mcTLS: enabling secure in-network functionality in TLS

David Naylor
Kyle Schomp
Matteo Varvello
Ilias Leontiadis
Jeremy Blackburn
Diego Lopez
Dina Papagiannaki
Pablo Rodriguez
Rodriguez
Peter Steenkiste
Observation 1:
Use of Encryption is Increasing
OBSERVATION 2: In-Network Functionality is Widespread
GOAL:

Encryption & In-Network Functionality
Value-Added Services
Opt-in services that benefit end users.

Administrator-Mandated
Help the company/network; for users, just a fact of life.

Unauthorized
Not necessary for network & not beneficial for user.
Value-Added Services
Opt-in services that benefit end users.

Administrator-Mandated
Help the company/network; for users, just a fact of life.

Unauthorized
Not necessary for network & not beneficial for user.
GOAL:

Encryption & In-Network Functionality
mcTLS

Encryption &
In-Network Functionality
mcTLS
Encryption & In-Network Functionality

TLS + Middleboxes
mcTLS Design Ideas
mcTLS Handshake
Performance Evaluation
TLS

CONSISTS OF:

- Handshake Protocol for session setup
  - &
  - Record Protocol for data transfer

AND GIVES US THREE SECURITY PROPERTIES:

1. Entity Authentication
2. Payload Secrecy
3. Payload Integrity
TLS + middleboxes is broken

1. Company installs root cert on client
2. Firewall fabricates a cert for foo.com
3. Client accepts fake cert because it’s signed by company’s root cert
4. Firewall opens separate TLS connection to foo.com
TLS was designed for 2 parties

1. No mechanism to authenticate middleboxes.
2. Client has no security guarantees past middlebox.
3. Middleboxes have full read/write access.
mcTLS
Encryption & In-Network Functionality

- TLS + Middleboxes
- mcTLS Design Ideas
- mcTLS Handshake
- Performance Evaluation
Design requirements for mcTLS

Maintain TLS security properties:

1. Entity Authentication
2. Payload Secrecy
3. Payload Integrity

Plus two new ones:

4. Visibility & Control
5. Least Privilege
Design requirements for mcTLS

MAINTAIN TLS SECURITY PROPERTIES:

1. Entity Authentication
2. Payload Secrecy
3. Payload Integrity

PLUS TWO NEW ONES:

4. Visibility & Control
5. Least Privilege
Most middleboxes do not need read/write access to all data

<table>
<thead>
<tr>
<th>Parental Filter</th>
<th>Packet Pacer</th>
<th>IDS</th>
<th>WAN Optimizer</th>
<th>Caching</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Headers</td>
<td>☀</td>
<td>☀</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td></td>
<td>☀</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| HTTP Response   |              |     |               |         |             |
| Headers         | ☀            | ☀   |               |         |             |
| Body            |              | ☀   |               |         |             |

○ = read only
● = read/write

Access all data
Write access

○ ○ ○ ○
○ ○ ○ ○
Idea #1: Encryption Contexts
(for access control)

send(data, context)

Context 1: “Request Headers”
  - Read-Only:
  - Read/Write:

Context 2: “Request Body”
  - Read-Only:
  - Read/Write:

Context 3: “Response Headers”
  - Read-Only:
  - Read/Write:

Context 4: “Response Body”
  - Read-Only:
  - Read/Write:
Idea #1: Encryption Contexts (for access control)

**TLS** uses *one key* for encryption and MAC:
Idea #1: Encryption Contexts (for access control)

mcTLS uses three keys to separate read-only and read/write access:

- $K_{\text{readers}}$
- $K_{\text{writers}}$
- $K_{\text{endpoints}}$

Readers, Writers, & Endpoints check to detect 3rd party changes

Writers & Endpoints check to detect reader changes

Endpoints check to detect writer changes
Idea #1: Encryption Contexts (for access control)

Each context has a read key and a write key:

Context 1: “Request Headers”
Read: \( K_{\text{readers}} \)
Write: \( K_{\text{writers}} \)

Context 2: “Request Body”
Read: \( K_{\text{readers}} \)
Write: \( K_{\text{writers}} \)
Encryption contexts example

Context 1: “Request Headers”
- Read: [key]
- Write: [key]

Context 2: “Request Body”
- Read: [key]
- Write: [key]

Context 3: “Response Headers”
- Read: [key]
- Write: [key]

Context 4: “Response Body”
- Read: [key]
- Write: [key]
Design requirements for mcTLS

MAINTAIN TLS SECURITY PROPERTIES:

1. Entity Authentication
2. Payload Secrecy
3. Payload Integrity

PLUS TWO NEW ONES:

4. Visibility & Control
5. Least Privilege

Multiple Encryption Contexts
Design requirements for mcTLS

**MAINTAIN TLS SECURITY PROPERTIES:**

1. Entity Authentication
2. Payload Secrecy
3. Payload Integrity

**PLUS TWO NEW ONES:**

4. Visibility & Control
5. Least Privilege

Multiple Encryption Contexts
Idea #1: Contributory Context Keys (for endpoint agreement)

Client and server generate part of each context key:

Middlebox only learns key if client and server agree on its permissions.
Design requirements for mcTLS

**MAINTAIN TLS SECURITY PROPERTIES:**

1. Entity Authentication
2. Payload Secrecy
3. Payload Integrity

**PLUS TWO NEW ONES:**

4. Contributory Context Keys
5. Visibility & Control
6. Least Privilege
7. Multiple Encryption Contexts
mcTLS
Encryption & In-Network Functionality

TLS + Middleboxes
mcTLS Design Ideas
mcTLS Handshake
Performance Evaluation
mcTLS Design Ideas

mcTLS Handshake

Performance Evaluation

TLS + Middleboxes

mcTLS

Encryption & In-Network Functionality
Handshake Goals

**TLS**
- Authenticate server ✓
- Establish session key ✓

**mcTLS**
- Authenticate server ✓
- Authenticate middlebox ✓
- Distribute context keys ✓
Hello

List of middleboxes, contexts, and permissions
Hello

Hello + Cert

Server Key Exchange

Server Hello Done

Middlebox Key Exchange

Middlebox Hello Done

✓ Auth server

✓ Auth mbox

CLIENT

MIDDLEBOX

SERVER
Hello

Client Key Exchange

Client Context Secrets

Change Cipher Spec

Finished

Hello + Cert

Server Key Exchange

Server Hello Done

Change Cipher Spec

Middlebox Hello Done

Client Key Exchange

Middlebox Key Exchange

Client Context Secrets

Middlebox

Context 1: read secret write secret

Context 2: read secret

...

Client

Context 1: read secret write secret

Context 2: read secret write secret

...

Mbox

K

Endpoints

K

Client - Mbox

K

Client secrets

K

Endpoints

K

Endpoints

K

Client - Mbox

K

Client secrets

K

Endpoints

K

Endpoints

K

Client secrets

K

Endpoints

K

Endpoints
For each context:

Client read secret + server read secret
client random + server random
“reader keys”

Client write secret + server write secret
client random + server random
“writer keys”
mcTLS
Encryption & In-Network Functionality

TLS + Middleboxes
mcTLS Design Ideas
mcTLS Handshake
Performance Evaluation
mcTLS
Encryption &
In-Network Functionality

- TLS + Middleboxes
- mcTLS Design Ideas
- mcTLS Handshake
- Performance Evaluation
mcTLS adds functionality to TLS. Does it add overhead?

**Data Overhead**
- context key material + certificates

**CPU Overhead**
- context key generation + key exchange

**Time Overhead**
- handshake duration
mcTLS increases handshake size

![Bar chart showing the size (kB) for different contexts and mailboxes with mcTLS and TLS compared.](image)
mcTLS can increase server load
mcTLS does not increase time to first byte
mcTLS
Encryption &
In-Network Functionality

- TLS + Middleboxes
- mcTLS Design Ideas
- mcTLS Handshake
- Performance Evaluation
Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS

David Naylor*, Kyle Schomp†, Matteo Varvello‡, Ilias Leontiadis†, Jeremy Blackburn‡, Diego Lopez‡, Konstantina Papagiannaki‡, Pablo Rodríguez Rodríguez‡, and Peter Steenkiste*

*Carnegie Mellon University †Case Western Reserve University ‡Telefónica Research

ABSTRACT
A significant fraction of Internet traffic is now encrypted and HTTPS will likely be the default in HTTP/2. However, Transport Layer Security (TLS), the standard protocol for encryption in the Internet, assumes that all

1. INTRODUCTION
The increased personalization of Internet services and rising concern over users’ privacy on the Internet has led to a number of services (e.g., Facebook, Twitter, and Google) offering access solely over HTTPS. HTTPS

mcTLS is a secure communication protocol that extends TLS to allow endpoints to incorporate trusted middleboxes into secure sessions.

- **No Transparent Middleboxes**: Both endpoints explicitly approve each middlebox.
- **Least Privilege**: Middleboxes see only what they need to do their jobs.
- **Middlebox Authentication**: Client and server can verify the identity of each middlebox.
- **No Custom Root Certificates**: Overall security is not undermined by requiring users to install root certificates.

Check out our SIGCOMM 2015 paper
mcTLS: enabling secure in-network functionality in TLS

David Naylor
Kyle Schomp
Matteo Varvello
Ilias Leontiadis
Jeremy Blackburn
Diego Lopez
Dina Papagiannaki
Pablo Rodriguez
Rodriguez
Peter Steenkiste