# A Trustworthy Middlebox-aware Networking Architecture

Hyunwoo Lee$_{\text{Student}}$[1], Zachary Smith$_{\text{Student}}$[2], Selin Chun$_{\text{Student}}$[1], and Ted “Taekyoung” Kwon[1]

[1]Seoul National University, [2]University of Luxembourg
hwlee2014@mmlab.snu.ac.kr, zach.smith@uni.lu, slchun@mmlab.snu.ac.kr, tkkwon@snu.ac.kr

## 1 Motivation

Using TLS middleboxes (a.k.a. TLS proxies) has been problematic since splitting TLS connections in a “middlebox-in-the-middle” fashion is risky [2, 3]. Currently, a TLS middlebox makes a TLS connection split (so, called *split TLS*) to perform its functionality in-between two endpoints. In particular, it intercepts a client’s **ClientHello** addressed to the intended server and plays the role of the server by performing a TLS handshake with the client. For this purpose, a client-side middlebox (e.g. an anti-virus software) uses a forged certificate (of the intended server) signed by the custom root certificate pre-installed on the client. Similarly, a server-side middlebox (e.g. a website application firewall) uses a certificate taken from the server for impersonation. In any case, a middlebox sends a **ClientHello** to the server, which initiates another TLS handshake between the middlebox and the intended server. The number of split TLS sessions can be more than two since there can be multiple middleboxes between the client and the server.

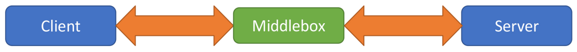


Figure 1: A middlebox, such as an anti-virus software, splits TLS into two separate TLS sessions, impersonating the server in the middle of the client and the server.

Although the *split TLS* approach (as in Figure 1) helps a middlebox intervene and process data on-the-fly, it breaks the security requirements of TLS: entity authentication, data secrecy, and data authentication. For instance, from a client’s standpoint, data is encrypted only between the client and the next middlebox; she has no idea of whether her data will have confidentiality and integrity after the next middlebox. Also, she cannot be assured of the authenticity of the intended server.

Thus, the client is forced to trust the “secure” indicator on her browser, even if the security is broken after the next middlebox. We seek to address this problem by making her explicitly aware of TLS interception [9]. After she becomes aware of the middleboxes, there should be a mechanism to manage the trustworthiness of the middleboxes.

There are several studies related to this issue [1, 4, 7, 8, 10]. All the proposals have been discussing either a certificate for a middlebox or a secure participation of a middlebox in TLS. Less attention, however, has been paid to how to address both issues together.

## 2 Middlebox-aware Architecture

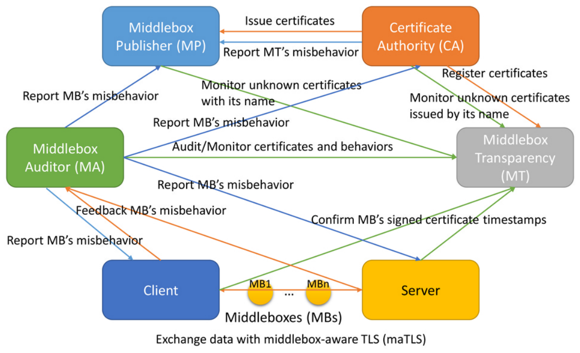

Middlebox Publisher (MP)
Issue certificates
Report MT’s misbehavior
Certificate Authority (CA)
Register certificates
Report MB’s misbehavior
Monitor unknown certificates with its name
Monitor unknown certificates issued by its name
Report MB’s misbehavior
Middlebox Auditor (MA)
Audit/Monitor certificates and behaviors
Middlebox Transparency (MT)
Report MB’s misbehavior
Feedback MB’s misbehavior
Confirm MB’s signed certificate timestamps
Report MB’s misbehavior
Client
MB1
...
MBn
Server
Middleboxes (MBs)
Exchange data with middlebox-aware TLS (maTLS)

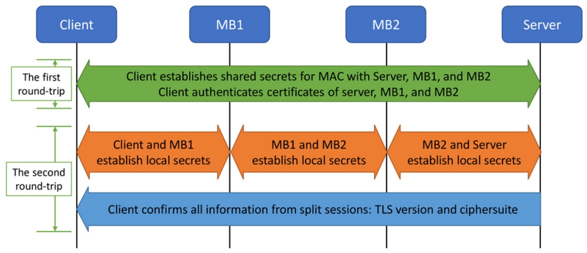
Figure 2: Our architecture seeks to ensure end-to-end secure connections through explicit authentication and modification check of each middlebox.

To this end, we propose a trustworthy middlebox-aware networking architecture as in Figure 2. The key concept of the architecture is trustworthiness management of middleboxes, which is substantiated by extending TLS in a middlebox-aware fashion (dubbed maTLS).

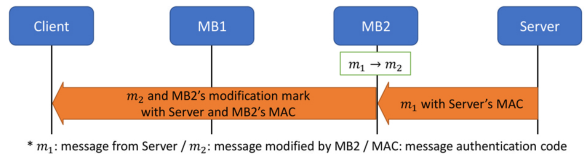
We define the trustworthiness of a middlebox as *public auditability*. Thus, trustworthiness management is a way to make a middlebox auditable in terms of its certificate and its read/write permission. This is feasible by introducing two mechanisms: (i) explicit authentication of a middlebox by its certificate, and (ii) permission check on a packet modification by a middlebox.

**Explicit authentication:** Every middlebox has its certificate that lists its publisher, its role, its read/write permission, and other fields. A certificate authority issues a middlebox certificate to a middlebox publisher and logs the certificate in a ‘middlebox transparency,’ which is audited by a middlebox auditor. (Note that the middlebox transparency is similar to the certificate transparency [5].) This certificate is used to authenticate the middlebox in maTLS. By verifying all the relevant certificates with signed certificate timestamps [5] during a maTLS handshake, a client can identify all intermediate middleboxes participating in the maTLS session.

**Modification check:** In the maTLS protocol, whenever a participating middlebox modifies a packet, it records this operation by adding a message authentication code (using a key established during the maTLS handshake). From the record of packet modifications, the client figures out all intermediate writers and detects invalid modifications by cross-checking the permission information in their certificates. The client may report invalid modifications to the auditor via an out-of-band channel. The auditor checks the corresponding certificate for the invalid modification, whose result is notified to the middlebox publisher and the certificate authority.


Client
MB1
MB2
Server
The first round-trip
Client establishes shared secrets for MAC with Server, MB1, and MB2
Client authenticates certificates of server, MB1, and MB2
The second round-trip
Client and MB1 establish local secrets
MB1 and MB2 establish local secrets
MB2 and Server establish local secrets
Client confirms all information from split sessions: TLS version and ciphersuite

(a) **maTLS handshake**. The client authenticates the server/middlebox(es) by their certificates, confirms TLS versions/ciphersuites of each session, and establishes the keys for message authentication codes with the server and the middlebox(es).


Client
MB1
MB2
Server
$m_1 \rightarrow m_2$
$m_2$ and MB2's modification mark with Server and MB2's MAC
$m_1$ with Server's MAC
* $m_1$: message from Server / $m_2$: message modified by MB2 / MAC: message authentication code

(b) **maTLS record**. The client verifies the sender and the intermediate writer by their message authentication codes.

Figure 3: Middlebox-aware TLS protocol overview

We design maTLS based on the modified security requirements of TLS. Entity authentication becomes server/middlebox authentication; data secrecy is extended to path secrecy; and data authentication is divided into data source authentication and modification accountability. The high-level overview of maTLS is shown in Figure 3.

**Server/Middlebox authentication:** A client authenticates the server and the middlebox(es) by their certificates. This prevents any untrusted middlebox from participating in a maTLS session.

**Path secrecy:** A client is aware of all the split sessions and can abort the connection if any split session with a low version or weak ciphersuite exists. This defends the client from passive attackers in every split session.

**Data source authentication:** A client confirms the message comes from the server without any invalid modifications, by checking the server's message authentication code. This prevents an active attacker from tampering the message.

**Modification accountability:** A client confirms any modifications are written by authorized middleboxes, by checking the writers' message authentication codes. This defends the client against an active attacker's modification.

## 3 Future Work

To demonstrate the feasibility of our architecture, we plan to perform two main tasks: formal verification and evaluation. We will prove the security of maTLS with the state-of-the-art formal verification tool Tamarin [6]. This will improve our protocol by identifying any security flaws based on the security requirements. Further, we will implement our architecture in C with the OpenSSL library and evaluate the performance overhead in terms of delay and memory usage of participating parties.

## 4 Conclusions

In this proposal, we design a trustworthy middlebox-aware networking architecture, whose main concept is the trustworthiness management of middleboxes in an open fashion. By verifying their certificates and read/write permission, the proposed architecture can enhance the security of end-to-end TLS communications in presence of middleboxes.

## References

[1] D. McGrew, D. Wing, Y. N. P. G. Tls proxy server extension. Internet-Draft draft-mcgrew-tls-proxy-server-01, IETF.

[2] de Carnavalet, X. d. C., and Mannan, M. Killed by proxy: Analyzing client-end tls interception software. In *Network and Distributed System Security Symposium* (2016).

[3] Durumeric, Z., Ma, Z., Springall, D., Barnes, R., Sullivan, N., Bursztein, E., Bailey, M., Halderman, J. A., and Paxson, V. The security impact of https interception. In *Network and Distributed Systems Symposium* (2017).

[4] Han, J., Kim, S., Ha, J., and Han, D. Sgx-box: Enabling visibility on encrypted traffic using a secure middlebox module. In *Proceedings of the First Asia-Pacific Workshop on Networking* (2017), ACM, pp. 99–105.

[5] Laurie, B., Langley, A., and Kasper, E. Certificate transparency. Tech. rep., 2013. RFC 6962, IETF.

[6] Meier, S., Schmidt, B., Cremers, C., and Basin, D. The tamarin prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification* (2013), Springer, pp. 696–701.

[7] Naylor, D., Li, R., Gkantsidis, C., Karagiannis, T., and Steenkiste, P. And then there were more: Secure communication for more than two parties. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies* (2017), ACM, pp. 88–100.

[8] Naylor, D., Schomp, K., Varvello, M., Leontiadis, I., Blackburn, J., López, D. R., Papagiannaki, K., Rodriguez Rodriguez, P., and Steenkiste, P. Multi-context tls (mctls): Enabling secure in-network functionality in tls. In *ACM SIGCOMM Computer Communication Review* (2015), vol. 45, ACM, pp. 199–212.

[9] Ruoti, S., O'Neill, M., Zappala, D., and Seamons, K. E. User attitudes toward the inspection of encrypted traffic. In *SOUPS* (2016), pp. 131–146.

[10] S. Loreto, J. Mattsson, R. S. H. S. G. G. D. D. Explicit trusted proxy in http/2.0. Internet-Draft draft-loreto-httpbis-trusted-proxy20-01, IETF.