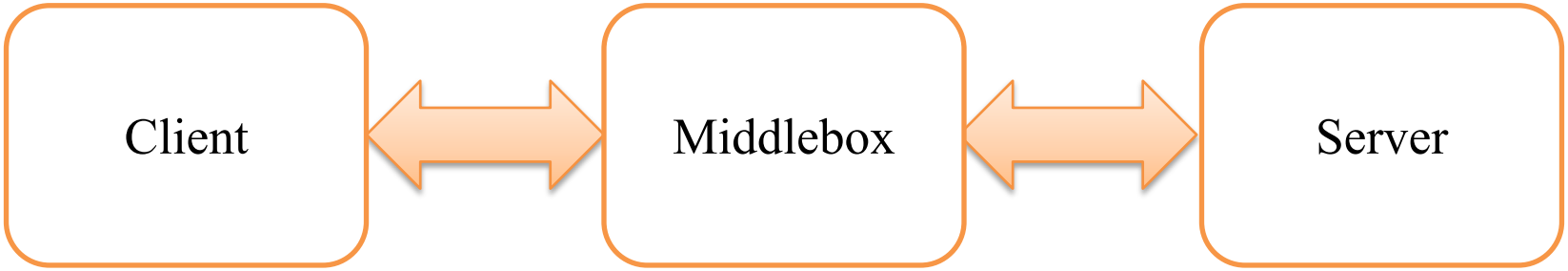
maTLS: How to Make TLS middlebox-aware?

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

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

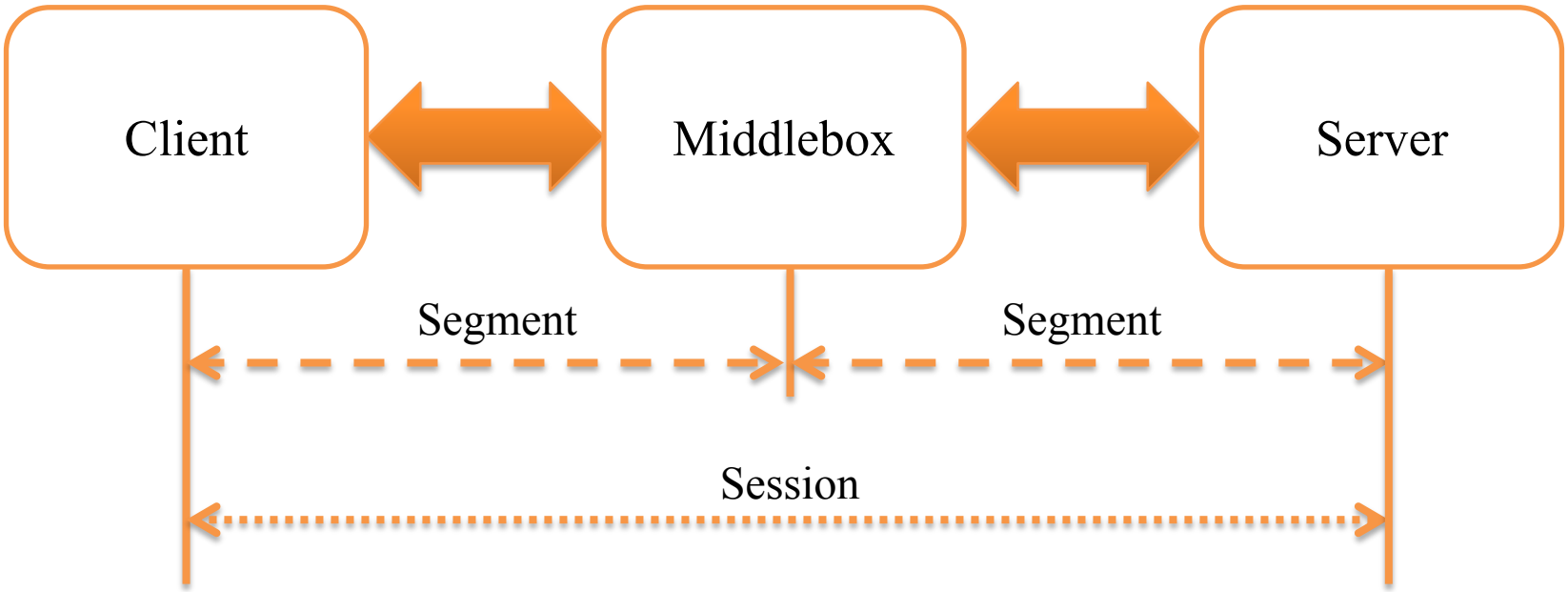
To perform their functions
Middleboxes **split** the TLS session

- Private key sharing
- Custom root certificate

Session and Segment

To perform their functions
Middleboxes **split** the TLS session

- Private key sharing
- Custom root certificate



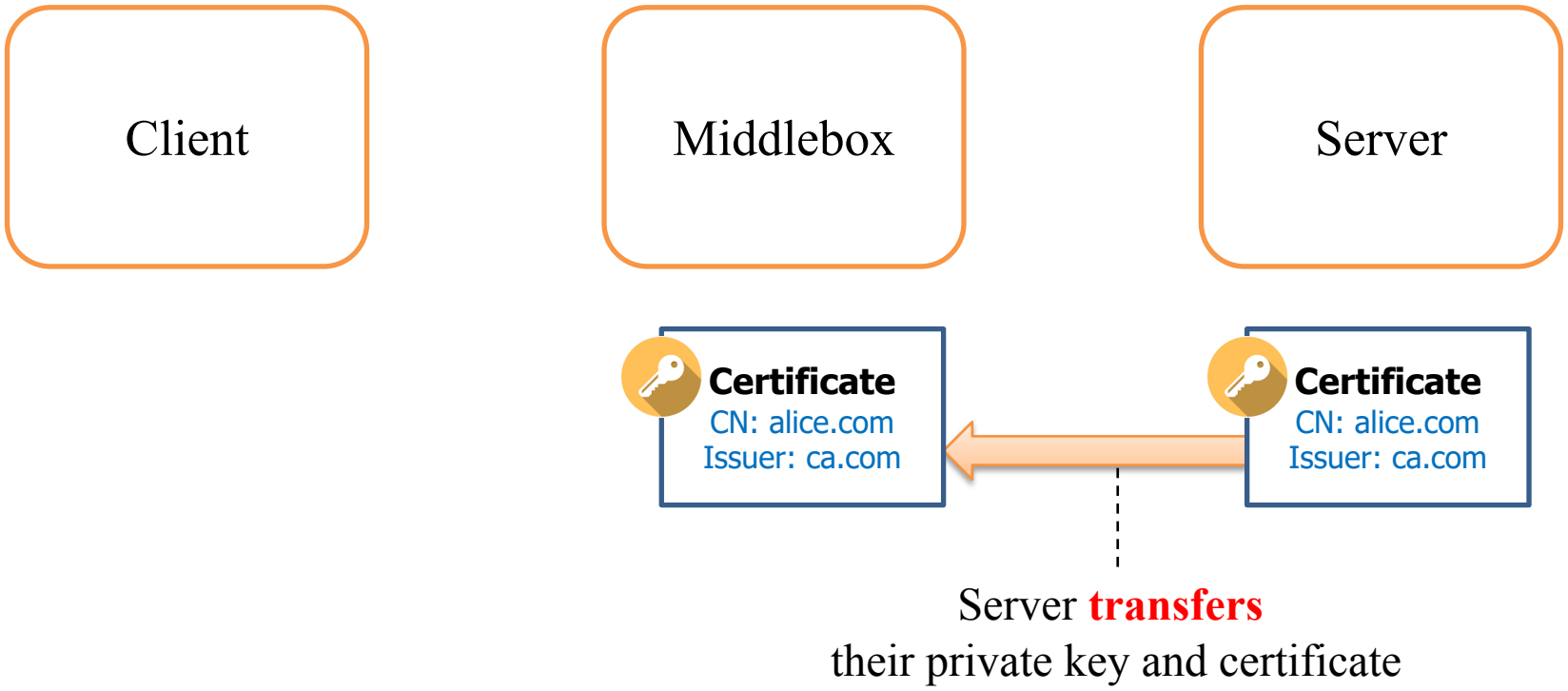
SplitTLS (1) Private Key Sharing

Client

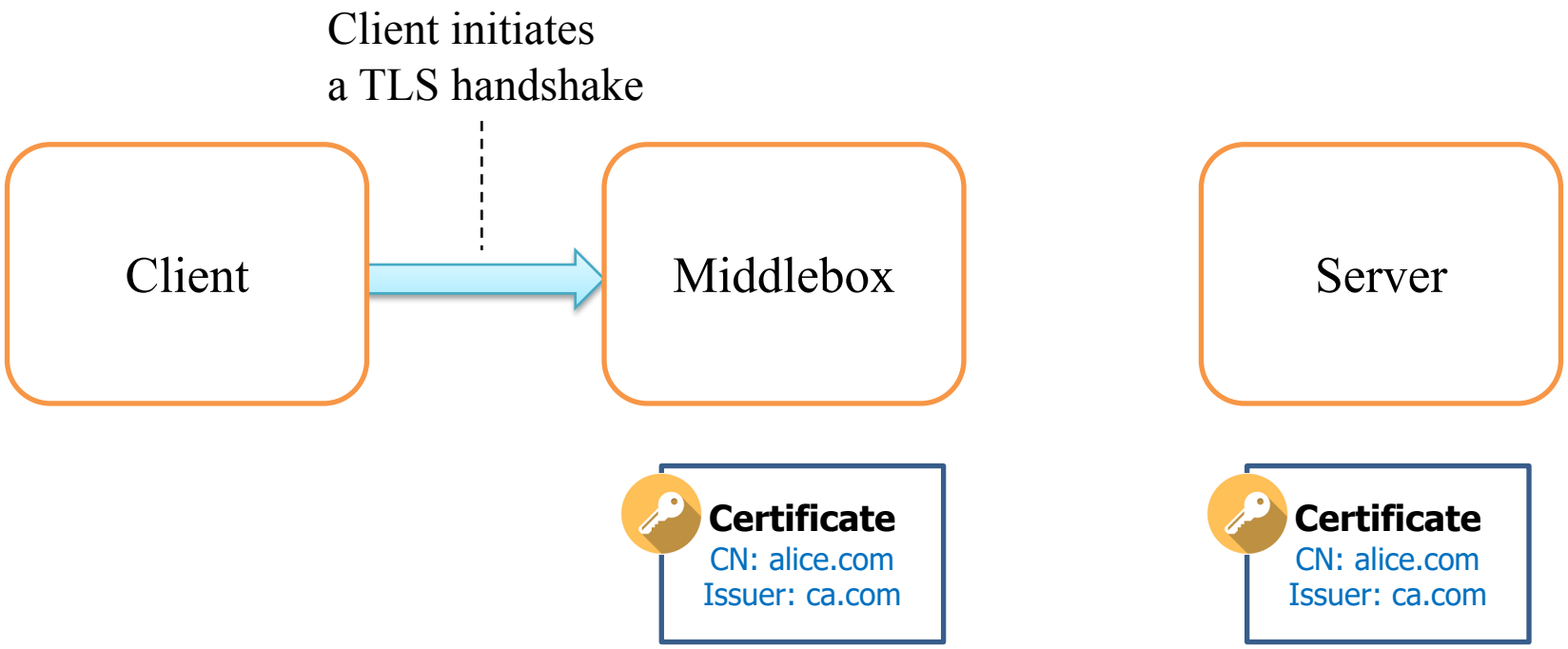
Middlebox

Server

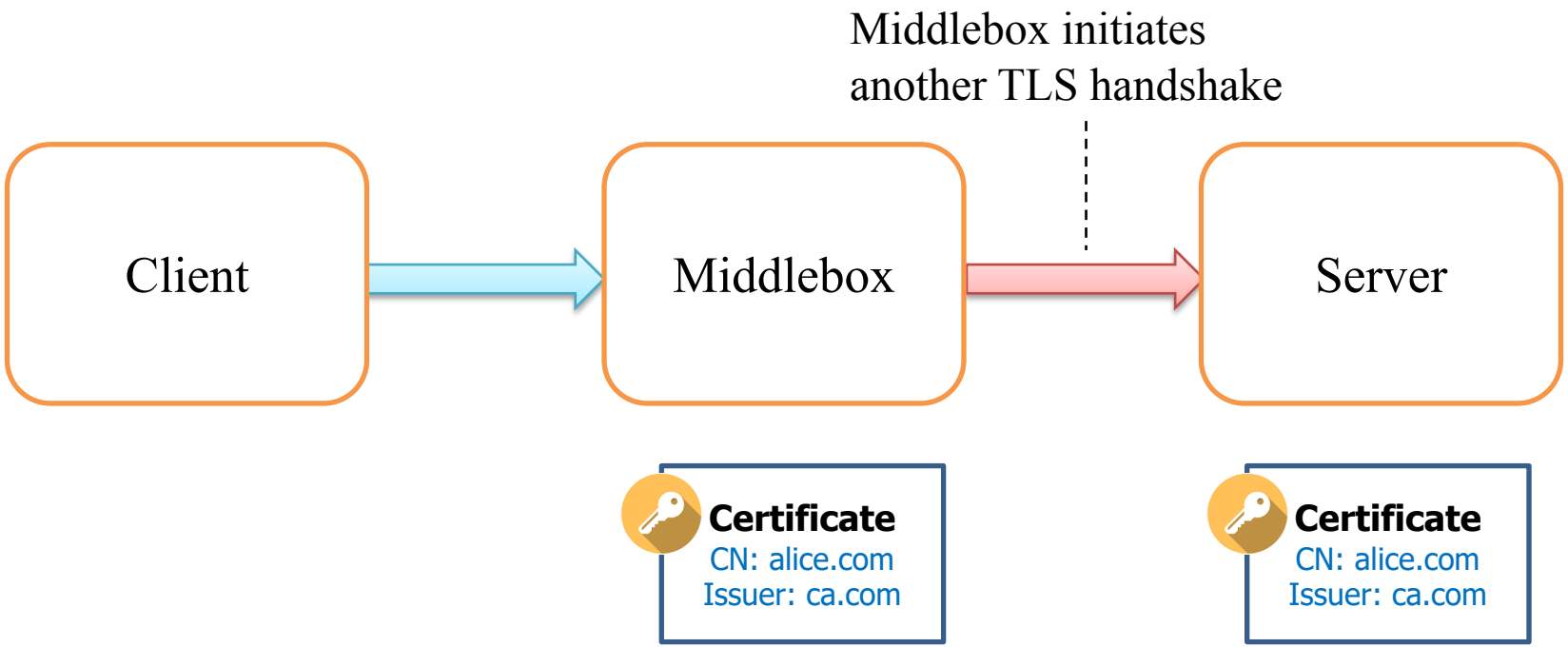
SplitTLS (1) Private Key Sharing



SplitTLS (1) Private Key Sharing

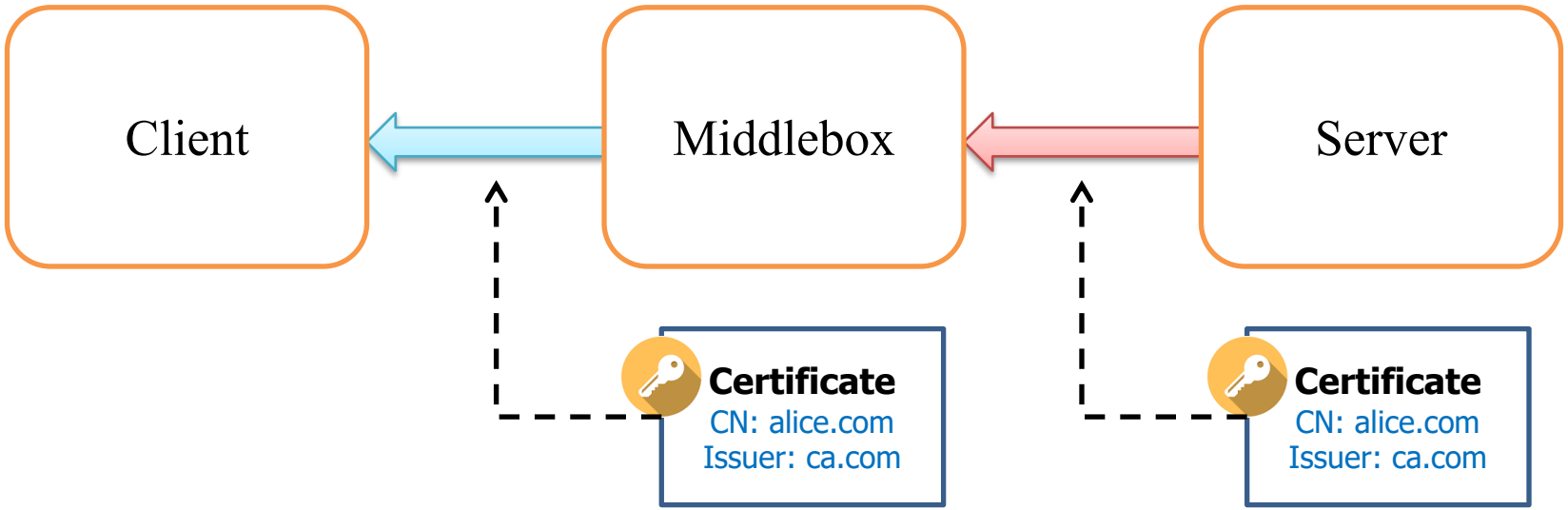


SplitTLS (1) Private Key Sharing



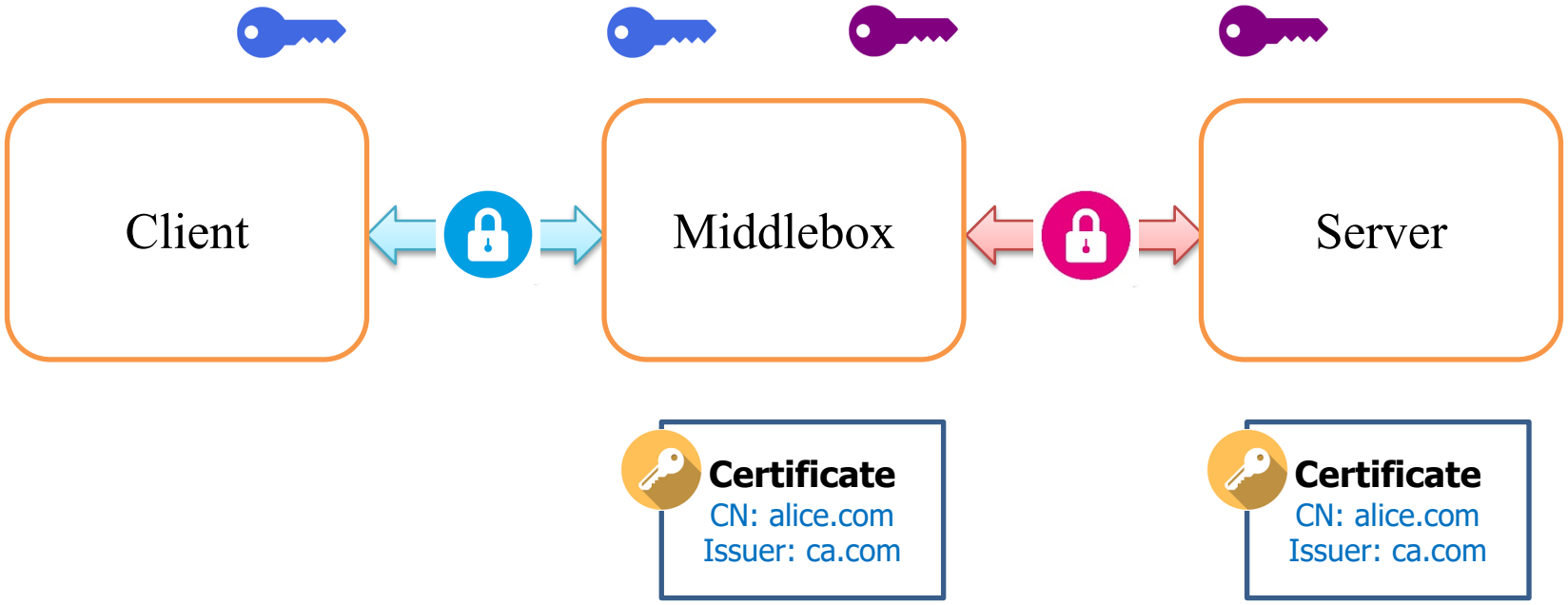
SplitTLS (1) Private Key Sharing

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

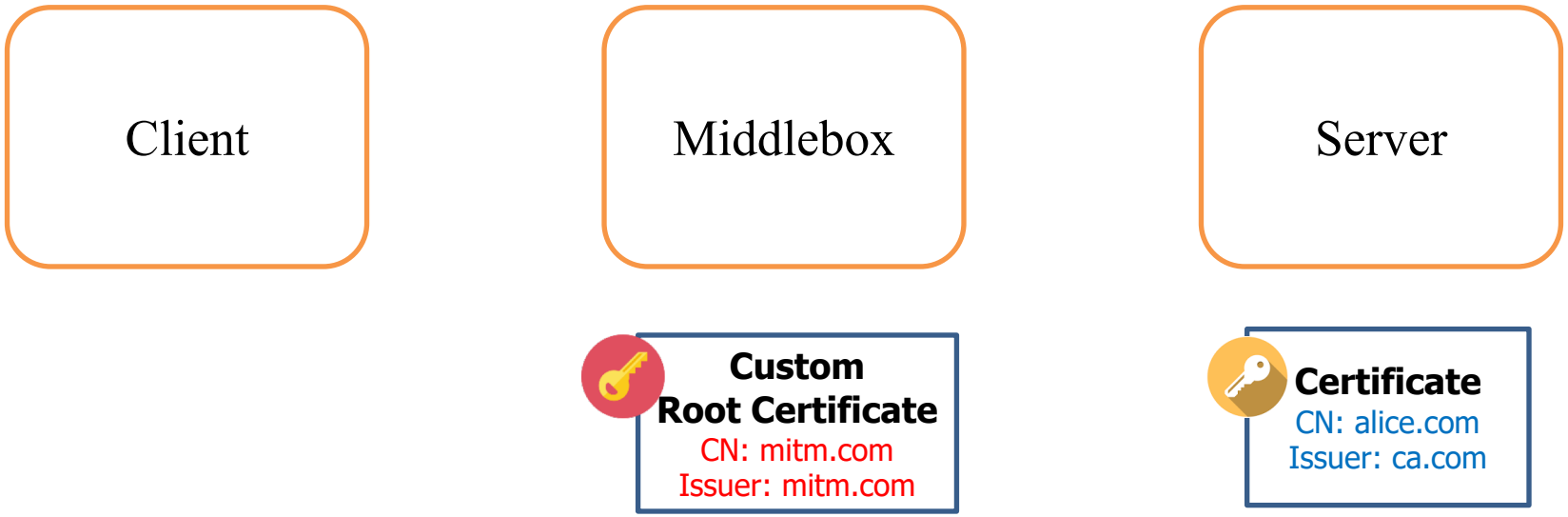


SplitTLS (1) Private Key Sharing

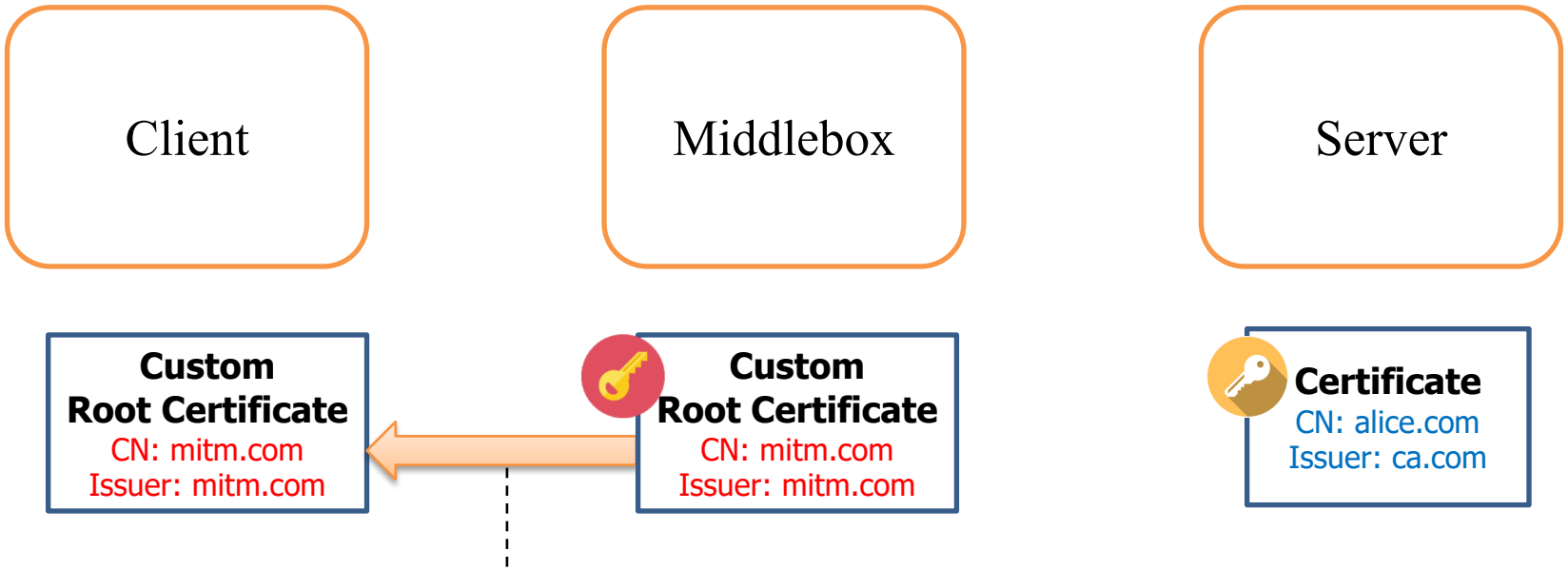
Client believes they have established a TLS session **with Server**, **not Middlebox!**



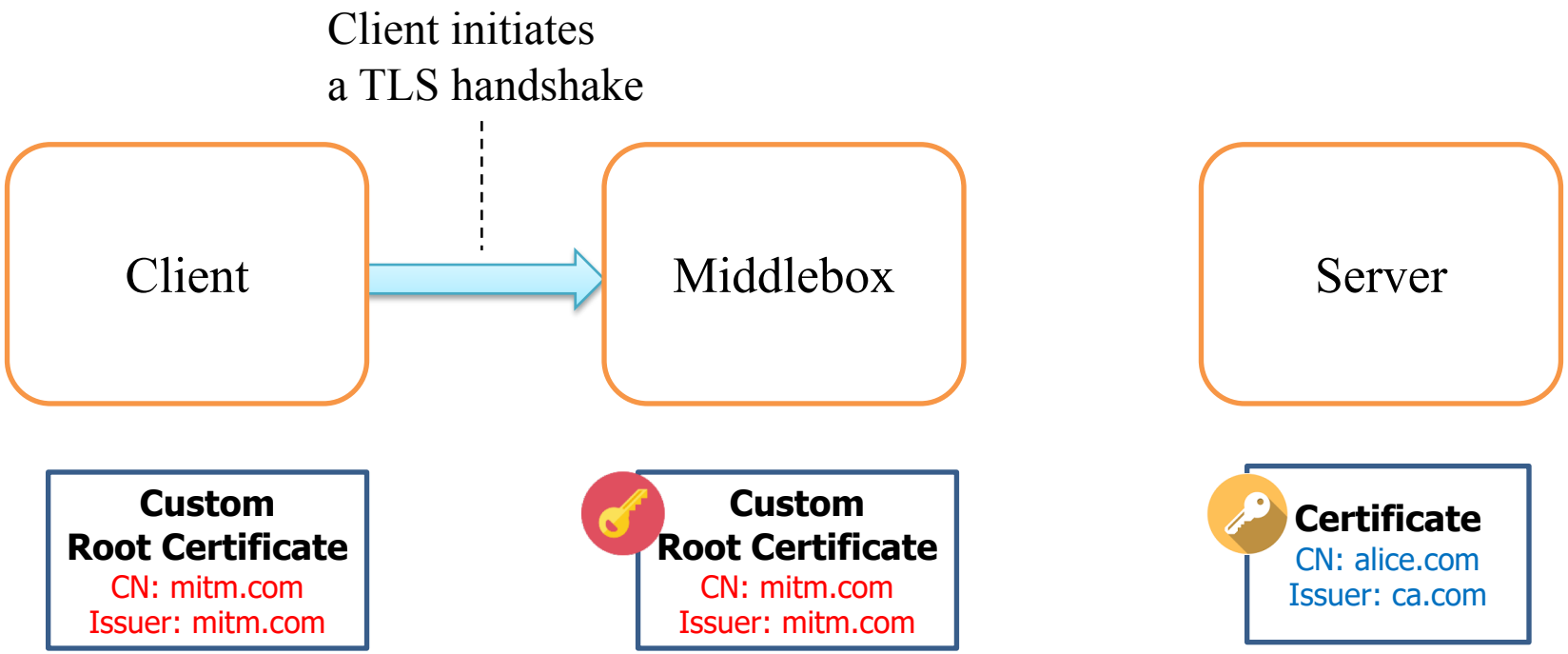
SplitTLS (2) Custom Root Certificate



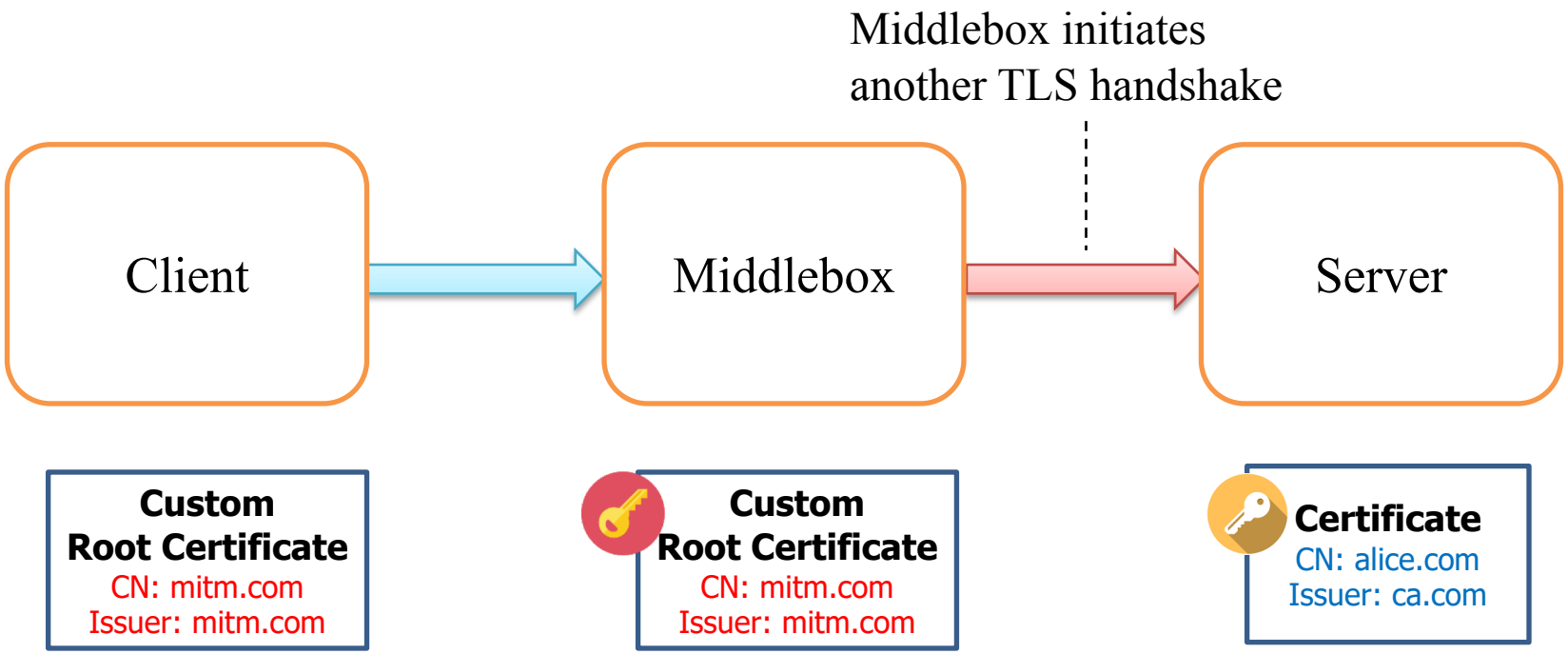
SplitTLS (2) Custom Root Certificate



SplitTLS (2) Custom Root Certificate

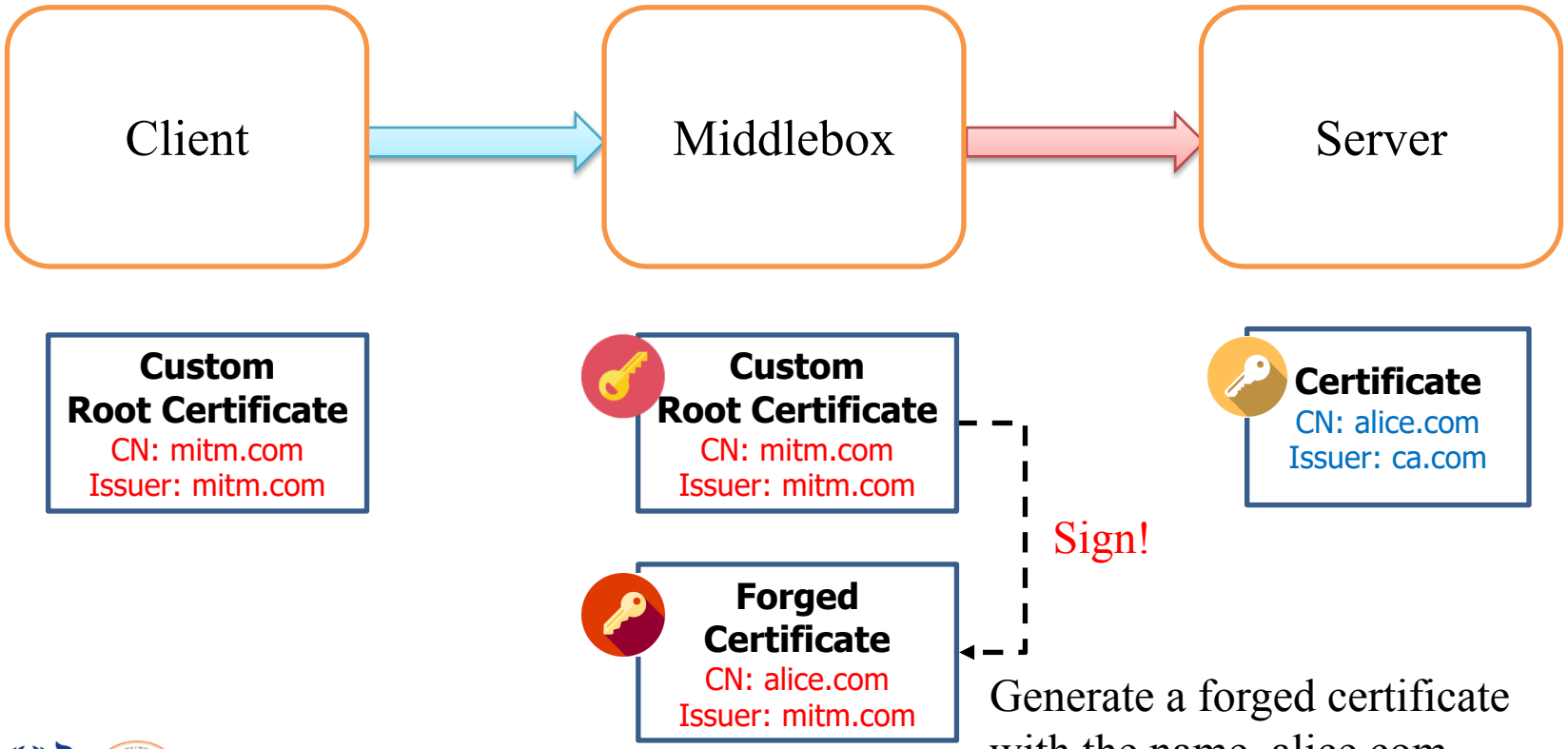


SplitTLS (2) Custom Root Certificate

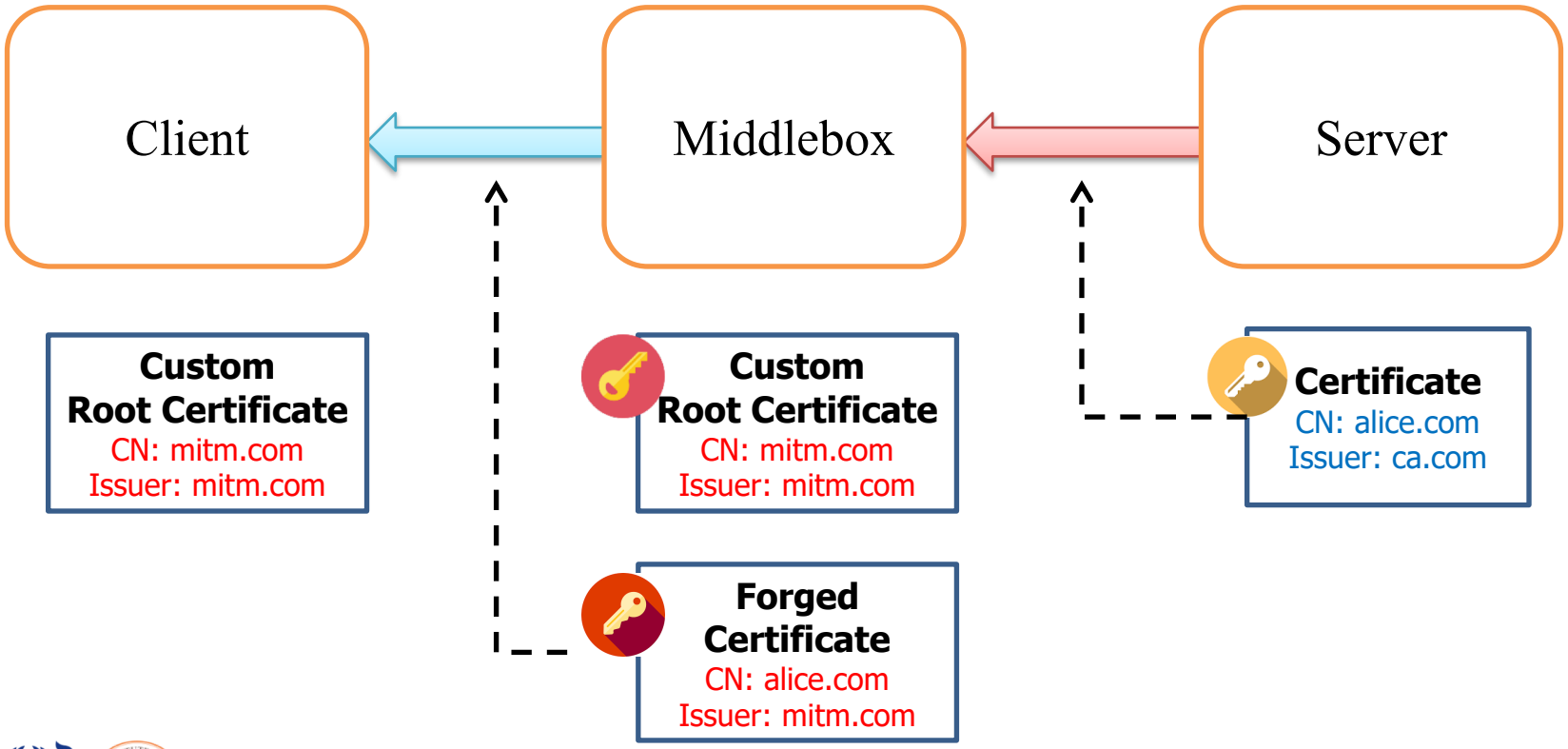


SplitTLS (2) Custom Root Certificate

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

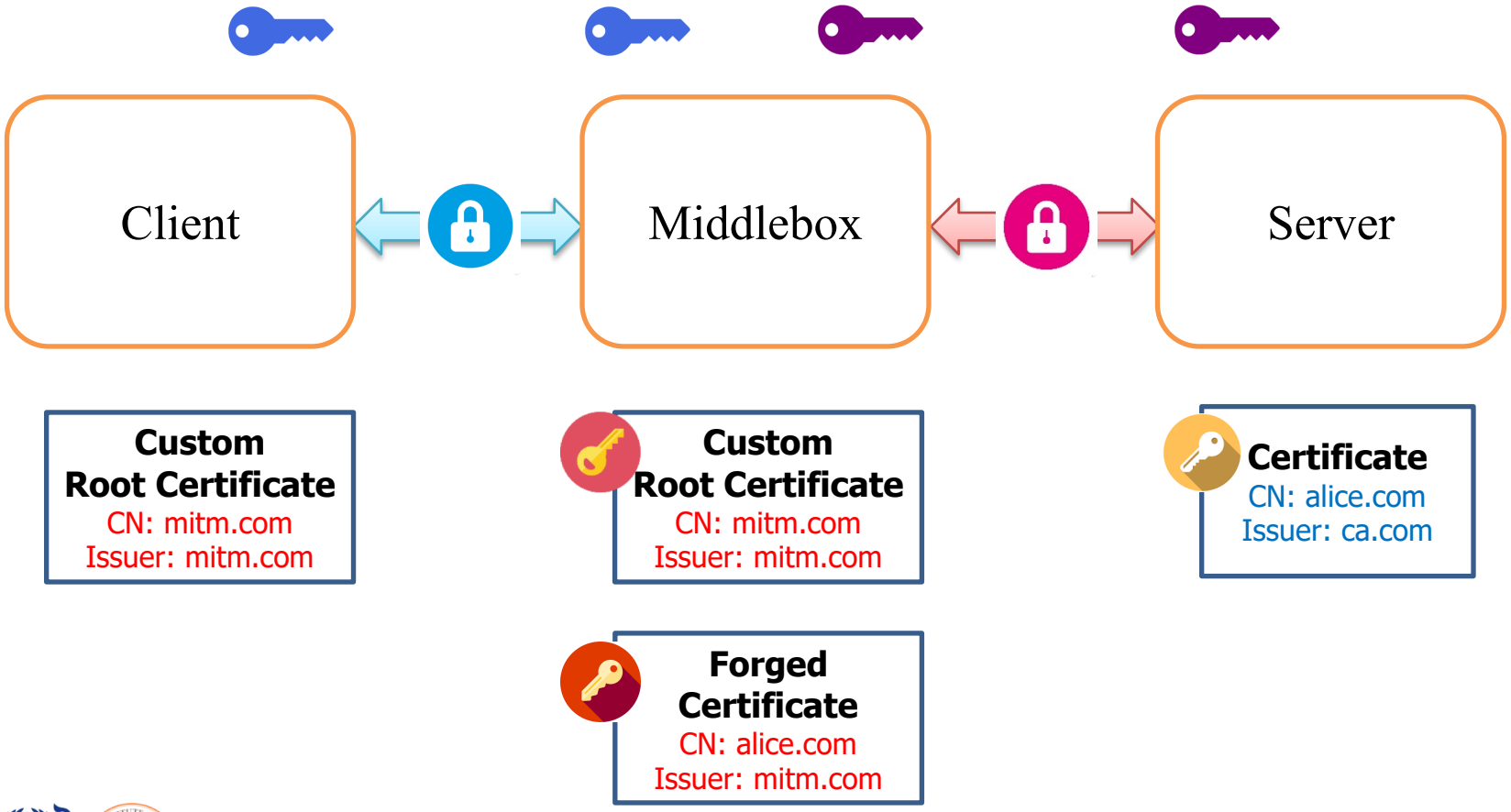


SplitTLS (2) Custom Root Certificate

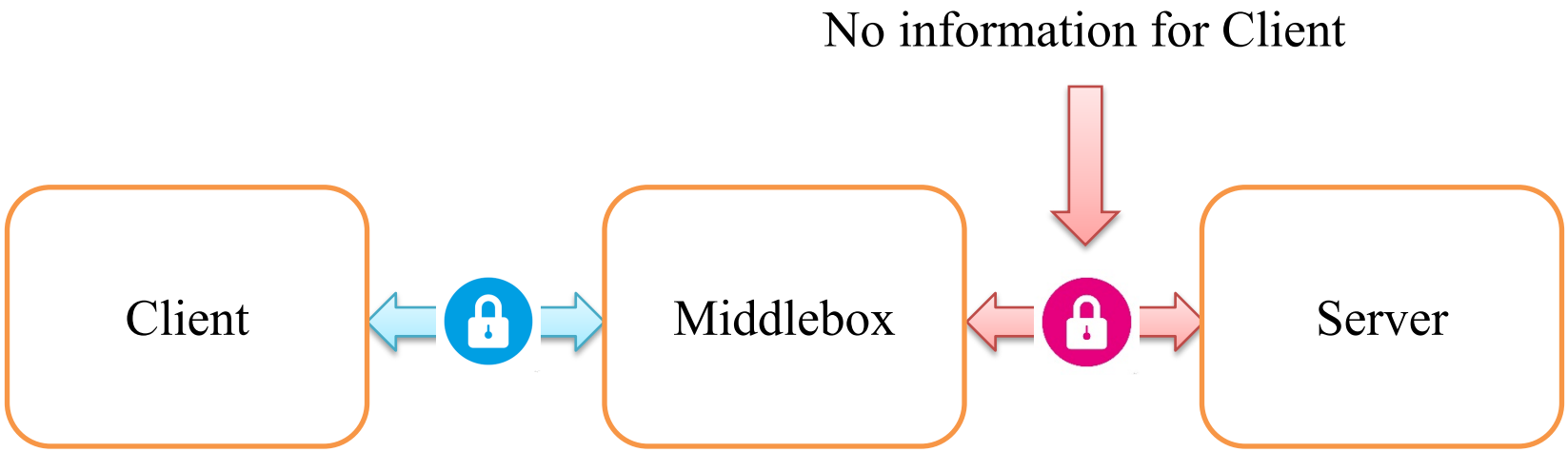


SplitTLS (2) Custom Root Certificate

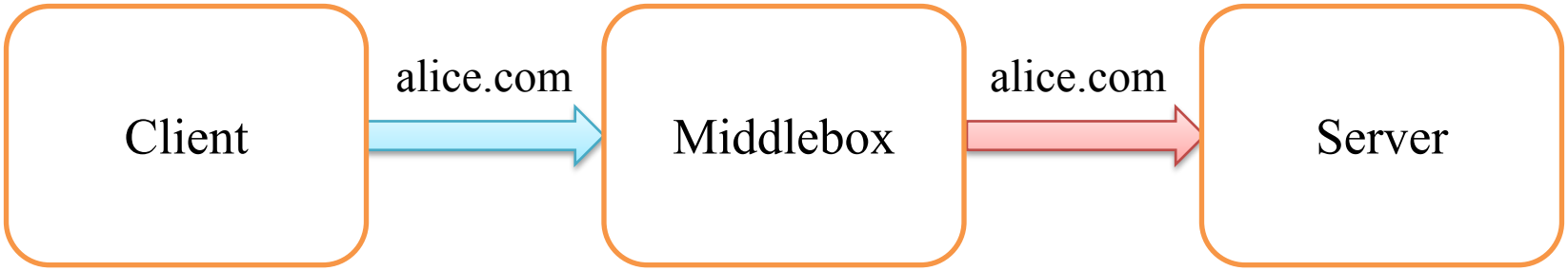
Client believes they have established a TLS session **with Server**, **not Middlebox!**



Problems in SplitTLS

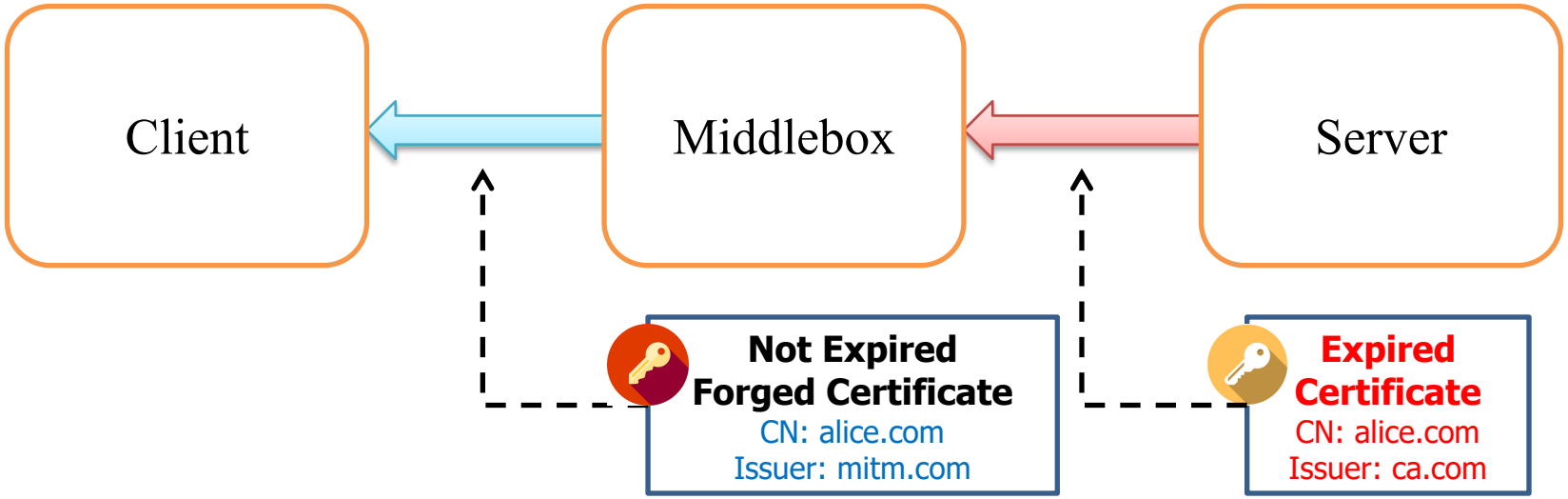


Problems in SplitTLS - Authentication



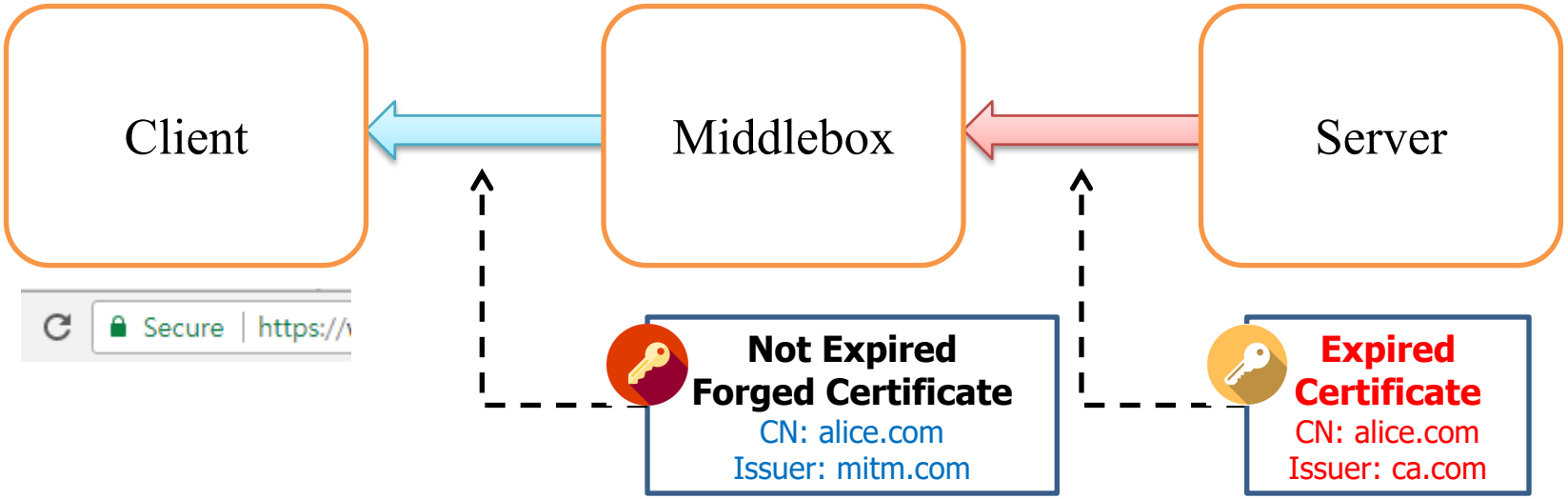
Authentication Client does not authenticate Server

Problems in SplitTLS - Authentication



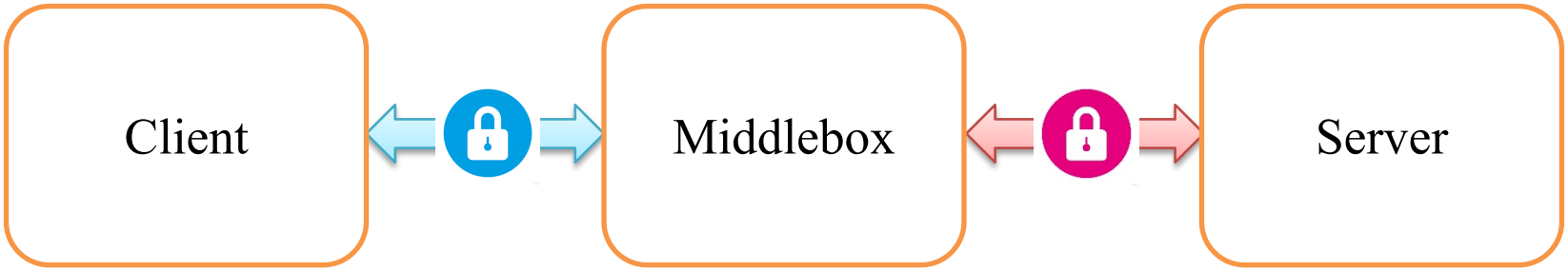
Authentication Client does not authenticate Server

Problems in SplitTLS - Authentication



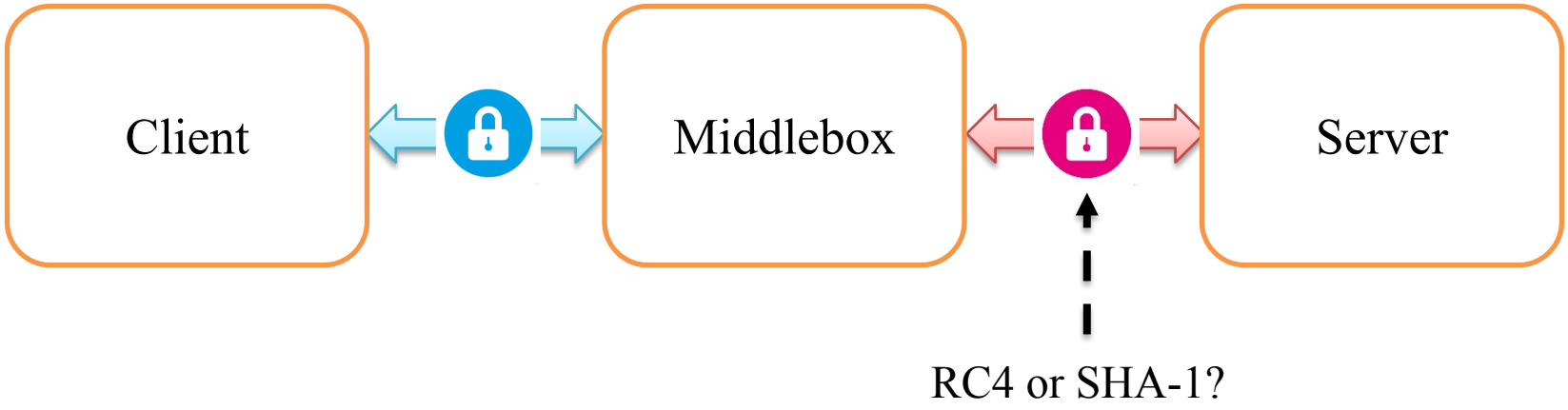
Authentication Client does not authenticate Server

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

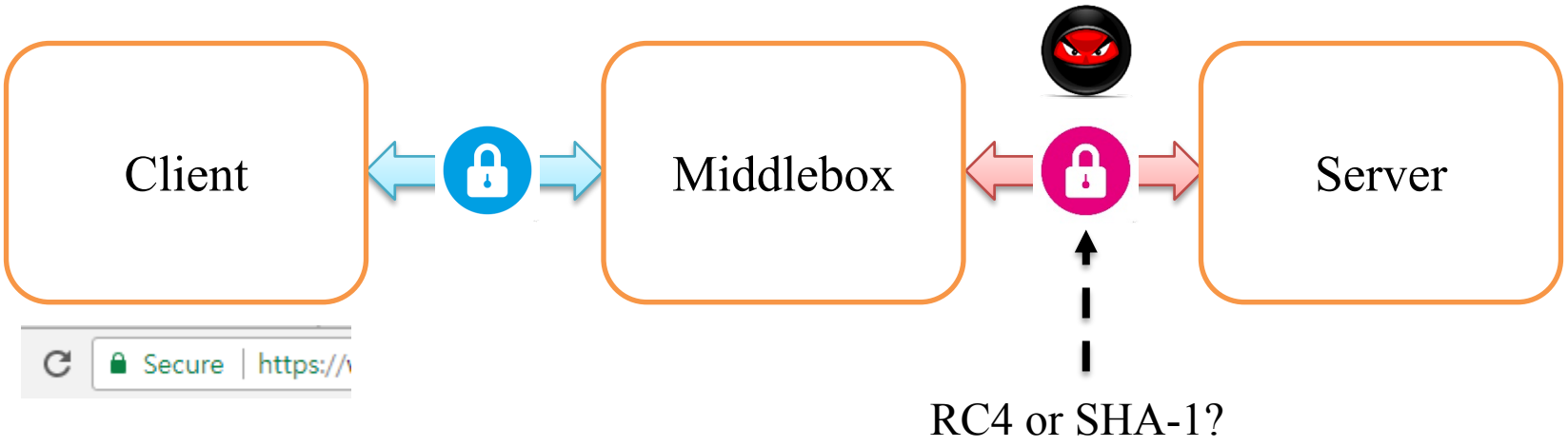
Problems in SplitTLS - Confidentiality



Authentication Client does not authenticate Server

Confidentiality Client does not know whether or not the segment is encrypted with a strong ciphersuite

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

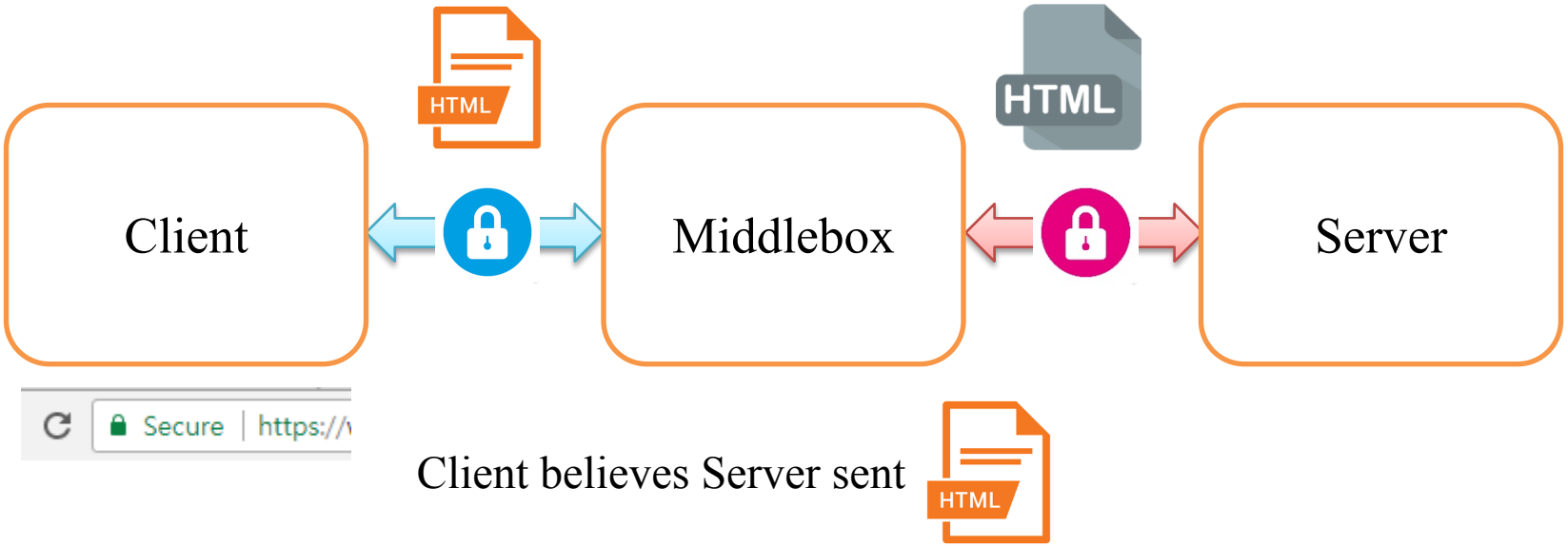
- ***Authentication*** — Client does not authenticate Server
- ***Confidentiality*** — Client does not know whether or not the segment is encrypted with a strong ciphersuite
- ***Integrity*** — Client cannot confirm that Server sent the message, or which middleboxes have modified it

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

	Problems in SplitTLS	Solution in maTLS
Authentication	Client can't authenticate Server	Explicit Authentication

Goal: Middlebox-aware TLS (maTLS)

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

	Problems in SplitTLS	Solution in maTLS
Authentication	Client can't authenticate Server	Explicit Authentication
Confidentiality	Client can't know if each of the segments has been encrypted with strong ciphersuites	Security Parameter Verification

Goal: Middlebox-aware TLS (maTLS)

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

	Problems in SplitTLS	Solution in maTLS
Authentication	Client can't authenticate Server	Explicit Authentication
Confidentiality	Client can't know if each of the segments has been encrypted with strong ciphersuites	Security Parameter Verification
Integrity	Client can't confirm (1) who actually sent the message (2) if it has been modified	Valid Modification Checks

Auditable Middleboxes

Certificate Authority

Middlebox Transparency Log Server

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

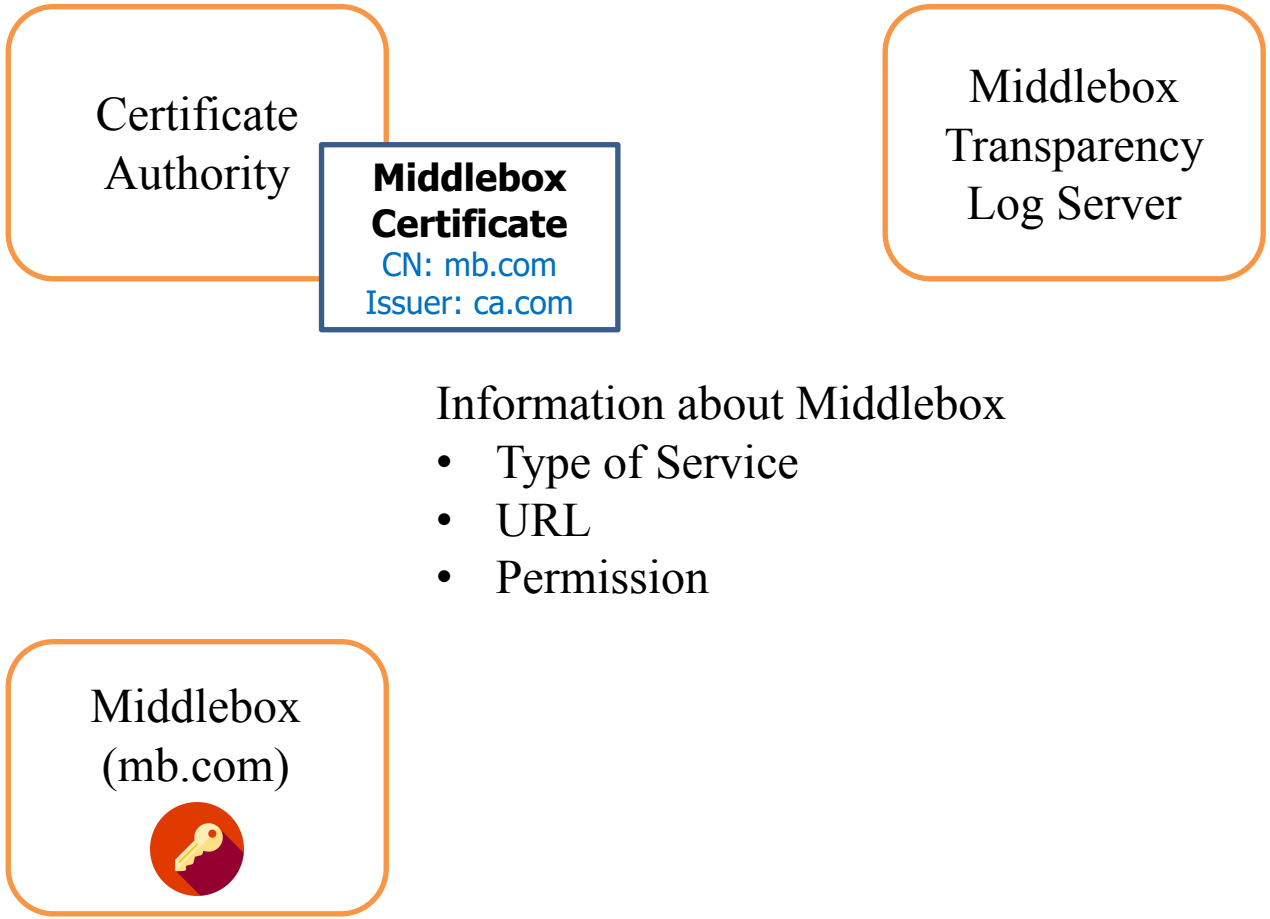
Middlebox (mb.com)

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

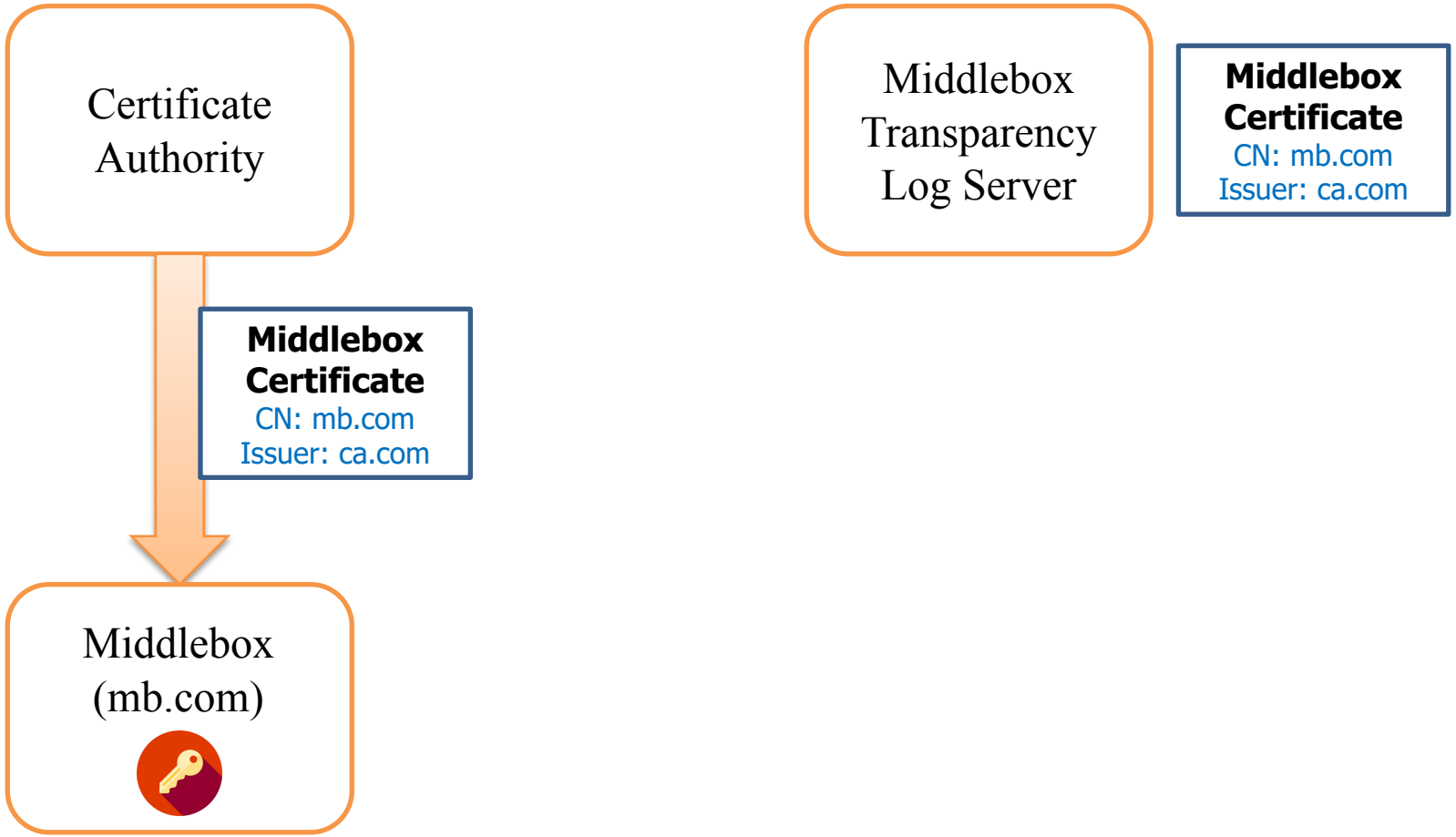
Auditable Middleboxes

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

Auditable Middleboxes



Auditable Middleboxes



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

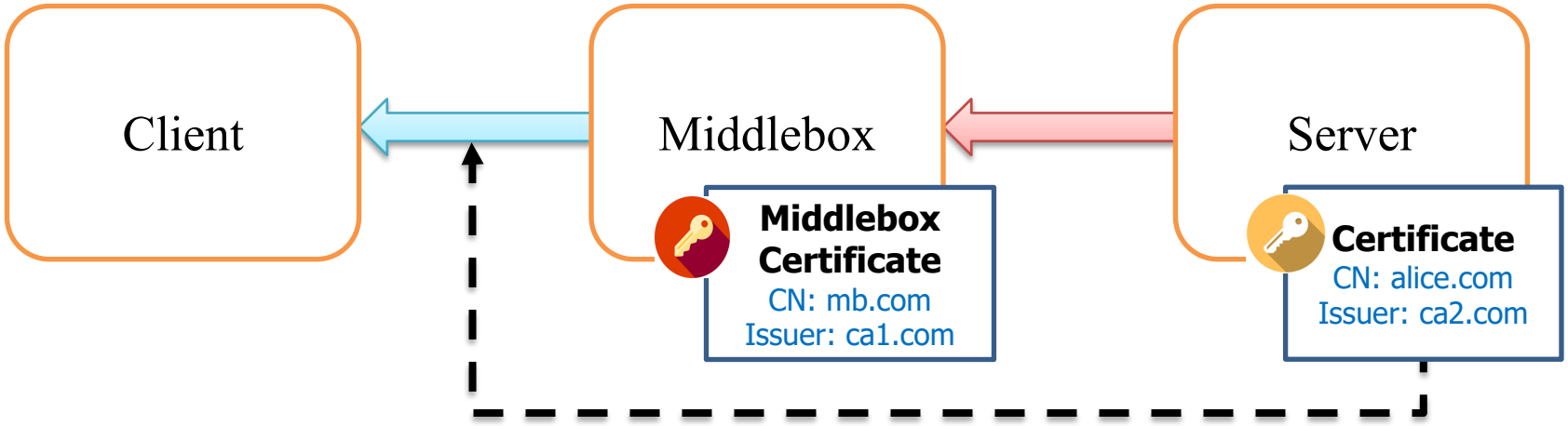
Revocability

Any incorrect middleboxes can be blocked following the ***certificate revocation mechanisms*** (e.g., CRL or OCSP)

Security Goals of maTLS

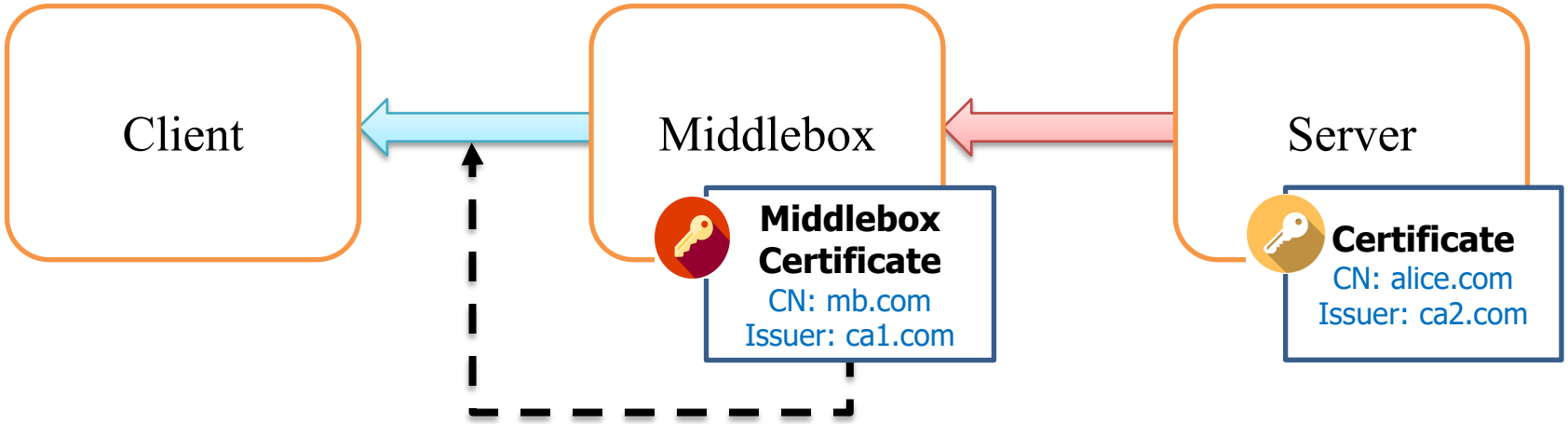
- ✓ Server Authentication
- ✓ Middlebox Authentication
- ✓ Segment Secrecy
- ✓ Individual Secrecy
- ✓ Data Source Authentication
- ✓ Modification Accountability
- ✓ Path Integrity

Security Goals of maTLS - Authentication



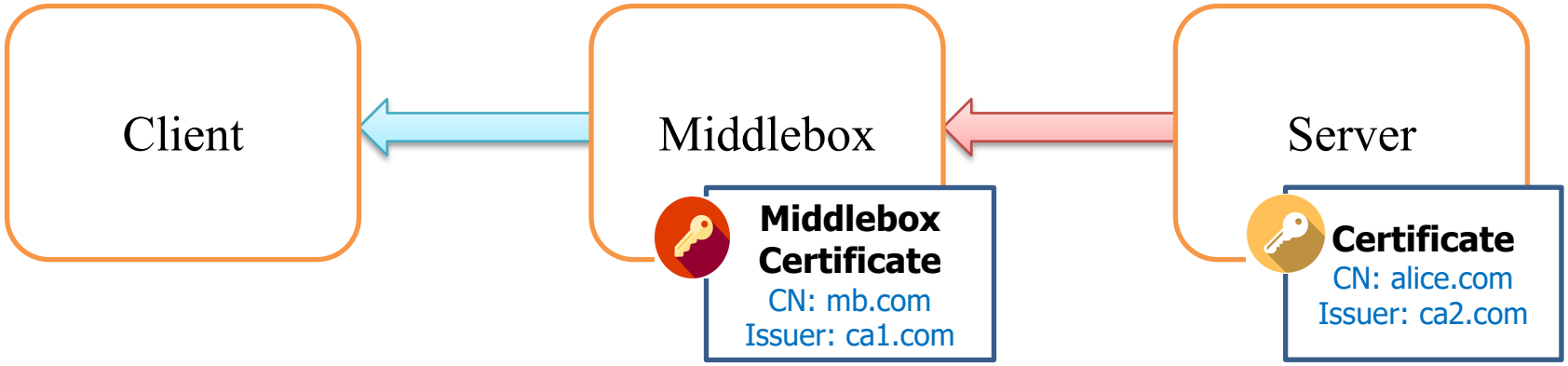
✓ Server Authentication

Security Goals of maTLS - Authentication



- ✓ Server Authentication
- ✓ Middlebox Authentication

Security Goals of maTLS - Authentication

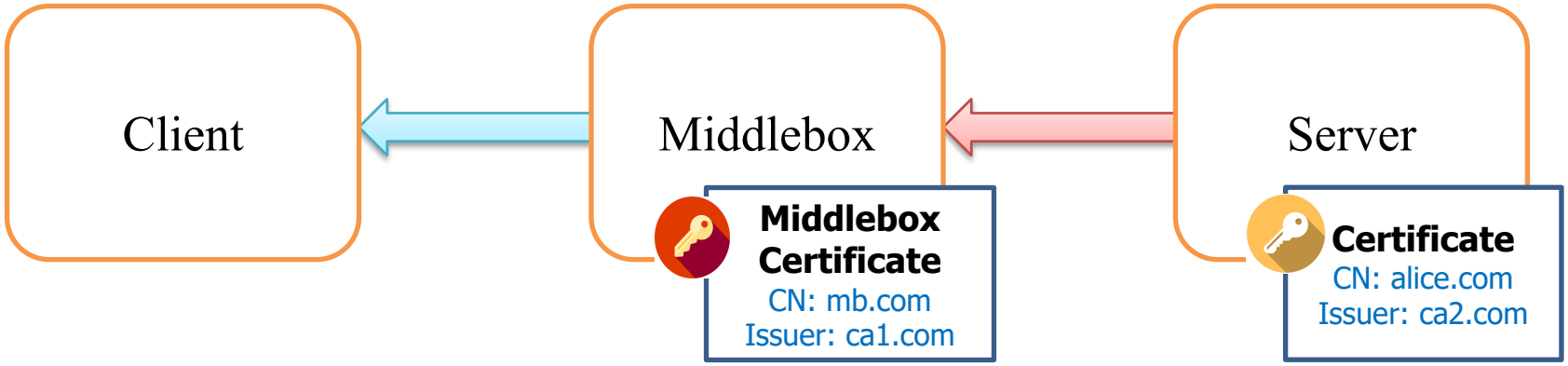


✓ Server Authentication

✓ Middlebox Authentication

⇒ Explicit Authentication

Explicit Authentication

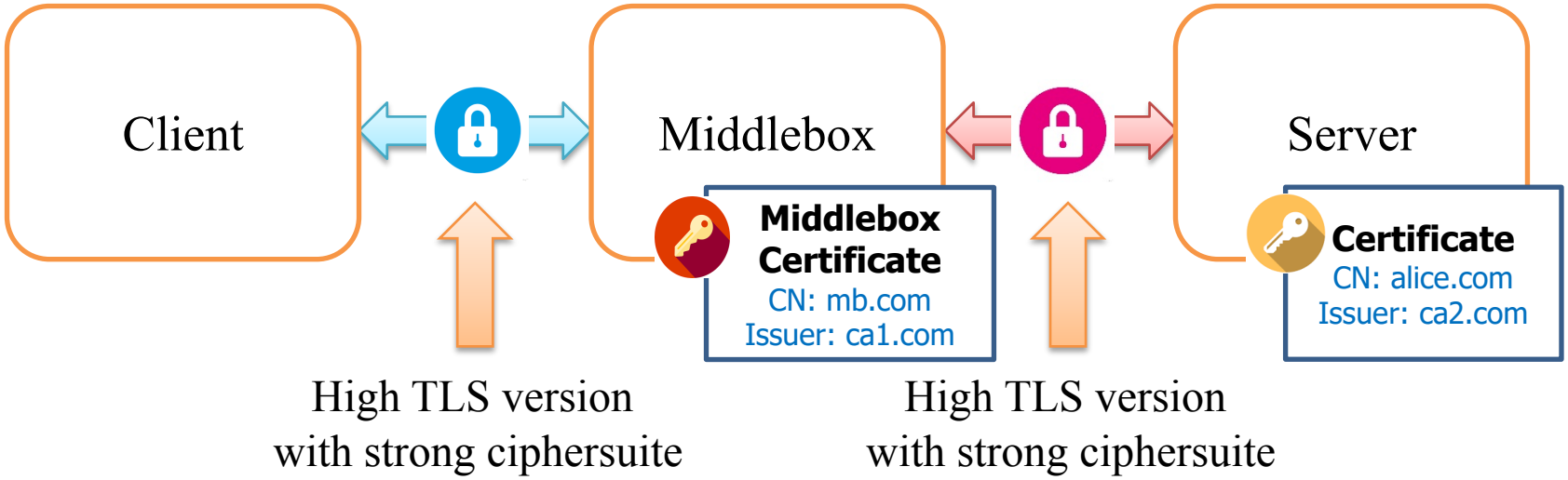


Certificate Blocks

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

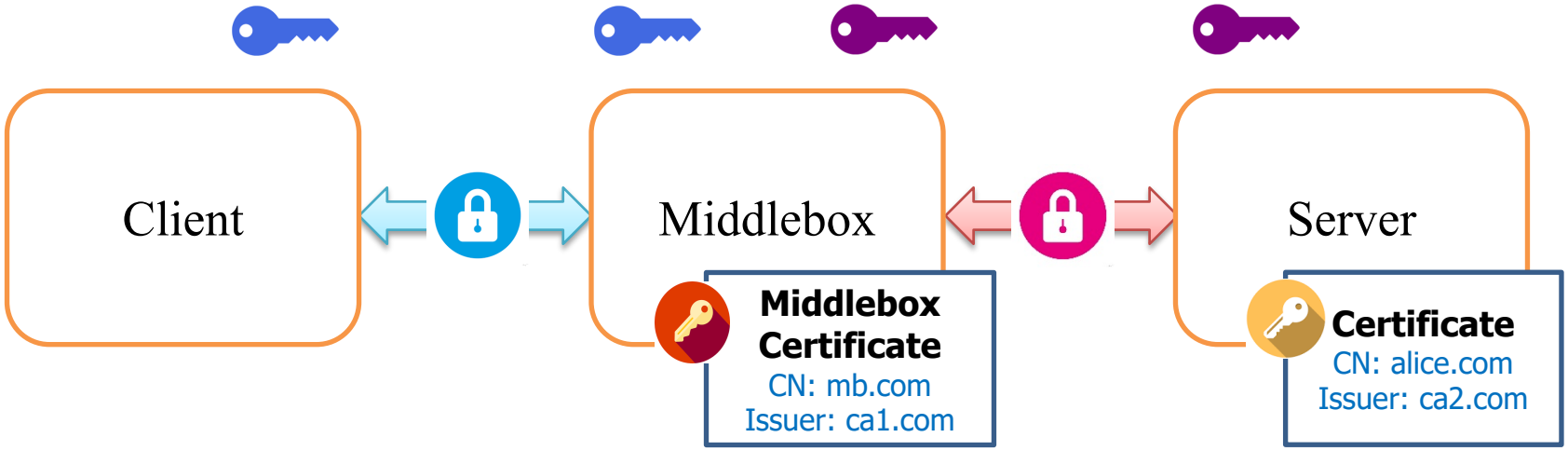
No impersonation

Security Goals of maTLS - Confidentiality



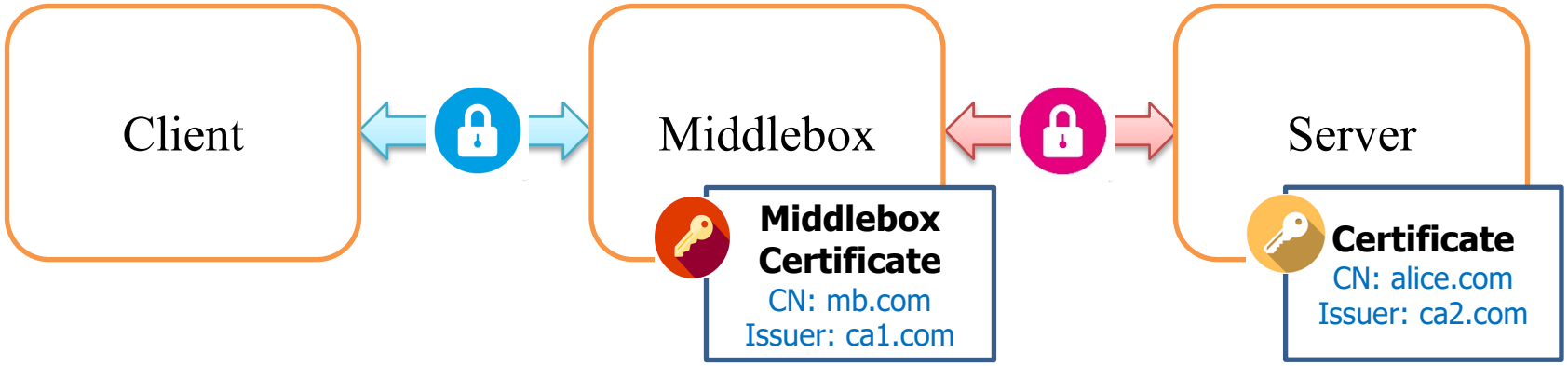
✓ Segment Secrecy

Security Goals of maTLS - Confidentiality



- ✓ Segment Secrecy
- ✓ Individual Secrecy

Security Goals of maTLS - Confidentiality

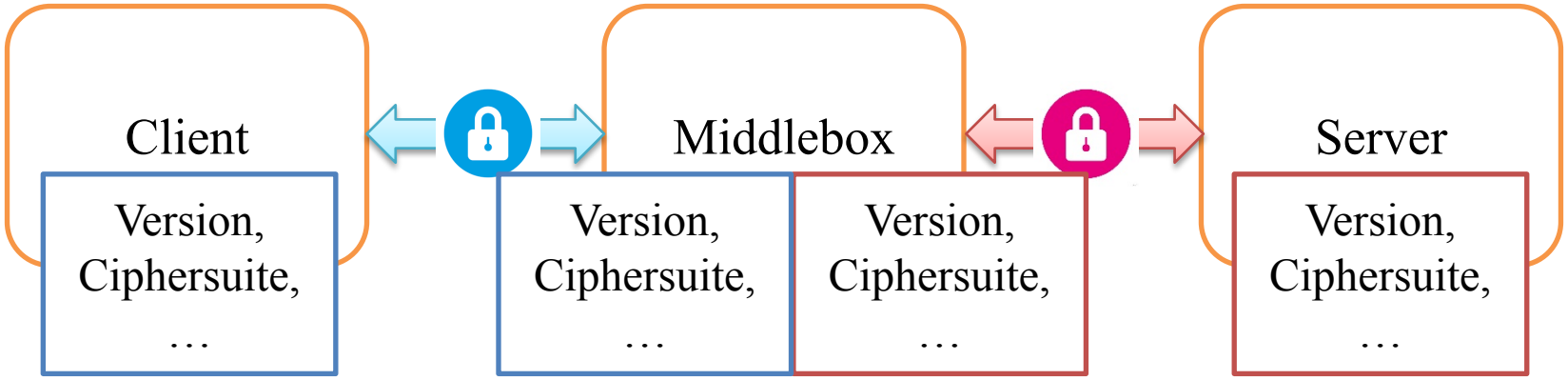


✓ Segment Secrecy

✓ Individual Secrecy

⇒ Security Parameter Verification

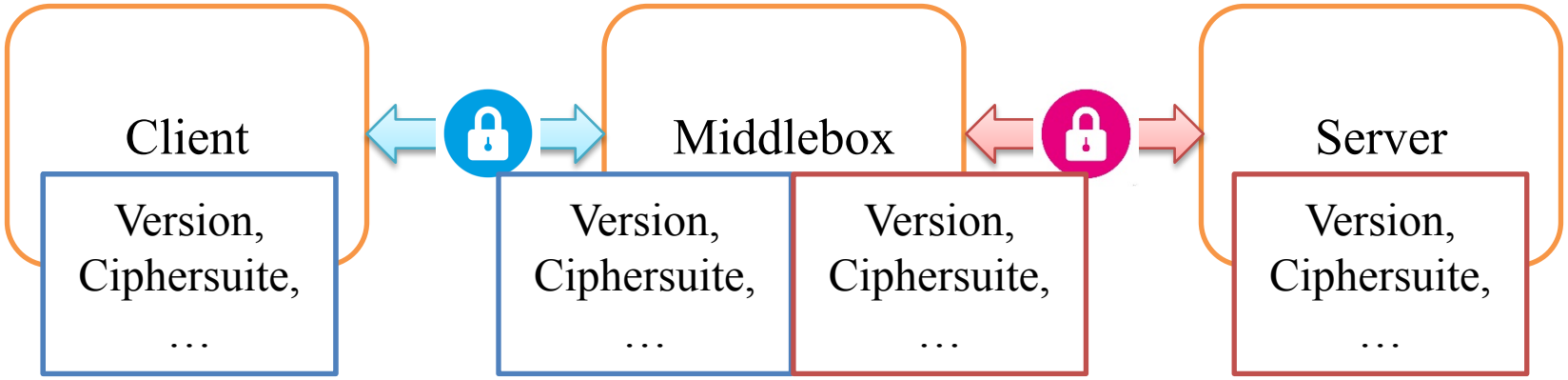
Security Parameter Verification



Security Parameter Blocks

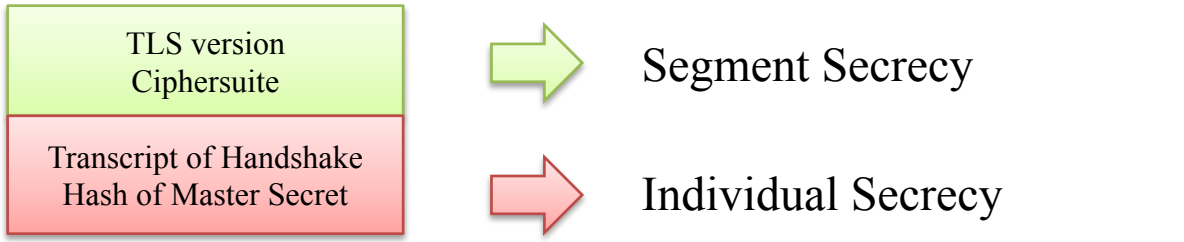
Each entity describes information about its related segment(s)

Security Parameter Verification

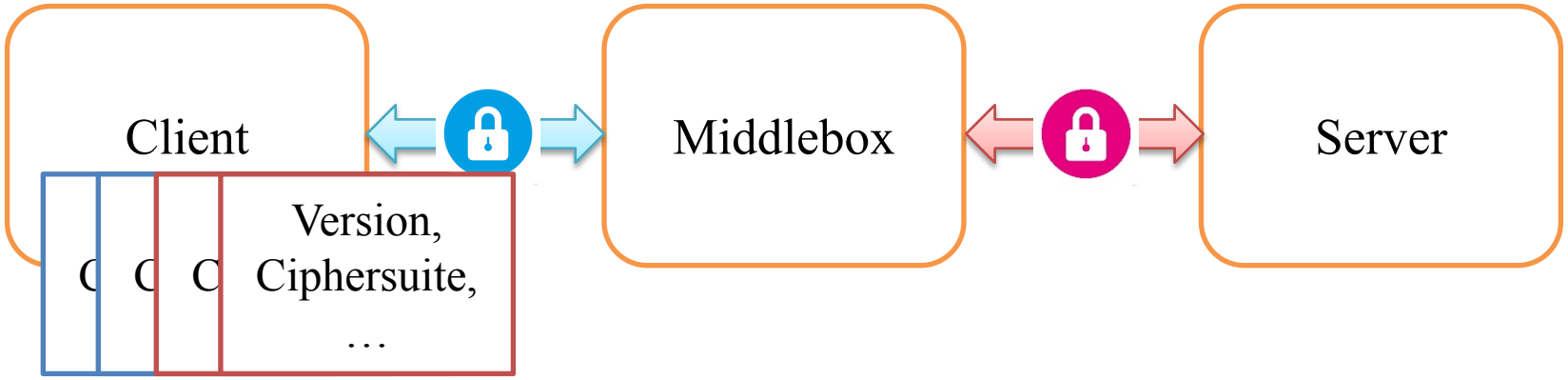


Security Parameter Blocks

Each entity describes information about its related segment(s)



Security Parameter Verification



Security Parameter Blocks

Each entity describes information about its related segment(s)

Report

TLS version
Ciphersuite

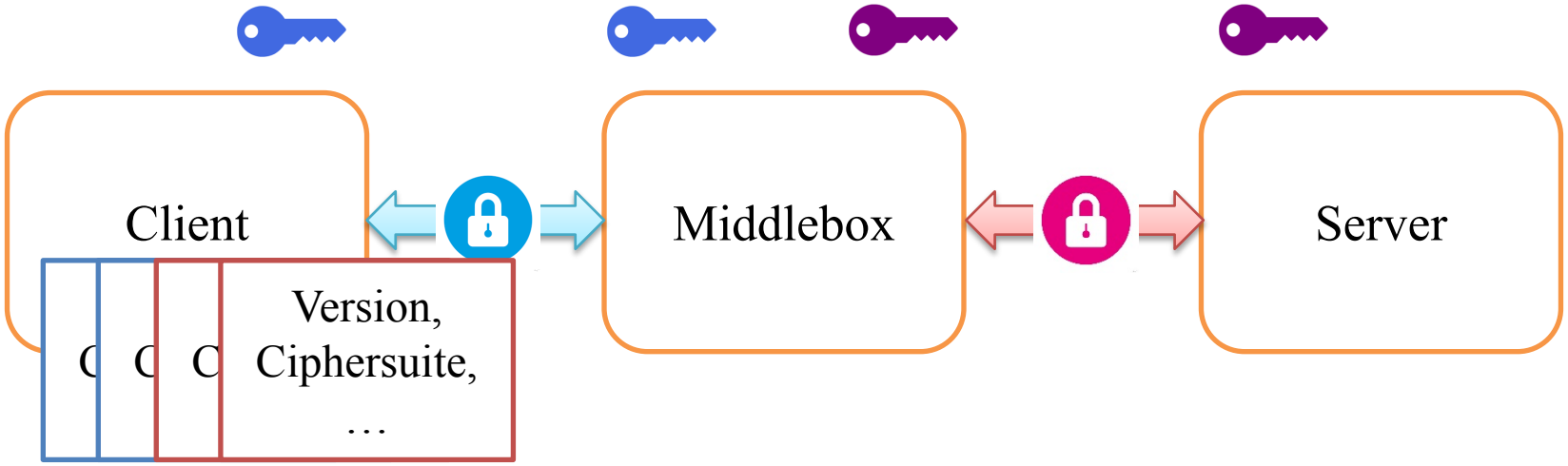
Transcript of Handshake
Hash of Master Secret

Segment Secrecy

Individual Secrecy

No low TLS versions and weak ciphersuites

Security Parameter Verification



Security Parameter Blocks

Each entity describes information about its related segment(s)

Report

TLS version
Ciphersuite

Transcript of Handshake
Hash of Master Secret

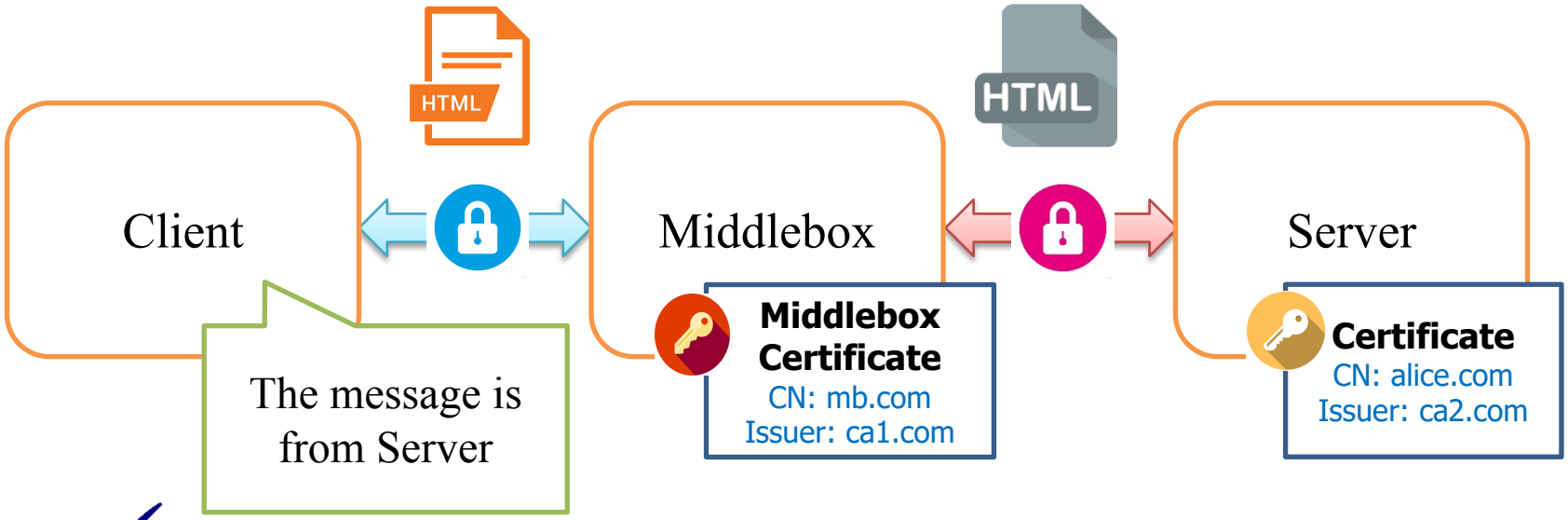
Segment Secrecy

Individual Secrecy

No low TLS versions and weak ciphersuites

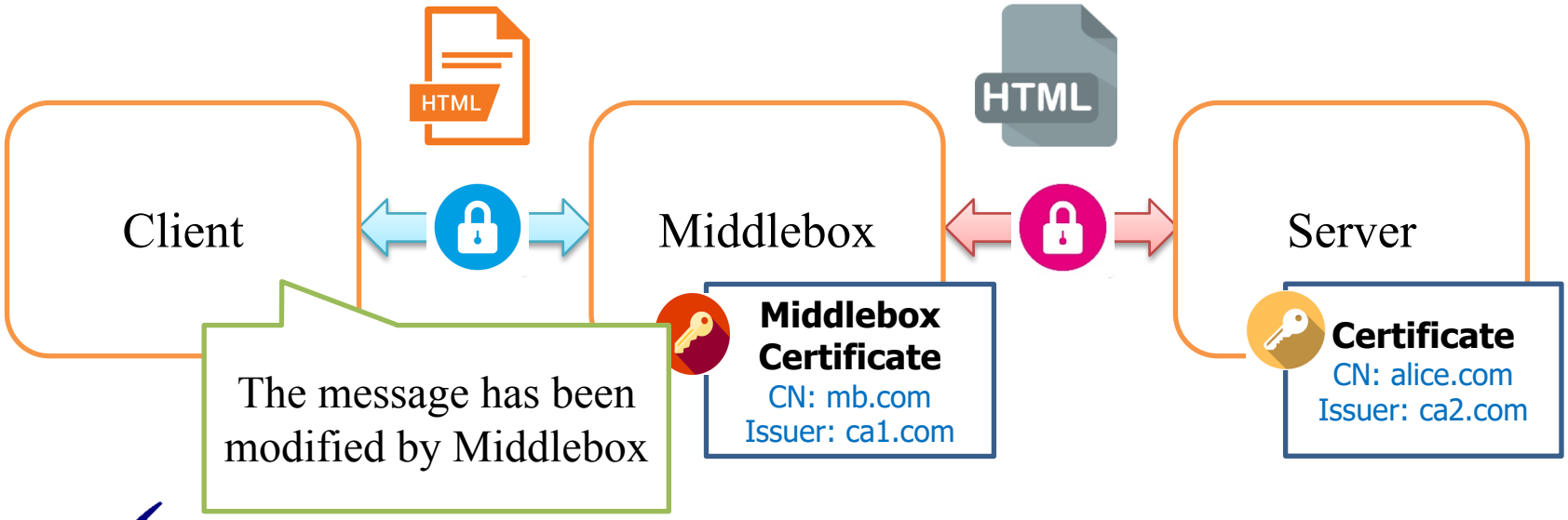
Confirmation of different segment keys

Security Goals of maTLS - Integrity



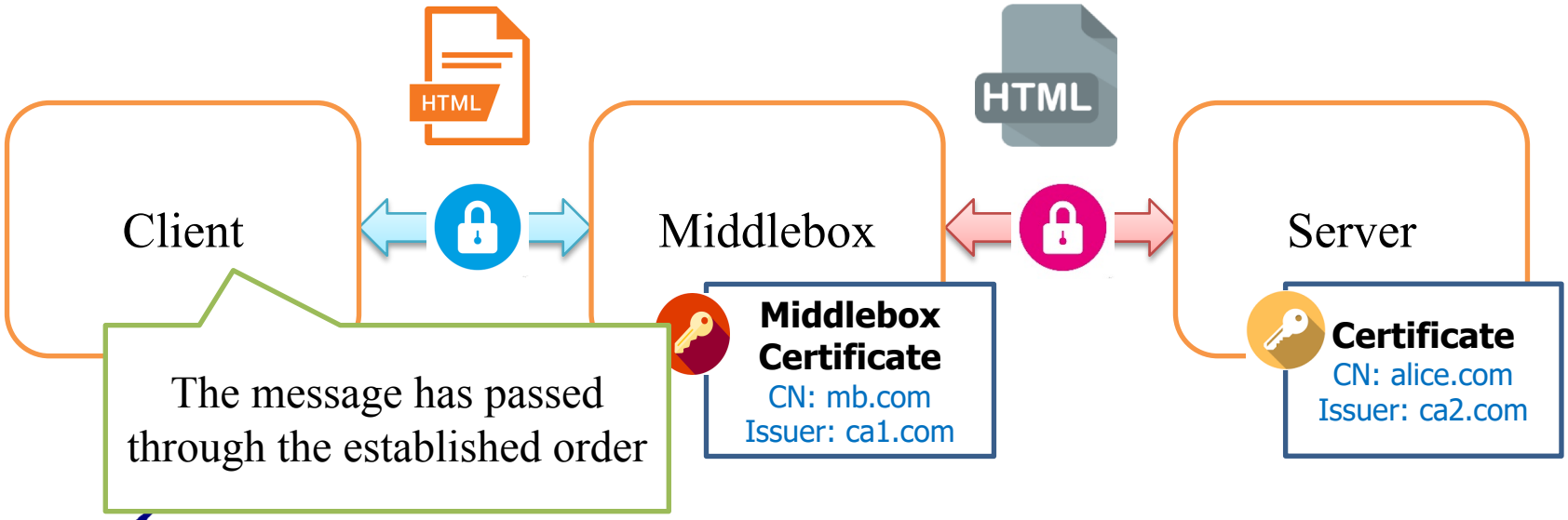
✓ Data Source Authentication

Security Goals of maTLS - Integrity



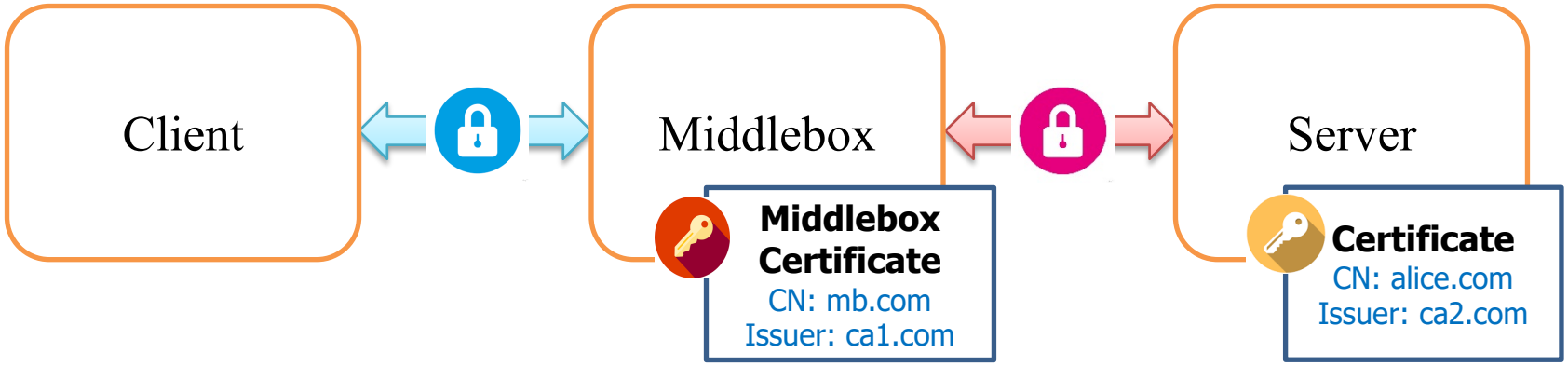
- ✓ Data Source Authentication
- ✓ Modification Accountability

Security Goals of maTLS - Integrity



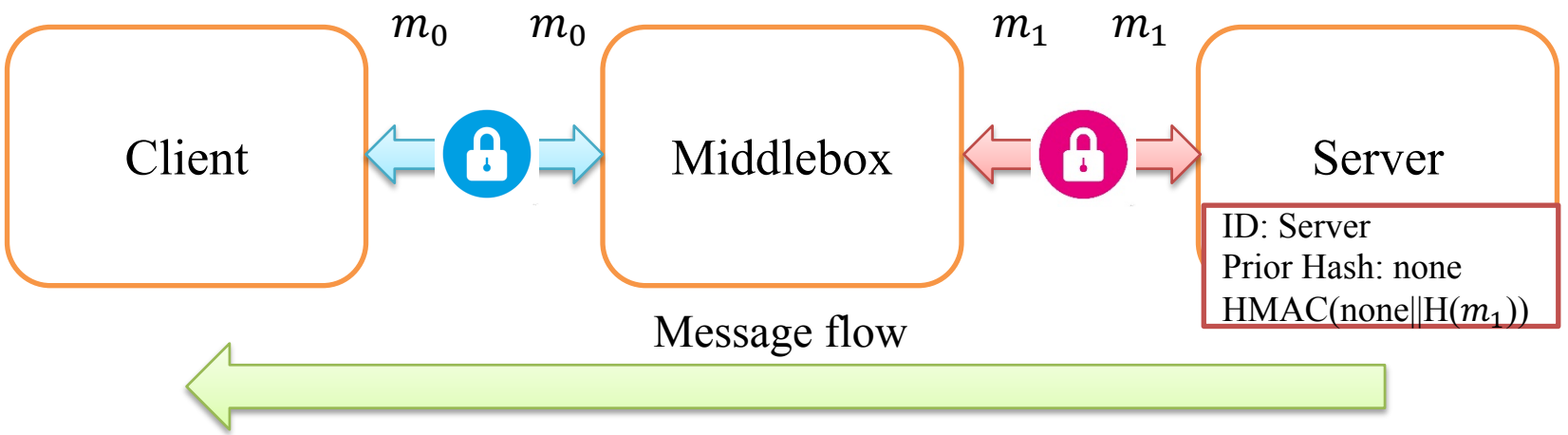
- ✓ Data Source Authentication
- ✓ Modification Accountability
- ✓ Path Integrity

Security Goals of maTLS - Integrity



- ✓ Data Source Authentication
- ✓ Modification Accountability → Valid Modification Checks
- ✓ Path Integrity

Valid Modification Checks



Modification Log Blocks

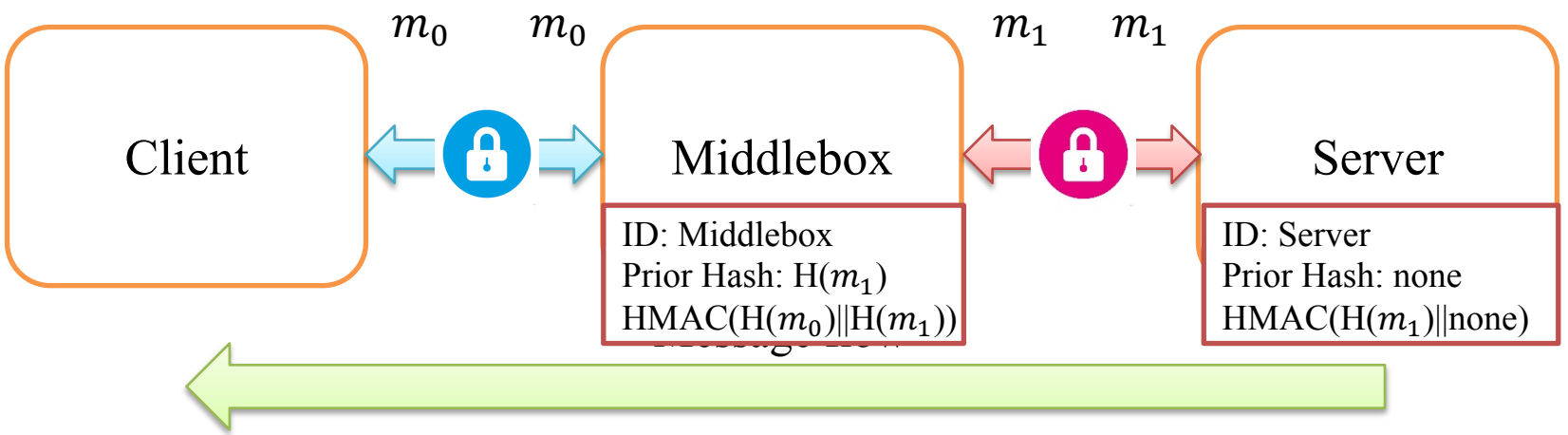
Each entity describes information about its modification by using HMAC (The HMAC key is called an ***accountability key***)

$m \rightarrow m'$

ID	$H(m)$	$HMAC(H(m')\|\|H(m))$

* Optimization on a Modification Log is described in the paper

Valid Modification Checks



Modification Log Blocks

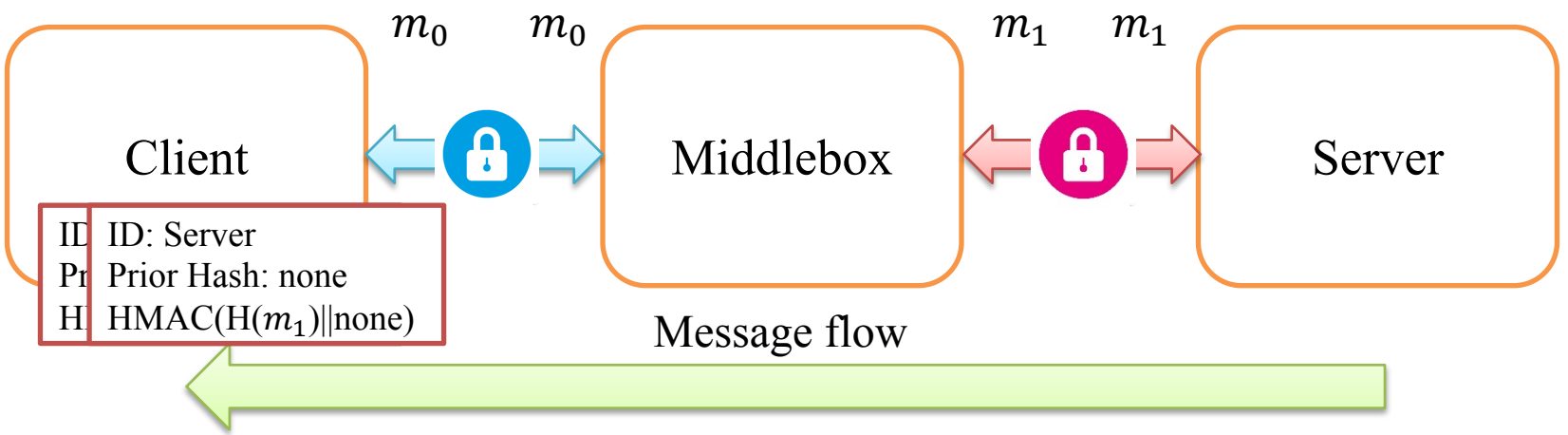
Each entity describes information about its modification by using HMAC (The HMAC key is called an ***accountability key***)

$m \rightarrow m'$

ID	$H(m)$	$HMAC(H(m')\|\|H(m))$

* Optimization on a Modification Log is described in the paper

Valid Modification Checks



Modification Log Blocks

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

Report

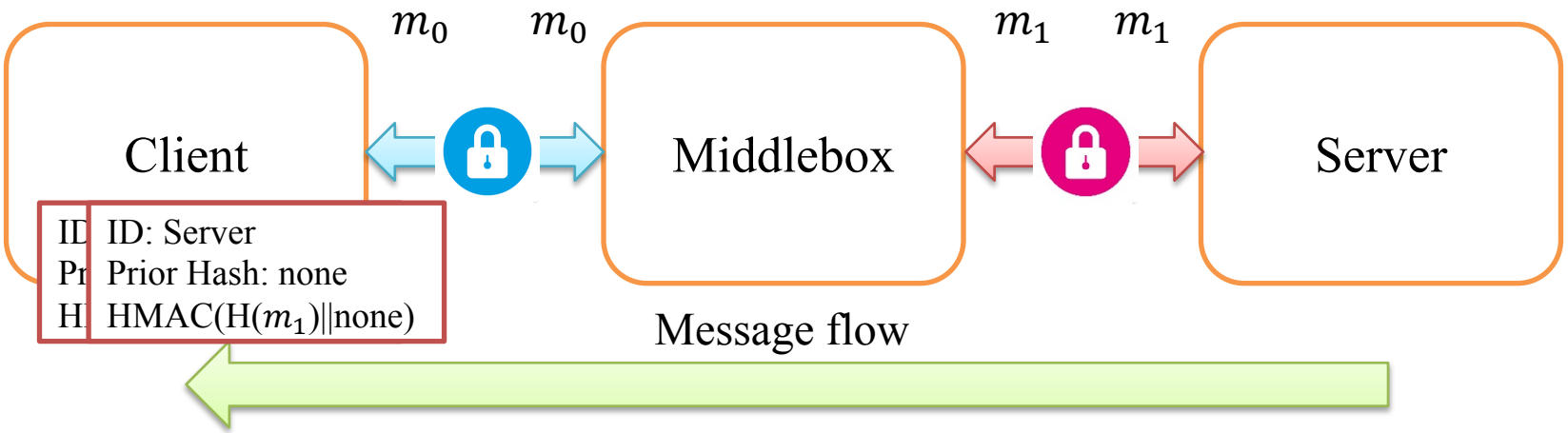
$m \rightarrow m'$

ID	$H(m)$	$HMAC(H(m')\|\|H(m))$

Confirmation of who sends and who modifies the message

* Optimization on a Modification Log is described in the paper

Valid Modification Checks



Modification Log Blocks

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

Report $m \rightarrow m'$

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

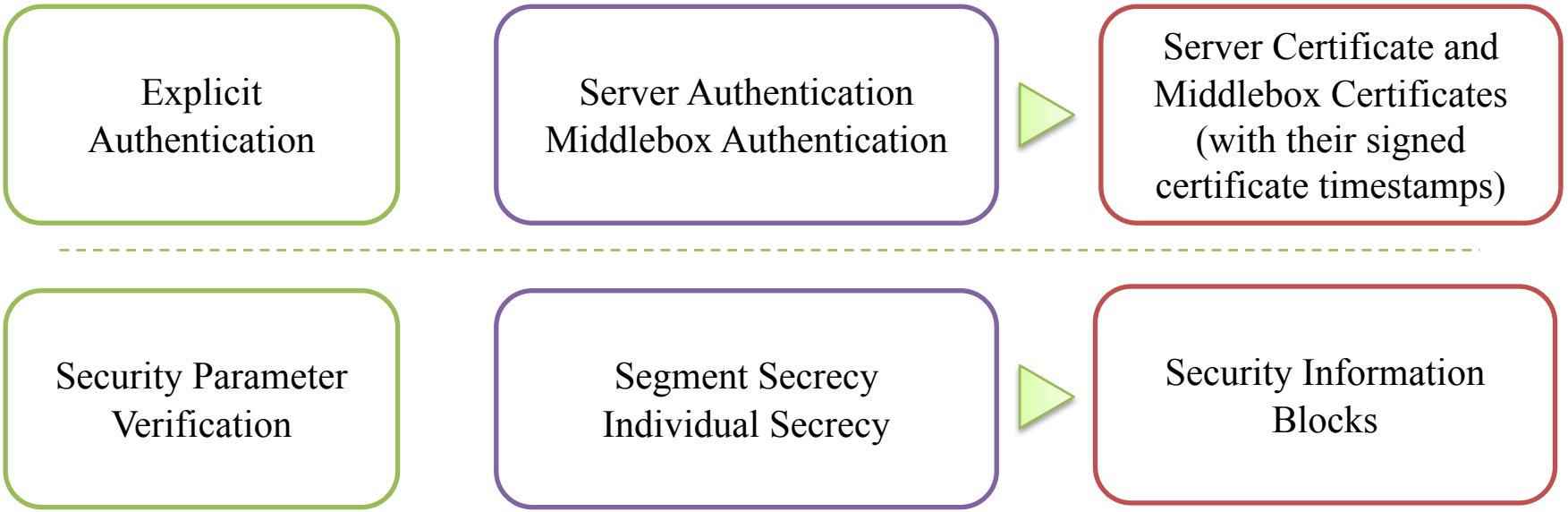
Summary of Audit Mechanisms

Explicit Authentication

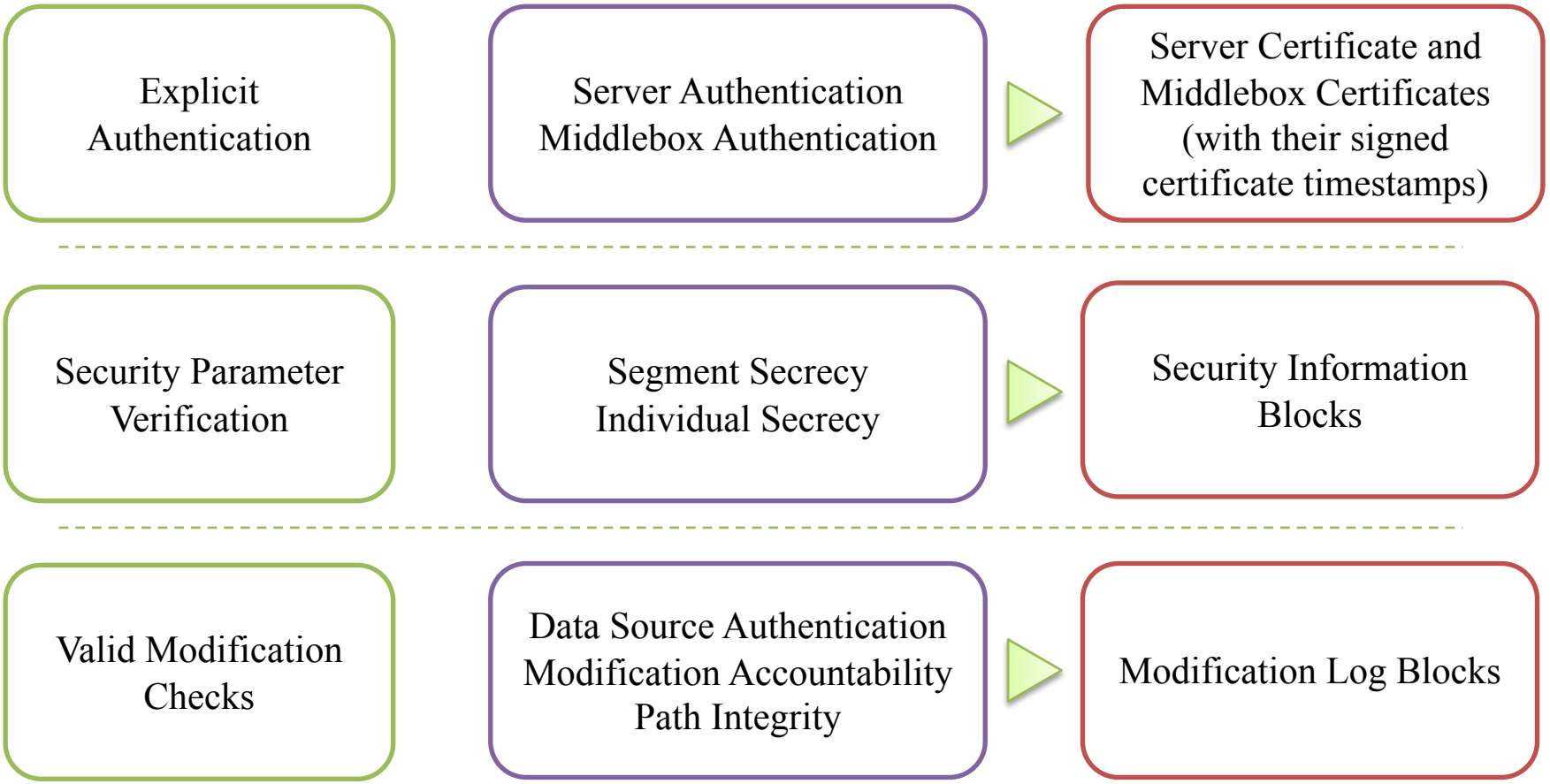
Server Authentication
Middlebox Authentication

Server Certificate and Middlebox Certificates (with their signed certificate timestamps)

Summary of Audit Mechanisms

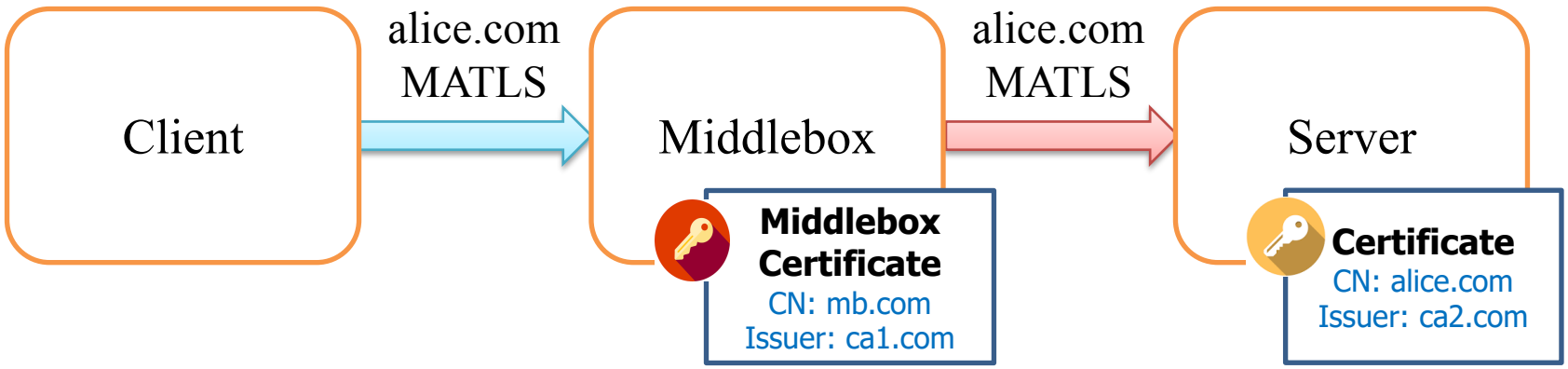


Summary of Audit Mechanisms



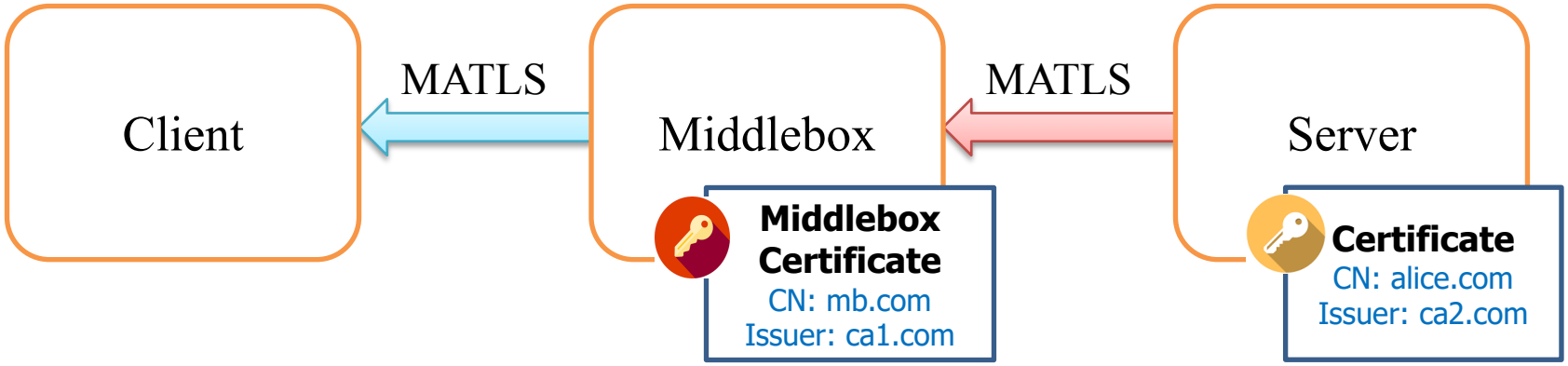
maTLS Handshake

maTLS Handshake



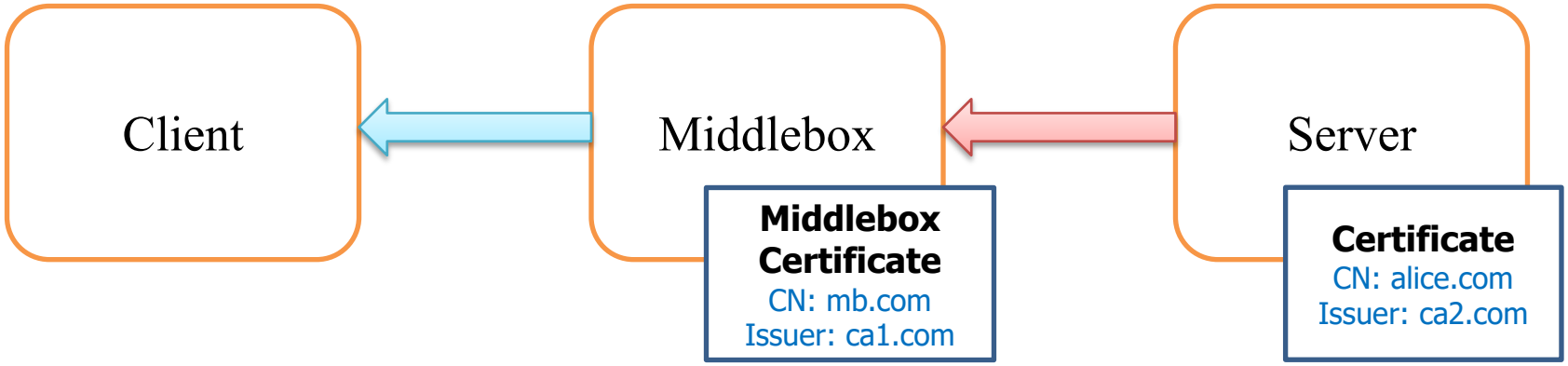
- ClientHello and ServerHello,

maTLS Handshake



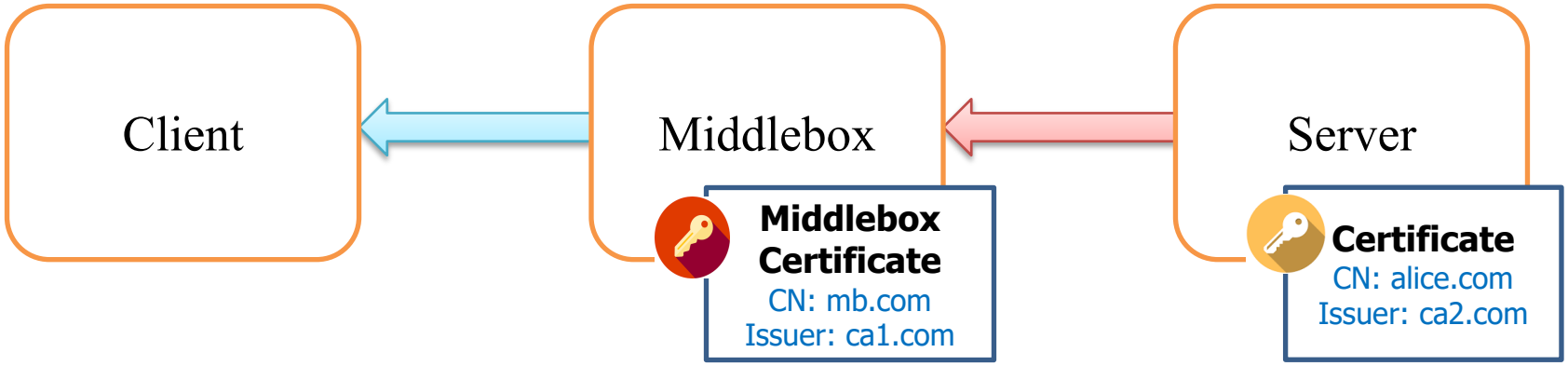
- ClientHello and ServerHello,
  - Each segment negotiates its TLS version and ciphersuite
  - Each entity establishes HMAC keys (accountability keys)

maTLS Handshake



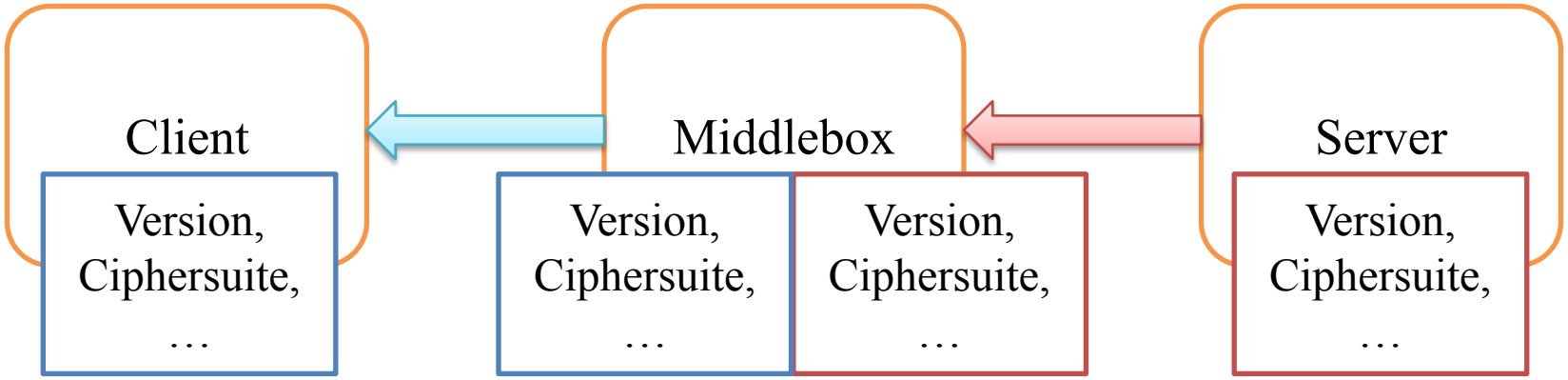
- ✓ ClientHello and ServerHello,
  - Each segment negotiates its TLS version and ciphersuite
  - Each entity establishes HMAC keys (accountability keys)
- ✓ Certificate, ⇒ Explicit Authentication

maTLS Handshake



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

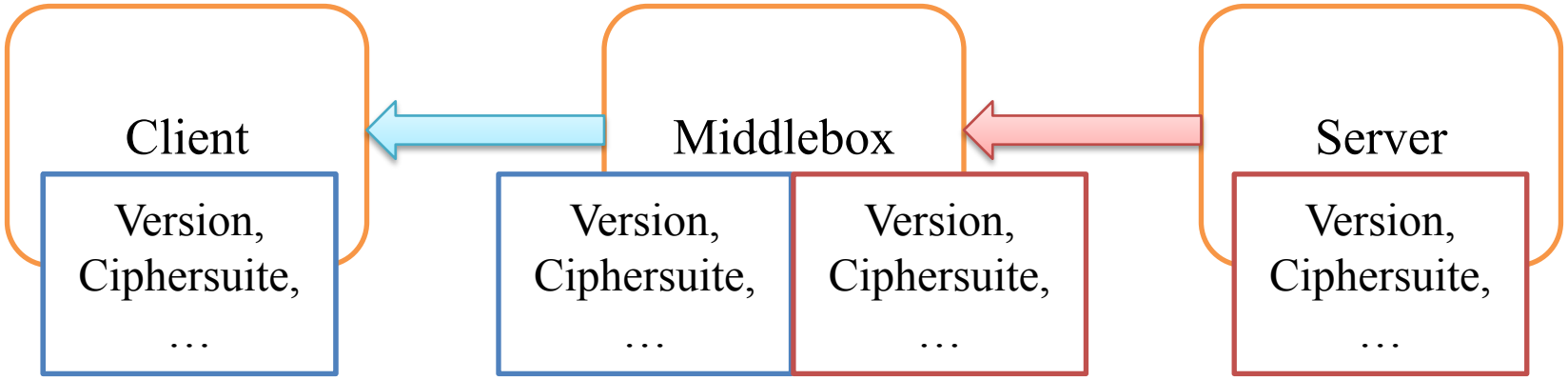
maTLS Handshake



✓ Finished

Each segment confirms the transcript of their handshake

maTLS Handshake

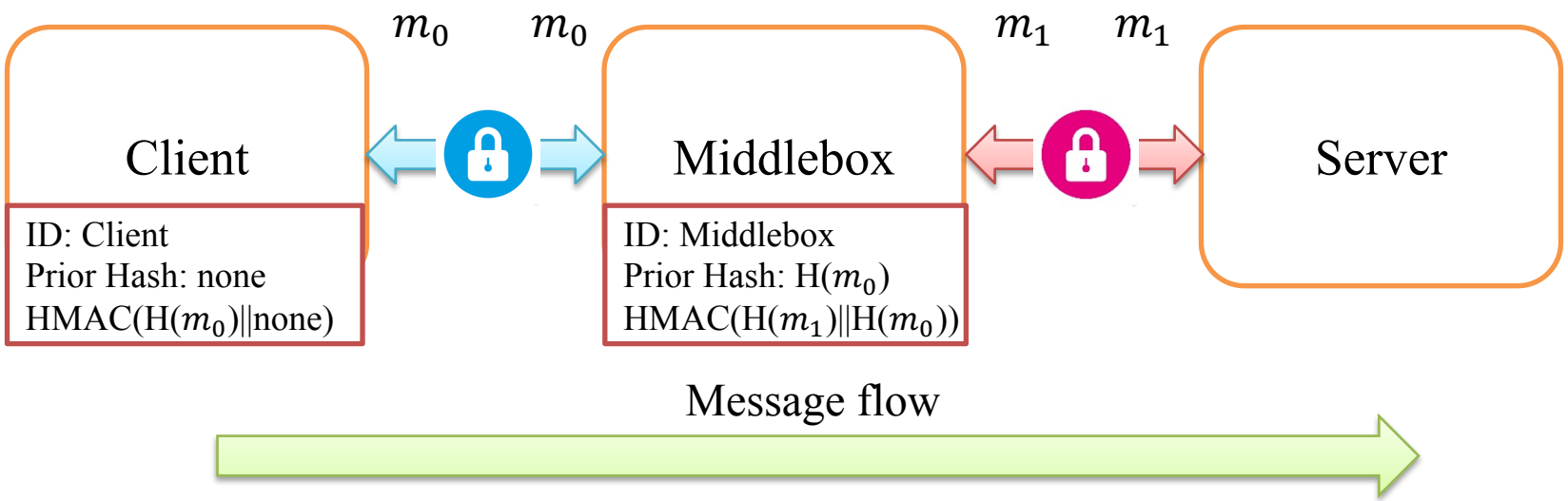


✓ Finished

Each segment confirms the transcript of their handshake

✓ ExtendedFinished ⇒ Security Parameter Verification

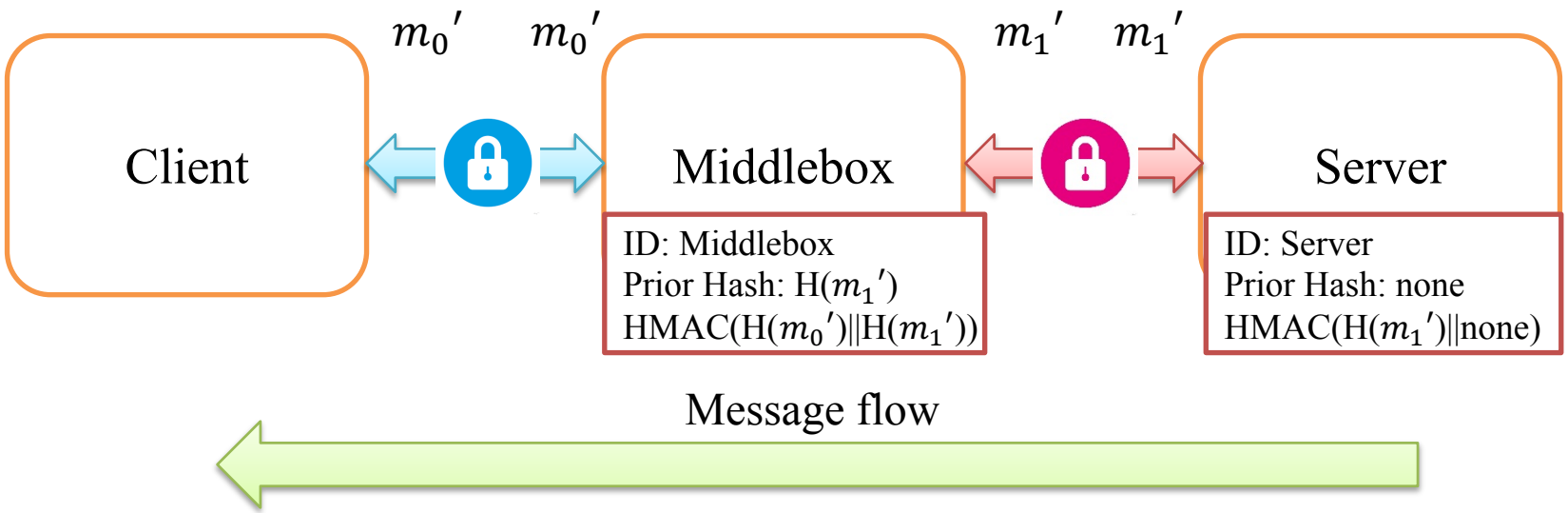
maTLS Record



✓ Data Exchange ⇨ Valid Modification Checks

* Optimization on a Modification Log is described in the paper

maTLS Record



✓ Data Exchange ⇨ Valid Modification Checks

* 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)

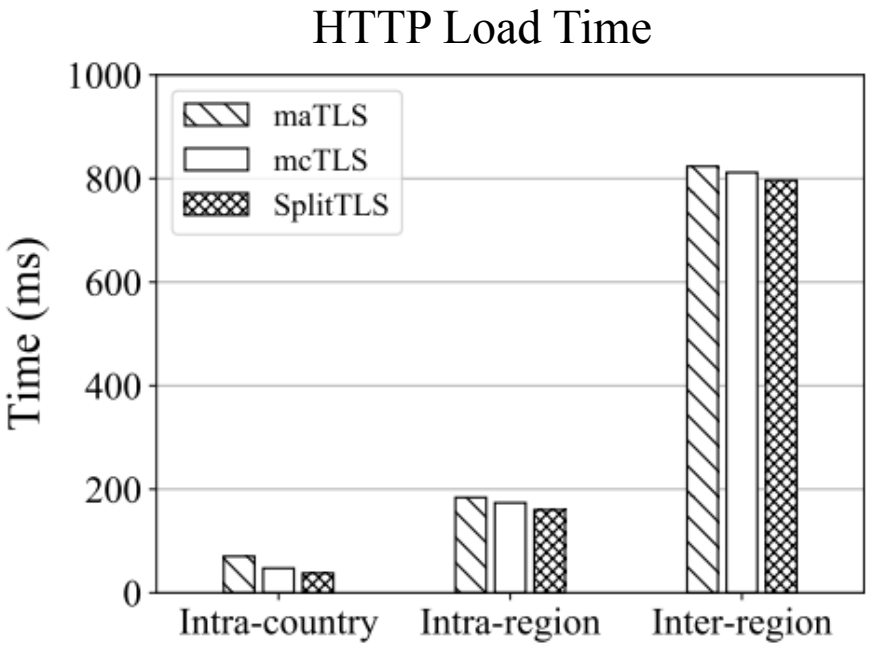
Server-side Middlebox

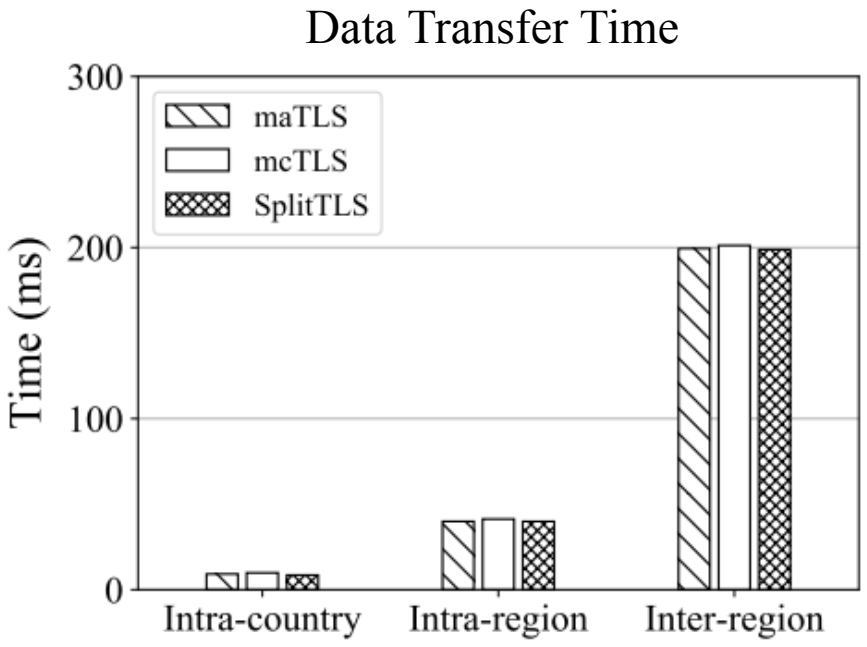
Server

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

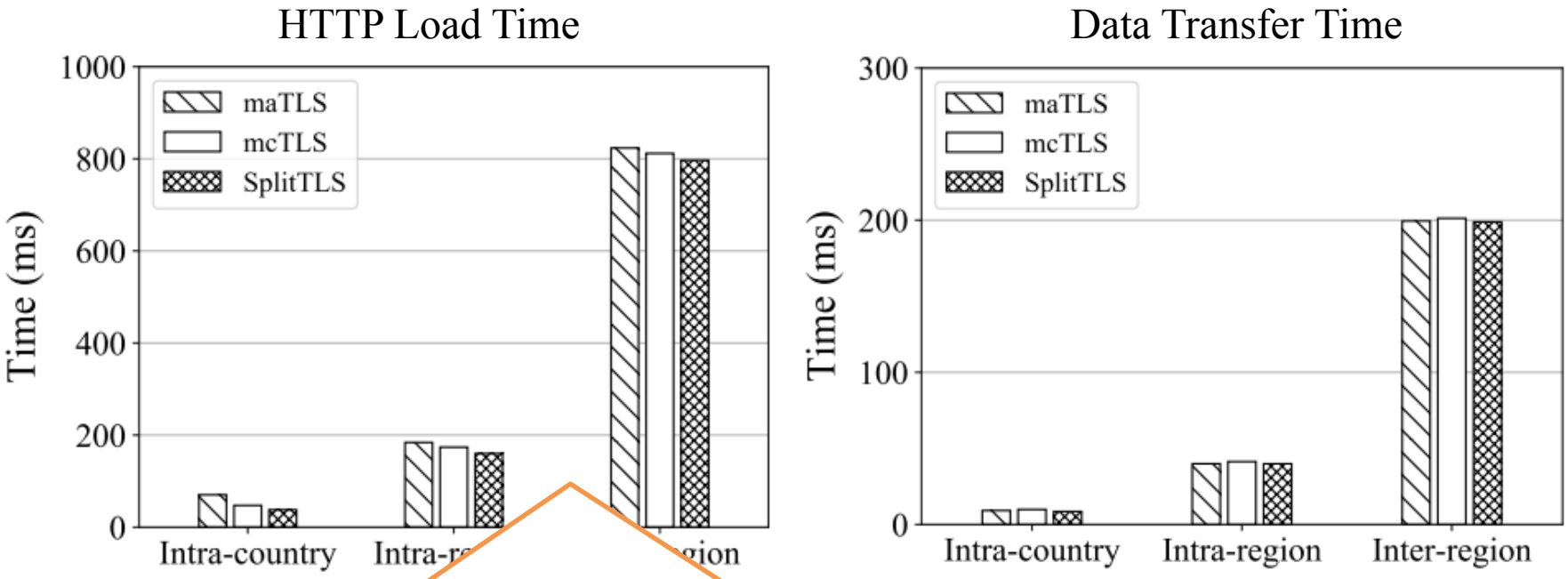




Testbed	$C\text{-}MB_C$	$MB_C\text{-}MB_S$	$MB_S - S$
Intra-country	1.136ms	4.944ms	0.551ms
Intra-region	1.136ms	35.896ms	0.537ms
Inter-region	1.136ms	192.818ms	0.610ms

- 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)

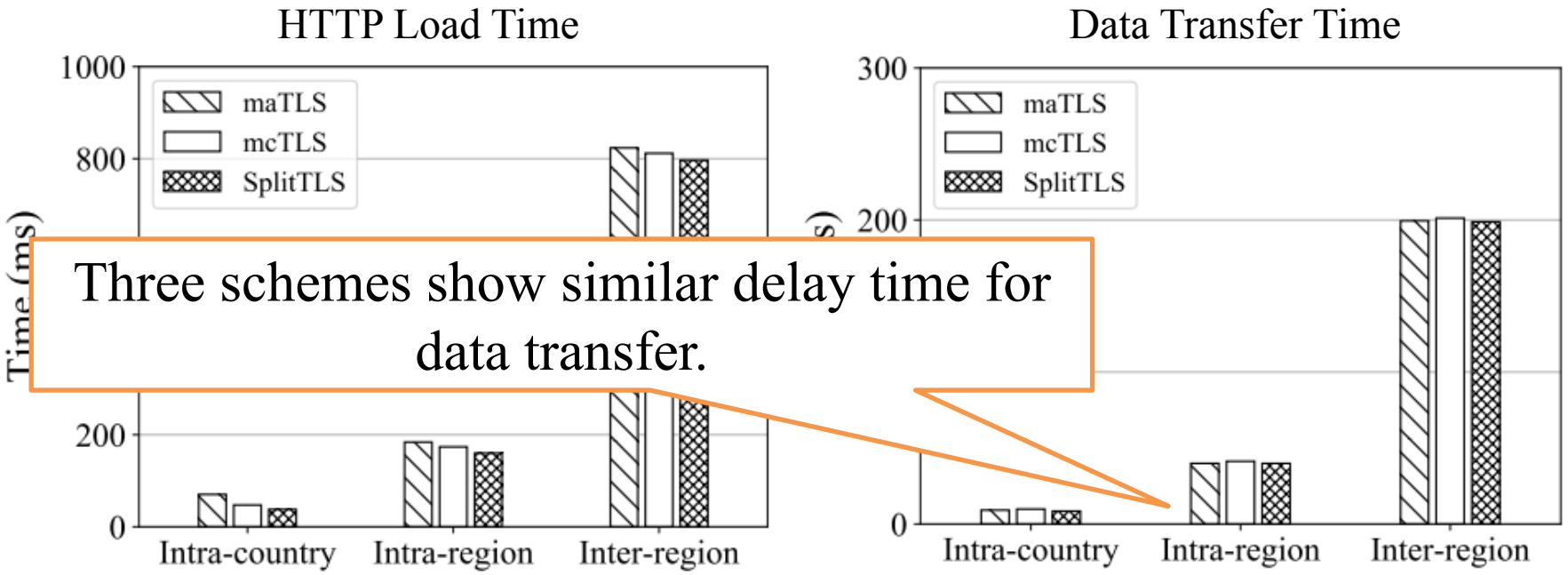
Evaluation – HTTP Load Time



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)

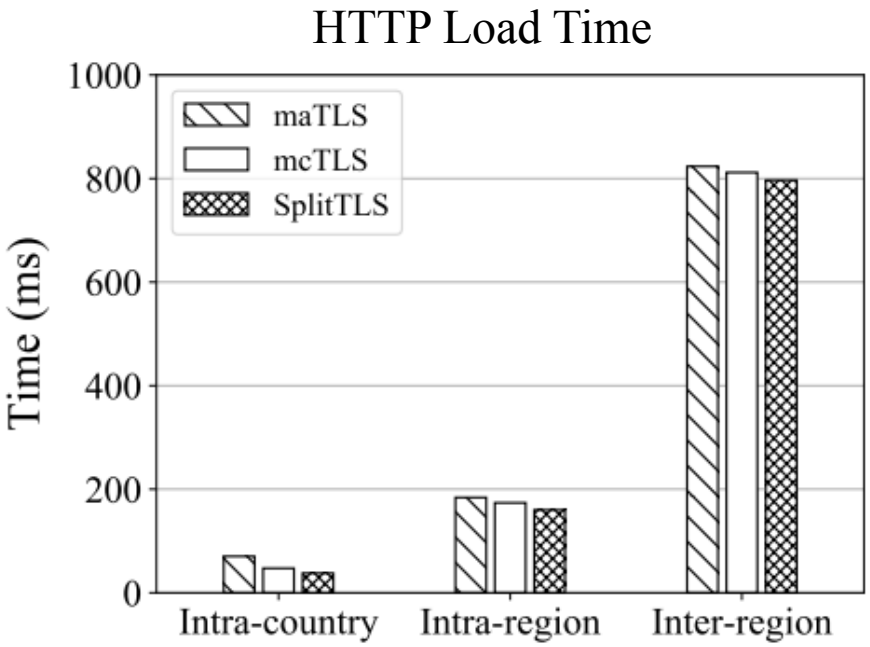
Evaluation – HTTP Load Time

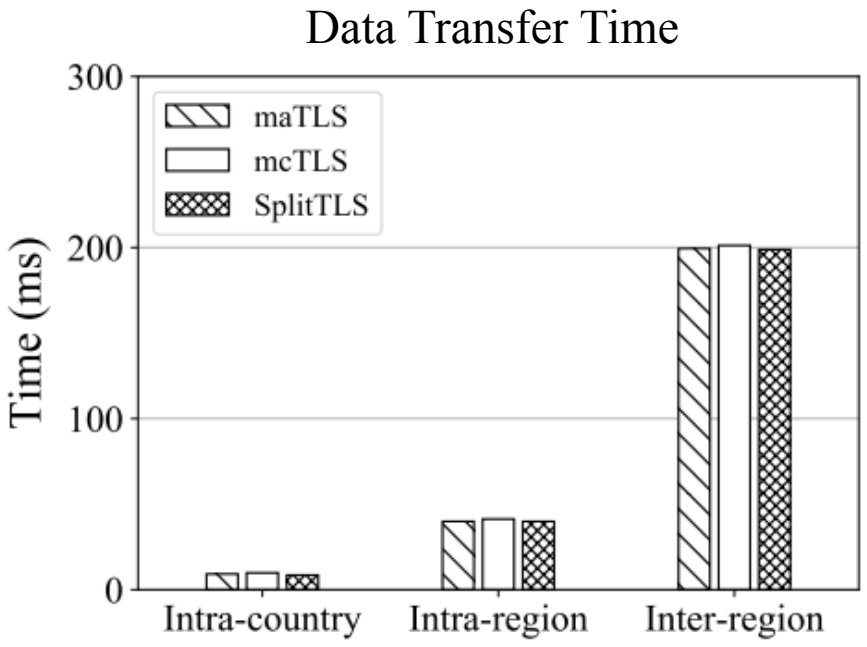


Testbed	$C\text{-}MB_C$	$MB_C\text{-}MB_S$	$MB_S - S$
Intra-country	1.136ms	4.944ms	0.551ms
Intra-region	1.136ms	35.896ms	0.537ms
Inter-region	1.136ms	192.818ms	0.610ms

- 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



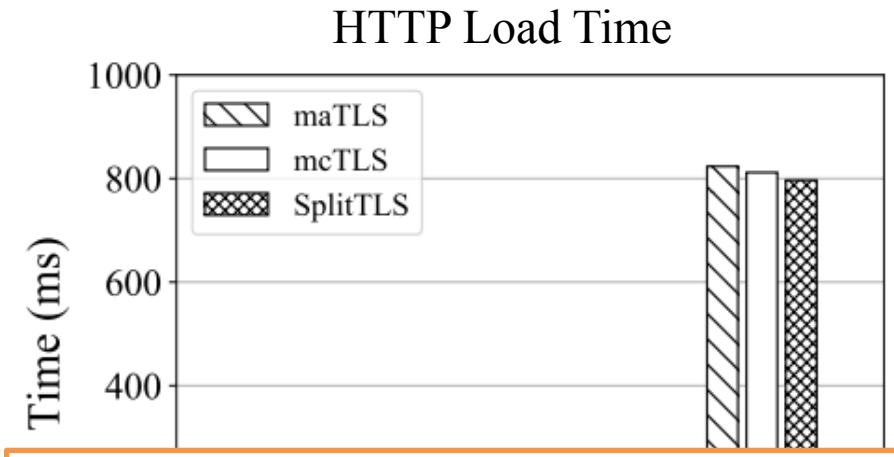


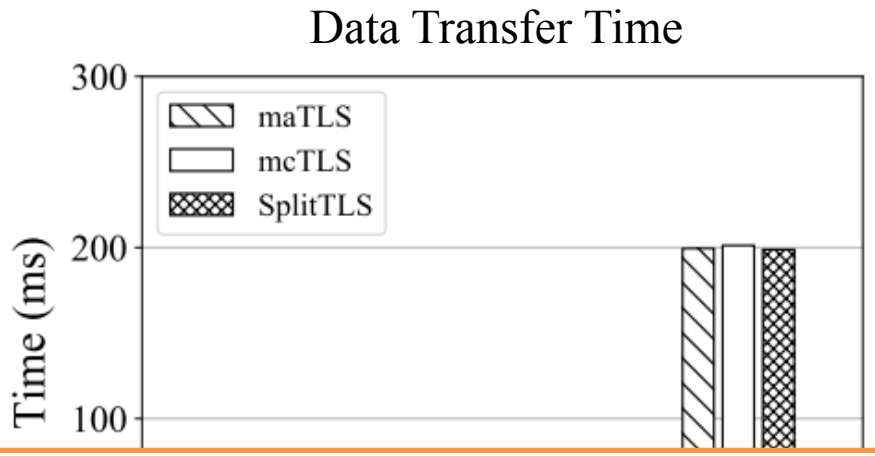
We conclude that the maTLS overhead is mainly due to the setup of an maTLS session

Inter-region	1.136ms	192.818ms	0.610ms

- 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





Once the session is established,
maTLS provides similar performance to the others
while preserving all security merits that we have discussed

Intra-country	1.136ms	4.944ms	0.551ms
Intra-region	1.136ms	35.896ms	0.537ms
Inter-region	1.136ms	192.818ms	0.610ms

- 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

Backup Slides

Why Middleboxes?

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

⋮

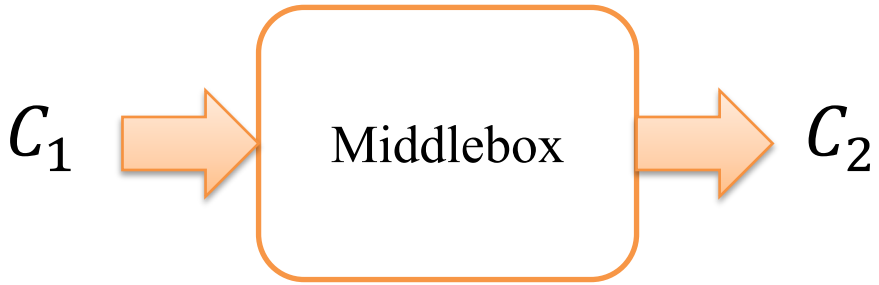
* I get the use cases from a draft of the RFC document titled “TLS 1.3 Impact on Network-Based Security”

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



$$C_1 = P_1 \oplus F(Key, IV)$$

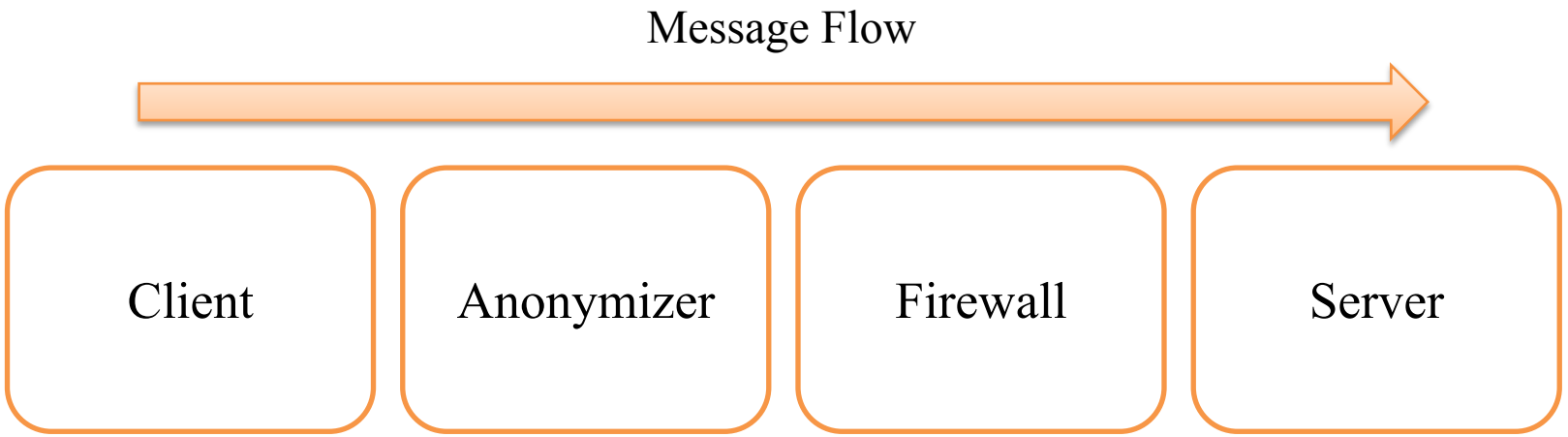
$$C_2 = P_2 \oplus F(Key, IV)$$

$$C_1 \oplus C_2 = P_1 \oplus P_2$$

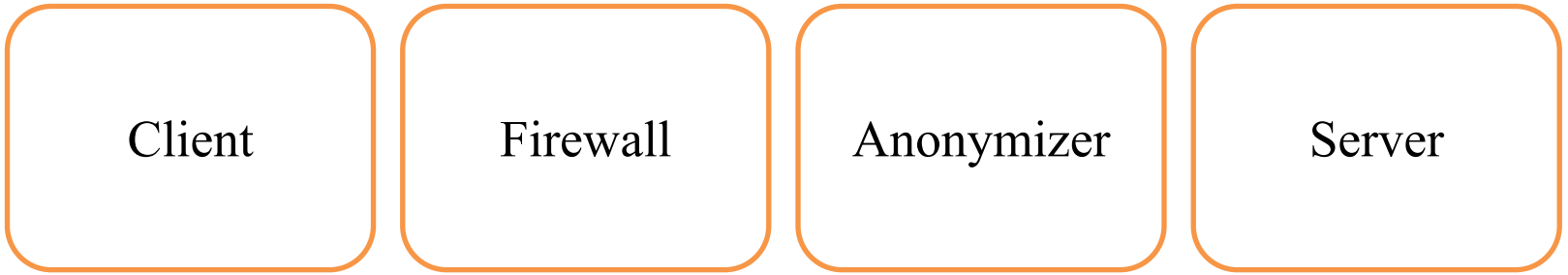
The same keystream with the different message

It is desirable to use different segment keys across the session

Why Path Integrity?



The data is anonymized and then the firewall read it

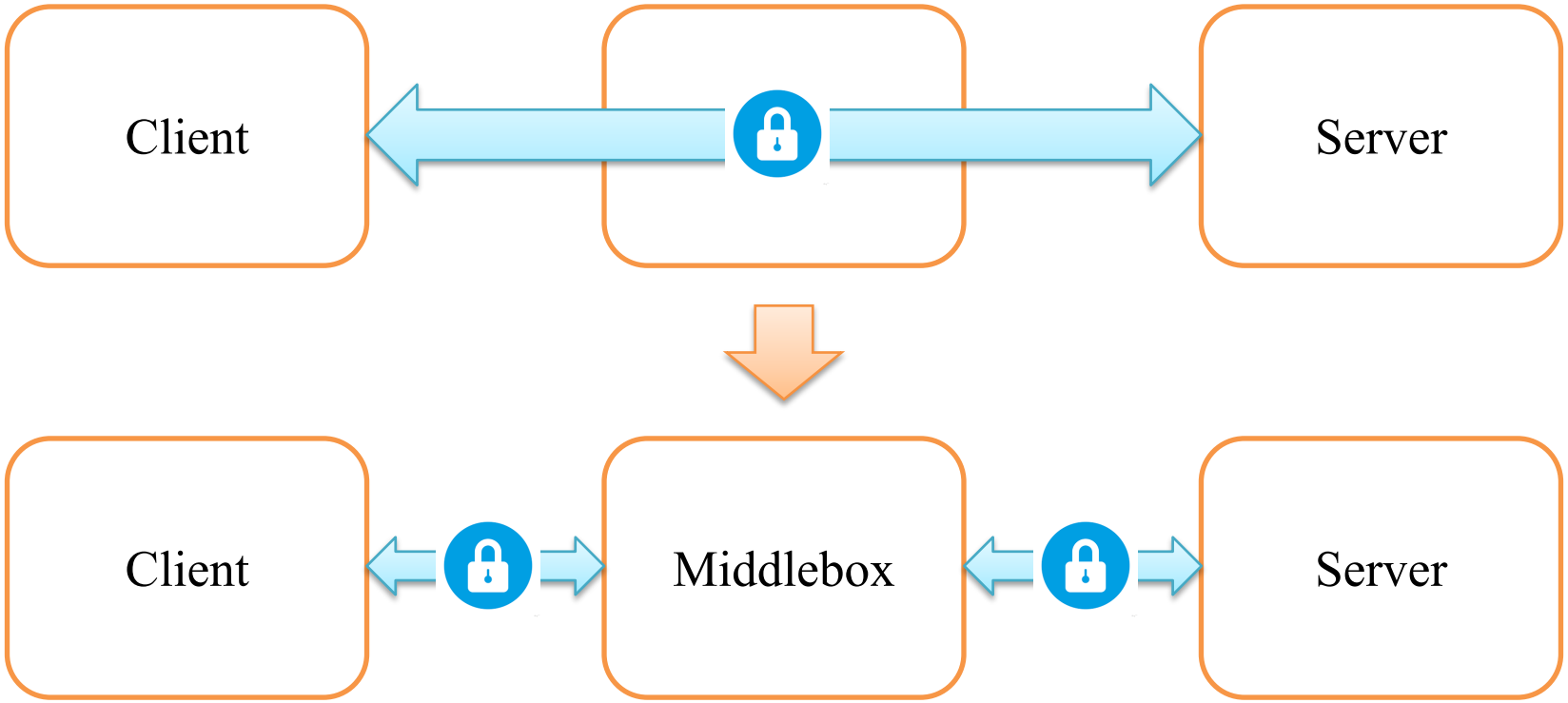


The firewall read the data and then it is anonymized

Session Establishment Approach (1)

Top-down approach

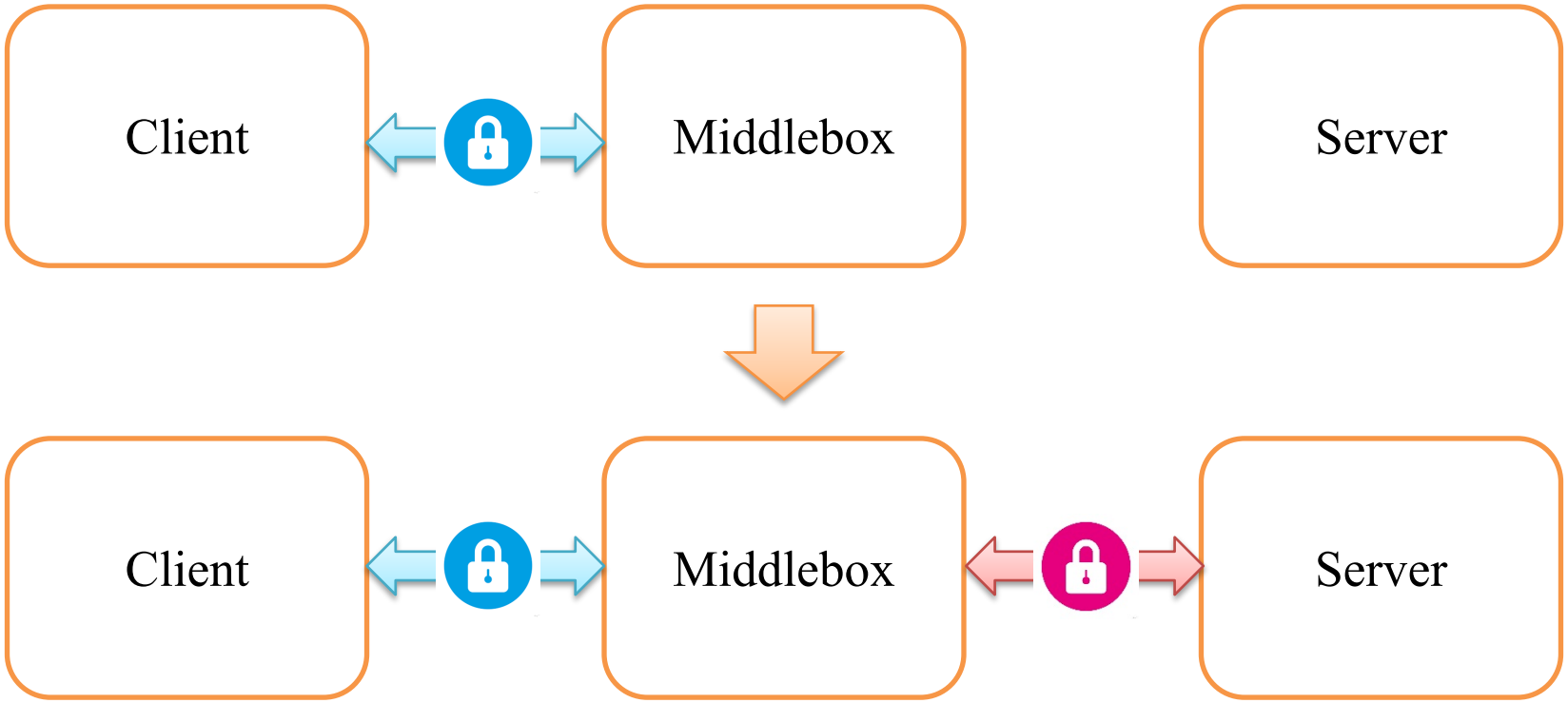
Server determines a TLS version, a ciphersuite, and extensions



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 achieves Individual Secrecy

The same keystream is used across the session, which might undermine the confidentiality of the session

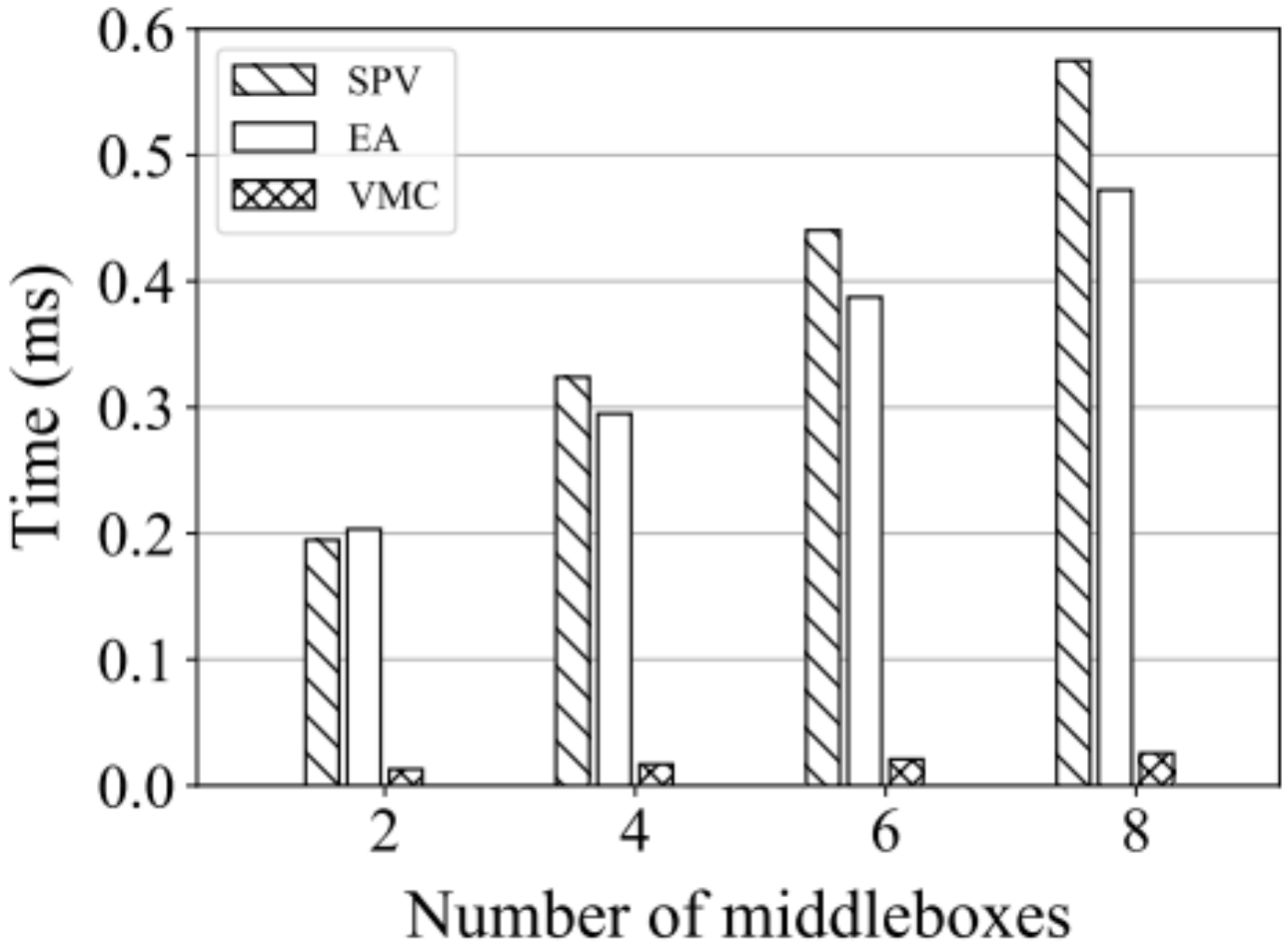
maTLS establishes different segment keys in different segments

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

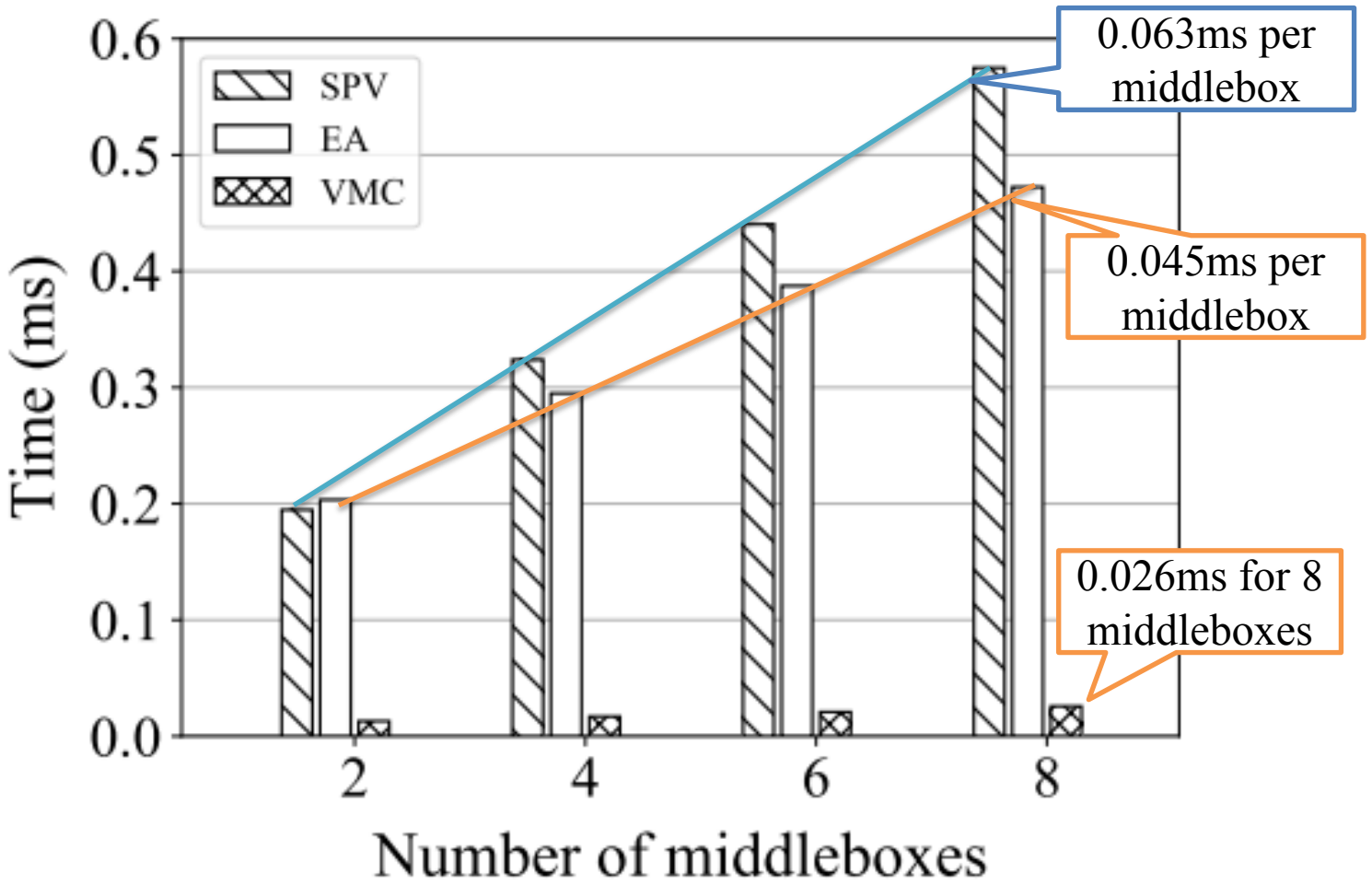
maTLS allows a partial maTLS session

Evaluation – Scalability of Three Audit Mechanisms



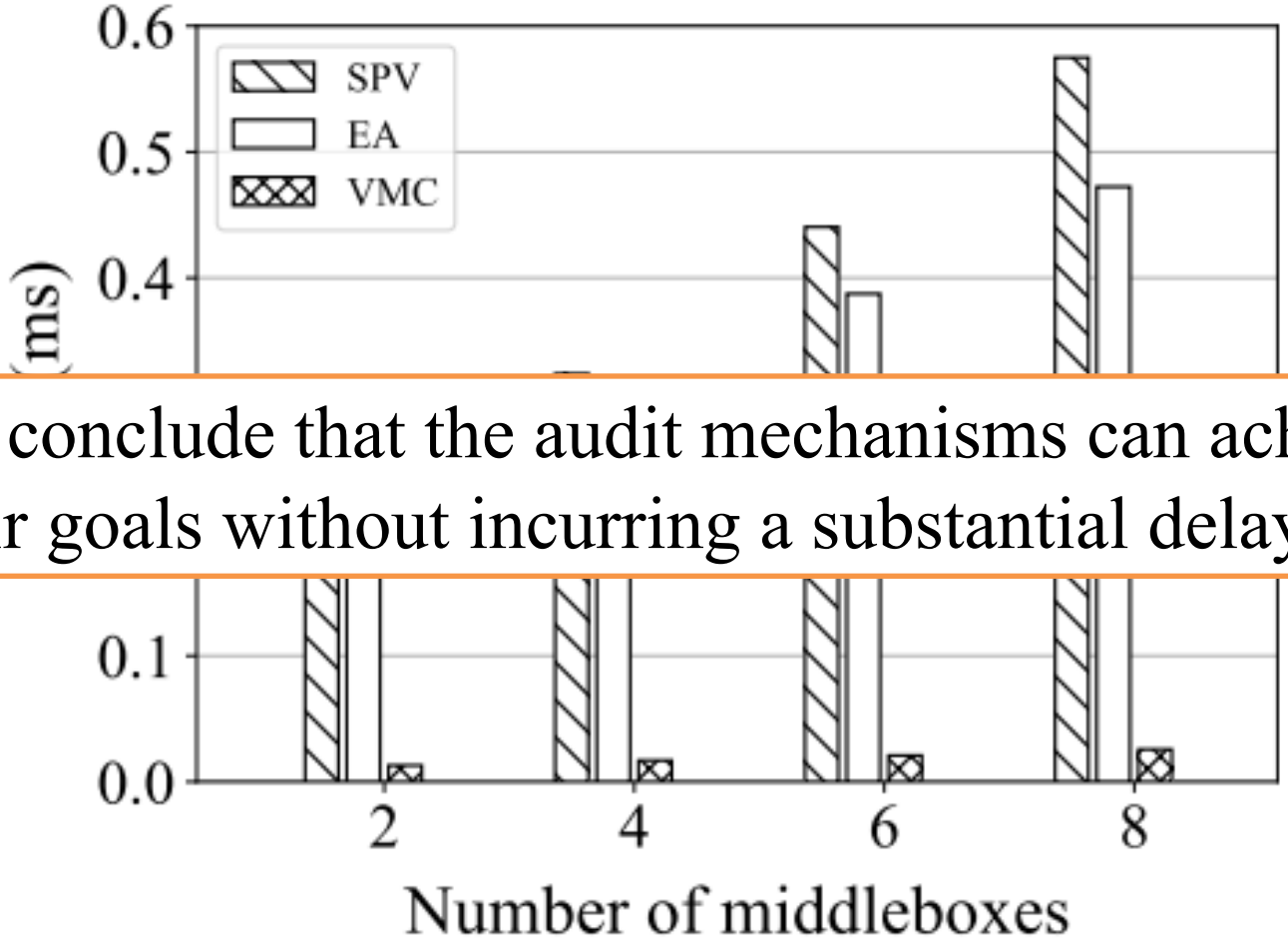
SPV: Security Parameter Verification / EA: Explicit Authentication / VMC: Valid Modification Checks

Evaluation – Scalability of Three Audit Mechanisms



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

Evaluation – Scalability of Three Audit Mechanisms



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

$ak_{s,c}$: Server's accountability key
$ak_{m,c}$: MB's accountability key (with client)

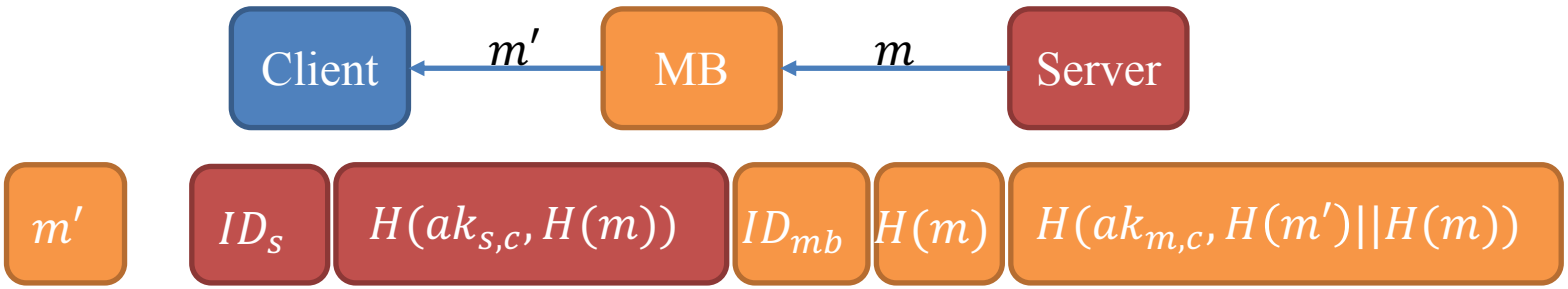
$H(k, m)$: The keyed hash function with k, applying to m
$H(m)$: The hash function, applying to m

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

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

ID_{mb} modifies m into m'

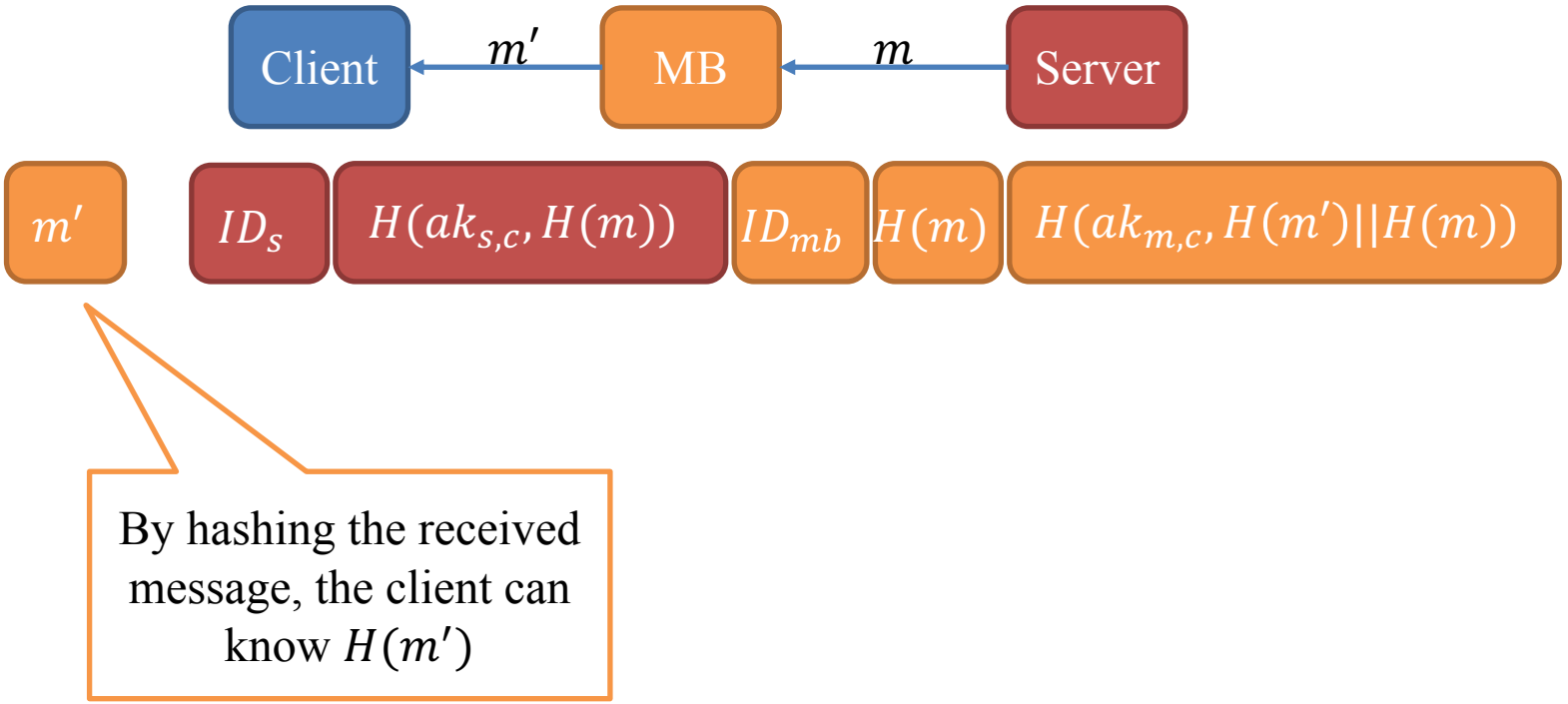
Modification Log Verification



Client knows

- $ak_{s,c}$: The accountability key with the server
- $ak_{m,c}$: The accountability key with the MB

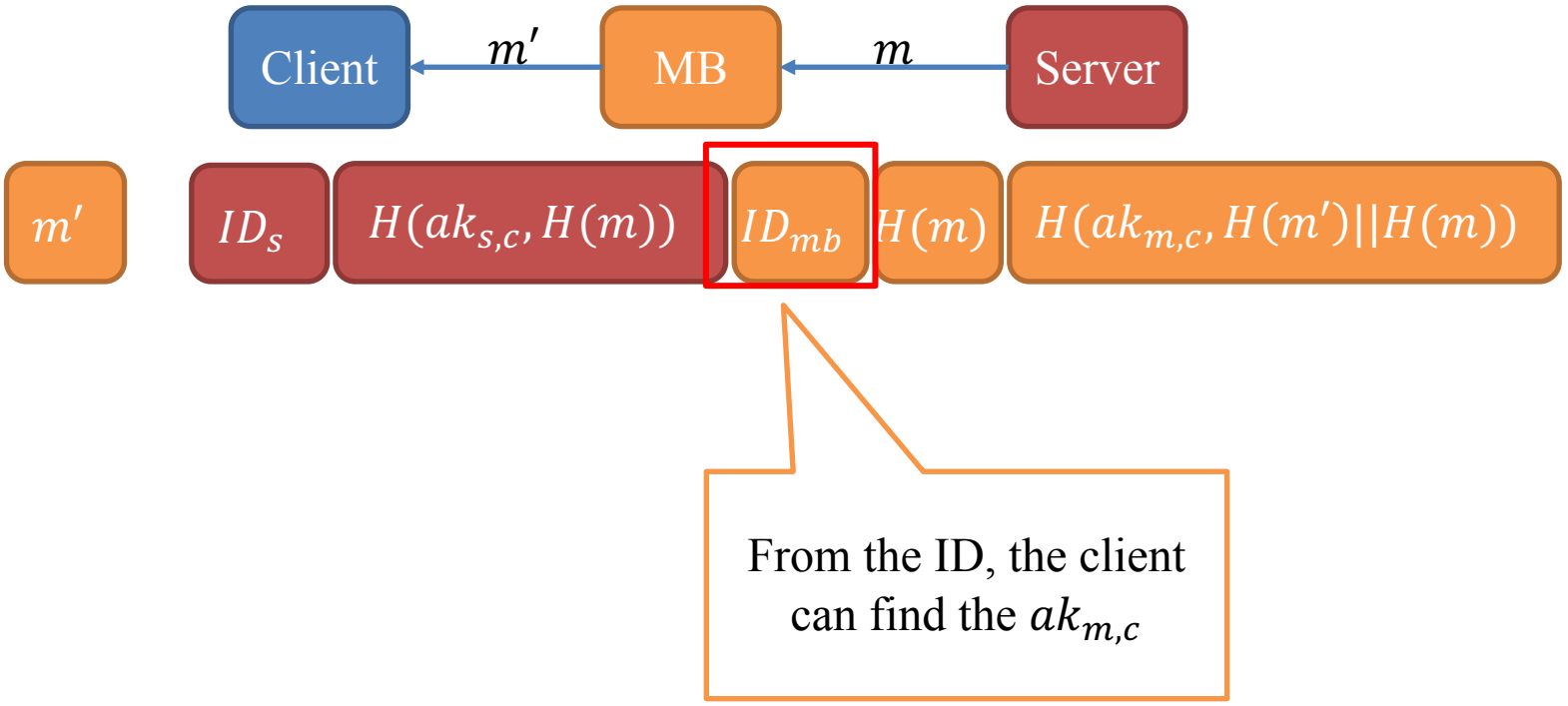
Modification Log Verification



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

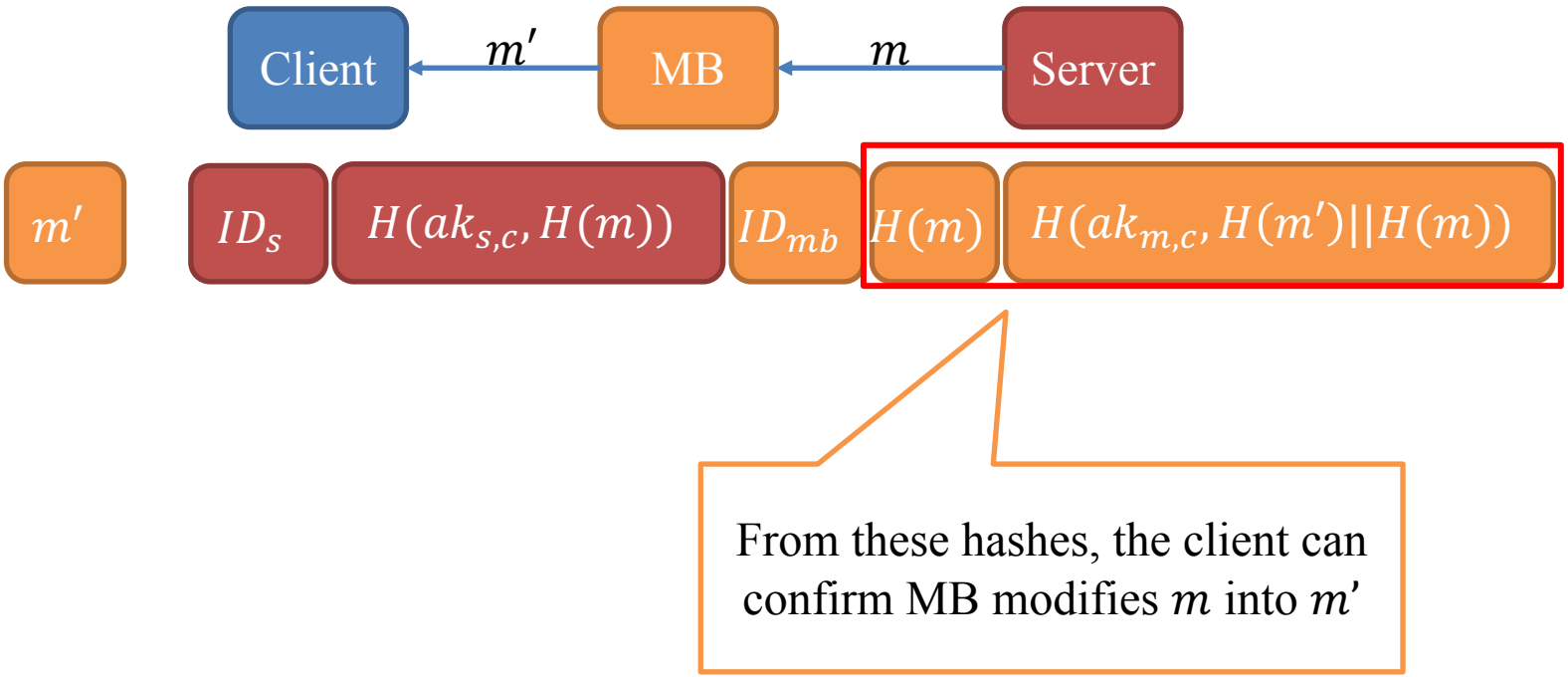
Modification Log Verification



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

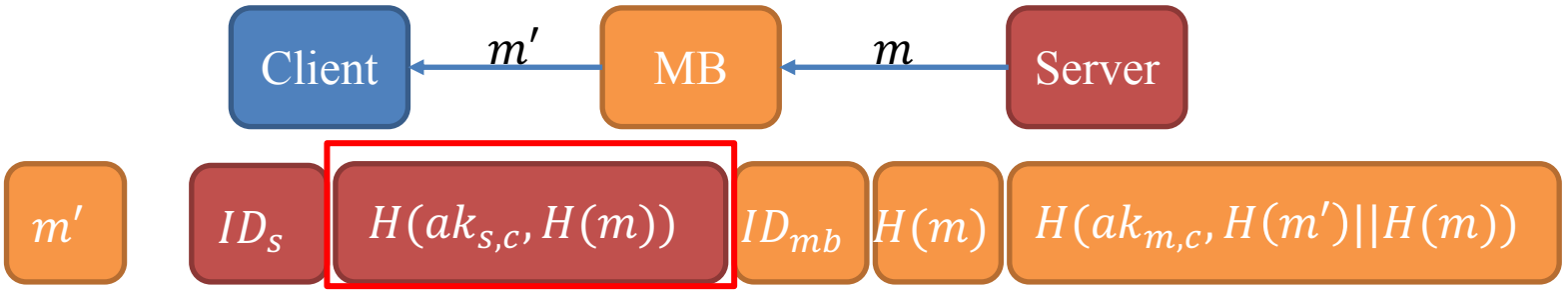
Modification Log Verification



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

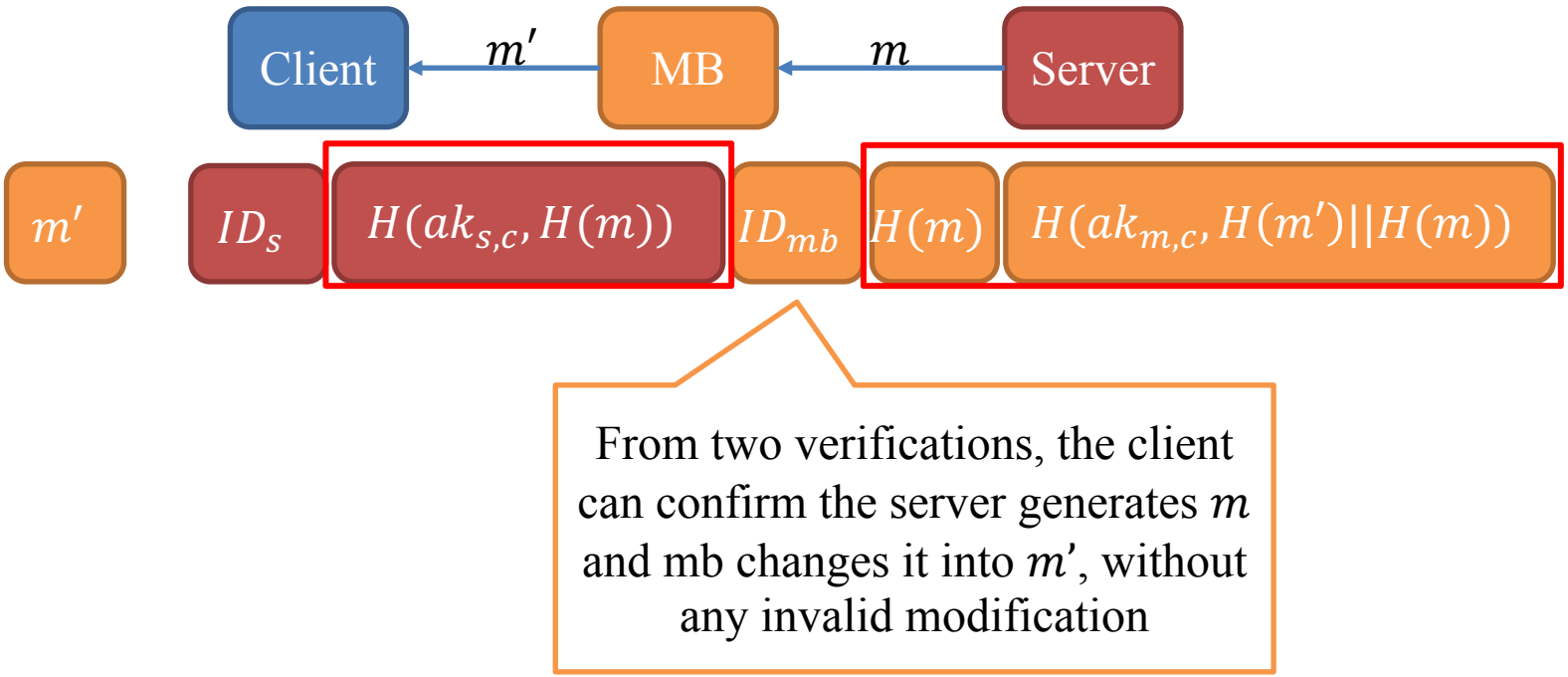


From this hash, the client can confirm the server generates m,
even though the client cannot confirm m itself

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



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