VISVESWARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-590018

TECHNICAL SEMINAR REPORT ON

"A MIDDLEBOX-ENHANCED TLS-BASED SECURITY FOR IOT SYSTEMS"

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE AND ENGINEERING
Submitted By

A Nitya Dyuthi
1BY18CS001

Under The Guidance Of
Mrs Durga Bhavani A
Assistant Professor
Department of CSE

BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Avalahalli , Yelahanka , Bengaluru – 560064.

2021-2022

VISVESWARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-590018

BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Avalahalli , Yelahanka , Bengaluru – 560064.

CERTIFICATE

This is to certify that the Seminar work entitled “A Middlebox Enhanced TLS-Based Security for IoT systems” has been carried out by Mr/Ms. A Nitya Dyuthi, 1BY18CS001, a bonafide student of BMS Institute of Technology and Management in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2021-2022. It is certified that all corrections/suggestions indicated for assessment have been incorporated in the report deposited in department library. The Seminar report has been approved as it satisfies the academic requirements in respect of Seminar work prescribed for the said degree.

_____	_____
Signature of the Guide	**Signature of the HOD**
Mrs Durga Bhavani A	**Dr. Bhuvaneshwari C M.**
Assistant Professor	**Professor& Head of department**
Department of CSE	**Department of CSE**
BMSIT & M	**BMSIT & M**

DECLARATION

I, A Nitya Dyuthi (1BY18CS001), student of VIII Semester BE, in Computer Science and Engineering, BMS Institute of Technology and Management hereby declare that the Seminar entitled "A Middlebox Enhanced TLS-Based Security for IoT systems**"** has been carried out by me and submitted in partial fulfillment of the requirements for the *VIII Semester degree of* ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2021-22.

Date :

Place :

A NITYA DYUTHI

1BY18CS001

ACKNOWLEDGEMENT

We are happy to present this Seminar report after completing it successfully. This seminar would not have been possible without the guidance, assistance and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and everyone who has helped us make this a success.

We heartily thank our **Principal, Dr MOHAN BABU G N, BMS Institute of Technology & Management,** for his constant encouragement and inspiration in taking up this project.

We heartily thank our **Head of the Department, Dr Bhuvaneshwari C.M, Department of Computer Science and Engineering, BMS Institute of Technology &Management,** for her constant encouragement and inspiration in taking up this project.

We gratefully thank our Seminar Guide, **Mrs Durga Bhavani A, Assistant Professor, Department of Computer Science and Engineering,** for his/her guidance, support, and advice.

Special thanks to all the staff members of the Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given to us in completing this precious work successfully.

A Nitya Dyuthi
1BY18CS001

ABSTRACT

IoT systems are widely used in our world and have turned into an indispensable technology. IoT aims to provide advanced and intelligent services but at the same time ensure security.

In-network middleboxes are vital for Internet-of-things system security, but the widely adopted Transport Layer Security (TLS) protocol blinds application-level middleboxes due to the encryption of traffic data. A “middlebox” is a computer networking device that transforms, inspects, filters, and manipulates traffic for purposes other than packet forwarding. To resolve this problem, many solutions have been proposed to date. SplitTLS is widely adopted in the industry by proxy manufacturers. It requires a TLS client to install custom root certificates and incurs additional security flaws, e.g., disabling server authentication and using weak cipher suites. Another approach is to customise the TLS protocol where middleboxes are enabled via either performing handshake directly with TLS endpoints, or receiving session key materials in an out-of-band manner. Overall, current solutions would either jeopardise the original TLS handshake procedure or incur additional overheads on the endpoints.

This research digs deep into a new TLS-based security method, the ME-TLS, or middlebox-enhanced TLS, which enables endpoints to introduce authenticated middleboxes into a TLS session.

CONTENTS

List Of Figures

Chapter 1
Introduction

The Internet of Things (IoT) is a massive group of devices connected over wired or wireless networks. The data required and processed by IoT systems is extremely sensitive and requires air-tight security as it involves processing data like controls to a home, health-related data from wearables, biometric data, etc. It is a challenge to find lightweight security models and techniques as the IoT devices have low resources and low computing power. The data involved in such systems is usually highly sensitive. Since IoT transactions are done over the internet, sensitive data is vulnerable to security threats intentionally or unintentionally. IoT devices became a target of cyber-attacks because of deficiencies in the safety mechanism. There are many threats facing the security of IoT systems.

In addition to their vulnerabilities, these devices do not have software routines for automatic updating and use unencrypted or poorly encrypted communication channels. For this reason, more and more smart devices (telephones, network devices, surveillance cameras or refrigerators) are involved in large scale cyber-attacks. The main cyber threats to IoT are DDoS attacks, Botnets and malware attacks, data breaches, etc. The threat of attacks goes from information manipulation to action control, which means these attacks transcend from the digital to the physical world.

Transport Layer Security or TLS, is a cryptographic protocol designed to provide communications security over a computer network. It makes use of “certificates”, which consist of a key pair made of a public key and a private key. These keys are important because they interact behind the scenes during website transactions. Every time a website is visited, the client-server and web browser communicates to ensure there is a secure TLS/SSL encrypted connection.

1.1 MOTIVATION

IoT systems are but a set of independent devices if they can not communicate with each other. The basis of IoT systems is communication and since the devices involved are low in computing power, they can not have heavy security software installed on them. IoT devices work with highly sensitive data. If not home controls, it would be anything ranging from human body data to sensitive medical devices. The IoT systems are extremely susceptible to attacks and hence need a strongly secured system when they communicate with each other.

One could note that in all current solutions, middleboxes are introduced into a TLS session either by involving them in the handshake phase directly (e.g., SplitTLS, mcTLS, maTLS) or by transmitting session key materials to them through secondary TLS connections (e.g., mbTLS). In TLS 1.3, involving middleboxes in the handshake procedure would jeopardise the original efficient handshake structure, and thus potentially harm connection set-up time and affect the user experience. On the other hand, the out-of-band key transmission would definitely incur additional communication and computation overheads on the endpoints.
The main question that is answered through this survey is:
“How to enable passive and authenticatable middleboxes over TLS 1.3 without modifying the TLS handshake structure?”

CHAPTER 2: LITERATURE SURVEY

2.1 SECURITY IN IOT SYSTEMS

The massive boom of the Internet of Things (IoT) has led to the explosion of smart IoT devices and the emergence of various applications such as smart cities, smart grids, smart mining, connected health, and more. While the proliferation of IoT systems promises many benefits for different sectors, it also exposes a large attack surface, raising an imperative need to put security in the first place. It is impractical to heavily rely on manual operations to deal with the security of massive IoT devices and applications[3]. Hence, there is a strong need for securing IoT systems with minimum human intervention

There are many categories of threats that IoT systems are known to be vulnerable to [4]:

- **Common IoT worms**: Generally limited to devices running operating systems such as Windows, Linux, iOS, Android
- "**Script Copies**" or other threats targeting residential IoT objects: unprotected webcams, content theft, burglary of home control systems
- **Organized crime**: access to intellectual property, sabotage and espionage
- **Cyber terrorism**: traffic monitoring, railways, critical infrastructure
- **Vulnerabilities due to Bluetooth connection**: BlueBorne attack vector

[5]Cybercriminals often use infected routers to launch distributed denial-of-service (DDoS) attacks on third parties. This is a tried and tested form of online extortion — making a machine on the internet inaccessible and refusing to stop doing so until a fee is paid. This is even more relevant nowadays when many businesses rely on their e-commerce revenue to survive.

In general, existing works for automation of threat modeling for IoT systems can be divided into

(i) non-adaptive threat modeling and

(ii) adaptive threat modeling

Works done in the first category [2] mainly rely on theoretical methods like game theory and graph theory to model the potential threats and vulnerabilities regarding communication, computation, and control among IoT devices. Works in the second category [3] focus on designing frameworks which can continuously evaluate the security using various security

metrics, learn and adapt to dynamic environments in IoT systems, and identify and respond to unknown threats.

Here we focus on a different, hardware-oriented approach involving in-network middleboxes. This solution is not new, but the way it is implemented is out of the box.

2.2 TRANSPORT LAYER SECURITY (TLS) AND ITS IMPLEMENTATION IN IOT

The TLS is a widely used protocol in IoT systems, which are vulnerable in one or more of the following ways: use of old/insecure protocol versions and/or cypher suites, lack of certificate validation, and poor maintenance of root stores [6].

According to a report from Fortinet Networks [8], HTTPS [9] traffic accounts for 72.2% of all Internet traffic in the third season of 2018 and the proportion keeps growing. In 2018, IETF released TLS 1.3 [7], which requires one less round trip to establish a connection than TLS 1.2.

Moreover, under session resumption, TLS 1.3 offers an even more efficient 0-RTT mode, where the application data can be sent by the client on the first flight. TLS 1.3 also made several security improvements including the removal of static RSA and Diffie-Hellman (DH) cipher suites, encryption of handshake messages, etc.

Many application-layer protocols for IoT also adopt TLS to establish underlying secure channels, including Message Queuing Telemetry Transport (MQTT) protocol [10], and Extensible Messaging and Presence Protocol (XMPP) [11], Advanced Message Queuing Protocol (AMQP) [12], etc.

The current TLS-based IoT security solutions either jeopardize the original TLS handshake procedure or incur additional overheads on the endpoints, both of which are undesirable and compromise the system.

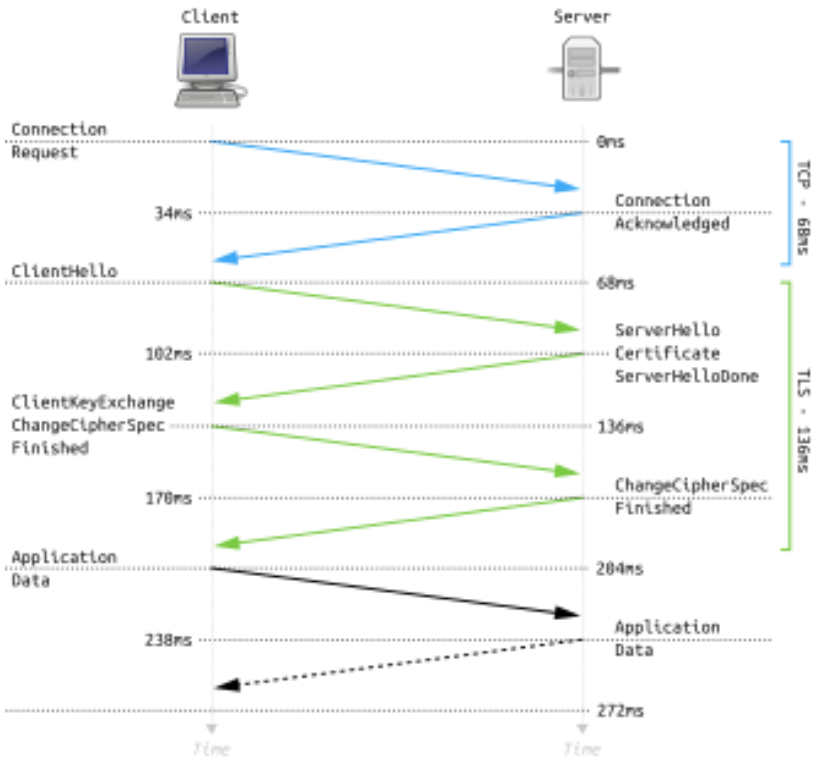

Fig 2.1: The handshake process in TLS 1.2 protocol

Unfortunately, although TLS encryption could legitimate user privacy, it naturally disables middleboxes which are ubiquitous in computer networks. Particularly, middleboxes usually require access to application data to provide functionalities, including performance optimization devices such as compression proxies and load balancers, and security devices such as firewalls and intrusion detection systems (IDS). It has been well known that middleboxes play a significant role in IoT systems where devices are usually resource-constrained, e.g., with weak computation power and limited battery capacity. Thus, it is practically infeasible to install endpoint-based programs such as virus scanners to protect IoT devices. Instead, in-network middleboxes should be deployed for IoT system protection.

Deploying in-network middleboxes also brings additional benefits including network-wide visibility and convenient management, a single middlebox is competent for handling all IoT devices in a local area network, while keeping the middlebox up to date is much easier than managing multiple scattered IoT devices.

Techniques	**SplitTLS**	**Multi-Context TLS (mcTLS)**	**Middlebox TLS (mbTLS)**	**Middlebox-aware TLS (maTLS)**
About	Used by proxy vendors to introduce client-side middleboxes. It needs the installation of a customized root certificate on the client, the middlebox splits the original one connection into two [13]	Multi-context TLS (mcTLS)[14] endpoints to perform handshaking with each introduced middlebox. multi-context TLS enables endpoints to have fine-grained access control over the introduced middleboxes.	Middlebox TLS (mbTLS) [15] enables the endpoints to introduce middleboxes into a session on each side independently	In middlebox-aware TLS (maTLS) [16], every two adjacent neighbors establish a TLS connection called a TLS segment
Drawbacks	1. The client is no longer able to authenticate the server 2. Middlebox vendors usually implement TLS improperly. The middlebox may create a less secure connection with the server using a weaker cipher 3. Middlebox may inject content such as advertisements into TLS traffic without client consent	Can not interoperate with standard TLS endpoints [14] and thus is not immediately deployable	During handshaking, each endpoint transmits key materials to middleboxes through secondary TLS connections [15], which will incur additional overheads on the endpoints	The middlebox certificates and TLS segments' security information are sent to the maTLS client for assessment, thus incurring additional computation and communication overheads

CHAPTER 3: WORKING PRINCIPLE OF ME-TLS

3.1 INTRODUCTION

The aim of the Middlebox-Enhanced TLS, or ME-TLS system is to customise the TLS protocol where middleboxes are enabled via either performing handshake directly with TLS endpoints or receiving session key materials in an out-of-band manner, ie, independent of the primary connection.

3.2 CRYPTOGRAPHIC PRINCIPLES AND BF-IBE

Standard TLS uses Public Key Infrastructure (PKI), but, unfortunately, the PKI system has a major security drawback: if a certificate authority (CA) is compromised, attackers could use it to issue false certificates and render the TLS insecure.

To overcome this drawback, the ME-TLS utilizes a distributed Boneh-Franklin identity-based encryption (BF-IBE) scheme. The BF-IBE scheme has two benefits:

1. In an IBE system, a user’s public key is bound with their identity and thus certificates are no longer needed in the ME-TLS for entity authentication
2. BF-IBE offers a particularly nice property in that the two entities can establish a shared secret in a non-interactive manner. In ME-TLS, this property is used to derive symmetric keys for entity authentication and session key material distribution

In an identity-based cryptographic system,

1. the user's identity is a string of arbitrary length and
2. each user has a private/public key pair,
    a. the public key can be computed by anyone from the system’s public parameters and the user’s identity
    b. the private key can only be generated by a private key generator (PKG)

The PKG used is (n,t)-distributed PKG, the master key is distributed among n PKG nodes and can not be computed by less than t+1 nodes.

3.3 SYSTEM ARCHITECTURE

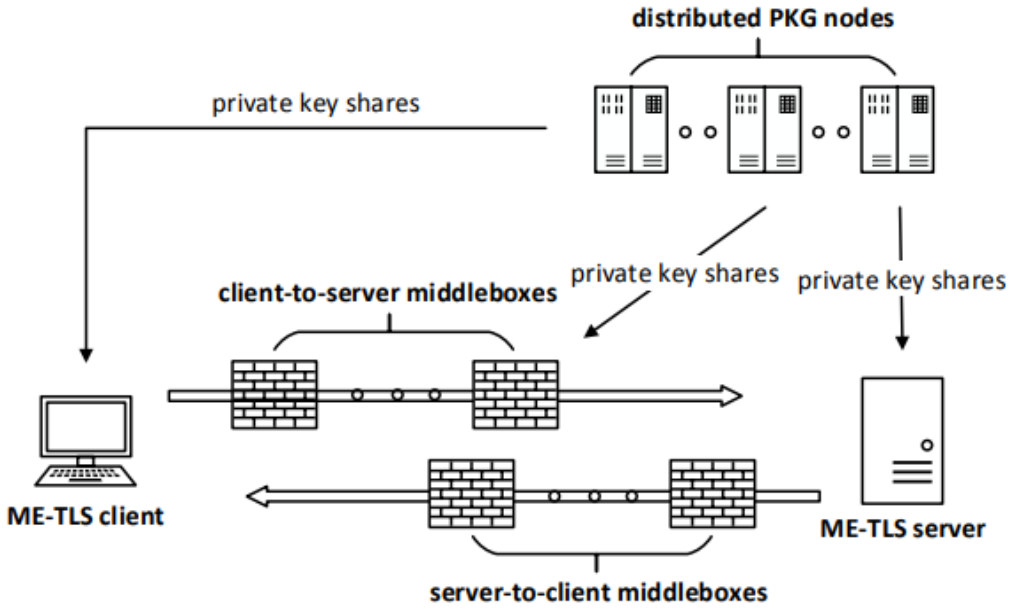

Fig. 3.1: System Architecture

There are three types of entities in the system model, as described below:

- **ME-TLS endpoint:** ME-TLS endpoints refer to the ME-TLS client and server. Each ME-TLS endpoint is associated with a unique publicly known identity ID, which can be generated from the endpoints' IP address in addition to some metadata, such as organization name, service name, etc. At system initialization, a ME-TLS endpoint authenticates itself to a set of distributed BF-IBE PKG nodes and obtains its private key.
- **Middleboxes:** Like the endpoints, each middlebox is also equipped with a unique publicly known identity. At initialization, each middlebox also authenticates itself to the distributed PKG nodes and obtains its private key. Then each middlebox also derives the shared secrets between itself and the endpoints. The shared secrets will be used by the middlebox to join future ME-TLS sessions. During handshaking, ME-TLS endpoints can choose middleboxes by their identities and involve them in a session for performance and security enhancements
- **Distributed PKG nodes:** In ME-TLS, endpoints authenticate each other and the introduced middleboxes by utilizing the BF-IBE scheme instead of certificates. At system initialization, each ME-TLS endpoint and middlebox authenticates itself to the distributed PKG nodes in the same way that a normal TLS server would authenticate to a Certificate Authority (CA).

CHAPTER 4: ME-TLS PROTOCOL

The protocol enables middleboxes to be introduced into a session in a passive manner, which preserves the original handshake structure of TLS and eliminates the requirement of secondary secure channels. The endpoints can decide which middlebox can read or modify traffic data, and further verify that the introduced middleboxes performed their tasks on ME-TLS traffic in a predefined order. ME-TLS is also equipped with an implicit version negotiation mechanism to achieve legacy TLS compatibility.

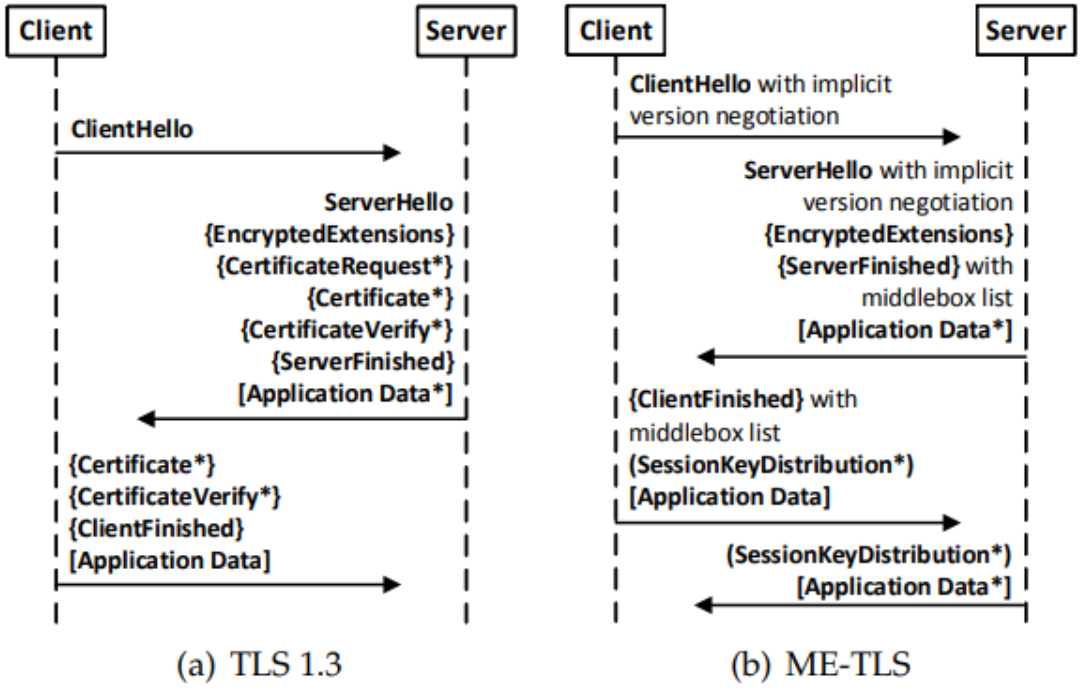

Fig. 4.1: TLS 1.3 and ME-TLS Handshake Overview

4.1 Protocol Overview

Handshake message flow in TLS 1.3 and ME-TLS are shown in Figure 3(a) and Figure 3(b) respectively. ME-TLS is designed to have the same handshake structure as TLS 1.3 to maintain its simplicity and clarity but differs from TLS 1.3 in the following ways

1. Implicit version negotiation mechanism in handshake messages: ME-TLS offers an implicit version negotiation mechanism to achieve legacy TLS compatibility. In customized TLS protocols such as mcTLS [15], a user can not interoperate with a standard TLS endpoint efficiently. If an mcTLS user tries to perform a handshake

with a standard TLS endpoint, he/she would fail and have to analyze handshake packets to find out what causes the failure, then the user needs to switch to standard TLS protocol and re-perform handshaking. While in the ME-TLS protocol, we offer an implicit version negotiation mechanism by utilizing the random field in ClientHello and ServerHello messages. The ClientHello and ServerHello messages in ME-TLS would look exactly like the ones in the standard TLS protocol, but they enable a ME-TLS endpoint to discover whether the other party supports ME-TLS. If so, the ME-TLS handshake procedure is performed, otherwise, the ME-TLS endpoint downgrades itself to a standard TLS endpoint and continues handshaking

2. Removal of certificate-related messages: When two ME-TLS endpoints are establishing a connection, they would utilize their private/public key pairs in the distributed BF-IBE system instead of certificates to authenticate each other, thus handshake messages including Certificate, CertificateVerify and CertificateRequest are no longer needed.
3. Middlebox negotiation and endpoint authentication via Finished message: In ME-TLS, we would like to conceal the identities of the middleboxes to be involved in a session from third parties. Since if an attacker finds out which middleboxes are introduced into an ME-TLS session, the middleboxes would become targets to the attacker. Note that even one compromised middlebox is a tremendous security threat to the entire system. In TLS 1.3, handshake messages after ServerHello are encrypted. We utilize this property and modify the Finished messages for ME-TLS endpoints to perform middlebox negotiation. In this way, the introduced middleboxes' identities are known only by the two endpoints. ME-TLS Finished messages are utilized to realize another important functionality: endpoint authentication. A Finished message carries an HMAC on previously sent handshake messages so that ME-TLS endpoints could authenticate each other.
4. Newly introduced SessionKeyDistribution message: When ME-TLS endpoints decide to involve middleboxes in a session, they need to distribute necessary key materials to middleboxes for them to perform their tasks. In ME-TLS, middleboxes are introduced into a session in an in-band fashion. That is, we do not utilize secondary secure channels such as TLS connections to transmit key materials. Instead, we introduce the new SessionKeyDistribution message to fulfill this task. The message contains key materials for middleboxes, encrypted by different symmetric keys derived from the shared secret between a ME-TLS endpoint and different middleboxes. In this way,

only the intended middlebox can extract its corresponding key materials. The session key materials are distributed to middleboxes under the least privilege principle: a middlebox gets only the key materials it needs. A read-only middlebox would not obtain key materials to perform legal modifications on ME-TLS traffic. We note that the SessionKeyDistribution message is not a handshake message since it is sent together with an application data record, the TLS 1.3 handshake structure is preserved in ME-TLS.

When the handshake is done, the introduced middleboxes utilize their retrieved session key materials to perform various tasks on the session traffic, such as virus detection, intrusion detection, load balancing, etc. When a middlebox finishes processing a ME-TLS application data record, it generates a unique hop-tag and updates the field in the record. What is more, a middlebox with write access generates a new HMAC for the record after processing it. When an endpoint receives the record, it uses the record's HMAC to perform an integrity check and the hop-tag to perform middlebox service chain verification.

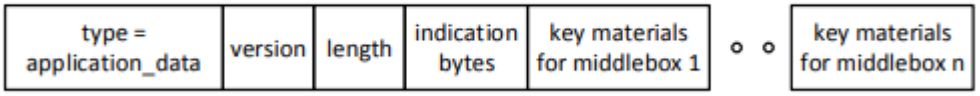

Fig. 4: Format of `SessionKeyDistribution` message

The middleboxes obtain their session key materials by monitoring handshake messages passively instead of joining handshake procedures and negotiating session keys with endpoints directly.

CHAPTER 5: EXPERIMENTAL EVALUATION

This section reviews the performance of ME-TLS, including connection setup time, handshake network consumption, data transmission throughput and HTTPS page load time. McTLS, mbTLS and maTLS are three representative academic works targeted at introducing middleboxes into a TLS session, but mbTLS requires Intel SGX for middlebox and traffic protection. We compare the evaluation results with mcTLS, maTLS and standard TLS 1.3 to demonstrate the practicability of our proposal.

Forward secrecy and entity authentication. In ME-TLS, session keys are derived from the ephemeral key exchange and the BF-IBE private/public key pairs of endpoints and middleboxes. If an attacker obtains history ME-TLS traffic and BF-IBE private keys of the entities in this session, he/she will not able to re-derive session keys since the ephemeral private keys generated during handshaking are discarded by the endpoints. Thus, TLS 1.3 perfect forward secrecy is also preserved in ME-TLS.

However, the attacker is able to impersonate itself as an endpoint to establish a connection, or as a middlebox to join a session by using the obtained BFIBE private keys, either way, system security is damaged.

The situation can be improved by making the distributed PKG nodes include time information in private key generation for endpoints and middleboxes. The endpoints and middleboxes update their private keys by obtaining private key shares from the PKG nodes periodically, for example, once a day. In such a case, if an entity’s private key is compromised, the attacker will only be able to conduct attacks within a particular time period.

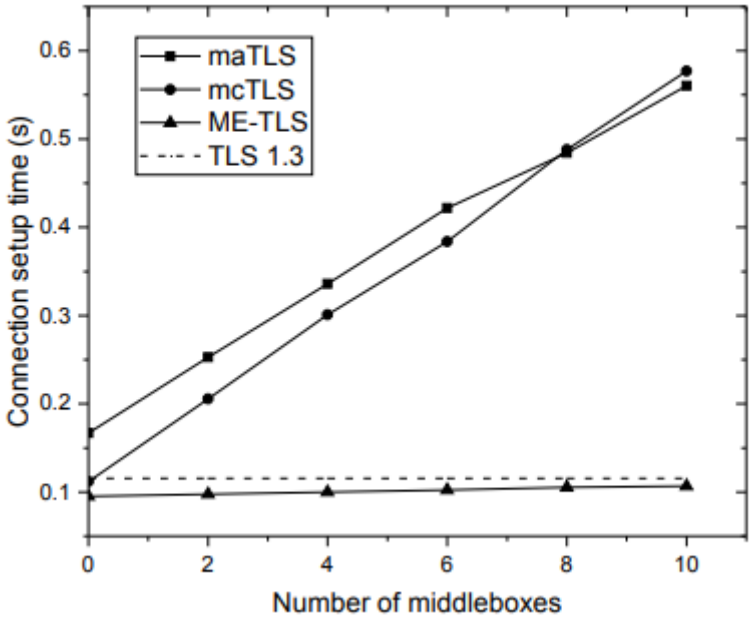

Fig. 5.1: Connection setup time

The connection setup time vs a number of middleboxes, as we can see from Fig. 5, is much less compared to maTLS and mcTLS, and is almost the same as TLS 1.3, which shows that, even as the number of middleboxes increases, the setup time does not increase as much, and this is because of the distributed PKG model.

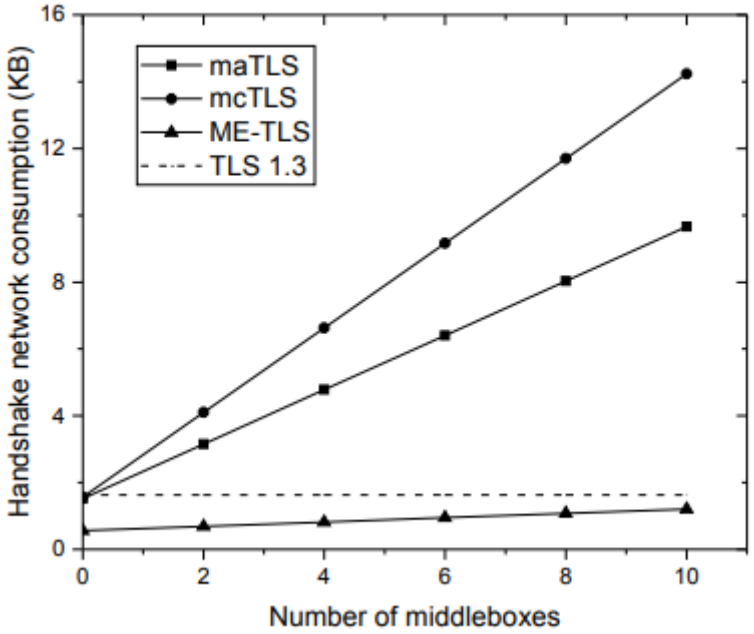

Fig. 5.2: Handshake network consumption

The network consumption for the handshake in the various protocols vs the number of middleboxes is shown in the figure. As we can see, the network consumption in ME-TLS is considerably lower than in its counterparts, the maTLS, mcTLS or in TLS 1.3.

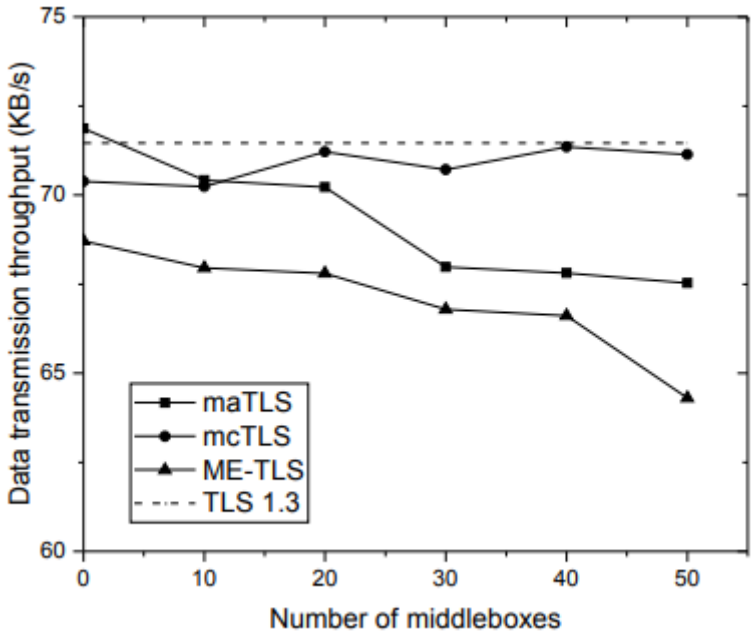

Fig. 5.3: Transmission throughput

The above figure shows a plot of Data transmission throughput and the number of middleboxes in the various protocols. As we can observe, the throughput of ME-TLS does not increase with the number of middleboxes, as one would normally expect to happen. In maTLS, mcTLS and TLS 1.3, the data transmission throughput does not decrease as much as in ME-TLS.

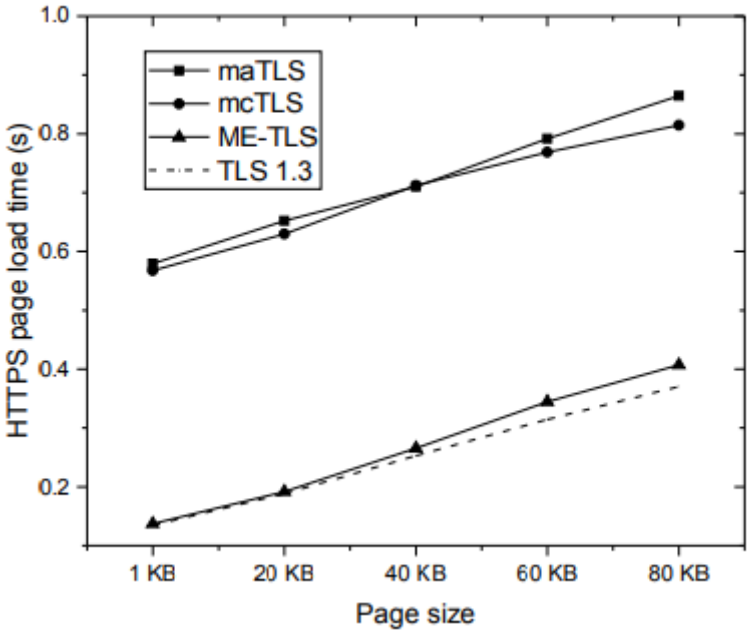

Fig. 5.4: HTTPS page load time

The page load time vs page size for HTTPS in ME-TLS is much better than in maTLS and mcTLS but TLS 1.3 seems better at loading pages faster, even as the page size increases.

Chapter 6
Conclusion

Security is of prime importance in IoT systems. IoT systems carry and process highly sensitive data that needs to remain airtight, as they have ramifications in the real, physical world. In this research, we have considered a more hardware-oriented approach, the middleboxes, since IoT devices are low on computing resources.

The ME-TLS protocol is designed and implemented, which aims to introduce middleboxes into TLS sessions in a passive, authenticated manner. ME-TLS endpoints can further perform traffic access control and service chain verification over the introduced middlebox,es. An implicit version negotiation mechanism is also provided to enable legacy TLS compatibility. Distributed BF-IBE infrastructure is introduced into ME-TLS to perform entity authentication and session key distribution, which eliminates the problems with PKI.

Finally, various performance evaluations are run on the proposal, the experimental results show that ME-TLS offers promising performances and is suitable for practical deployment in IoT systems.

References:

[1] 2020 Yifeng Zheng; Arindam Pal; Sharif Abuadbba; Shiva Raj Pokhrel; Surya Nepal; Helge Janicke “Towards IoT Security Automation and Orchestration”

[2] 2017 Mengmeng Ge, Jin B. Hong, Walter Guttmann, Dong Seong Kim “A framework for automating security analysis of the internet of things”

[3] 2015 Kashif Habib, Wolfgang Leister “Threats Identification for the Smart Internet of Things in eHealth and Adaptive Security Countermeasures”

[4] 2020 Jie Li; Rongmao Chen; Jinshu Su; Xinyi Huang; Xiaofeng Wang “ME-TLS: Middlebox-Enhanced TLS for Internet-of-Things Devices”

[5] 2020 Stephen Hilt, Fernando Mercês, Mayra Rosario, and David Sancho “Worm War: The Botnet Battle for IoT Territory”

[6] 2021 Muhammad Talha Paracha, Narseo Vallina-Rodriguez, Daniel J. Dubois, David Choffnes “IoTLS: Understanding TLS Usage in Consumer IoT Devices”

[7] 2018 “Rfc 8446 - the transport layer security (TLS) protocol version 1.3”

[8] 2019 EtherealMind.com, “Percentage of https (TLS) encrypted traffic on the internet?”

[9] 2000 “RFC 2818 - HTTP over TLS”

[10] 2016 S. A. Shinde, P. A. Nimkar, S. P. Singh, V. D. Salpe, and Y. R. Jadhav, “Mqtt-message queuing telemetry transport protocol,” International Journal of Research, vol. 3, no. 3, pp. 240–244, 2016.

[11] 2011 P. Saint-Andre, “Extensible Messaging and presence protocol (XMPP): Core,”.

[12] S. Vinoski, “Advanced message queuing protocol,” IEEE Internet Computing, no. 6, pp. 87–89, 2006.

[13] T. Chung, D. Choffnes, and A. Mislove, “Tunnelling for transparency: A large-scale analysis of end-to-end violations on the internet,” in Proceedings of the 2016 Internet Measurement Conference. ACM, 2016, pp. 199–213.

[14] D. Naylor, K. Schomp, M. Varvello, I. Leontiadis, J. Blackburn, D. R. Lopez, K. Papagiannaki, P. Rodriguez Rodriguez, ´ and P. Steenkiste, “Multi-context TLS (mcTLS): Enabling secure in-network functionality in TLS,” ACM SIGCOMM Computer Communication Review, vol. 45, no. 4, pp. 199–212, 2015.

[15] D. Naylor, R. Li, C. Gkantsidis, T. Karagiannis, and P. Steenkiste, “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. ACM, 2017, pp. 88–100

[16] H. Lee, Z. Smith, J. Lim, G. Choi, S. Chun, T. Chung, and T. T. Kwon, “maTLS: How to make TLS middlebox-aware?” in NDSS, 2019.