
  - Issuer: ca2.com
Security Goals of maTLS - Integrity

- Data Source Authentication
- Modification Accountability

The message has been modified by Middlebox
Security Goals of maTLS - Integrity

- Data Source Authentication
- Modification Accountability
- Path Integrity

The message has passed through the established order.

Client

Middlebox

Server

Certificate
CN: alice.com
Issuer: ca2.com

Certificate
CN: mb.com
Issuer: ca1.com
Security Goals of maTLS - Integrity

- Data Source Authentication
- Modification Accountability
- Path Integrity

Client → Middlebox
- Middlebox Certificate
  CN: mb.com
  Issuer: ca1.com

Middlebox → Server
- Certificate
  CN: alice.com
  Issuer: ca2.com

Valid Modification Checks
Valid Modification Checks

Each entity describes information about its modification by using HMAC (The HMAC key is called an *accountability key*).

$$m \rightarrow m'$$

| ID | $H(m)$ | $\text{HMAC}(H(m')||H(m))$ |

* Optimization on a Modification Log is described in the paper
Valid Modification Checks

Each entity describes information about its modification by using HMAC (The HMAC key is called an *accountability key*).

\[ m \rightarrow m' \]

| ID   | \( H(m) \) | \( \text{HMAC}(H(m') || H(m)) \) |

*Optimization on a Modification Log is described in the paper.*
**Valid Modification Checks**

Each entity describes information about its modification by using HMAC (The HMAC key is called an *accountability key*).

| ID   | H(m) | HMAC(H(m')||H(m)) |
|------|------|------------------|
| m    | m'   |                  |

Confirmation of who sends and who modifies the message.

* Optimization on a Modification Log is described in the paper.
Valid Modification Checks

Each entity describes information about its modification by using HMAC (The HMAC key is called an *accountability key*)

| m̃ → m' | ID | H(m) | HMAC(H(m')||H(m)) |
|---------|----|------|-------------------|

**Confirmation of who sends and who modifies the message**

**Confirmation of the order of middleboxes**

*Optimization on a Modification Log is described in the paper*
Summary of Audit Mechanisms

- Explicit Authentication
- Server Authentication
- Middlebox Authentication
- Server Certificate and Middlebox Certificates (with their signed certificate timestamps)
Summary of Audit Mechanisms

- Explicit Authentication
- Server Authentication
  - Middlebox Authentication
- Security Parameter Verification
- Segment Secrecy
  - Individual Secrecy

- Server Certificate and Middlebox Certificates
  (with their signed certificate timestamps)
- Security Information Blocks
Summary of Audit Mechanisms

- Explicit Authentication
- Server Authentication
- Middlebox Authentication
- Server Certificate and Middlebox Certificates (with their signed certificate timestamps)
- Security Parameter Verification
- Segment Secrecy
- Individual Secrecy
- Security Information Blocks
- Valid Modification Checks
- Data Source Authentication
- Modification Accountability
- Path Integrity
- Modification Log Blocks
maTLS Handshake

Client

Middlebox

Certificate
CN: mb.com
Issuer: ca1.com

Server

Certificate
CN: alice.com
Issuer: ca2.com
maTLS Handshake

Client\hspace{1cm} alice.com MATLS \hspace{1cm} Middlebox\hspace{1cm} alice.com MATLS \hspace{1cm} Server

- ClientHello and ServerHello,
ClientHello and ServerHello,
Each segment negotiates its TLS version and ciphersuite
Each entity establishes HMAC keys (accountability keys)
**maTLS Handshake**

- Client
- Middlebox
- Server

**Middlebox Certificate**
- CN: mb.com
- Issuer: ca1.com

**Certificate**
- CN: alice.com
- Issuer: ca2.com

- ✓ ClientHello and ServerHello,
  Each segment negotiates its TLS version and ciphersuite
  Each entity establishes HMAC keys (accountability keys)

- ✓ Certificate, ➡️ Explicit Authentication
maTLS Handshake

- **ClientHello and ServerHello**, Each segment negotiates its TLS version and ciphersuite
- **Certificate**, Each entity establishes HMAC keys (accountability keys)
- **Explicit Authentication**
- **ServerKeyExchange and ClientKeyExchange**, Each segment establishes its master secret
maTLS Handshake

Client
Version, Ciphersuite, ...

Middlebox
Version, Ciphersuite, ...

Server
Version, Ciphersuite, ...

✔ Finished
Each segment confirms the transcript of their handshake
maTLS Handshake

Client
Version, Ciphersuite, ...

Middlebox
Version, Ciphersuite, ...
Version, Ciphersuite, ...

Server
Version, Ciphersuite, ...

✅ Finished
Each segment confirms the transcript of their handshake

✅ ExtendedFinished ➔ Security Parameter Verification
maTLS Record

Client
ID: Client
Prior Hash: none
HMAC(H(m_0)||none)

Middlebox
ID: Middlebox
Prior Hash: H(m_0)
HMAC(H(m_1)||H(m_0))

Server

Message flow

Data Exchange
Valid Modification Checks

* Optimization on a Modification Log is described in the paper
Optimization on a Modification Log is described in the paper.
Security Verification

✓ Security verification of maTLS through Tamarin

✓ Dolev-Yao adversary

- Can capture all the messages delivered on the air
- Can insert/drop/alter/reorder messages
- Can corrupt long-term keys

✓ Seven lemmas (security goals in first-order logic)

Example of Server Authentication

```
All C S nonces #tc.
C_HandshakeComplete(C, S, nonces)@tc
==> Ex #ts.
S_HandshakeComplete(C, S, nonces)@ts & (#ts < #tc)
```

✓ The result shows that the maTLS protocol is secure

* The implementation can be found at https://github.com/middlebox-aware-tls/matls-tamarin.git
Evaluation Setting

All the applications are implemented in C with OpenSSL (for maTLS)

Client

Client-side Middlebox

Located in Seoul National University

Client: Intel Broadwell CPU at 3.30GHz with 1GB Memory
Client-side Middlebox: Intel Core i7 at 2.30GHz with 1GB Memory

Located in
1) AWS Seoul (Intra-Country)
2) AWS Tokyo (Intra-Region)
3) AWS Virginia (Inter-Region)

Server and Server-side Middlebox: Intel Xeon CPU E5-3676 at 2.40GHz with 1GB Memory

* The implementation can be found at https://github.com/middlebox-aware-tls/maTLS-implementation.git
Evaluation – HTTP Load Time

- **HTTP Load Time**: The TLS handshake and the HTTP message exchange (GET and RESPONSE)
- **Data Transfer Time**: Only the HTTP message exchange (GET and RESPONSE)

<table>
<thead>
<tr>
<th>Testbed</th>
<th>$C-MB_C$</th>
<th>$MB_C-MB_S$</th>
<th>$MB_S - S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra-country</td>
<td>1.136ms</td>
<td>4.944ms</td>
<td>0.551ms</td>
</tr>
<tr>
<td>Intra-region</td>
<td>1.136ms</td>
<td>35.896ms</td>
<td>0.537ms</td>
</tr>
<tr>
<td>Inter-region</td>
<td>1.136ms</td>
<td>192.818ms</td>
<td>0.610ms</td>
</tr>
</tbody>
</table>
Evaluation – HTTP Load Time

The maTLS protocol introduces a slight delay (10.22ms – 32.52ms) compared to SplitTLS and mcTLS

- **HTTP Load Time**: The TLS handshake and the HTTP message exchange (GET and RESPONSE)
- **Data Transfer Time**: Only the HTTP message exchange (GET and RESPONSE)
Evaluation – HTTP Load Time

- **HTTP Load Time:** The TLS handshake and the HTTP message exchange (GET and RESPONSE)
- **Data Transfer Time:** Only the HTTP message exchange (GET and RESPONSE)

Three schemes show similar delay time for data transfer.

<table>
<thead>
<tr>
<th>Testbed</th>
<th>$C - MB_C$</th>
<th>$MB_C - MB_S$</th>
<th>$MB_S - S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra-country</td>
<td>1.136ms</td>
<td>4.944ms</td>
<td>0.551ms</td>
</tr>
<tr>
<td>Intra-region</td>
<td>1.136ms</td>
<td>35.896ms</td>
<td>0.537ms</td>
</tr>
<tr>
<td>Inter-region</td>
<td>1.136ms</td>
<td>192.818ms</td>
<td>0.610ms</td>
</tr>
</tbody>
</table>
Evaluation – HTTP Load Time

HTTP Load Time: The TLS handshake and the HTTP message exchange (GET and RESPONSE)

Data Transfer Time: Only the HTTP message exchange (GET and RESPONSE)

We conclude that the maTLS overhead is mainly due to the setup of an maTLS session

<table>
<thead>
<tr>
<th></th>
<th>Intra-country</th>
<th>Intra-region</th>
<th>Inter-region</th>
</tr>
</thead>
<tbody>
<tr>
<td>maTLS</td>
<td>0.00ms</td>
<td>0.00ms</td>
<td>0.00ms</td>
</tr>
<tr>
<td>mcTLS</td>
<td>0.00ms</td>
<td>0.00ms</td>
<td>0.00ms</td>
</tr>
<tr>
<td>SplitTLS</td>
<td>0.00ms</td>
<td>0.00ms</td>
<td>0.00ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Time (ms)</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-region</td>
<td>1.136ms</td>
<td>192.818ms</td>
<td>0.610ms</td>
</tr>
</tbody>
</table>
Once the session is established, maTLS provides similar performance to the others while preserving all security merits that we have discussed.

- HTTP Load Time: The TLS handshake and the HTTP message exchange (GET and RESPONSE)
- Data Transfer Time: Only the HTTP message exchange (GET and RESPONSE)
Conclusion

SplitTLS is risky
- Client is not aware of the middleboxes involved
- Client is forced to fully trust behavior of middleboxes

Auditable Middlebox
- Middlebox Certificate
- Middlebox Transparency System

Middlebox-aware TLS (maTLS)
- Explicit Authentication
- Security Parameter Verification
- Valid Modification Checks
fin.

email: hwlee2014@mmlab.snu.ac.kr

project webpage: https://middlebox-aware-tls.github.io

source codes: https://github.com/middlebox-aware-tls
Why Middleboxes?

- Acceptable Use Policy
- Marware and Threat Protection
- IoT Endpoint Protection
- Unpatched Endpoint Protection
- Crypto Security Audit

* I get the use cases from a draft of the RFC document titled “TLS 1.3 Impact on Network-Based Security”
**Why Individual Secrecy?**

⚠️ It is known that initialization vector should not be reused

⚠️ Without Individual Secrecy, confidentiality is undermined

This happened when the same keystream is used across the session and the middlebox modified the message

\[
C_1 = P_1 \oplus F(\text{Key}, IV) \\
C_2 = P_2 \oplus F(\text{Key}, IV) \\
C_1 \oplus C_2 = P_1 \oplus P_2
\]

Middlebox

The same keystream with the different message

✅ It is desirable to use different segment keys across the session
Why Path Integrity?

Message Flow

Client → Anonymizer → Firewall → Server

The data is anonymized and then the firewall read it

Client → Firewall → Anonymizer → Server

The firewall read the data and then it is anonymized
Session Establishment Approach (1)

Top-down approach

Server determines a TLS version, a ciphersuite, and extensions
Session Establishment Approach (2)

**Bottom-up approach**

A TLS version, a ciphersuite, and extensions are selected on a segment basis.
Difference from mcTLS

mcTLS does not achieve Individual Secrecy
The same keystream is used across the session, which might undermine the confidentiality of the session

maTLS establishes different segment keys in different segments

mcTLS requires all the entities support the protocol
Since the server determines the extensions among the “intersection” of the supported extensions by all the entities

maTLS allows a partial maTLS session
Evaluation – Scalability of Three Audit Mechanisms

SPV: Security Parameter Verification / EA: Explicit Authentication / VMC: Valid Modification Checks
Evaluation – Scalability of Three Audit Mechanisms

SPV: Security Parameter Verification / EA: Explicit Authentication / VMC: Valid Modification Checks

0.063ms per middlebox

0.045ms per middlebox

0.026ms for 8 middleboxes
We conclude that the audit mechanisms can achieve their goals without incurring a substantial delay.

SPV: Security Parameter Verification / EA: Explicit Authentication / VMC: Valid Modification Checks
Modification Log

- A series of HMACs
- End point: Server, Client, or a valid end-point middlebox such as a cache proxy

$H(k, m)$: The keyed hash function with $k$, applying to $m$
$H(m)$: The hash function, applying to $m$

- Writer: HTTP Header Enrichment, Optimizer (adding JavaScript) ($m \rightarrow m'$)

$ID_{mb}$ modifies $m$ into $m'$
Modification Log Verification

Client knows
- \(ak_{s,c}\): The accountability key with the server
- \(ak_{m,c}\): The accountability key with the MB
Client knows

- $ak_{s,c}$: The accountability key with the server
- $ak_{m,c}$: The accountability key with the MB
- $H(m')$: The hash value of the received message

By hashing the received message, the client can know $H(m')$
Modification Log Verification

Client knows
- $ak_{s,c}$: The accountability key with the server
- $ak_{m,c}$: The accountability key with the MB
- $H(m')$: The hash value of the received message
Modification Log Verification

Client knows
- $ak_{s,c}$: The accountability key with the server
- $ak_{m,c}$: The accountability key with the MB
- $H(m')$: The hash value of the received message

From these hashes, the client can confirm MB modifies $m$ into $m'$
Modification Log Verification

From this hash, the client can confirm the server generates $m$, even though the client cannot confirm $m$ itself.

Client knows
- $ak_{s,c}$: The accountability key with the server
- $ak_{m,c}$: The accountability key with the MB
- $H(m')$: The hash value of the received message
Modification Log Verification

From two verifications, the client can confirm the server generates $m$ and mb changes it into $m'$, without any invalid modification.

Client knows

- $ak_{s,c}$: The accountability key with the server
- $ak_{m,c}$: The accountability key with the MB
- $H(m')$: The hash value of the received message