Extending TLS with Mutual Attestation for Platform Integrity Assurance

NorazahAbd Aziz[1, 2], NurIzura Udzir[1], and Ramlan Mahmod[1]
[1]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia,
43400 UPM Serdang, SelangorDarulEhsan, Malaysia
[2]MIMOS Berhad, Technology Park Malaysia, 57000 Kuala Lumpur
Email: azahaa@mimos.my;izura@fsktm.upm.edu.my; ramlan@fsktm.upm.edu.my

Abstract—Normally, secure communication between client-server applications is established using secure channel technologies such as Transport Layer Security (TLS). TLS is cryptographic protocol which ensures secure transmission of data and authenticity of communication at each endpoint platform. However, the protocol does not provide any trustworthiness assurance of the involved endpoint. This paper incorporates remote attestation in the TLS key exchange protocol to solve this issue.The proposed embedded attestation extensionin TLS protocolwill provide assurance of sender's platforms integrity to receiver, and vice versa.The CA responsibility in TLSis replaced using own Trusted Certificate Authority (TCA) in our protocol. The credibility of the proposed protocol is studied to secure against replay attack and collusion attack. The proof is performed using AVISPA with High Level Protocol Specification (HLPSL) through Dolev-Yao intruder model implementation of the proposed protocol.

Index Terms—SSL/TLS extension, integrity, TPM, remote attestation, certificate Authority (CA).

I. Introduction

TLS and its predecessor SSL are cryptographic protocols that are used by web services to transmit sensitive information between client and server applications. The protocols provides authenticated key exchange using asymmetric cryptography, data confidentiality using symmetric encryption and message integrity using message authentication codes scheme. However, these protocols' foundation does not provide any trustworthiness assurance of the involved endpoint. The drawback in this technology can be used by malicious software to create sophisticated spoofs such as SSL certificate spoofing and SSL padlock, and may lead to the replay attack.In the absence of remote attestation protocol, the secure connection protocol is vulnerable to more attacks.

In order to protect against such protocol attacks, the Trusted Computing Group (TCG)has proposedan initiative to sustain the integrity of the computing platform. The technology proposed aims to enhance the security of hardware and software building block called trusted platform through Trusted Platform Module (TPM) [1]. TPM isa built-in hardware security chip embedded in the motherboard and separate from the CPU(s). It contains cryptographic mechanisms to certify the integrity of the software running on a host and to protect sensitive information inside the host [2]. The TPM feature used by the attestation mechanism is the system measurements stored on TPM and the cryptographic functions.

Attestation is a process of assuring the correctness of the information. It provides remote assurance of the state of the hardware component running on a computing device. The trust in the system depends on measurement validation which indicates that the platform is not trusted if it fails the attestation.

Remote attestation mechanism requiresa protocol that can convince a communication partner that a trusted hardware is indeed a trusted hardware. Due to that, the TCG has come out with two protocols, i.e. Privacy CA and Direct Anonymous Attestation (DAA).

This paper proposes a new approach that embed remote attestation component in TLS protocol. The approach also proposes a Trusted Certificate Authority (TCA) component that acts as a Privacy CA to provide the remote identification protocols to users of the platform. The embedded attestation consists of the TCA is able to solve the privacy issue and efficiently defending against malicious code. It also provides confidence to the user about the computing platform's trustworthiness.

This paper starts with part II that discusses the related work. Part III briefly describes the TLS handshake and attestation protocol. The new protocol is demonstrated in part IV and followed by the detailed description on the extended TLS. Part V of the paper discusses about the security analysis of the approach. The paper ends with a conclusion.

II. Related Work

Many discussions on remote attestation focus on methods to enable a remote peer to be securely and

Manuscript received August 8, 2013; revised October 20, 2013.
This work was supported in part by the MimosBerhad and UPMunder NBvTPM project.
Corresponding author email: azahaa@mimos.my
doi:10.12720/jcm.9.1.63-72

reliably informed if the local platform has loaded new software and whether the new local configuration adheres to the remote platform acceptable configurations.[3, "in press" 4, 5]. Dietrich [5] proposed a reliable method to provide a proof for a configuration change. He claimed the method can be implemented with only minor modifications of the TPM specification and the TLS protocol. The simple concept allows a reliable and secure reporting of PCR changes and was designed to easily integrate with TLS implementation. His method requires some modification of TLC MAC computation within TPM. However, no further explanations about the linkage with the TLS protocol were provided.

Goldman et al. "in press" [6] were the first to propose the inclusion of attestation in the setting up of secure tunnel between two hosts. Specifically they look into putting together SSL and remote attestation protocols. The proposed protocol aims to establish an authenticated shared key between two hosts and obtain assurance of the server's platform integrity. Their protocol uses three certificates, namely, SSL endpoint certificate, AIK certificate and Platform Property certificate. The first two certificates are required for SSL and remote attestation protocols, respectively. The Platform Property certificate is introduced to bind the endpoint SSL certificate to the AIK certificate. Unlike our protocol, the SSL certificate is signed by the TPM using the AIK private key, instead of a Certification Authority.

In 2007, Gasmi et al. "in press" [7] proposes some changes in TLS handshake in order to insert attestation protocol. Then in 2008, Armknecth et al."in press" [8] propose similar protocol but adhere to TLS protocol standard toallow backward compatibility. Specifically, the attestation protocol messages are sent over in the TLS handshake extension. The protocol is defined for mutual authentication and mutual attestation and is designed to resist replay attack. Here each entity's signing key pair is generated and sealed in its TPM. In this way the protocol provides assurance that the signature is produced by the entity's platform, and not another. These keys are very similar to AIK. The difference is that Armknecth et al. propose the signature verification keys to be included in the TLS certificates.

Stumpf etal. [9] performed masquerading attack on Sadeghi et al."in press" [10] remote attestation protocol. It happened when the server request for the attacker to attest its platform, the attacker relay messages between the server and a good client. So, the attacker succeeds in attestation protocol although his platform is compromised. This attack is successful because the server does not know that the replies were not originally produced by the attacker's platform. To overcome this vulnerability, Stumpf et al. added the Diffie-Hellman (DH) key exchange to the remote attestation protocol.Our approach differs in that we do not use Diffie-Hellman algorithm to perform attestation in TLS handshake protocol.

Zhou and Zhang [11] performed collusion attack on protocols in [8,9]. The attacker sends its DH public key to the good client to be included in the hash and signed by the good client's AIK private key. This is the only way the attacker can obtain a shared key with the server after attestation using the good client's platform. This attack is applicable if there is a method to get the TPM to sign any given data through valid software. Their approach differs from ours because the SSL libraries do not need to be modified when integrating with the attestation protocol. Moreover, we also focus on preventing collusion attack.

III. PRELIMINARIES

A. TLS

SSL was developed in 1994 by Netscape Communications Corporation to secure Internet transaction. Then, SSL 3.0 became the basis of the standard protocol, the TLS. The TLS authentication protocol suite is based on public key cryptography. This protocol allows client and server to authenticate themselves to each other and allows both hosts to establish an encrypted connection.

There are four components/parts of the TLS protocol, separated into two layers. The first layer is TLS Handshake protocol consisting of Handshake, Change Cipher Spec and Alert protocol. The second layer is the TLS record protocol where the data are transmitted securely.The Handshake protocol uses the Record layer protocol to establish a TLS connection by allowing authentication and negotiation of a cryptographic algorithm between a TLS-enabled server and a TLS-enabled client.

TLS does not aim to provide the endpoint integrity issue so it cannot assure the trustworthiness of involved endpoints. Lack of the trustworthiness of client-server system may lead to the Reply Illusion attack on TLS Protocol. This can expose the private key to the attacker. Hence, it is necessary to enhance the security of TLS communication by enabling platform integrity between client and server to ensure no malware or spyware is present. The new TLS extensions protocol proposed is embedded with Remote Attestation protocol process in TLS to ensure the endpoint integrity of host platform. The protocol approach will be explained in the next section.

B. Attestation

Trusted platform module (TPM) is a hardware based security chip which is developed in accord to the specifications published by TCG. Currently, they are manufactured by Infineon, Atmel and Broadcom. The TPM provides root of trust and establishes chain of trust of the platform. The chain of trust is the approach where every link in the chain has been measured by the prior one.

The TPM operations based on TCG architecture is divided into three main categories; trusted measurement, trusted reporting, and trusted storage [12]. Trusted or integrity measurement consists of a service that measures

platform characteristics beforea measured software component gets executed or run by producing a metric or fingerprint using SHA-1 computed by the TPM. The metrics will be stored in Storage Message Log (SML) and Platform Configuration Registers (PCR). Integrity measurement incorporates the same chain of trust process into it, but instead of verifying, it makes the measurement.

Establishing trust of a computer system begins with enhancing the computer BIOS with a Core Root of Trust for Measurement (CRTM). CRTM is a piece of written code stored in the boot block of the BIOS. This piece of code is considered trustworthy. It reliably measures integrity value of other entities, and stays unchanged during the lifetime of the platform "in press" [13]. CRTM will be the first to run to measure other parts of the BIOS block before passing control. The BIOS then measures hardware and the boot-loader and passes control to the boot-loader. The boot-loader measures OS kernel and pass control to the OS. After the OS is loaded or during the boot process,the administrator can check for the PCR values to see if it is running in the expected configuration.Up to this point, the system configuration has been measured and it is possible to verify the current system configuration by examining the content of the PCR.

The security of the attestation report is the value of PCR digitally signed with the TPM public AIK. The Endorsement Key (EK) public key is used to uniquely identify a TPM and certified by an appropriate CA. The EK private key can be used to sign assertions of the trusted computer's state. A remote computer can verify that those assertions have been signed by a trusted TPM [14].

Trusted storage is where cryptographic keys and data are stored in the TPM. It consists of volatile and non-volatile memory. The volatile memory is for the usage of cryptographic protocols and storage of sensitive information like passwords. The non-volatile memory contains the AIK.

Remote attestation was adopted as the method for remote assurance of the state of the hardware and software running on the computing device. Attestation is a process of assuring accurateinformationand closely related to authentication. It is obviously a critical concept for the trusted platform because the trust in the system is based on taking measurements and checking the measurements. If a system is not able to attest the accuracy of the information, then the trust on the platform cannot be established.

The TCG proposed two protocols for a client to convince its remote communication partner that a piece of hardware is trustworthy. In return, this enables two communication partners to establish the secure computing platform and therefore it is safe to exchange data. These specified protocols are Privacy CA "unpublished" [14] and Direct Anonymous Attestation (DAA)"in press" [13, 15], [16].

The Privacy CA similar to Public Key Infrastructure (PKI) which involves TTP in each transaction and the party must be fully trusted by all other parties. The Privacy CA is assumed to know the public parts of the Endorsement Keys of all valid TPM. This valid TPM refers to an uncompromised TPM. In contrasts, a rogue TPM is a TPM that has been compromised and had its secrets extracted. Now, whenever a TPM needs to authenticate itself to a verifier, it generates a second RSA key pair, called an Attestation Identity Key (AIK). Then, it sends the AIK public key to the Privacy CA, and authenticates this public key relating to the EK. The Privacy CA will issue a certificate on the TPM's AIK if it finds the EK in its list.

DAA implements the Camenisch-Lysyanskaya signature scheme [16] and zero-knowledge proof [17] to exclude the usage of a Privacy CA, but various exponential operations in its protocol leads to the lack of efficiency. In this paper, we focus on built-in Privacy CA implementationnamed as Trusted CA (TCA).

IV. New Extension Approach

Our approach protocol does not require modification of the TLS library and hence no changes to plug-ins in user's browser or application. Through this protocol, we achieve the mutual authentication and mutual attestation between two parties with the following advantages:

- Easy deployment because there are no changes required in the TLS library. Theattestation protocol is not yet standardized, so attestation only need to be appliedasadditional plugin while using existing TLS plugin.
- Backward compatible - if a host does not have TPM, it can still connect toa server without attestation.
- Achieve anonymity and unlinkability via TPM and TCA.

TABLE I: NOTATIONS OF PROTOCOLS

n_{host}	non-predictable nonce of host getting from TLS
$cert_{RSA}$	public key certificate signed by CA i.eVerisign
id_{host}	unique Identifier of host for $cert_{RSA}$
id_{AIK}	identity label for the new AIK (AIK identifier)
S_{PCR}	selection of PCR values verifier (client or server) is interested in
PK_{AIK}	public verification key of AIK
SK_{AIK}	private verification key of AIK
pk_{TCA}	public verification key of the Trusted CA
PK_{EK}	public verification key of EK
SK_{EK}	private verification key of EK
N_v	Nonce (anti-replay value) chosen by the verifier
I_V	Indicator whether verifier is interested in TPM version information
TPM_{info}	TPM version and revision information
$cert_{AIK}$	Attestation Credential (from Trusted CA)
Mlog	TPM Platform measurement log

The extended protocol consists of three defined protocols: registration (P1), AIK certificate creation (P2) and TPM-based attestation (P3) protocol. Table I defines the notations of the protocols.

A. Registration Protocol (P1)

This protocol must take place first before any transaction between client and server. It requires the host to request EK certificate from the TCA by providing the necessary TPM EK public key. The TCA signs the certificate request using its private key if the TPM EK public key is valid and thenthe identity of host is added to the list of acknowledged clients. The EK certificate also indicates that the EK has been properly created and embedded into a TPM. In this paper, the TCA acts as the TPM manufacturer and is fully trusted not to reveal any information to any other party. This is important to protect any correlation of AIKs with corresponding platform identity (EK).

In this phase, the host needs to register to other communicating entity to acknowledge each other's policies such as integrity measurement reference point. Based on the policies, the host computes the core integrity measurement values and sends the valuesto the endpoint entity for future reference. The entity can use TPM sealing mechanism to secure the core values. As we know, the TPM can generate authentic reports of current PCR contents that contain current integrity measurement values. These current values will be compared with the core values in the attestation channel.

B. AIK Certificate Creation Protocol (P2)

TABLE II: AIK CERTIFICATE CREATION PROTOCOL (P2)

Protocol : AIK Certificate Creation Phase (P2)		
SUMMARY: Host(H) creates AIK and signing the AIK with TCA (T) RESULT: Unilateral AIK certificate creation with TCA		
H	:	Retrieve n_{host}
H	:	Generate $id_{AIK} = hash(id_{host} \| n_{host})$
H	:	TPM loadKey(sk_{AIK})
H	:	σTPM = TPM MakeIdentity= sign($hash(id_{AIK} \| pk_{TCA})$)
H→T	:	$enc_{TCA}(pk_{AIK}, cert_{RSA}, n_{host}, \sigma TPM, pk_{EK})$
T	:	Verify EK credential, σTPM
T	:	Verify $cert_{RSA}$, generate $id_{AIK} = hash(id_{host} \| n_{host})$
T	:	Issue $cert_{AIK} = sign_{TCA}(pk_{AIK}, id_{AIK})$
T	:	Create symmetric encryption key, K
H←T	:	$C1 = enc_{EK}(K, hash(pk_{AIK}))$, $C2 = enc_K(cert_{AIK})$
H	:	K, $hash(pk_{AIK})$ =TPM ActivateIdentity=$dec_{EK}(C1)$
H	:	Verify pk_{AIK}, $hash(pk_{AIK})$
H	:	Retrieve $cert_{AIK} = dec_K(C2)$

Table II illustrates the AIK generation process adopted from [12]. In order to prevent replay and collusion attack, we have proposed the creation of identity label for new AIK as:

$$id_{AIK} = hash\ (id_{certRSA} \| n_{host})$$

This AIK unique identity is linkable with TLS connection which the n_{host} is generated during TLS handshake and $id_{certRSA}$ is unique identifier of the host signed by other CA. Hence, $id_{certRSA}$ is represented by id_{host}as in Table II.

C. TPM-based Attestation Protocol (P3)

The integrity reporting protocol represent as P3 which was extracted from [19], [20]. This protocol is based on TPM challenge-response authentication and able to prevent non-authentic integrity information execution by maintaining freshness and authenticity of the integrity information. The random number nonce of client and server are used to verify freshness. MAC of TLS session key-establishment is used as encryption key for all messages between client and server including integrity information in order to ensure the messages' authenticity. Table III simplifies this TPM-based attestation protocol [9], [18] after each host performs P2 to get $Cert_{AIK}$.

TABLE III: TPM-BASED ATTESTATION PROTOCOL (P3)

Protocol : AIK Certificate Creation Phase (P3)		
SUMMARY: S answers the attestation challenge of platform C RESULT: Integrity reporting - unilateral attestation		
C ← S	:	N_S, I_V, S_{PCR}
C	:	TPM loadKey(sk_{AIK})
C	:	σTPM=TPM_Quote=sign{PCR, N_S, TPM_{info}} SK_{AIK}
C	:	Retrieve M_{log}
C → S	:	Q = (Mlog, σTPM, TPM_{info}, $cert_{AIK}$), MAC_{sess}(Q)
S	:	Verify $Cert_{AIK}$using PK_{AIK}
S	:	Verify σTPMusing TPM_{info}, N_S, and PCR_{SPCR}

D. Mutual Attestation on TLS

In our protocol, we used the AIK certificate for the TLS handshake in order to achieve anonymity from server and unlinkability by eavesdropper active listener. After the TLS handshake, we use the session key to perform remote attestation to handle the endpoint integrity issue. We have overcome the following issues:

- The client and server host without integrity verification cannot be trusted because secret information can be revealed to any adversary platform by unverified applications inside the host system itself.
- The unverified applications can be compromised either by malicious codes.

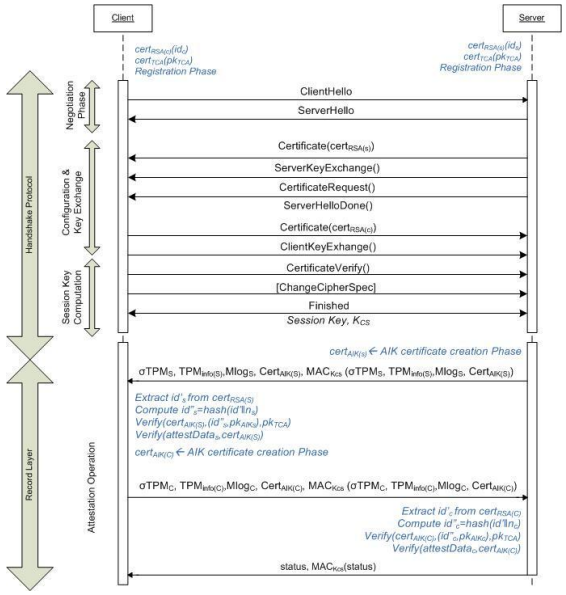

Fig. 1. New extend TLS with attestation approach

TABLE IV: EXTENDED TLS WITH MUTUAL ATTESTATION PROTOCOL

Protocol : Achieving mutual attestation on TLS				
SUMMARY: C and S sends attestation challenge to each other				
RESULT: Mutual authentication and mutual attestation between C and S				
1. System Setup				
C and S must generate id_x and sign it with CA (i.e. Verisign) to get $Cert_{RSA(x)}$ then perform registration protocol (P1) to S to obtain $Cert_{EK}$.				
2. Protocol Messages				
1	C → S	:	nc, Sid, Pc	(client hello)
1	C ← S	:	ns, Sid, Ps	(server hello)
2	C ← S	:	$cert_{RSA(s)}$	(server cert)
3	C	:	$Verifycert_{RSA(c)}(pk_{CA})$	(certverify)
4	C→S	:	$cert_{RSA(c)}$	(clientcert)
4	C → S	:	Finished sessionkey, k_{cs}	(client finished)
5	S	:	Verify $cert_{RSA(c)}(pk_{CA})$	(cert verify)
6	C ← S	:	Finished sessionkey, k_{cs}	(server finished)
7'	S	:	id_{AIKs} = hash(id_s|| n_s)	
7'	S	:	$cert_{AIK(s)}$← AIK creation Protocol (P2)	
7'	S	:	$attestData_S$← Attestation Protocol (P3)	
8'	C ← S	:	$attestData_S$, $MAC_{kcs}(attestData_S)$	
9'	C	:	id'$_s$←$cert_{RSA(s)}$	
9'	C	:	id"$_s$← hash (id'$_s$|| n_s)	
9'	C	:	Verify [$cert_{AIK(s)}$, (id"$_s$,pk_{AIKs}), pk_{TCA}]	
9'	C	:	Verify $attestData_S$←Attestation Protocol (P3)	
10'	C	:	id_{AIKc} = hash(id_c || n_c)	
10'	C	:	$cert_{AIK(c)}$← AIK creation Protocol (P2)	
10'	C	:	$attestData_C$← Attestation Protocol (P3)	
11'	C → S	:	$attestData_C$, $MAC_{kcs}(attestData_C)$	
12'	S	:	id'$_c$←$cert_{RSA(c)}$	
12'	S	:	id"$_c$← hash (id'$_c$|| n_c)	
12'	S	:	Verify [$cert_{AIK(c)}$, (id"$_c$,pk_{AIKc}), pk_{TCA}]	
12'	S	:	Verify $attestData_C$←Attestation Protocol (P3)	
12'	C ← S	:	status, MAC_{kcs} (status)	

We have illustrated this protocol sequence as Fig. 1 and detail protocol message in Table IV.

V. SECURITY ANALYSIS

This paper will discuss the security analysis using informal analysis followed by formal analysis using the AVISPA protocol prover [21].

A. Informal Security Analysis

For informal security analysis, we focus on replay attack and collusion attack that are handled by our approach. Replay attack is an attack where data packets are intercepted between two parties and are later re-sent by the attacker to one or more of the parties. It is also called as freshness attack because the loss of freshness of the information during transmission. In this case, the adversary does not necessarily distinguish the message content since he only retransmit the older message and blocking any new fresh messages to receiver. This hacking technique usually was played by the adversary called as Man-in-the-Middle attack.

Collusion attack is a new type of attack during which a compromised host asks the information from a trusted host in order to answer attestation challenges. This attack also utilizes the replay attack technique with collaborative mechanism. As example, we adopted [9] protocol that is vulnerable to this attack as discussed in [11] as in Fig. 2. The figure shows that the server M passed attestation challenge request of client C without revealing M's session key information.

1.	$C \rightarrow M$	:	$nonce, DH^c_{pub}$
2.	$M \rightarrow S$	:	$nonce, DH^m_{pub}$
3.	S	:	Compute and retrieve $\sigma data{=}DH^m_{pub}, SML, cert_{AIK}$
4.	$S \rightarrow M$	:	$\sigma data, Sig_{AIK_s}(PCR_s, H(nonce, DH^m_{pub}))$
5.	$M \rightarrow C$	:	$\sigma data, Sig_{AIK_s}(PCR_s, H(nonce, DH^m_{pub}))$
6.	C	:	Verify the signature
6.	C	:	Compute $K_{cm} = (DH^m_{pub})^c$
7.	$C \rightarrow M$	:	$nonce_1$
8.	M	:	Compute $K_{cm} = (DH^c_{pub})^m$
9.	$M \rightarrow C$	:	$enc(nonce_1)K_{cm}$
10.	C	:	Validate $nonce_1$ with K_{cm}

Fig. 2. Collusion attack scenario example

Replay attack can be launched by the client or the server. For our protocol, the aim of the attack is for a compromised host to attest successfully. This is achieved by making another host to provide good attestation data and relaying it to the verifier. As such, the compromised host will be accepted as a good platform. Suppose the server S wants to launch a replay attack in the protocol. The server S will begin the attack after the TLS ends and the attestation stage begins. Server S will coerce another host, V to provide attestation data, *attestData* which includes PCR values and SML values which will then be signed with V's AIK signing key. When server S obtains the values, signature and V's AIK certificate, S computes the Message Authentication Code (MAC) of the whole message using the session key k.Then S sends all of the information to the client.

We argue that the AIK certificate verification will fail because

$$id_s = hash(id_s \| n_s) \neq id_V$$

where id_s is the server's identifier in its RSA certificate while id_V is the victimized host AIK identifier in the AIK certificate. At this point the protocol will be aborted and hence the attack fails.

The only way for S to make the attack successful is by registering V's RSA certificate to the TCA to be associated with S's endorsement key. Assuming that the TCA carefully verifies the submitted RSA certificate against the identity of the host during registration phase, attempts to associate one host's RSA certificate to another host's endorsement key will fail. Clearly, here the TCA is a fully trusted party.

Collusion attack is similar to replay attack, but the third host is now an accomplice of the adversarial host. As in the replay attack example, let the adversary be in the server S. For collusion attack, the host V cooperates with S to enable S to successfully attest itself to the client. Clearly, in this attack, the best chance to be successful is to register S's RSA certificate with V's TPM EK. The alternatives are to register S's RSA certificate with its own EK or register V's RSA certificate with V's EK to the TCA. If the first option is performed, then verification of attestation data will fail because it will be signed using V's AIK signing key. The second choice will fail at the AIK certificate verification stage because TLS will only be successful if S's RSA certificate is presented, while the AIK certificate contains V's identifier.

Here we attack by having server S registers its RSA certificate with V's endorsement key EK. During registration, there will be discrepancy between the identifier in the RSA certificate which holds S's name and the TPM (or EK) owner name which contains V's name. The TCA is expected to detect such discrepancies and reject the registration.

Otherwise, during the protocol, V generates AIK key pair and then S relay all messages between TCA and V to obtain AIK certificate. Then, S performs TLS with the client using its RSA certificate as usual. TLS run will be successful and the session key will be established. However, at the attestation stage, S again relays the message to V who would provide the attestation data.

In this case, the AIK certificate is found valid by the client because the AIK certificate contains S's identifier. Hence, the protocol will establish a trusted channel between the client and server S. Therefore, the TCA must be honest and trusted to perform identity checks during registration.

B. Formal Security Analysis

We used the Automated Validation of Internet Security Protocols and Applications (AVISPA) [22] tool in order to prove our security protocol. AVISPA uses High Level Protocol Specification (HLPSL) [23] as the specification language to present the analyzed protocols and to specify the protocol's security properties. AVISPA is very expressive with great flexibility to analyze cryptographic protocols. This tool doesnot require expertise or skills in mathematics. However, it is so complex to define implementation environment of a protocol and user-defined intrusion model in order to convert the protocol into HLPSL

All formal cryptographic protocols analyzers generally implement Dolev-Yao intruder model [24]. The Dolev-Yao model assumesan intruder can control the entire network, performs cryptographic operations available to authentic users of a system, and is in association with a subset of dishonest principals. But, the special types of insider attacks which are able to create authentic message might not be detected in the Doley-Yao model. The insider attacks are a very potent attack type because the adversary takes over the honest platform including the TPM. However, through AVISPA, our protocol's security properties can be achieved using the authentication phase and key-establishment verification and against the Dolev-Yao attackers.

The HLPSL is a role-based language consisting of basic roles and composition roles. Basic roles are defined as communicating entities which participate in the protocol. Other than the communicating entities, environment and session of protocol execution also declared as roles. Composition roles represent scenario of the basic roles. Each role has its own variables. Communication networks in the protocol are represented by the variables which transmits different properties of a particular environment. In HLPSL model of our protocol, we define several roles; A (modelling the client) and B (modelling the server), intruder model and environment. Since our protocol integrates with TLS protocol, we modified the HLPSL model of TLS protocol adopted from [23].The complete code of the HLPSL model for this protocol is presented in Appendix A.

The basic roles defined the information of initial parameters, state, and the state transition. Refer the role of client (A) and server (B) with their global variables in Appendix Afor the state transition.

Currently, AVISPA tool only supportsDolev-Yao model which is handled by (dy) channels. As mentioned earlier, the intruder in Dolev-Yao model is an authentic user of the network who has full ability to control all transmission messages over the network. In the HLPSL, we named the channel as SA, RA which denotes send or receive for Client (A) and SB, RB denotes as send or receive for Server (B). Since the channels comprise of changes values, whether it is empty or not, so we assume these values are represented by a global parameter. This parameter aremodeled as variable V which is also known as potential attacker.*role session (A,B: agent, V: text,*

```
Ka, Kb, Ks, Kca, KaikA, KaikB: public_key,
      H, PRF, KeyGen: hash_func)
def=
      local  SA, SB, RA, RB: channel (dy)
composition
   client(A,B,H,PRF,KeyGen,V,Ka,Ks,KaikA,Kca,SA,RA)
```

```
/\server(A,B,H,PRF,KeyGen,V,Kb,Ks,KaikB,Kca,SB,RB)
  end role
```

The environment role contains the knowledge of the intruder behaviour and composition of session. The knowledge specified to the intruder is defined by global constants of assigned variables. The composition of session is modeled in order to make the intruder able to play as legitimate user.

The section goal is to define the security properties using predefines macros. The available macros relate to the secrecy of some information and the strong or weak authentication of agents of some information. The goal is identified by predefined predicates (secret, witness, request, and wrequest) in the state transition. The protocol goal is mutual authentication and to establish a secret key between the client and server. The intruder should not be able to accomplish valid authentication since he could not learn the secret session key, ClientK and ServerK.

```
goal
secrecy_ofsec_clientk,sec_serverk
  %Client authenticates Server on na_nb1
authentication_on na_nb1
  %Server authenticates Client  on na_nb2
authentication_on na_nb2
authentication_onsk_verify
secrecy_ofsec_smla, sec_smlb
authentication_onsmla_verify, smlb_verify
```

We have specified different security goals based on state transitions in the events using secret, witness and request structure which are verified through AVISPA as follows:

- Client authenticates Server on the value of $N_c$whereby only Server is able to sign N_c using his own private key.
- Server authenticates Client on the value of $N_s$where only Server is able to sign N_s using his own private key.
- Client and Server authenticate the TPM attestation data of each other on the Finished messages where Client and Server share the key K_{cs}. The K_{cs} remains secret and confidential.
- Client and Server share the Stored Measurement Log (SML) to re-compute value for attestation validation which is kept secret and privacy related.

Through AVISPA, we are able to specify the authentication goals using witness and request command. The secrecy goal also can be attained using secret command. We are going to discuss these commands as unilateral authentication. For normal authentication goal as statement [1a][1b][2a][2b] and attestation goal which involves TPM as statement [1c][2c].

```
/\ witness(A,B,na_nb2,Na.Nb')          [1a]
/\ witness(A,B,sk_verify,ClientK)      [1b]
```

The statement [1a] means that agent A creates*Na.Nb'* value for agent B and wants agent B to agree to the value. The value created is for the *na_nb2* purpose, which means that it is secure from intruder. Similarly, statement [1b] means that agent A declares the *sk_verify* purpose and he wants agent B to agree on the value *ClientK*.

```
/\ request(B,A,na_nb2,Na.Nb)           [2a]
/\ request(B,A,sk_verify,ServerK)      [2b]
```

The statement [2a] implies that agent B believes that he communicates with agent A and accepts the value of *Na_Nb* for the *na_nb2* purpose. The statement [2b] is read as agent B requests a check of *ServerK* value (where *ClientK* is equal to *ServerK*), agrees with agent A on this valueswhich relies on the guarantee that agent A exists.

```
/\ request(A,B,smlb_verify,SMLb)       [1c]
/\ witness(B,A,smlb_verify,SMLb')      [2c]
```

The attestation goal starts with the statement [2c] which declares that agent B is the witness for the information *SMLb*and wants agent A to agree with the information for *smlb_verify* purposes. Then in statement [1c], agent A requests a check of the information *SMLb* and agrees with the value.

We analyzed the HLPSL model of protocol using SPAN [25]. The result showed that no attack trace is found. So, the security properties of the protocol are fulfilled and secure against Dolev-Yao attackers.

VI. Conclusion

This paper proposed extending TLS protocol with mutual attestation in order to guarantee platform integrity of client-server environment. Many existing solution required the TLS library modification since the attestation embedded in the TLS handshake protocol. Our solution does not require the changes since the attestation protocol can be applied as additional plugin to theexisting TLS library. This solution achieves anonymity and unlinkability through TPM and Trusted CA and provides linkagebetween identity and integrity of endpoint platform. Similar with Zhang et.Al., our protocol is also resistant to thenew type of collusion attack as well as replay attack. Through the informal security analysis, we discussed the probability of the protocol to prevent the attacks. Using the AVISPA, the result shows that our protocol is secure against the Dolev-Yao attackers where no attack trace is found.

Appendix A Avispa Source for Extended Tls with Mutual Attestation Protocol

```
role client(A, B : agent,  H, PRF, KeyGen: hash_func, V: text,
Ka, Ks, KaikA,Kca: public_key,
          %% Ks is the public key of a T3P (ie. CA)
         SND, RCV: channel (dy))
played_by A

def=
      local Na, Nb, Sid, Pa, Nw: text,
      State: nat,
      Finished: hash(hash(text.text.text).agent.agent.text.text.text),
ClientK, ServerK: hash(agent.text.text.hash(text.text.text)),
Kb,KaikB: public_key,
```

```
        % certificates for the private key inv(KaikA)
CertAikA: {agent.public_key.text}_inv(public_key),
        % certificates for the private key inv(KaikB)
CertAikB: {agent.public_key.text}_inv(public_key),
KaikSetA, KaikSetB:public_key set,
KcaSet:public_key set,
SMLa,SMLb:hash(agent.nat.text.text),
PCRa, PCRb:{hash(agent.nat.text.text)}_inv(public_key),
M: hash(text.text.text)

constsec_clientk, sec_serverk : protocol_id
init  State := 0
  transition
  1.  State = 0
     /\ RCV(start)
     =|>
     State' := 2
     /\ Na' := new()
     /\ Pa' := new()
     /\ Sid' := new()
     /\ SND(A.Na'.Sid'.Pa')

  2.  State = 2
     /\ RCV(Nb'.Sid.Pa.{B.Kb'}_(inv(Ks)))
     =|>
     State' := 4
     /\ Nw' := new()
     /\ M' := PRF(Nw'.Na.Nb')
     /\ Finished' := H(PRF(Nw'.Na.Nb').A.B.Na.Pa.Sid)
     /\ ClientK' := KeyGen(A.Na.Nb'.PRF(Nw'.Na.Nb'))
     /\ ServerK' := KeyGen(B.Na.Nb'.PRF(Nw'.Na.Nb'))
     /\ SND({Nw'}_Kb'.
           {A.Ka}_(inv(Ks)).
           {H(Nb'.B.Nw')}_(inv(Ka)).
           {H(PRF(Nw'.Na.Nb').
A.B.Na.Pa.Sid)
           }_KeyGen(A.Na.Nb'.PRF(Nw'.Na.Nb')))
     /\ witness(A,B,na_nb2,Na.Nb')

  3.  State = 4
     /\ RCV({Finished}_ServerK)
     =|>
     State' := 6
     /\ request(A,B,na_nb1,Na.Nb)
     /\ secret(ClientK,sec_clientk,{A,B})
     /\ secret(ServerK,sec_serverk,{A,B})

 4. State=6
 /\ RCV(B.PCRb'.SMLb'.CertAikB'.
{PRF(PCRa'.SMLb'.CertAikB)}_ClientK)
    /\ CertAikB'={B.KaikB'.KaikSetB'}_inv(Kca')
 /\ in(Kca',KcaSet)
   /\ request(A,B,smlb_verify,SMLb)
     =|>

   State' := 8
     /\ SMLa':= new()
     /\ PCRa':= {SMLa}_inv(KaikA)
     /\ KaikA' := new()
     /\ CertAikA':={A.KaikA'.KaikSetA'}_inv(Kca')
     /\ SND(A.PCRa'.SMLa'.CertAikA'.
{PRF(PCRa'.SMLa'.CertAikA')}_ClientK)
     /\ witness(A,B,sk_verify,ClientK)
     /\ secret(SMLa,sec_smla,{A,B})
     /\ witness(A,B,smla_verify,SMLa)

end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role server(A, B : agent,
            H, PRF, KeyGen: hash_func,
           V: text,
            Kb, Ks,Kca,KaikB: public_key,
            SND, RCV: channel (dy))
played_by B

def=
  local Na, Nb, Sid, Pa, Nw: text,
State:nat,
Ka,KaikA: public_key,
ServerK:hash(agent.text.text.hash(text.text.text)),
       % certificates for the private key inv(KaikA)
CertAikA: {agent.public_key.text}_inv(public_key),
      % certificates for the private key inv(KaikB)
CertAikB: {agent.public_key.text}_inv(public_key),
KaikSetA, KaikSetB:public_key set,
KcaSet:public_key set,
SMLa, SMLb:hash(agent.nat.text.text),
PCRa,PCRb:{hash(agent.nat.text.text)}_inv(public_key)

init  State := 1
  transition
  1.  State = 1
     /\ RCV(A.Na'.Sid'.Pa')
     =|>
     State' := 3
     /\ Nb' := new()
     /\ SND(Nb'.Sid'.Pa'.{B.Kb}_(inv(Ks)))
     /\ witness(B,A,na_nb1,Na'.Nb')

  2.  State = 3
     /\ RCV({Nw'}_Kb.{A.Ka'}_(inv(Ks)).
           {H(Nb.B.Nw')}_(inv(Ka')).
           {H(PRF(Nw'.Na.Nb).
A.B.Na.Pa.Sid)
           }_KeyGen(A.Na.Nb.PRF(Nw'.Na.Nb)))
     =|>
     State' := 5
     /\ ServerK':=KeyGen(B.Na.Nb.PRF(Nw'.Na.Nb))
     /\ SND({H(PRF(Nw'.Na.Nb).
A.B.Na.Pa.Sid)
           }_KeyGen(B.Na.Nb.PRF(Nw'.Na.Nb)))
     /\ request(B,A,na_nb2,Na.Nb)

  3.  State = 5
      /\ RCV(start)
      =|>
     State' := 7
     /\ SMLb':= new()
     /\ PCRb':= {SMLb}_inv(KaikB)
     /\ CertAikB':={A.KaikB'.KaikSetB'}_inv(Kca')
     /\ SND(A.PCRb'.SMLb'.CertAikB.{
PRF(PCRb'.SMLb'.CertAikB)}_ServerK)
      /\ witness(B,A,smlb_verify,SMLb)
      /\ secret(SMLb,sec_smlb,{A,B})
      /\ request(B,A,sk_verify,ServerK)

  4. State= 7
     /\ RCV(A.PCRa'.SMLa'.CertAikA.{
PRF(PCRa'.SMLa'.CertAikA)}_ServerK)
     /\ CertAikA'={A.KaikA'.KaikSetA'}_inv(Kca')
     /\ in(Kca',KcaSet)
      =|>
    State' := 9
     /\ request(B,A,smla_verify,SMLa)

end role

role session(A,B: agent, V: text,
             Ka, Kb, Ks, Kca, KaikA, KaikB: public_key,
                H, PRF, KeyGen: hash_func)
def=
  local  SA, SB, RA, RB: channel (dy)
  composition
        client(A,B,H,PRF,KeyGen,V,Ka,Ks,KaikA,Kca,SA,RA)
     /\ server(A,B,H,PRF,KeyGen,V,Kb,Ks,KaikB,Kca,SB,RB)
end role
```

```
role environment()
def=
const
        na_nb1, na_nb2, sec_smla, sec_smlb,
        smla_verify,  smlb_verify, sk_verify: protocol_id,
        h, prf, keygen : hash_func,
        a, b        : agent,
        v : text,
        ka, kb, ki, ks, kaika, kaikb, kca: public_key
intruder_knowledge =
        { a, b, ka, kb, ks, kca, kaika, kaikb, ki, inv(ki),
        {i.ki}_(inv(ks)),  {i.ki}_(inv(kca)) ,v}
  composition
        session(a,b,v,ka,kb,ks,kca, kaika,kaikb,h,prf,keygen)
  /\    session(a,i,v,ka,ki,ks,kca,kaika,kaikb,h,prf,keygen)
  /\    session(i,b,v,ki,kb,ks,kca,kaika,kaikb,h,prf,keygen)

end role


goal
secrecy_ofsec_clientk,sec_serverk
authentication_on na_nb1
authentication_on na_nb2
authentication_onsk_verify
secrecy_ofsec_smla, sec_smlb
authentication_onsmla_verify,smlb_verify
end goal

environment()
```

REFERENCES

[1] Trusted Computing Group. Trusted Platform Module (TPM) Specifications. Technical Report. [Online]. Available: https://www.trustedcomputinggroup.org/specs/TPM, 2008.

[2] Trusted Computing Group, TCG TPM Specification Version 1.2 Revision 103, Design Principles, Technical report, TCG, July 2007.

[3] A. Bottoni and D. Gianluca, "Credentials and beliefs in remote trusted platforms attestation," in *Proc. International Symposium on a World of Wireless, Mobile and Multimedia Networks*, IEEE, University of Pisa, Italy, 2006.

[4] D. Schellekens, B. Wyseur, and B. Preneel, "Remote attestation on legacy operating system with trusted platform modules," *Elsevier Science of Computer Programming*, vol. 74, no. 1-2, pp. 13-22, 2008.

[5] K. Dietrich, "A secure and reliable platform configuration change reporting mechanism for trusted IEEE computing enhanced secure channels," presented at the 9th International Conference for Young Computer Scientists, 2008.

[6] K. Goldman, R. Perez, and R. Sailer, "Linking remote attestation to secure tunnel endpoints," in *First ACM Workshop on Scalable Trusted Computing*, New York, ACM, November 2006, pp. 21-24.

[7] Y. Gasmi, A. R. Sadeghi, P. Stewin, M. Unger, and N. Asokan, "Beyond secure channels,"in *Proc. ACM Workshop on Scalable Trusted Computing*, 2007.

[8] F. Armknecht, Y. Gasmi, A-R. Sadeghi, P. Stewin, M. Unger, G. Ramunno, and D. Vernizzi, "An efficient implementation of trusted channels based on open SSL," in *Proc. 3rd ACM Workshop on Scalable Trusted Computing*, New York, ACM, 2008, pp. 41-50.

[9] F. Stumpf, O. Tafreschi, P. Röder, and C. Eckert, "A robust integrity reporting protocol for remote attestation," presented at Second Workshop on Advances in Trusted Computing, Tokyo, Japan, November 2006.

[10] A. R. Sadeghi and C. Stueble. "Property-based attestation for computing platforms: Caring about properties, not mechanisms," in *Proc. Workshop on New Security Paradigms*, ACM, 2004, pp. 66-77.

[11] L. Zhou and Z. Zhang, "Trusted channels with password-based authentication and TPM-based attestation," in *Proc. International Conference on Communications and Mobile Computing*, 2010, vol. 1, pp. 223-227.

[12] A. R. Sadeghi, "Trusted computing–special aspects and challenges," in *Proc. SOFSEM 2008: Theory and Practice of Computer Science*, Lecture Notes in Computer Science, 2008, vol. 4910, pp. 98-117.

[13] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proc. 11th ACM Conference on Computer and Communications Security*, New York, 2004, pp. 132-145.

[14] J. Farris, "Remote attestation," presented at Urbana-Champaign, University of Illinois, Dec 6, 2005,

[15] L. Q. Chen, R. Landfermann, H. Löhr, M. Rohe, A-R. Sadeghi, C. Stueble, and Horst Görtz, "A protocol for property-based attestation," in *Proc. First ACM workshop on Scalable Trusted Computing*, New York, 2006, pp. 7-16.

[16] J. Camenisch, "Direct anonymous attestation: Achieving privacy in remote authentication," Technical report, Information Security Colloquium, IBM Research, Zurich Research Laboratory, 2004.

[17] J. Camenisch, "Better privacy for trusted computing platforms," Technical Report, IBM Research, Zurich Research Laboratory, 2005.

[18] J. Reid, J. M. G. Nieto, and Ed Dawson, "Privacy and trusted computing," in *Proc. 14th International Workshop on Database and Expert Systems Application*, *IEEE*, 2003.

[19] F. Stumpf, "Leveraging attestation techniques for trust establishment in distributed systems," Ph.D dissertation, Department of Computer Science, Technische Universitat Darmstadt, Germany, 2010.

[20] R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," presented at 13th USENIX Security Symposium, IBM T. J. Watson Research Center, 2004.

[21] S. Goldwasser, S. Micali, and C. Racko, "The knowledge complexity of interactive proofs," *SIAM J. Computer*, vol. 18, no. 1, pp. 186-208, 1989.

[22] Automated Validation of Infinite-State Systems. (2002). AVISPA v1.1 User Manual, Document version: 1.1 edn., Information Society Technologies Programme (1998-2002). [Online]. Available: www.avispa-project.org

[23] The AVISPA Team. (2006). HLPSL Tutorial-Beginners Guide to Modelling and Analysing Internet Security Protocols, 1st edn., European Community under the Information Society Technologies Programme (1998-2002). [Online]. Available: www.avispa-project.org

[24] D. Dolev, and A. C. Yao, "On the security of public key protocols," presented at the IEEE 22nd Annual Symposium on Foundations of Computer Science, 1981.

[25] Y. Gloucher, T. Genet, and E. Houssay, "SPAN a security protocol animator for AVISPA," INRIA/IRISA LANDE Project, 2008.

NorazahAbd Aziz is Senior Researcher at MIMOS Berhad since 2001. She graduated from the University Technology Malaysia with a bachelor's degree in Computer Science in 2003. Currently, she is pursuing her Masters degree at UPM. Her research focus is on utilisation of Trusted Platform module, specificallyon attestation protocol.

Assoc. Prof. Dr. NurIzuraUdzir is an academic staff at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM) since 1998. She received her Bachelor of Computer Science (1995) and Master of Science (1998) from UPM, and her PhD in Computer Science from theUniversityof York, UK (2006). She is a member of IEEE Computer Society and a Committee Member of Information Security Professionals Association of Malaysia (ISPA.my). Her areas of specialization are access control, secure operating systems, intrusion detection systems, coordination models and languages, and distributed systems. She is currently the Leader of the Information Security Group at the faculty.

Professor DrRamlanMahmod obtained his degree in Computer Science from Michigan State University, USA and his Master in Computer Science from Central Michigan University, USA. His PhD is in Artificial Intelligence from Bradford University, United Kingdom. He has been a lecturer atUniversiti Putra Malaysia1985 and is currently the Dean of Computer Science Faculty, Universiti Putra Malaysia. He was seconded to MIMOS Berhad for two years from 2008 -2009 to help R&D in Trusted Computing and Information Security. He has published more than 75 journal papers and more than 110 articles in conference proceedings. He has filed 10 patents and holding more than 10 software copyrights. More than 25 PhD and Master student graduated under his supervision and currently supervising and co-supervising more than 15 PhD and Master student. His current research interest is Information Security especially in Cryptographic Algorithms, Steganography, Digital Forensics and Trusted Computing.