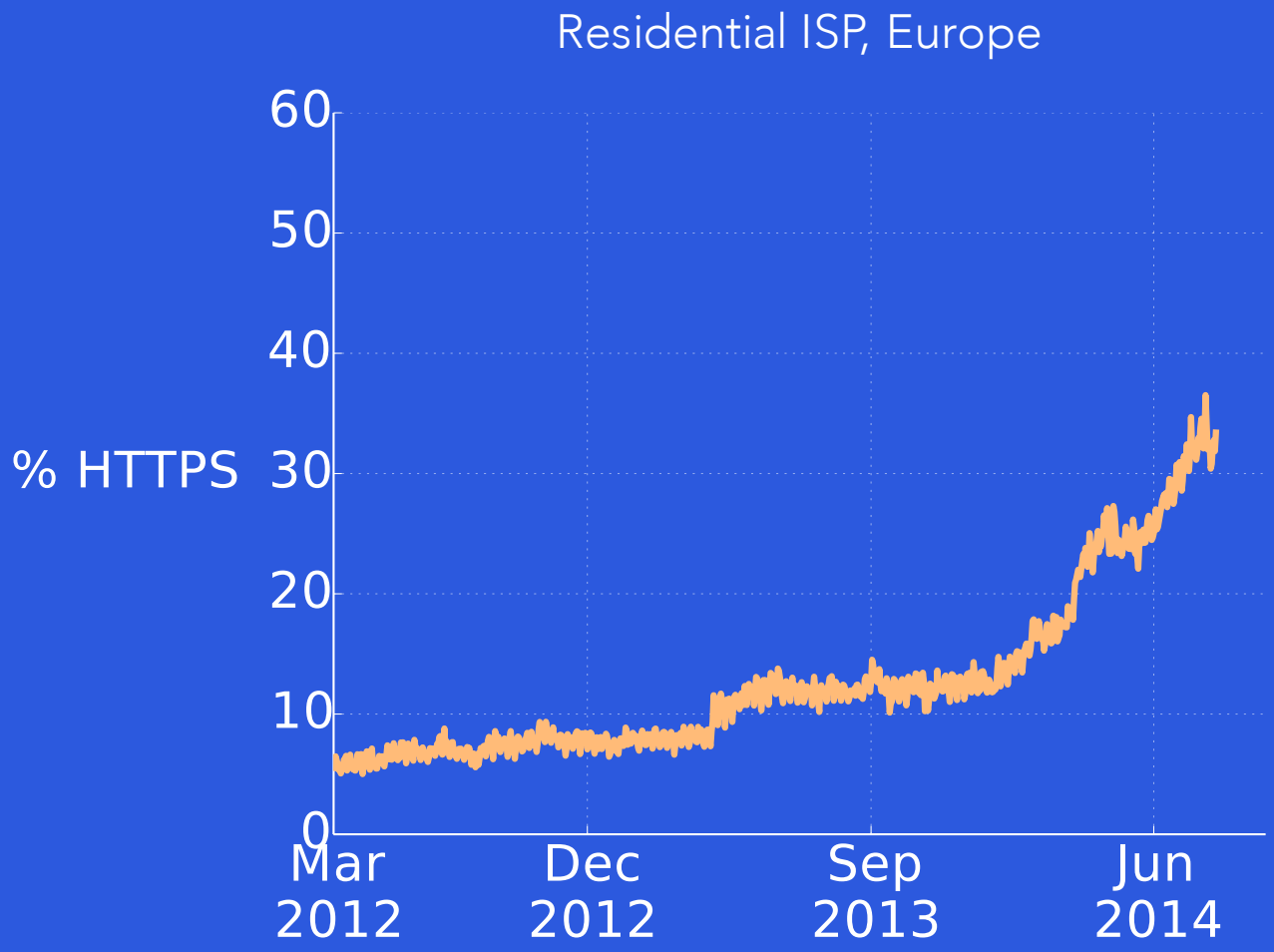
mcTLS:
enabling secure in-network functionality in TLS

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

Carnegie Mellon University
Case Western Reserve University
Telefonica



OBSERVATION 1:

Use of Encryption is Increasing

CACHING

COMPRESSION

PARENTAL FILTER

VIRUS SCANNER

PACKET PACING

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

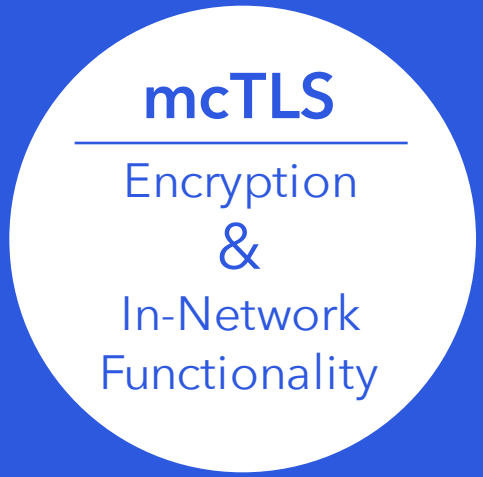
&

In-Network Functionality

mcTLS

Encryption
&
In-Network
Functionality

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



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

TLS

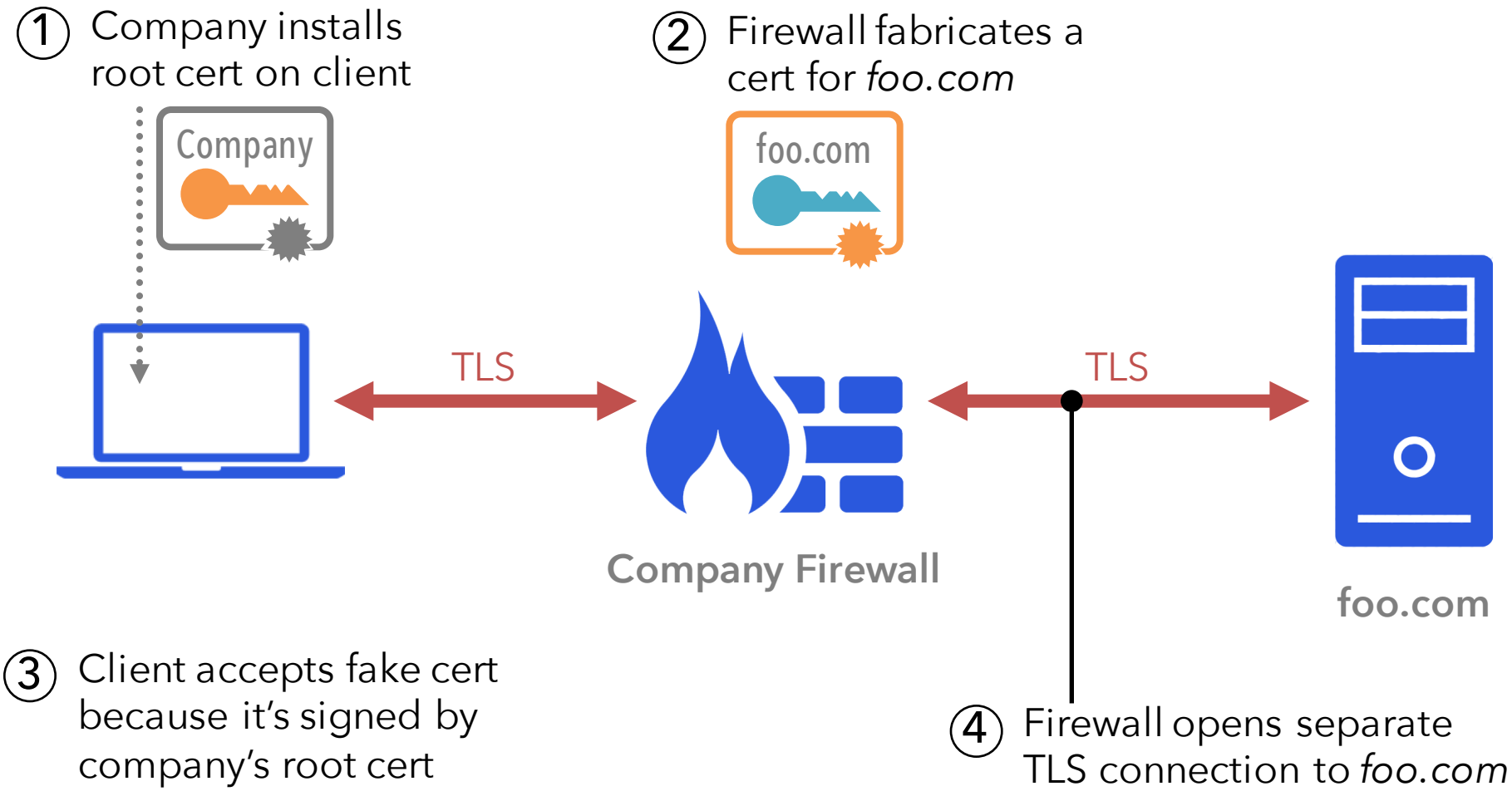
CONSISTS OF:

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

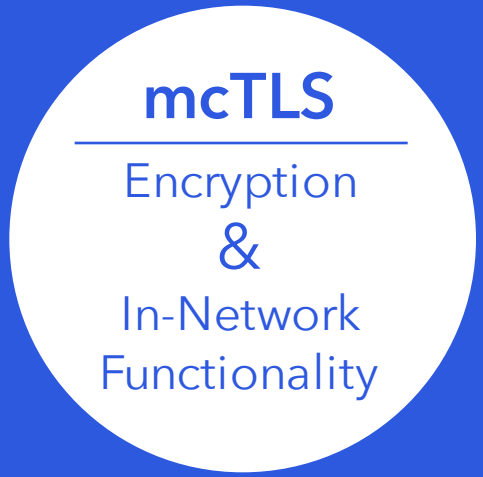


TLS was designed for 2 parties

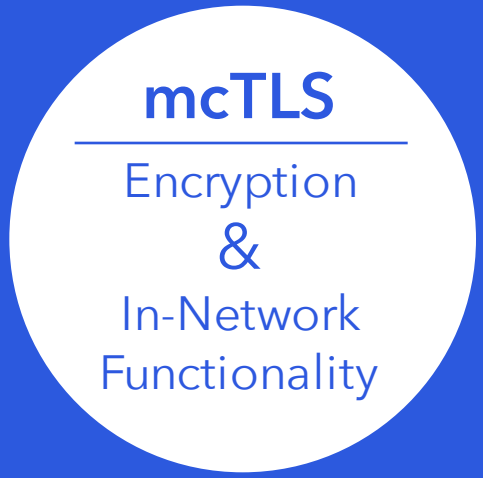
1. No mechanism to authenticate middleboxes.

2. Client has no security guarantees past middlebox.

3. Middleboxes have full read/write access.



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



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

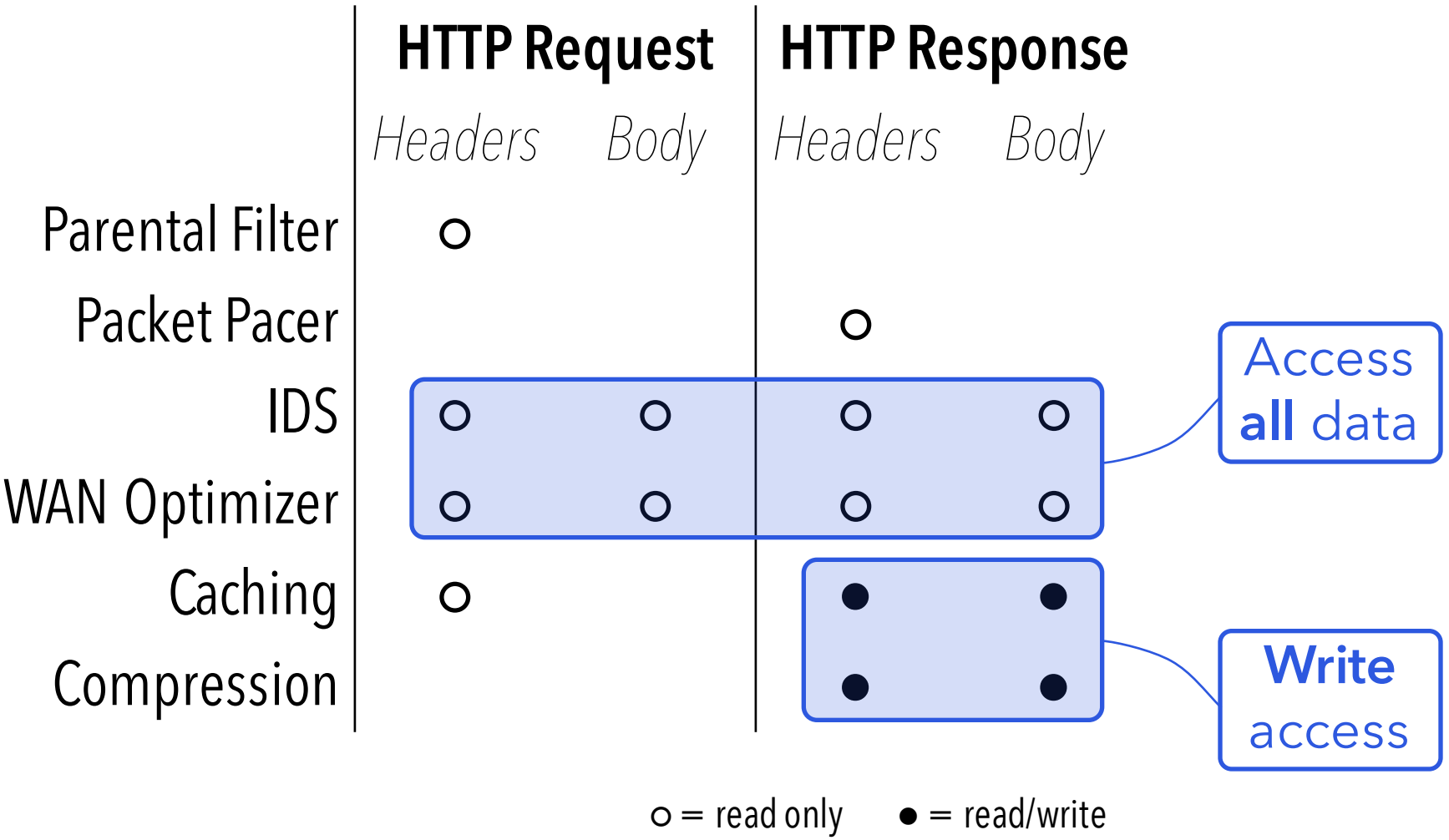
MAINTAIN TLS SECURITY PROPERTIES:

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

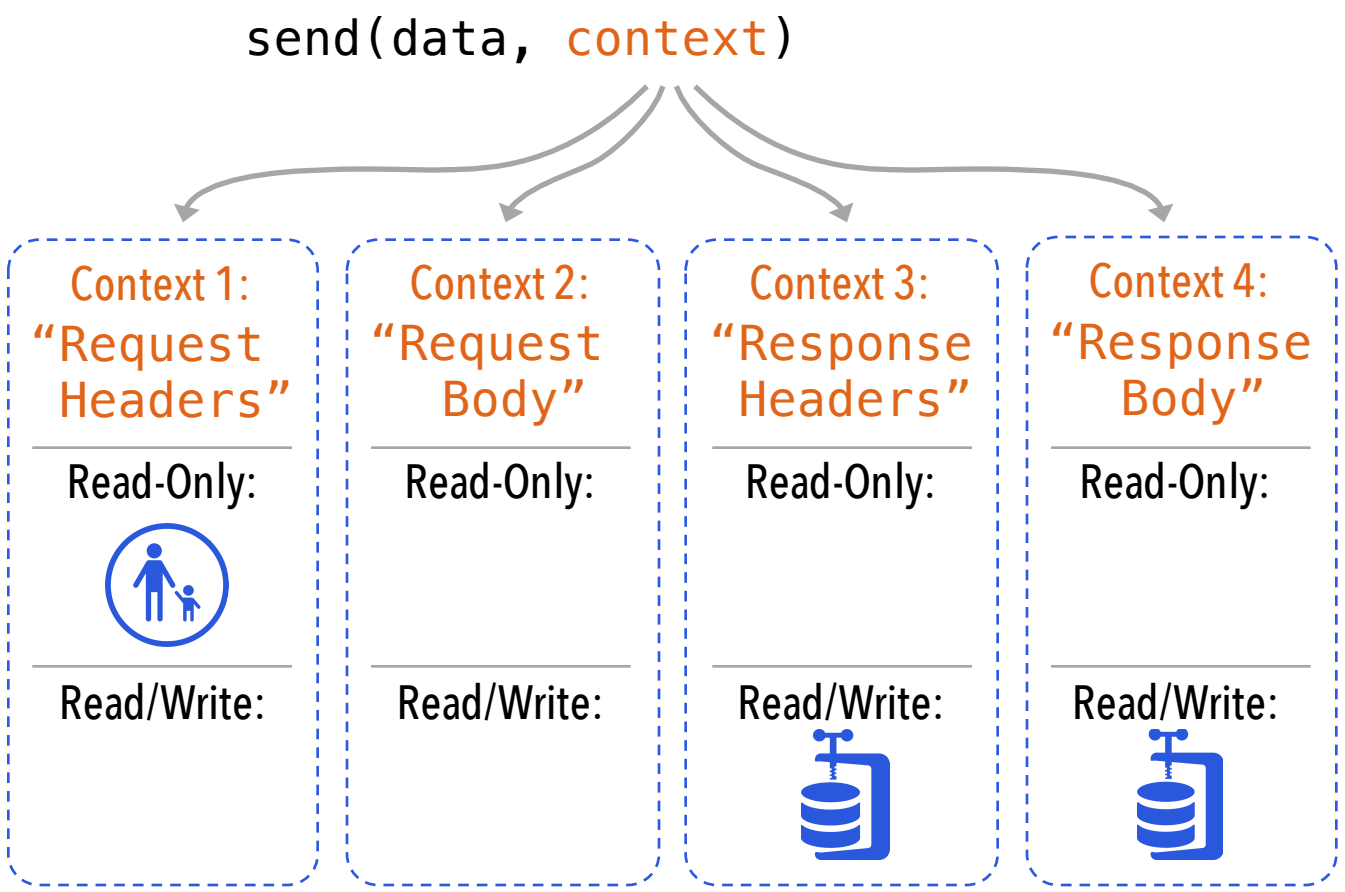
PLUS TWO NEW ONES:

4. Visibility & Control
5. Least Privilege

Most middleboxes do not need *read/write* access to *all* data

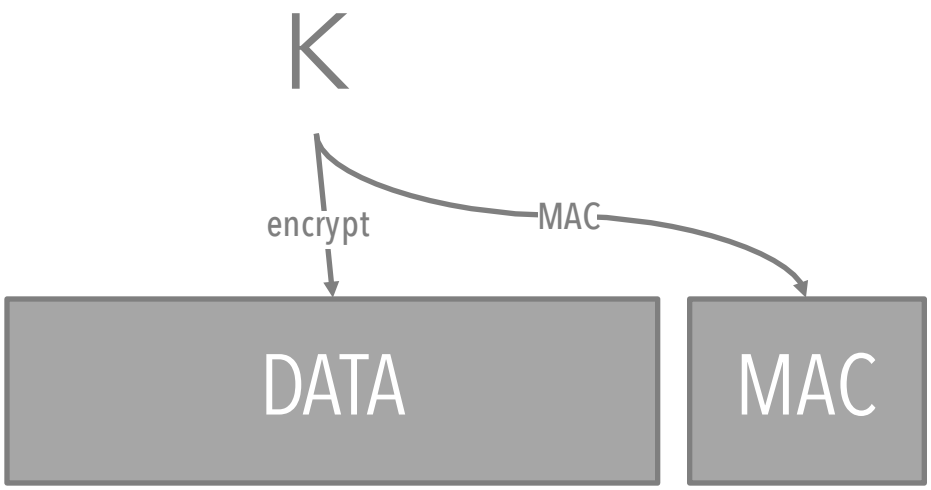


Idea #1: Encryption Contexts
(for access control)



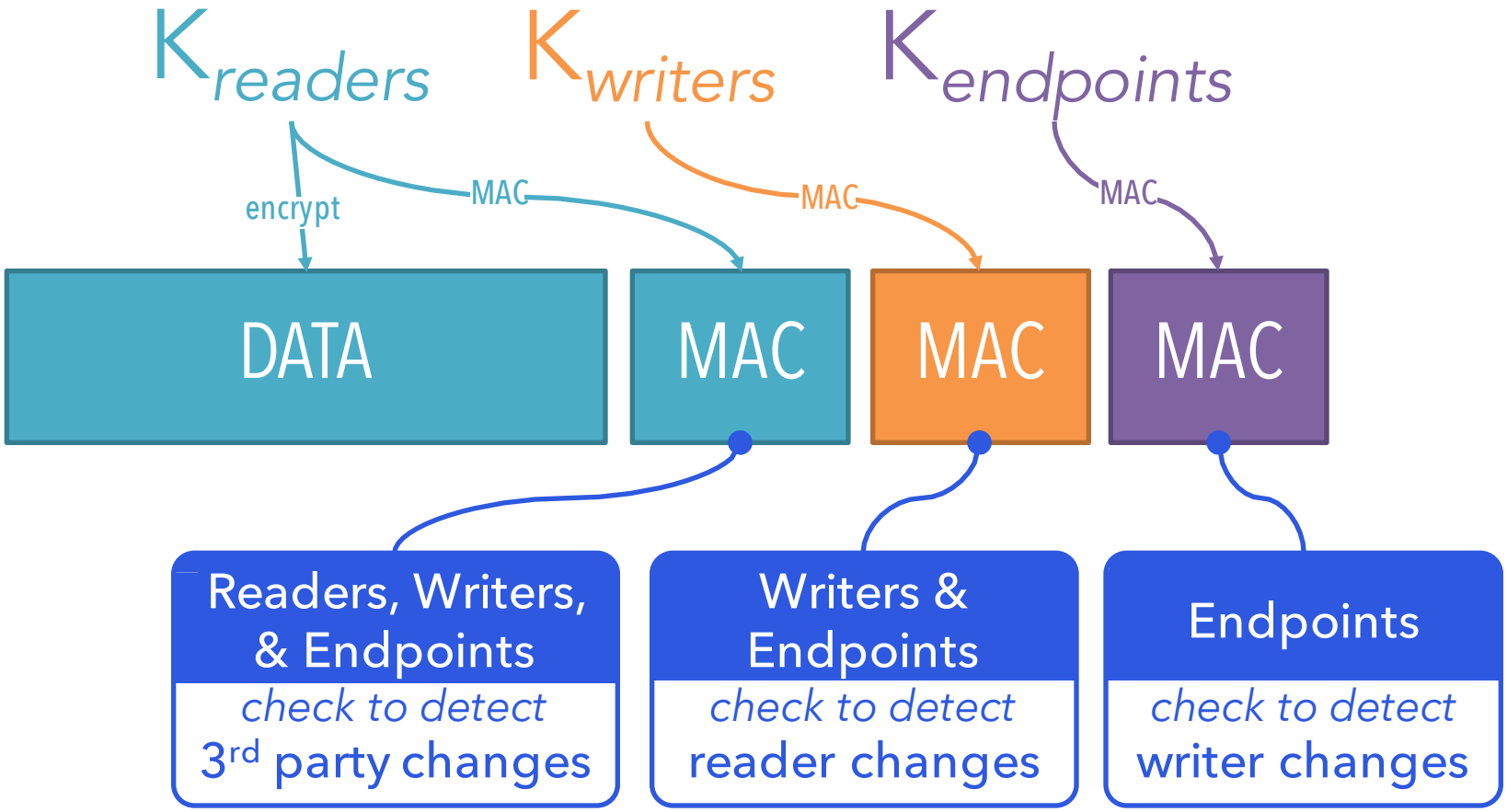
Idea #1: Encryption Contexts
(for access control)

TLS uses *one key* for encryption and MAC:



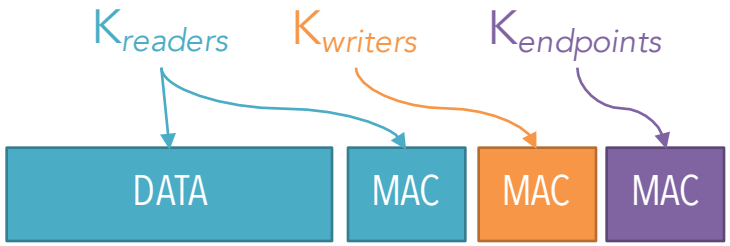
Idea #1: Encryption Contexts
(for access control)

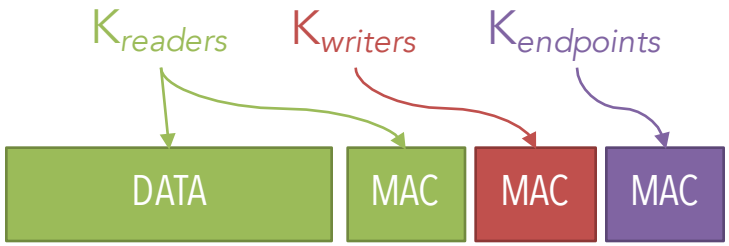
mcTLS uses *three keys* to separate read-only and read/write access:



Idea #1: Encryption Contexts
(for access control)

Each context has a read key and a write key:





Context 1:
"Request Headers"

Read:

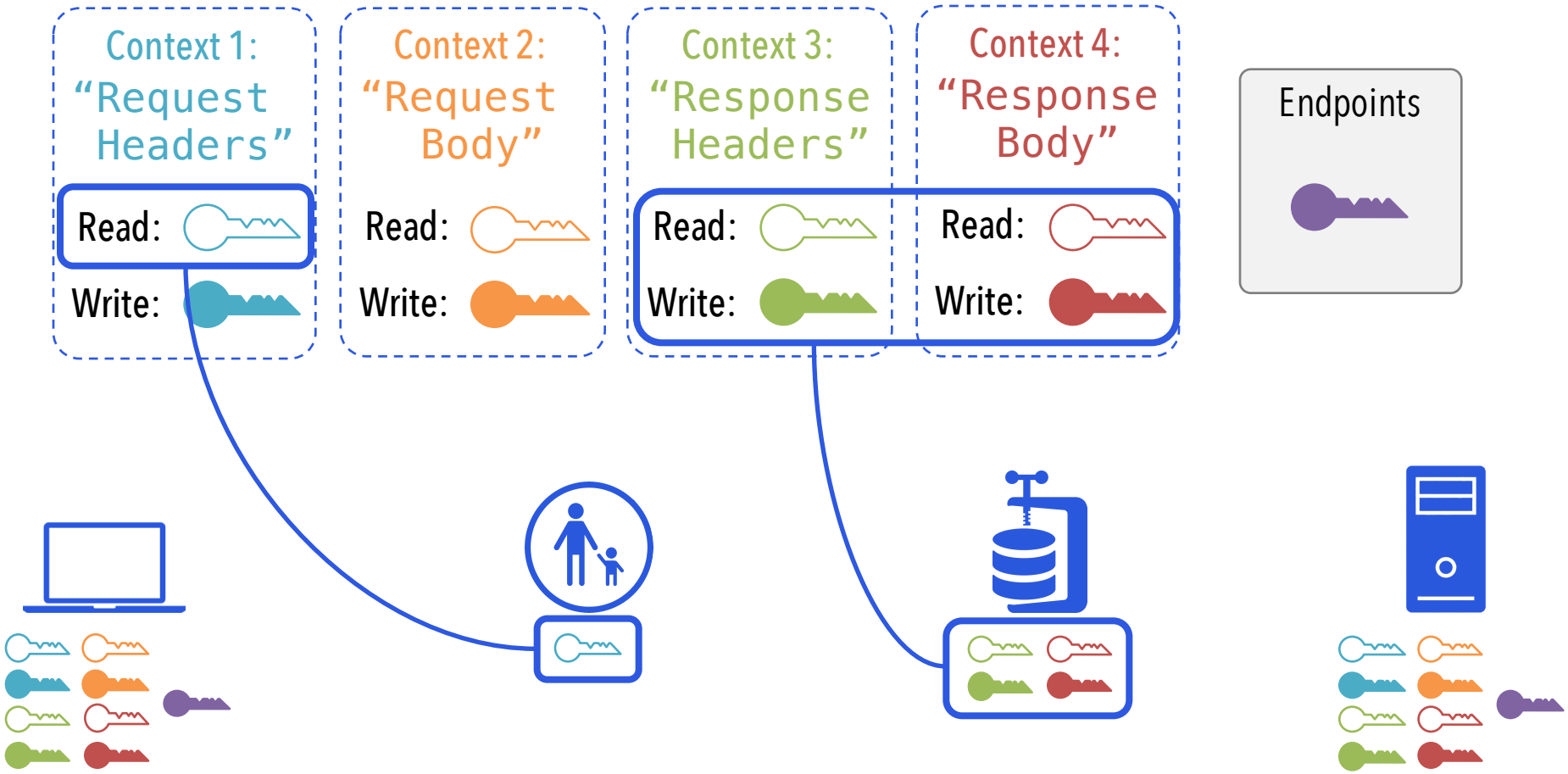
Write:

Context 2:
"Request Body"

Read:

Write:

Encryption contexts example



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

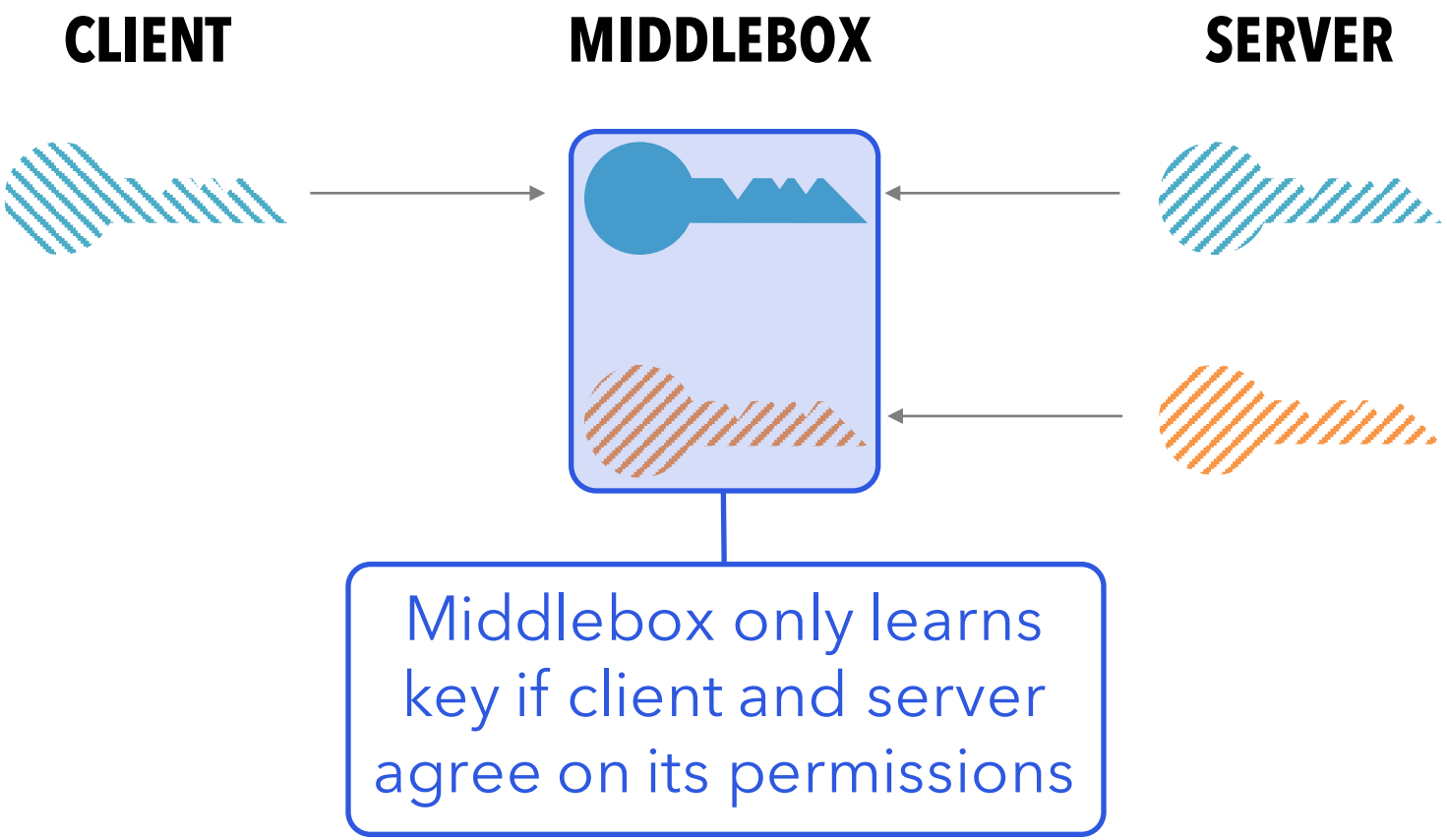
PLUS TWO NEW ONES:

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:



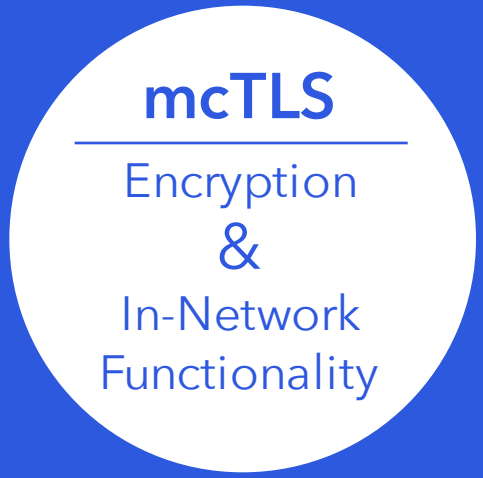
Design requirements for mcTLS

MAINTAIN TLS SECURITY PROPERTIES:

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

PLUS TWO NEW ONES:

4. Visibility & Control
   - Contributory Context Keys
5. Least Privilege
   - Multiple Encryption Contexts

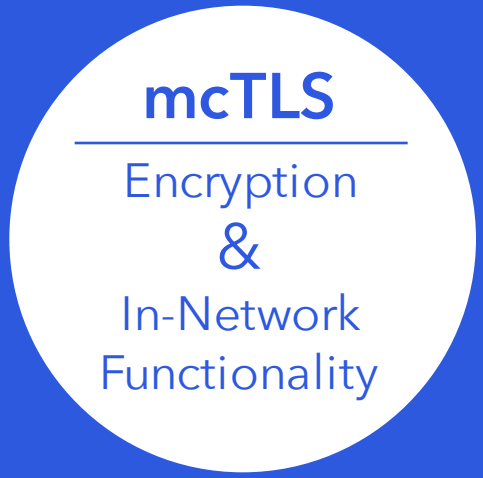


TLS + Middleboxes

mcTLS Design Ideas

mcTLS Handshake

Performance Evaluation



TLS + Middleboxes

mcTLS Design Ideas

mcTLS Handshake

Performance Evaluation

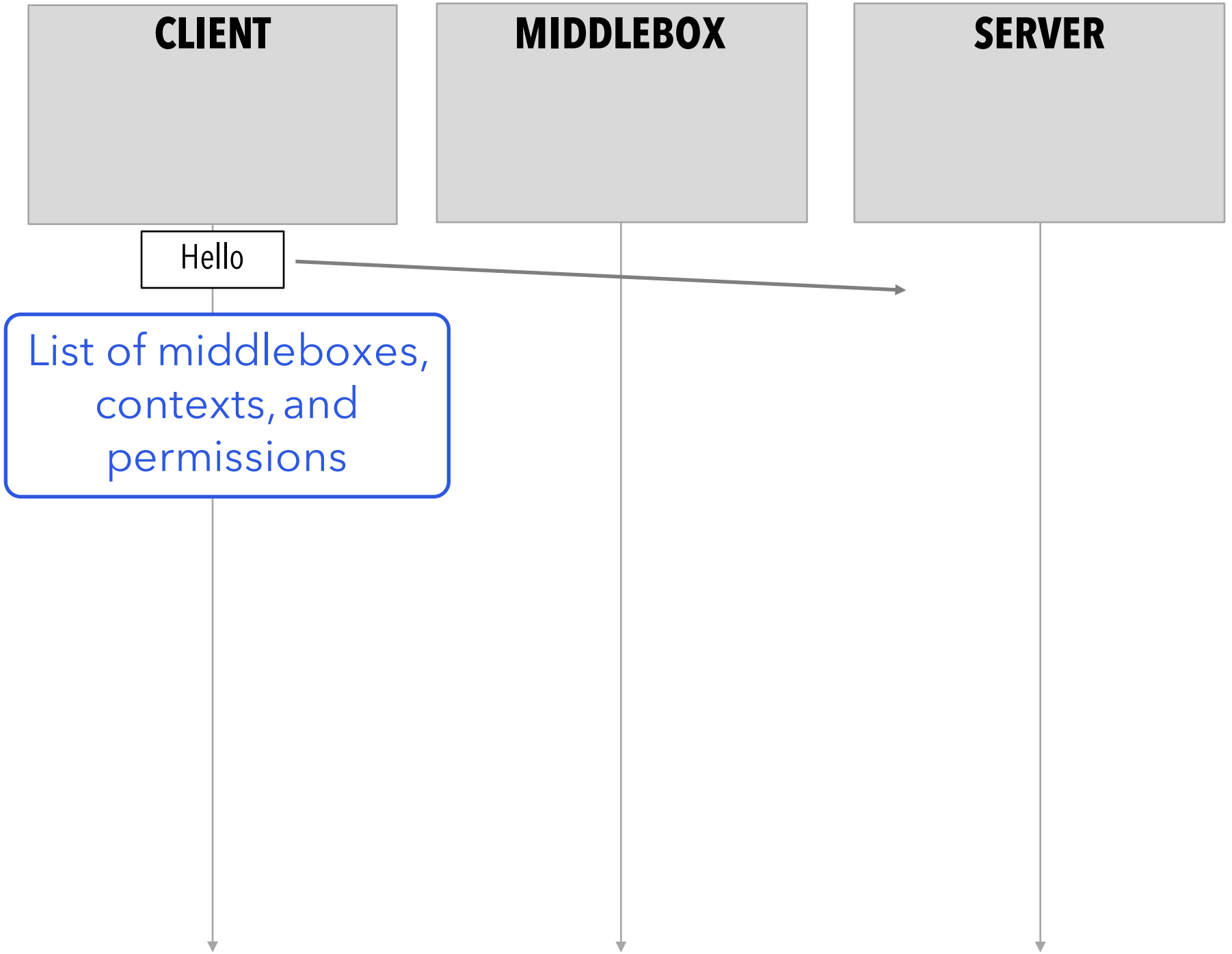
Handshake Goals

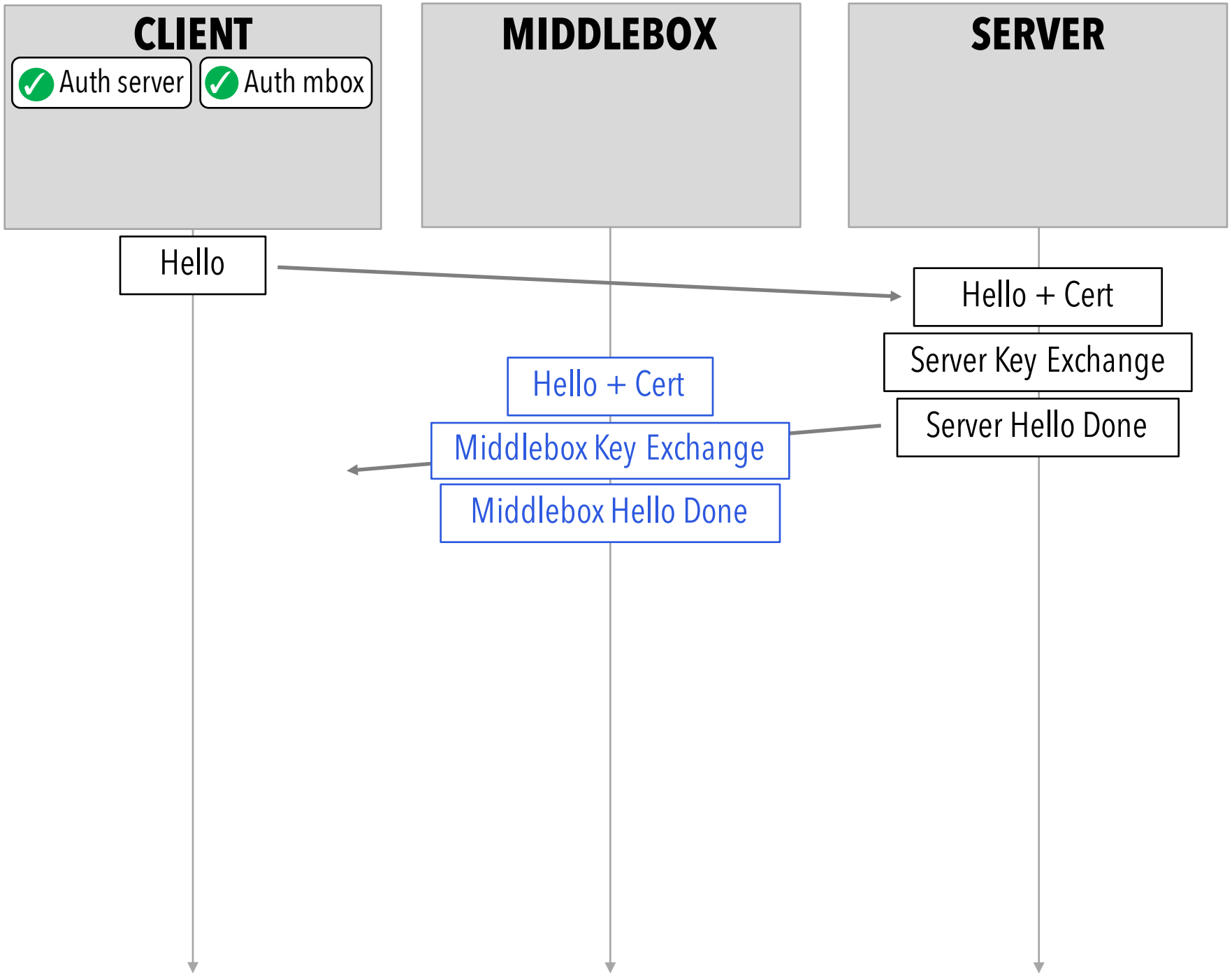
TLS

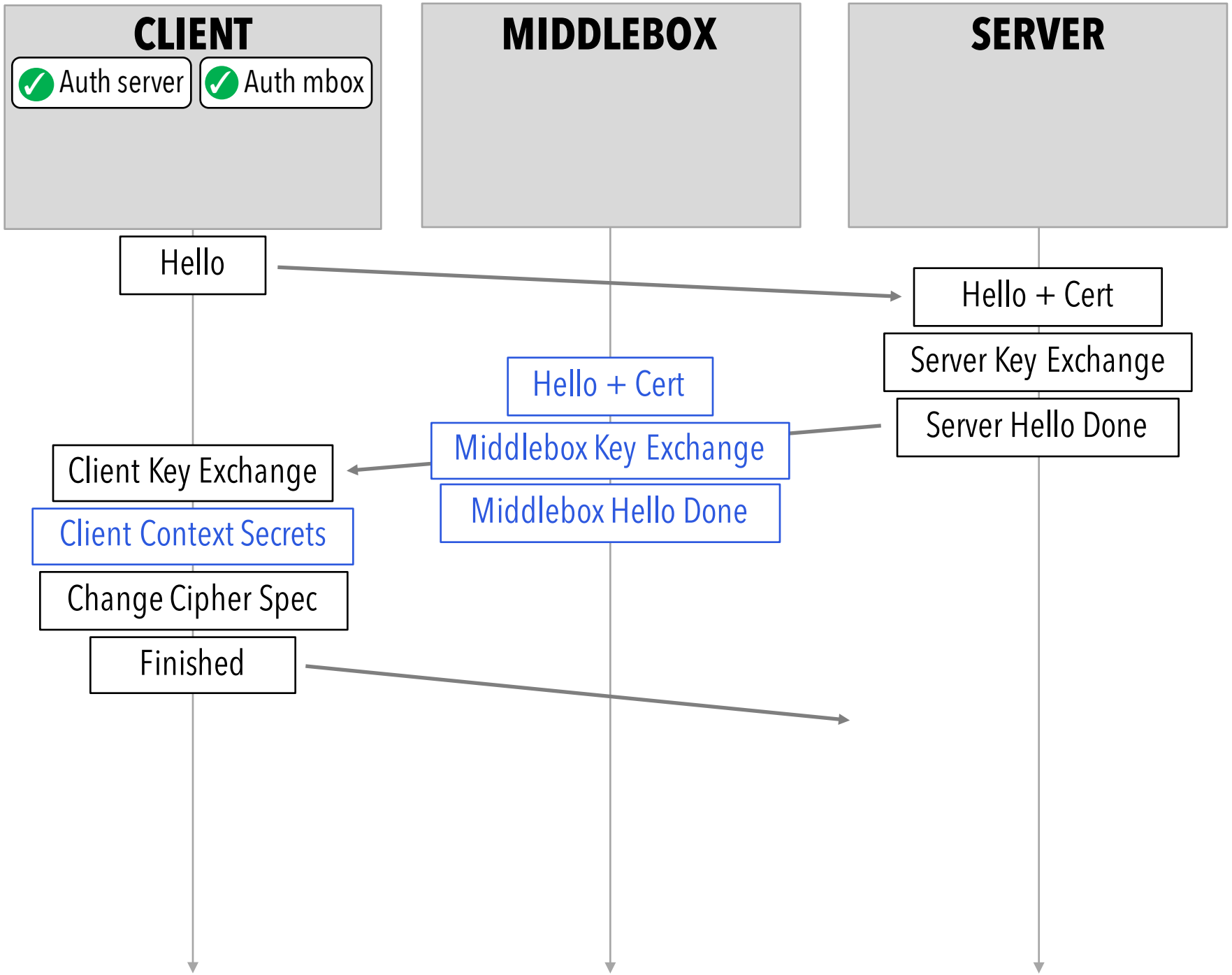
- ✓ Authenticate server
- ✓ Establish session key

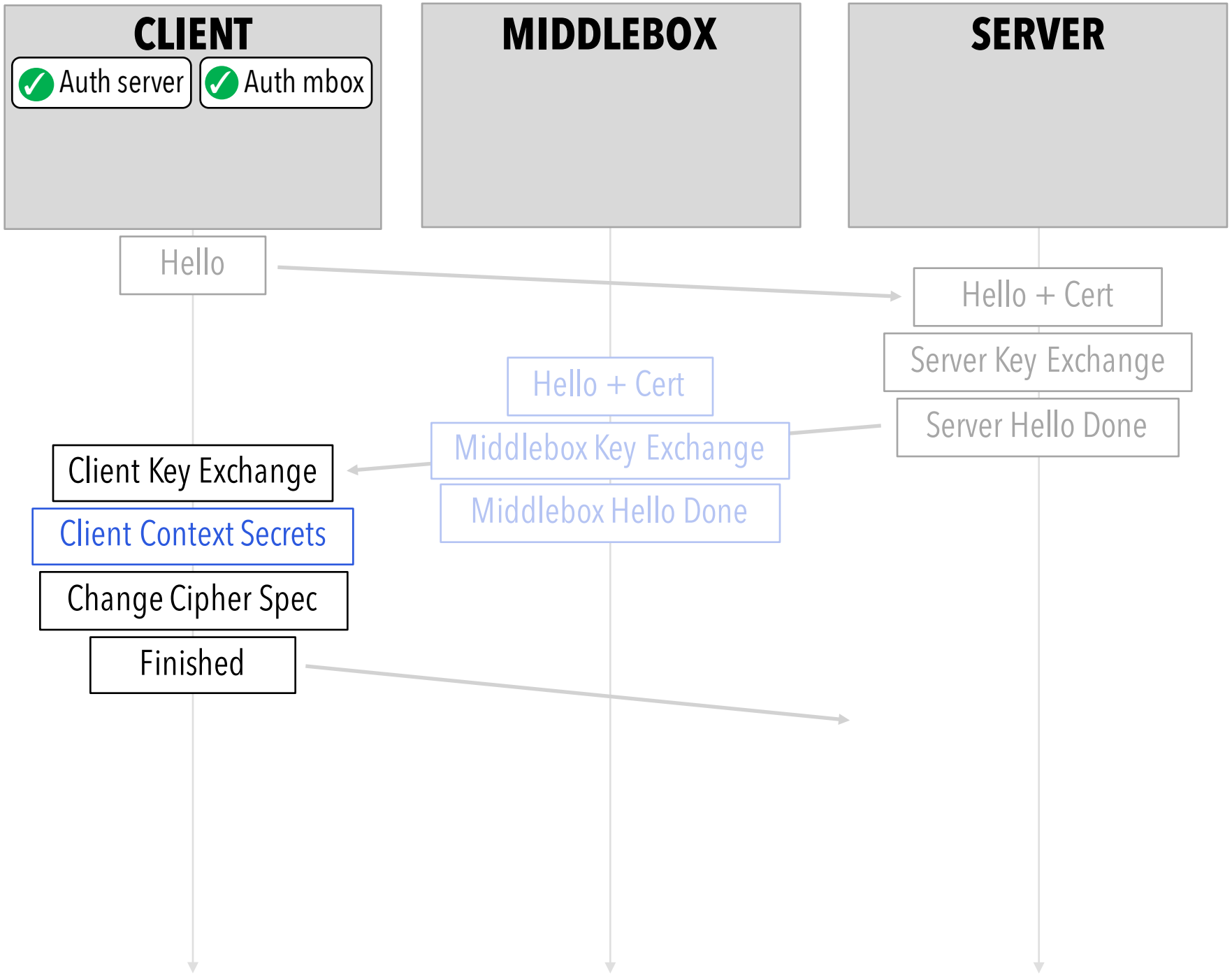
mcTLS

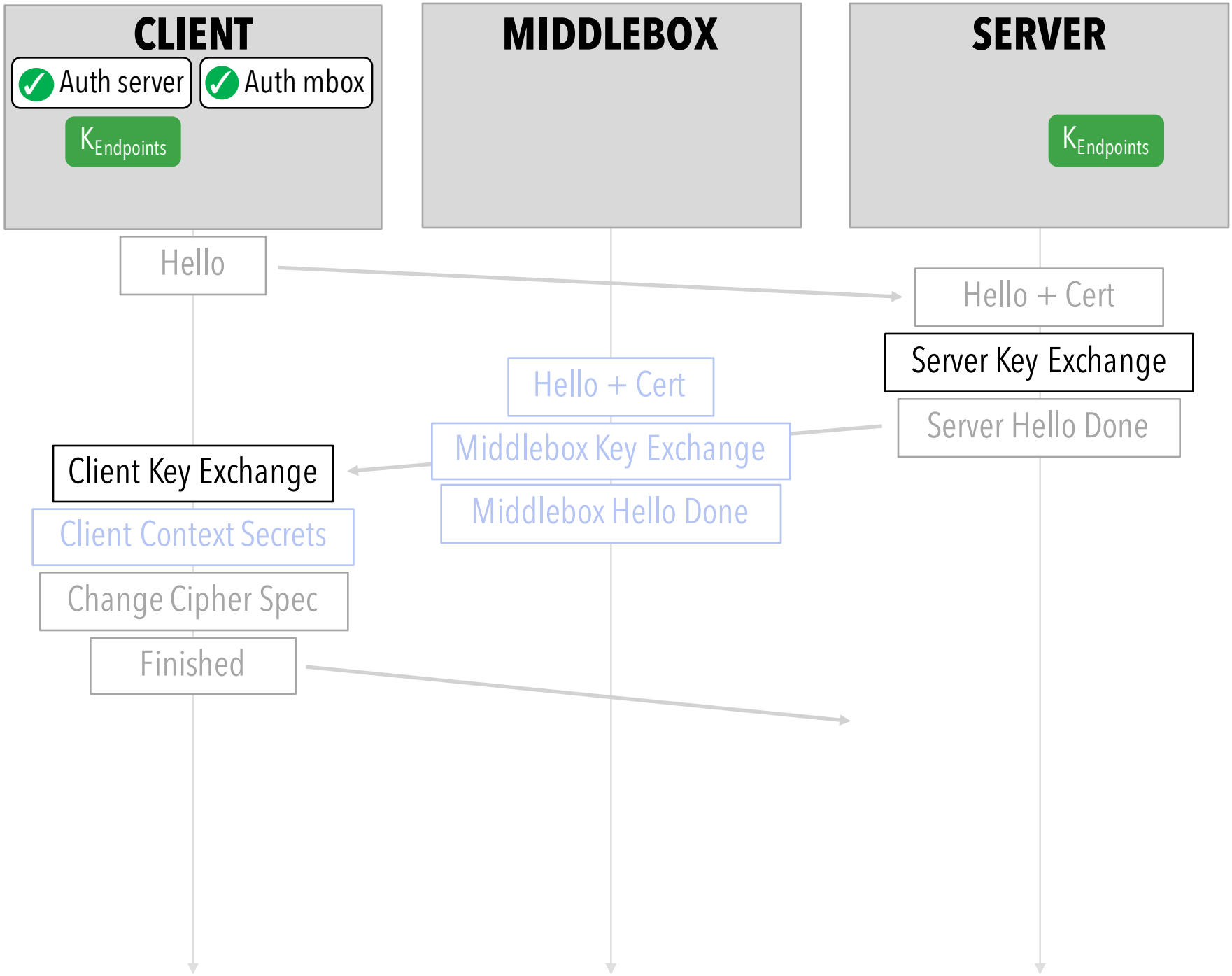
- ✓ Authenticate server
- ✓ Authenticate middlebox
- ✓ Distribute context keys

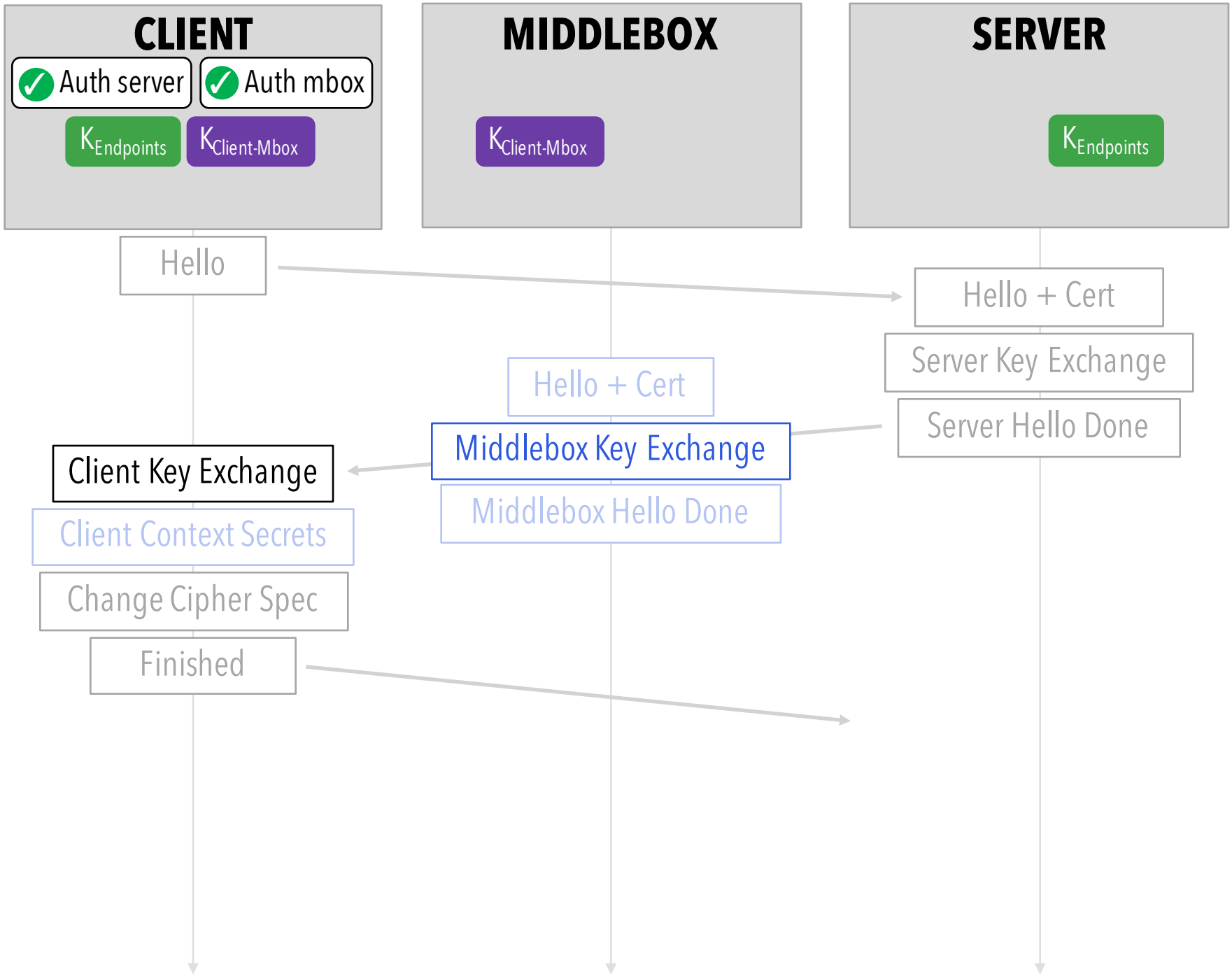

CLIENT
MIDDLEBOX
SERVER
Hello
List of middleboxes, contexts, and permissions

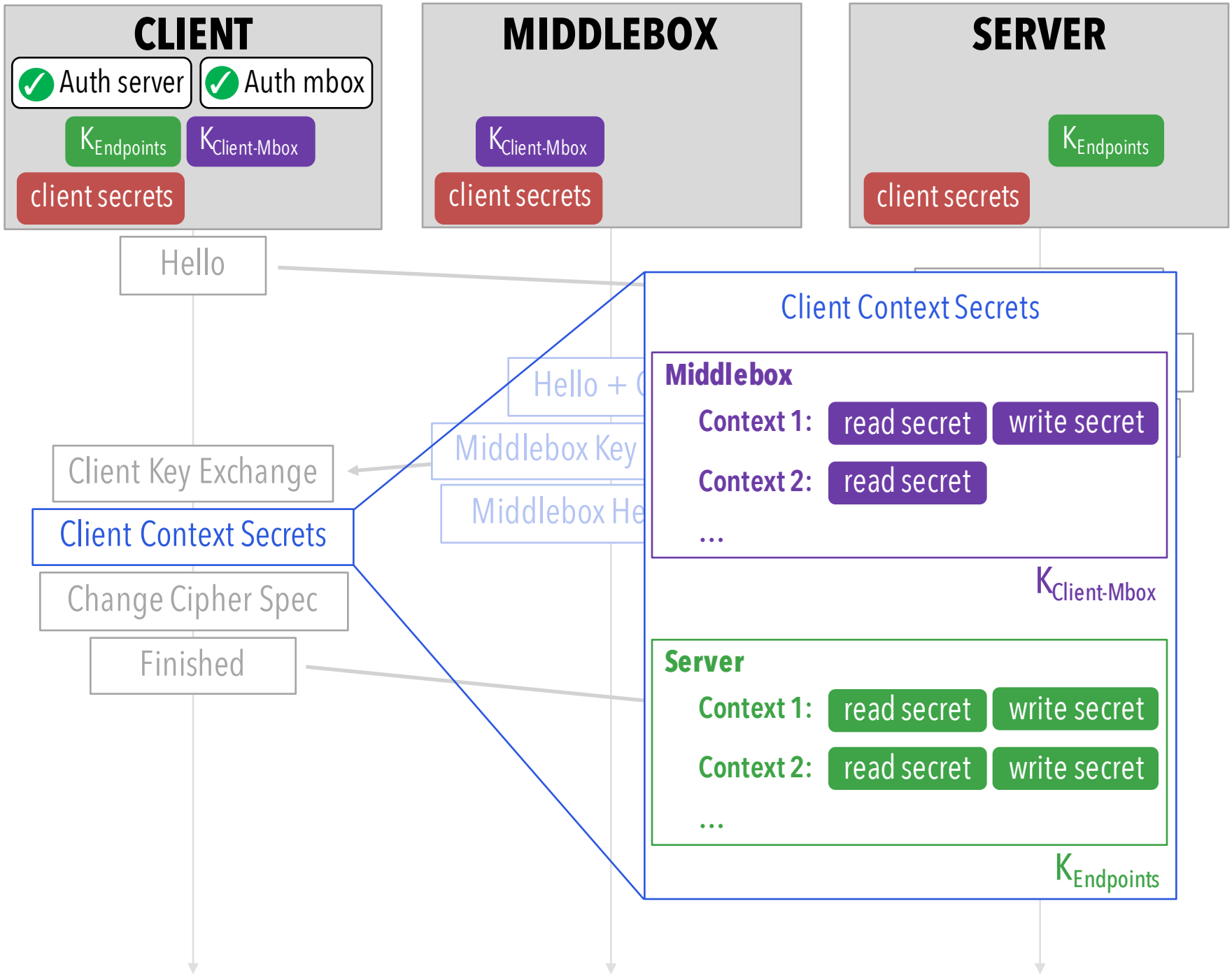

CLIENT
Auth server
Auth mbox
MIDDLEBOX
SERVER
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done

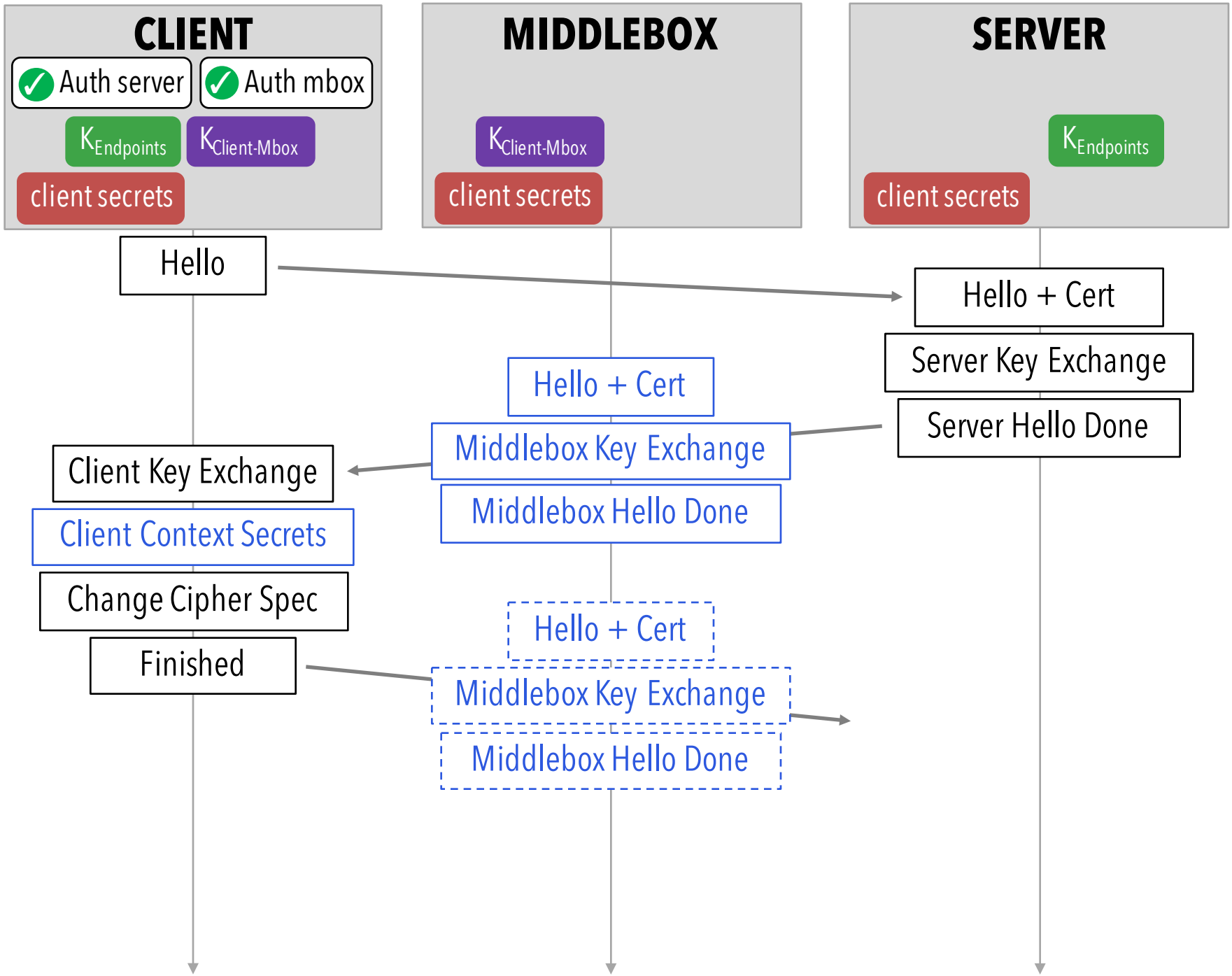

CLIENT
Auth server
Auth mbox
MIDDLEBOX
SERVER
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished

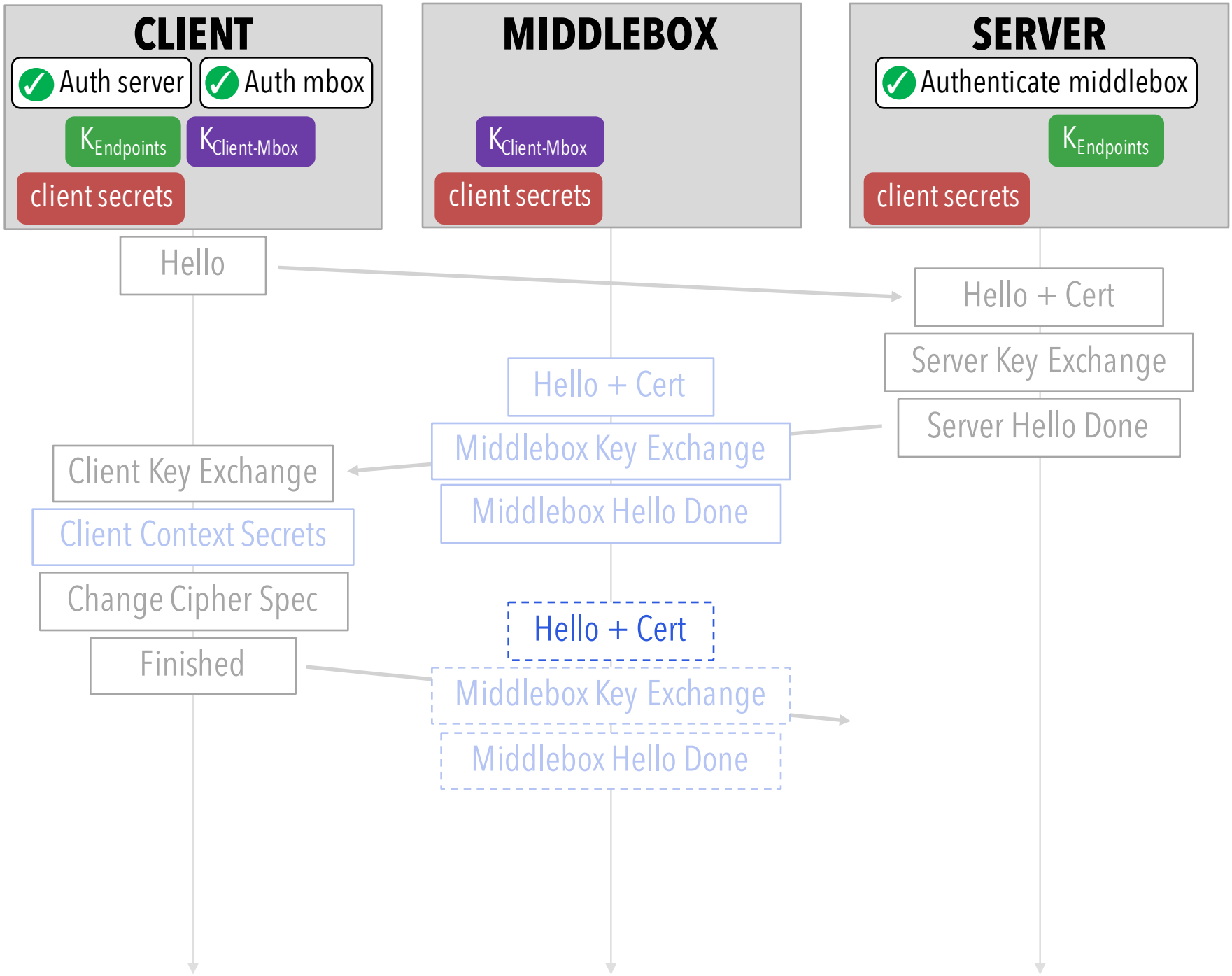

CLIENT
Auth server
Auth mbox
MIDDLEBOX
SERVER
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished

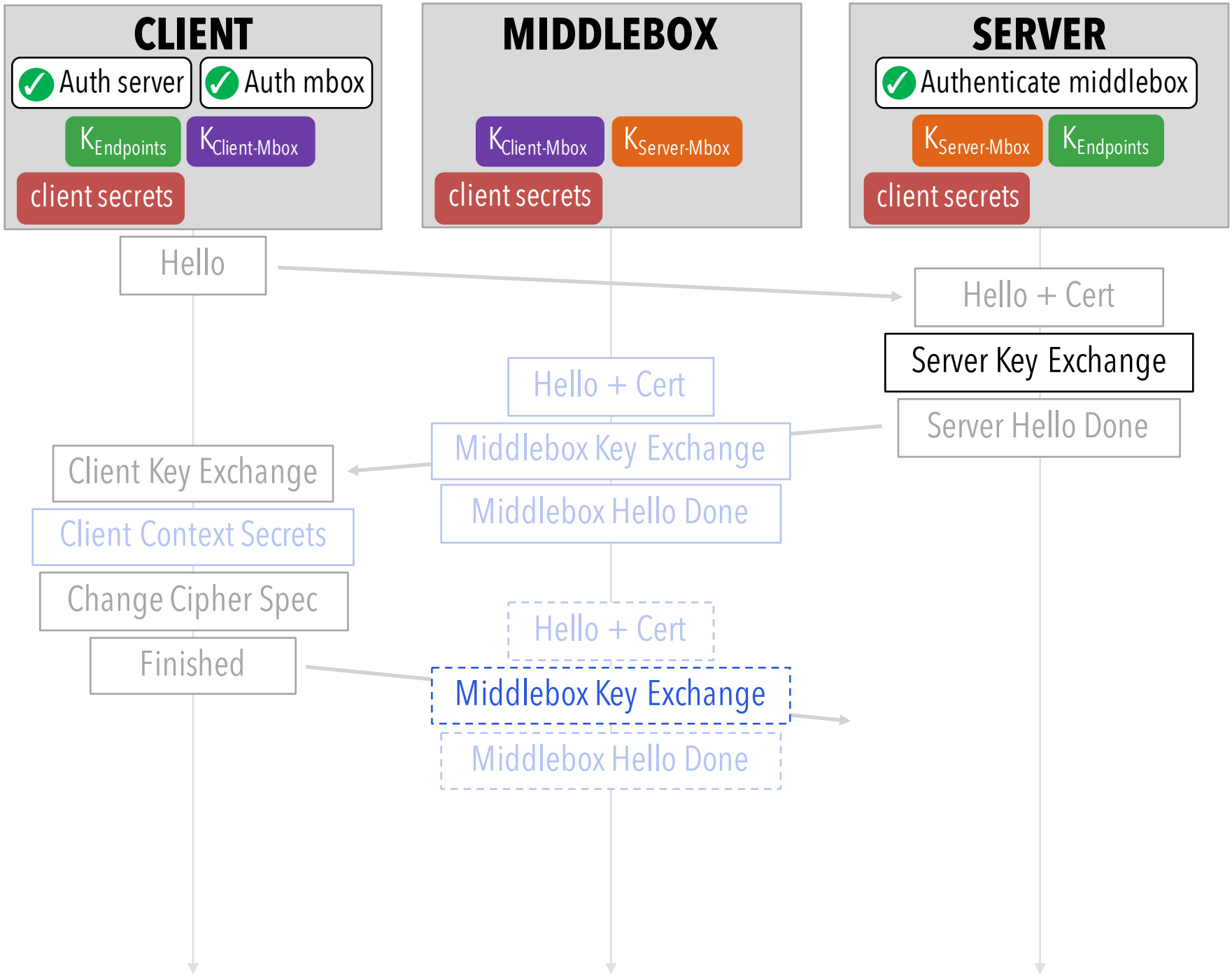

CLIENT
Auth server
Auth mbox
K Endpoints
MIDDLEBOX
SERVER
K Endpoints
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished

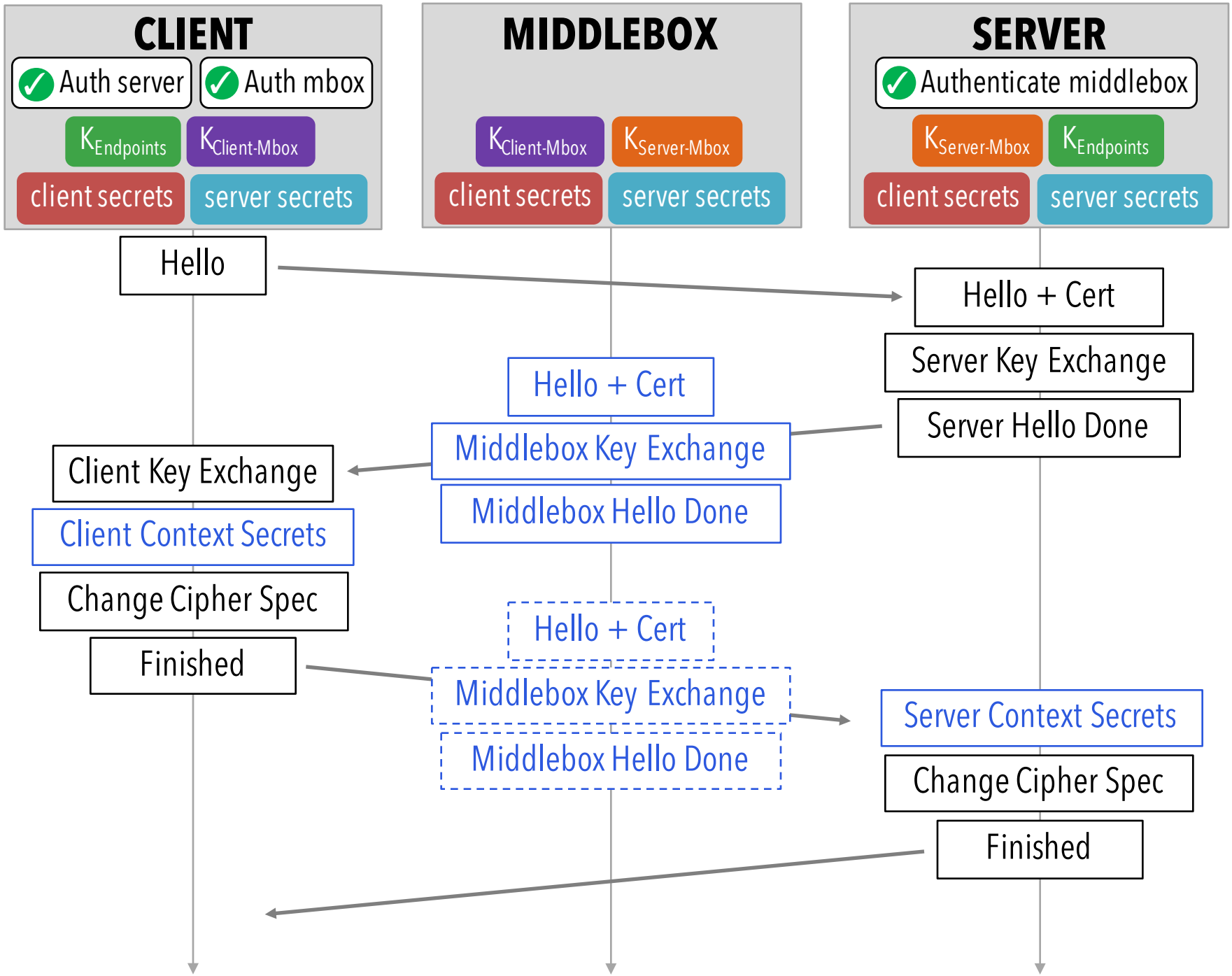

CLIENT
Auth server
Auth mbox
K Endpoints
K Client-Mbox
MIDDLEBOX
K Client-Mbox
SERVER
K Endpoints
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished

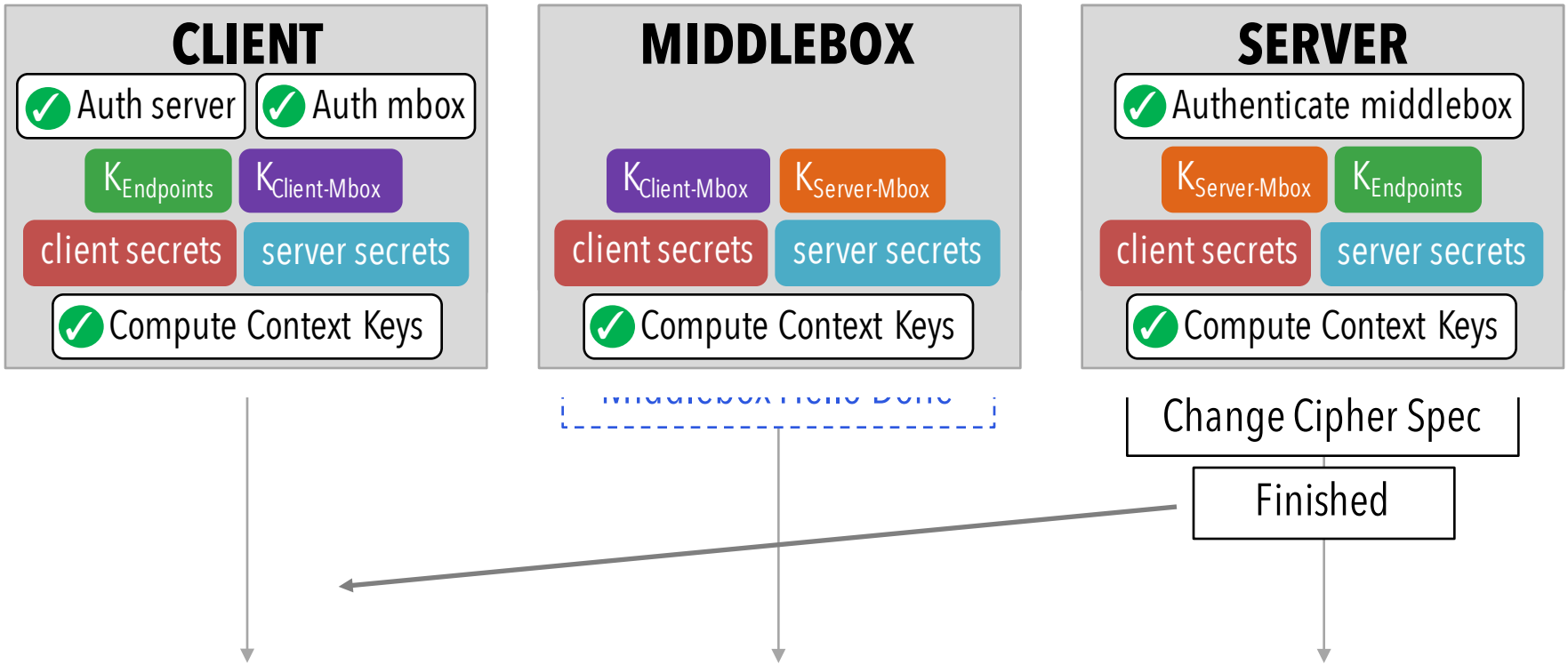

CLIENT
Auth server
Auth mbox
K Endpoints
K Client-Mbox
client secrets
MIDDLEBOX
K Client-Mbox
client secrets
SERVER
K Endpoints
client secrets
Hello
Hello + C
Middlebox Key
Middlebox He
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished
Client Context Secrets
Middlebox
Context 1:
read secret
write secret
Context 2:
read secret
...
K Client-Mbox
Server
Context 1:
read secret
write secret
Context 2:
read secret
write secret
...
K Endpoints


CLIENT
Auth server
Auth mbox
$K_{Endpoints}$
$K_{Client\text{-}Mbox}$
client secrets
MIDDLEBOX
$K_{Client\text{-}Mbox}$
client secrets
SERVER
$K_{Endpoints}$
client secrets
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done


CLIENT
Auth server
Auth mbox
$K_{Endpoints}$
$K_{Client-Mbox}$
client secrets
MIDDLEBOX
$K_{Client-Mbox}$
client secrets
SERVER
Authenticate middlebox
$K_{Endpoints}$
client secrets
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done


CLIENT
Auth server
Auth mbox
K Endpoints
K Client-Mbox
client secrets
MIDDLEBOX
K Client-Mbox
K Server-Mbox
client secrets
SERVER
Authenticate middlebox
K Server-Mbox
K Endpoints
client secrets
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done


CLIENT
Auth server
Auth mbox
$K_{Endpoints}$
$K_{Client\text{-}Mbox}$
client secrets
server secrets
MIDDLEBOX
$K_{Client\text{-}Mbox}$
$K_{Server\text{-}Mbox}$
client secrets
server secrets
SERVER
Authenticate middlebox
$K_{Server\text{-}Mbox}$
$K_{Endpoints}$
client secrets
server secrets
Hello
Hello + Cert
Server Key Exchange
Server Hello Done
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Client Key Exchange
Client Context Secrets
Change Cipher Spec
Finished
Hello + Cert
Middlebox Key Exchange
Middlebox Hello Done
Server Context Secrets
Change Cipher Spec
Finished



For each context:

client read secret + server read secret

client random + server random

"reader keys"

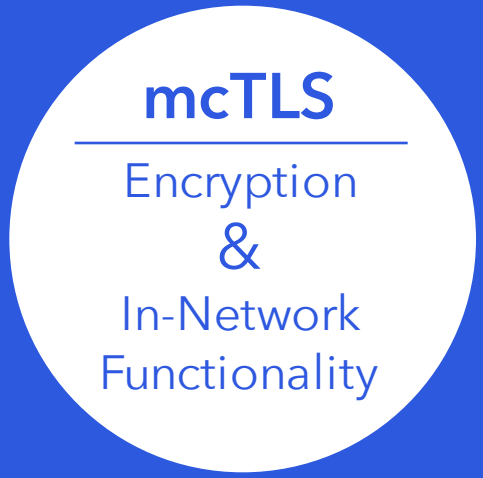
PRF → read key block

client write secret + server write secret

client random + server random

"writer keys"

PRF → write key block



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

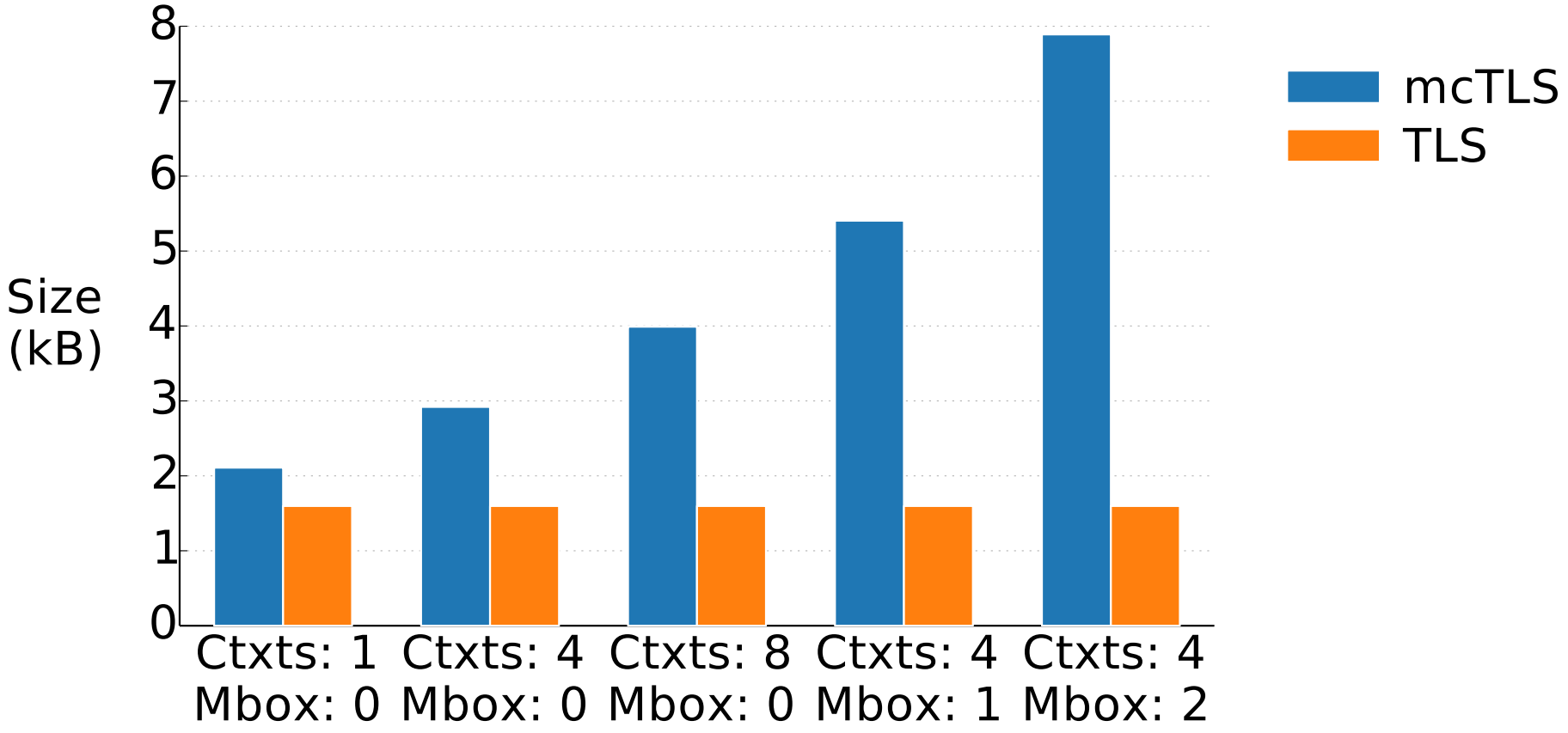
CPU Overhead

context key generation + key exchange

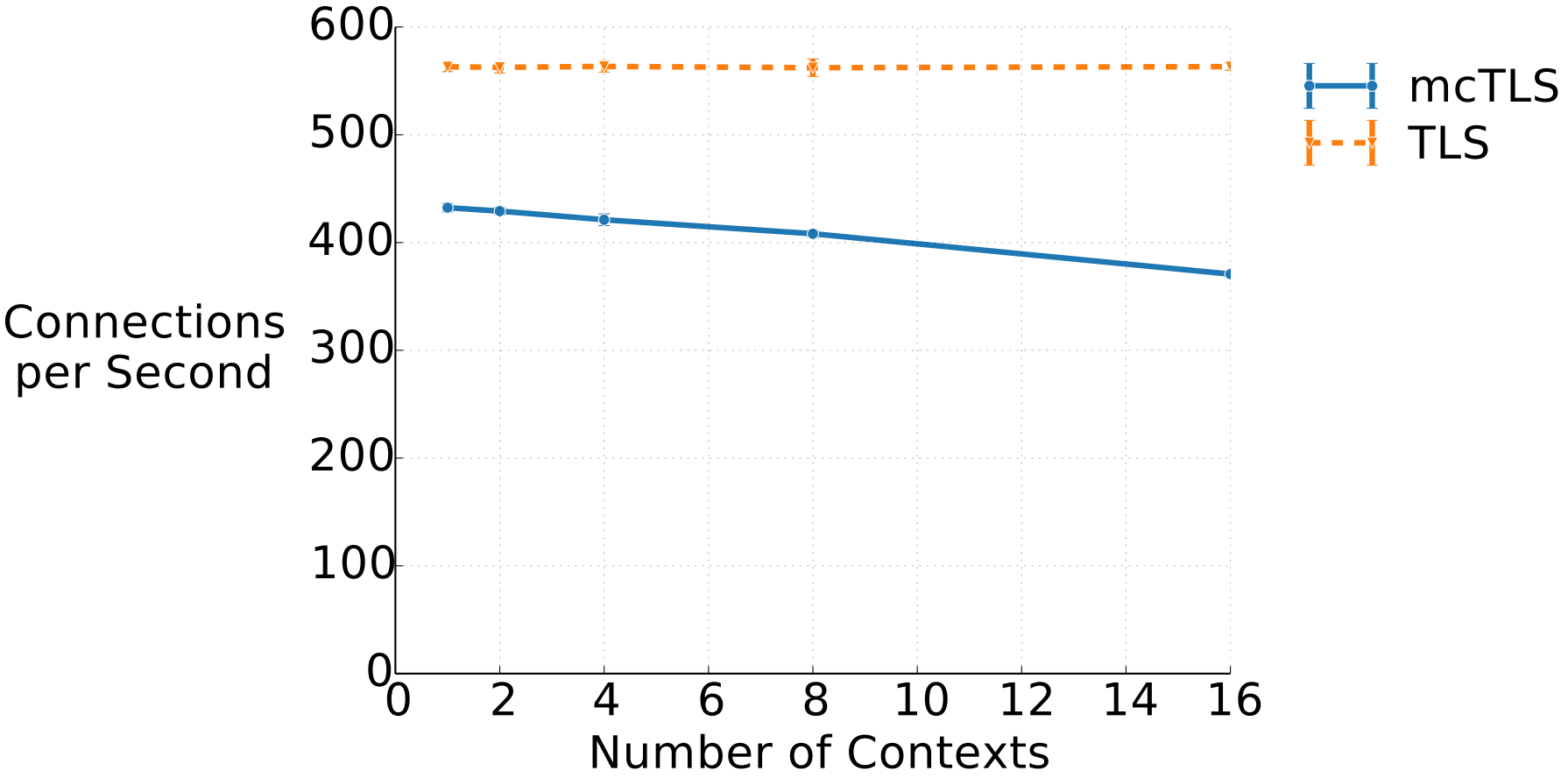
Time Overhead

handshake duration

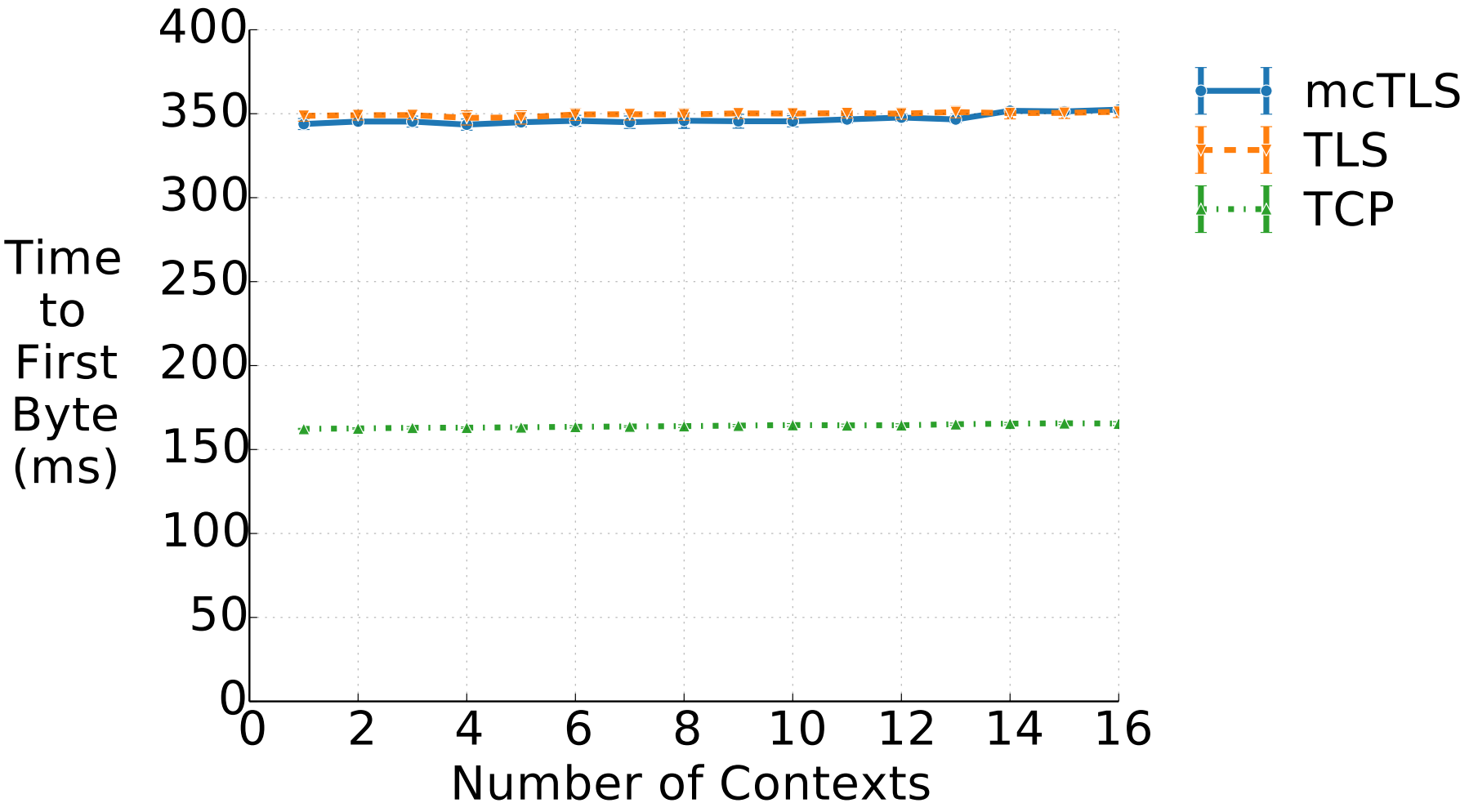
mcTLS increases handshake size



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

mcTLS

Encryption
&
In-Network
Functionality

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

In the Paper

crypto details
threat model
using encryption contexts
application use cases
detailed performance evaluation
future work

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 Rodriguez Rodriguez‡, and Peter Steenkiste*

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

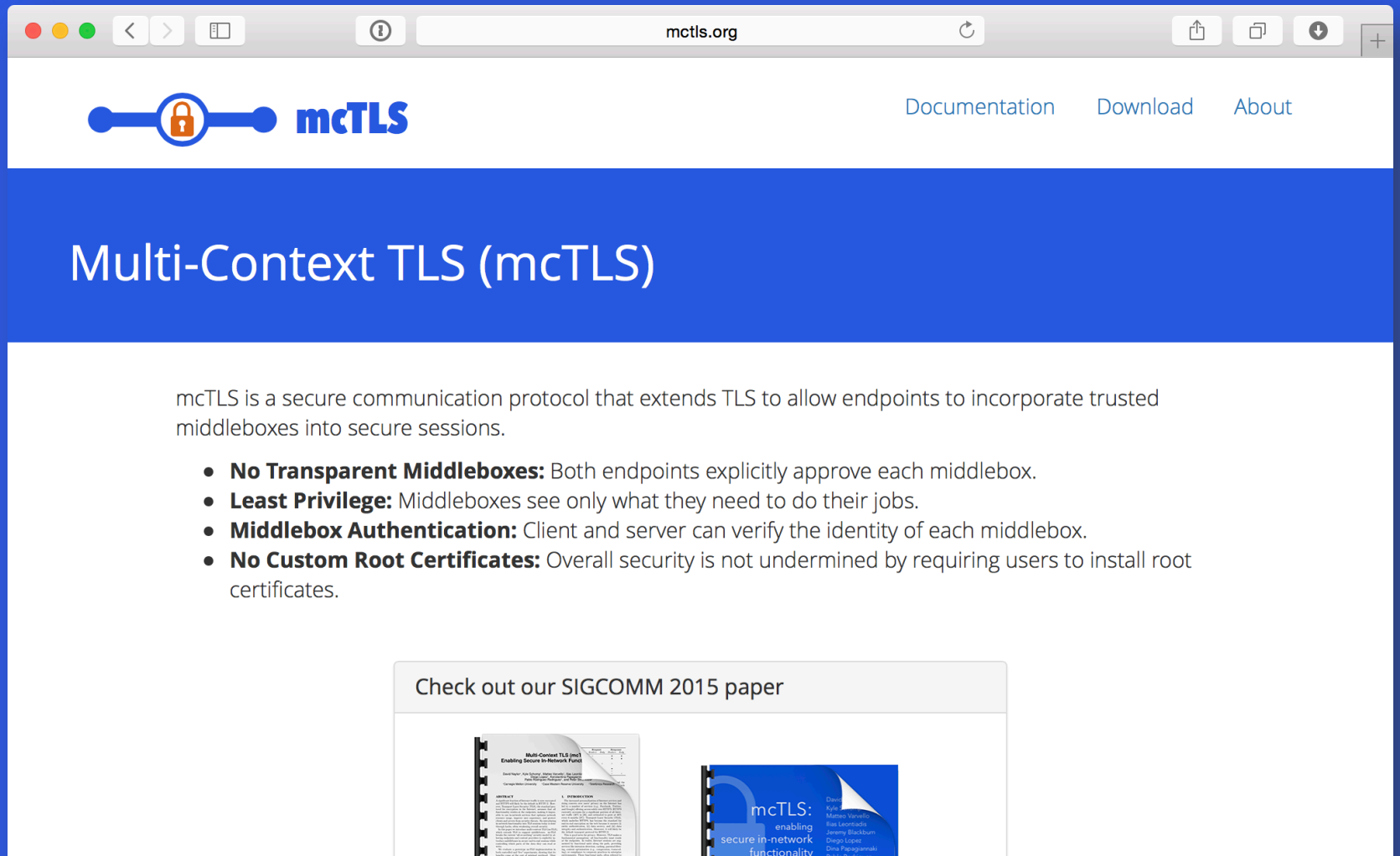
ABSTRACT

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.org


mctls.org
mcTLS
Documentation
Download
About
Multi-Context TLS (mcTLS)
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

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

Carnegie Mellon University
Case Western Reserve University
Telefonica