Universally Composable secure TNC protocol based on IF-T binding to TLS

(Full Version)*

Shijun Zhao, Qianying Zhang, Yu Qin, and Dengguo Feng

Institute of Software Chinese Academy of Sciences,
ISCAS, Beijing, China.
{zqyzsj@gmail.com},{zhangqy,qin_yu,fengdengguo@tca.iscas.ac.cn}

Abstract. Trusted Network Connect (TNC) requires both user authentication and integrity validation of an endpoint before it connects to the internet or accesses some web service. However, as the user authentication and integrity validation are usually done via independent protocols, TNC is vulnerable to the Man-in-the-Middle (MitM) attack. This paper analyzes TNC which uses keys with Subject Key Attestation Evidence (SKAE) extension to perform user authentication and the IF-T protocol binding to TLS to carry integrity measurement messages in the Universally Composable (UC) framework. Our analysis result shows that TNC using keys with SKAE extension can resist the MitM attack. In this paper, we introduce two primitive ideal functionalities for TNC: an ideal dual-authentication certification functionality which binds messages and both the user and platform identities, and an ideal platform attestation functionality which formalizes the integrity verification of a platform. We prove that the SKAE extension protocol and the basic TCG platform attestation protocol, both of which are defined by TCG specifications, UC-realizes the two primitive functionalities respectively. In the end, we introduce a general ideal TNC functionality and prove that the complete TNC protocol, combining the IF-T binding to TLS which uses keys with SKAE extension for client authentication and the basic TCG platform attestation platform protocol, securely realizes the TNC functionality in the hybrid model.

Key words: Universally Composable security, Trusted Network Connect, SKAE, TLS

1 Introduction

Many security solutions have been introduced to protect computers from attacks in the network, such as firewalls, virus scan engines and intrusion detection systems. However, as more and more security incidents ascend in numbers, these traditional solutions seem to be not sufficient to counter the current attacks. TNC, an open network access architecture enabling the network operators to authenticate the identity of the platform and perform the integrity verification of the platform before it connects to the network, is promoted and standardized by TCG to build a clean network environment. TNC aims to ensure that the integrity status of all the endpoints in the network are safe. The integrity information of an endpoint is collected and stored in a cost-effective, tamper-resistant Trusted Platform Module (TPM).

* An extended abstract of this paper appears in NSS'14.

When an endpoint wants to connect to the network or access some web service, it first calls the IF-T protocol [14, 18] to establish a mutually authenticated secure channel with the TNC server, then it runs platform attestation protocol to attest its integrity status to the TNC server. The platform attestation protocol messages are transported in the established secure channel. However, Askan et al. [1] find that running an authentication protocol in a tunneling protocol is vulnerable to Man-in-the-Middle (MitM) attacks, and such attacks can apply to TNC using IF-T protocol. To prevent MitM attacks, TCG promotes the Subject Key Attestation Evidence (SKAE) extension, which enables a cryptographic binding of a platform identity key (which signs integrity information in the platform attestation protocol) with a user certificate. Both the user identity and SKAE extension should be authenticated in the user authentication. The SKAE extension implies that the user authentication and platform attestation happen on the same platform, so MitM attacks won't work.

The UC framework defines the security goal of a protocol by an ideal functionality, which acts as a trusted third party. A good property of UC is the composability: a protocol π communicating with an ideal functionality $\mathcal{F}$ is identical to π calling a subroutine protocol ρ if ρ securely realizes π in the UC framework. This property suits the analysis of layered protocols very well: the high layer protocol invokes the ideal functionality realized by the lower layer protocol without considering the implementation details of the lower layer protocol. TNC is a layered architecture: the bottom layer is IF-T protocol, which establishes a mutual authentication secure channel, and the top layer is platform attestation protocol. So UC suits the analysis of TNC very well, and a proved TNC functionality will benefit the analysis of the protocols above TNC.

1.1 Related Work

To the best of our knowledge, few works on the formal analysis of TNC have been done since the publication of TNC. Zhang et al. [21] provide the first ideal TNC functionality $\mathcal{F}_{\text{TNC}}$, and analyze the EAP-TNC attestation protocol with Diffie-Hellman Pre-Negotiation (D-H PN) [14]. They find a MitM attack on the D-H PN EAP-TNC protocol and patch it by authenticating the Diffie-Hellman keys. However, their analysis is based on the assumption that the tunneled EAP protocol has provided an ideal mutually authenticated secure channel functionality. Some analysis of TNC using the symbolic logic method is proposed recently. Zhang et al. [22] propose a computationally sound symbolic analysis of D-H PN EAP-TNC protocol but not the complete TNC protocol. Xiao et al. [20] analyze the authentication property of the complete TNC protocol based on the IF-T protocol binding to TLS in their extended strand space model. They find that the complete TNC protocol can resist MitM attacks in the case the user is authenticated through a certificate with SKAE extension during the TLS setup phase. This result is in accord with our analysis result in the UC framework.

Until now, most basic ideal cryptography functionalities, such as public key encryption, digital signature, authentication communication, key exchange, and secure channel, have been realized in the UC or similar framework (see, e.g., [3, 4, 8, 11]. Gajek et al. [6] presented the first security analysis of the TLS protocol in the UC framework. They analyzed the key exchange functionality realized by TLS handshake and the secure channel functionality realized by the complete TLS protocol. In our analysis, the functionality provided by IF-T protocol binding to TLS is a variant of secure channel functionality which provides not only user authentication but also platform authentication. Our analysis of IF-T is based on the work of [6].

1.2 Our Contributions

In this paper, we investigate the complete TNC protocol in the UC framework. We adopt the composability of the UC framework in our analysis. We first separate the Complete TNC protocol into IF-T protocol and platform attestation protocol. Then we analyze the two protocols separately. Finally, we analyze the complete TNC protocol in the hybrid model. The following lists our contributions more specifically.

1. We introduce two primitive ideal functionalities for TNC. The first is a dual-authentication certification functionality $\mathcal{F}_{\text{D-Cert}}$ that authenticates both the user and platform identity. This functionality is necessary in the analysis of TNC as authenticating both the user and platform is a basic security policy requirement in TNC. The second is a platform attestation functionality $\mathcal{F}_{\text{P-A}}$ that captures the security requirement of the platform integrity status validation. These two primitive functionalities enable to analyze complex protocols based on SKAE extension and platform attestation in a modular way, and simplify the analysis.
2. We consider the realization of $\mathcal{F}_{\text{D-Cert}}$ and $\mathcal{F}_{\text{P-A}}$. We prove that 1) the SKAE extension creation and processing protocol (which we call SKAE-EX for short), defined in the SKAE specification [13], securely realizes $\mathcal{F}_{\text{D-Cert}}$; and 2) the basic TCG platform attestation protocol realizes $\mathcal{F}_{\text{P-A}}$. On the basis of $\mathcal{F}_{\text{D-Cert}}$, we show that the IF-T protocol binding to TLS which uses keys with SKAE extension for client authentication (which we call IF-TLS-SKAE for short) realizes a dual-authentication secure channel functionality $\mathcal{F}_{\text{D-SC}}$.
3. We introduce a general TNC functionality $\mathcal{F}_{\text{TNC}}$ and show that the complete TNC protocol, combining IF-TLS-SKAE and the basic TCG platform attestation protocol, UC-realizes $\mathcal{F}_{\text{TNC}}$.

1.3 Organization

Section 2 gives a brief overview of the background of this paper, i.e., TNC and the UC framework. Section 3 introduces the dual-authentication certification functionality $\mathcal{F}_{\text{D-Cert}}$, and show that 1) the SKAE-EX protocol securely realizes $\mathcal{F}_{\text{D-Cert}}$, and 2) the IF-TLS-SKAE protocol securely realizes the dual-authentication secure channel functionality $\mathcal{F}_{\text{D-SC}}$. Section 4 introduces the platform attestation functionality $\mathcal{F}_{\text{P-A}}$ and proves that the basic TCG platform attestation protocol realizes $\mathcal{F}_{\text{P-A}}$. Section 5 designs the general TNC functionality $\mathcal{F}_{\text{TNC}}$ and analyzes the complete TNC protocol in hybrid model. Section 6 concludes this work.

2 Background

This section briefly describes the TNC architecture and the UC framework on which our analysis is based.

2.1 TNC in a nutshell

TNC is an open network architecture that enables network operators to assess the integrity status of endpoints and verify the user and platform identities of endpoints in order to determine endpoints whether to grant access to the network or web services. The integrity information can be collected by the TCG-based integrity measurement architectures [12, 9] and stands for the security status of

the current system. The integrity information is stored in a TPM, which cannot be compromised by a potentially malicious host system. In order to validate the integrity information of an endpoint, the existence and genuineness of a TPM should be authenticated, i.e., the platform identity authentication.

TCG defines a series of specifications for TNC. The interoperability architecture specification [19] describes how TNC architecture can be implemented and integrated with existing network access control mechanisms such as 802.1X [7]. The IF-T specifications [14, 18] define how IF-T can be implemented over other lower layer protocols such as tunneled Extensible Authentication Protocol (EAP) and TLS. The IF-TNCCS messages, carrying the integrity measurement messages, are transported in the IF-T protocol, and its message format is defined in the IF-TNCCS specification [17]. The following gives a brief TNC flow based on the IF-T protocol binding to TLS:

0. Requirement. Before running TNC, the Network Access Requestor (NAR), which represents the endpoint that wants to get access to a TNC protected service, is already on the network thus has an IP address assigned.
1. TLS Setup. NAR establishes a TLS session with the Network Access Authority (NAA), who assesses NAR and decides whether NAR should be granted to access. In this phase, NAR must validate the certificate of NAA, e.g., NAA is authenticated to NAR. The NAR might use a client certificate with SKAE extension for client authentication, and this is the case analyzed in this paper.
2. User Authentication. If NAR isn't authenticated to NAA in the TLS setup phase, user authentication must be performed after TLS setup phase over the established TLS session. The second version of [18] defines that the user authentication must be performed in the Simple Authentication and Security Layer (SASL) [10] framework, and some standards-based authentication mechanisms are provided in SASL.
3. Platform Attestation. In this phase, the IF-T session is available. NAA assesses the integrity of NAR in the IF-T session established in the above phases. The assessment messages are encapsulated in IF-TNCCS messages. Finally, NAA decides whether to grant NAR to access the protected service.

2.2 The UC framework

The UC framework [2] is a kind of formal method for the modular design and analysis of multi-party protocols. UC defines an additional entity called the environment $\mathcal{Z}$. It feeds arbitrary inputs to the parties and the adversary, then collects the outputs from the parties and the adversary. $\mathcal{Z}$ interacts with two worlds: the real world and the ideal world. The real world is composed of honest parties running a real protocol and an adversary $\mathcal{A}$ which controls the communication between parties and may corrupt honest parties. The ideal world is composed of dummy parties and a simulator $\mathcal{S}$. The dummy parties simply receive inputs from $\mathcal{Z}$, and forward them to the ideal functionality $\mathcal{F}$, which is a trusted party and performs the ideal cryptographic task. After completing the cryptographic task, $\mathcal{F}$ hands the desired outputs to the dummy parties. The security notion of UC is defined in an indistinguishable way as follows.

Definition 1. *A protocol π UC-realizes (or emulates) an ideal functionality $\mathcal{F}$ if for any adversary $\mathcal{A}$ in the real world, there exists an adversary (simulator) $\mathcal{S}$ in the ideal world such that for any environment $\mathcal{Z}$, the probability that $\mathcal{Z}$ distinguishs whether it is interacting with the real protocol π and $\mathcal{A}$ or with the ideal functionality $\mathcal{F}$ and $\mathcal{S}$ is at most a negligible probability, i.e.,* $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} \approx \mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

A good feature of UC framework is its composition theorem.

Definition 2 (Composition theorem). *Let π be a protocol that uses subroutine calls to an ideal functionality f. We call π is an f-hybrid protocol. If protocol ρ realizes f, then the composed protocol $\pi^{\rho/f}$, in which each invocation of f is replaced by an invocation of ρ, UC-realizes π. Another way of saying it is, $\pi^{\rho/f}$ UC-realizes π in f-hybrid model, i.e.,* $\mathrm{EXEC}_{\pi^{\rho/f},\mathcal{A},\mathcal{Z}} \approx \mathrm{EXEC}^{f}_{\pi,\mathcal{S},\mathcal{Z}}$.

3 SKAE Certification Functionality

Functionality $\mathcal{F}_{\text{D-Cert}}$

Signature Generation: Upon receiving a value (Sign, sid, m) from party S, verify that $sid = (U, P, s)$ for some s (We denote by U the user identity and P the platform identity). If not, then ignore the request. Else send (Sign, sid, m) to the adversary. Upon receiving $(\mathsf{Signature}, sid, m, \sigma)$ from the adversary, verify that no entry $(m, \sigma, 0)$ is recorded. If it is, then output an error message to S and halt. Else, output $(\mathsf{Signature}, sid, m, \sigma)$ to S, and record the entry $(m, \sigma, 1)$.

Signature Verification: Upon receiving a value $(\mathsf{Verify}, sid, m, \sigma)$ from some party S', hand $(\mathsf{Verify}, sid, m, \sigma)$ to the adversary. Upon receiving $(\mathsf{Verified}, sid, m, \phi)$ from the adversary, do:

1. If $(m, \sigma, 1)$ is recorded then set $f = 1$.
2. Else, if the signer is not corrupted, and no entry $(m, \sigma', 1)$ for any σ' is recorded, then set $f = 0$ and record the entry $(m, \sigma, 0)$.
3. Else, if there is an entry (m, σ, f') recorded, then set $f = f'$.
4. Else, set $f = \phi$, and record the entry (m, σ', ϕ).

Output $(\mathsf{Verified}, sid, m, f)$ to S'.

Fig. 1. The Dual-authentication Certification Functionality, $\mathcal{F}_{\text{D-Cert}}$

This section presents the dual-authentication certification functionality $\mathcal{F}_{\text{D-Cert}}$ authenticating both the user and platform identity. We analyze the SKAE extension creation and processing protocol (SKAE-EX) defined in [13]. SKAE-EX runs in a setting of the existence of 1) an Attestation Identity Key (AIK), which certifies that the user key with SKAE extension is under the protection of the TPM where AIK resides, 2) a trusted certificate authority that registers user identities together with public keys. We formalize the global assumptions by utilizing the certification functionality $\mathcal{F}_{\text{Cert}}$ and the certificate authority functionality $\mathcal{F}_{\text{CA}}$ presented in [3]. We assume that the signature of the user key with SKAE extension is secure, so we add the signature functionality $\mathcal{F}_{\text{SIG}}$ presented in [3] to our global assumptions. Our analysis shows that the SKAE extension creation and processing protocol defined in [13] UC-realizes $\mathcal{F}_{\text{D-Cert}}$ in the $(\mathcal{F}_{\text{Cert}}, \mathcal{F}_{\text{CA}}, \mathcal{F}_{\text{SIG}})$-hybrid model. We don't describe $\mathcal{F}_{\text{Cert}}$, $\mathcal{F}_{\text{CA}}$, and $\mathcal{F}_{\text{SIG}}$ in this paper, and for readers who are interested in them please consult [3] for details.

It's reasonable to assume that the AIK certification capability provided by TPM (see [15, 16] for details) securely realizes $\mathcal{F}_{\text{Cert}}$, as the AIK certification with the help of a PrivacyCA [5] is the same as the CAS protocol [3] which UC-realizes $\mathcal{F}_{\text{Cert}}$.

3.1 The Dual-authentication Certification Functionality, $\mathcal{F}_{\text{D-Cert}}$

The ideal dual-authentication certification functionality, $\mathcal{F}_{\text{D-Cert}}$ is presented in Figure 1. It is similar to $\mathcal{F}_{\text{Cert}}$, except that it binds a signature for a message with the user identify and the platform identify, which are encoded in the *sid*. The dual binding relationship is critical in the complete TNC protocol, and is one of the two essential approaches used in TNC to prevent MitM attacks. This binding relationship shows that the key used in the user authentication is protected by some TPM which is identified by an AIK. If the later platform attestation protocol uses the same AIK, then the user authentication and the platform attestation must operate in the same platform, which prevents MitM attacks.

3.2 Analysis of SKAE-EX

We show that the SKAE-EX protocol UC-realizes $\mathcal{F}_{\text{D-Cert}}$ with the aid of an AIK certified by a trusted PrivacyCA and ideally authenticated communication with a "trusted certification authority". The certified AIK is formalized as $\mathcal{F}_{\text{Cert}}$, and this formalization is reasonable as we have explained that the AIK certification capability is the same as the CAS protocol[3], which UC-realizes $\mathcal{F}_{\text{Cert}}$. The trusted certification authority assumption is formalized as $\mathcal{F}_{\text{CA}}$ which registers user identities with public values. We stress that $\mathcal{F}_{\text{CA}}$ doesn't check the possession of secret key corresponding to the registered public value, which models most practical CAs.

The formal description of SKAE-EX protocol is given in Figure 2.

Theorem 1. *Protocol SKAE-EX securely realizes functionality $\mathcal{F}_{\text{D-Cert}}$ in the $(\mathcal{F}_{\text{Cert}}, \mathcal{F}_{\text{CA}}, \mathcal{F}_{\text{SIG}})$-hybrid model.*

Proof. Let $\mathcal{A}$ be an adversary that interacts with parties running SKAE-EX in the $(\mathcal{F}_{\text{Cert}}, \mathcal{F}_{\text{CA}}, \mathcal{F}_{\text{SIG}})$-hybrid model. We construct an ideal adversary (simulator) $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish whether it's interacting with $\mathcal{A}$ and SKAE-EX in the real world or it's interacting with $\mathcal{S}$ and $\mathcal{F}_{\text{D-Cert}}$ in the ideal world. $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$ and simulates for this copy interaction with parties running SKAE-EX. All messages from $\mathcal{Z}$ to $\mathcal{A}$ and back are forwarded. The operations of $\mathcal{S}$ are shown below:

Simulating signature generation. When $\mathcal{S}$ receives a message (Sign, sid, m) from $\mathcal{F}_{\text{D-Cert}}$, where $sid = (U, P, s)$ and both U and P are not corrupted, it proceeds as follows:

1. If this is the first time that U generates a signature, then simulates for $\mathcal{A}$:
   a) The process of key generation: send $(\mathsf{KeyGen}, (U, s))$ to $\mathcal{A}$ (in the name of $\mathcal{F}_{\text{SIG}}$), and obtains $(\mathsf{Verification\ key}, (U, s), v)$ from $\mathcal{A}$.
   b) The AIK certification: send $(\mathsf{Sign}, (P, s), v)$ to $\mathcal{A}$, and obtain $(\mathsf{Signature}, (P, s), v, \sigma_{\text{AIK}})$ from $\mathcal{A}$, and record (v, σ_{AIK}).
   c) The registration of user key: send to $\mathcal{A}$ the message $(\mathsf{Registered}, U, v, P, \sigma_{\text{AIK}})$ in the term of $\mathcal{F}_{\text{CA}}$, and record $(U, v, P, \sigma_{\text{AIK}})$ after receiving Ok from $\mathcal{A}$.
2. Simulate for $\mathcal{A}$ the processing of signing m. That is, in the name of $\mathcal{F}_{\text{SIG}}$, send to $\mathcal{A}$ the message $(\mathsf{Sign}, (U, s), m)$. After receiving $(\mathsf{Signature}, (U, s), m, \sigma)$ from $\mathcal{A}$, records the pair (m, σ) and sends $(\mathsf{Signature}, sid, m, \sigma)$ to $\mathcal{F}_{\text{D-Cert}}$.

$\mathcal{S}$ simulates the interaction of a party whose user identity is corrupted or platform identity is corrupted (or both are corrupted) as follows:

[3] Readers who are interested in the details of the CAS protocol please consult [3].

1. If the user identity U is corrupted, the process of key generation and signature generation is similar to the uncorrupted user, except that $\mathcal{A}$ can register a different public value v' to $\mathcal{F}_{\text{CA}}$. Note that the simulation is still valid, as if the user is corrupted, $\mathcal{F}_{\text{SIG}}$ follows the instruction of $\mathcal{A}$ to verify a signature.
2. If the platform identity P is corrupted, all that $\mathcal{S}$ has to do is to simulate for $\mathcal{A}$ the interaction with $\mathcal{F}_{\text{Cert}}$. That is, whenever a party sends a messages $(\mathsf{Sign}, (P, s), v')$ to $\mathcal{F}_{\text{Cert}}$, $\mathcal{F}_{\text{Cert}}$ queries $\mathcal{A}$ to get a signature σ_{AIK}.

Simulating signature verification. When $\mathcal{S}$ receives a verification request made by an uncorrupted party S' (both U' and P' are uncorrupted) from $\mathcal{F}_{\text{D-Cert}}$, $\mathcal{S}$ proceeds as follows:

1. If this is the first verification request from S', then simulates for $\mathcal{A}$:
   a) The exchange between S' and $\mathcal{F}_{\text{CA}}$. That is, send a message $(\mathsf{Retrieve}, (U, P), S')$ in the name of $\mathcal{F}_{\text{CA}}$. When $\mathcal{A}$ responds with Ok, if $\mathcal{F}_{\text{CA}}$ has a tuple $(U, v, P, \sigma_{\text{AIK}})$ recorded then record this tuple for the simulated S', else record a $(U, P, \perp)$.
   b) The AIK verification. That is, $\mathcal{S}$ simulates for $\mathcal{A}$ following the logic of $\mathcal{F}_{\text{Cert}}$: when $\mathcal{F}_{\text{Cert}}$ decides to send $(\mathsf{Verify}, (P, s), v, \sigma_{\text{AIK}})$ to $\mathcal{A}$ then send this message to $\mathcal{A}$ and obtain $(\mathsf{Verified}, (P, s), m, \phi)$ from $\mathcal{A}$; when $\mathcal{F}_{\text{Cert}}$ decides to output to S', if neither U nor P is corrupted then follow the logic of $\mathcal{F}_{\text{Cert}}$ to output $(\mathsf{Verified}, (P, s), v, f)$ to S' (If (v, σ_{AIK}) is recorded previously, then $f = 1$, else $f = 0$), else output $(\mathsf{Verified}, (P, s), v, \phi)$ to S'.
2. The process of signature verification. Denote the message from $\mathcal{F}_{\text{D-Cert}}$ by $(\mathsf{Verify}, sid, m, \sigma)$ where $sid = (U, P, s)$. Forward $(\mathsf{Verify}, (U, s), m, \sigma, v)$ to $\mathcal{A}$ in the name of $\mathcal{F}_{\text{SIG}}$, and forward $\mathcal{A}$'s response to $\mathcal{F}_{\text{D-Cert}}$.

Simulating party corruptions. When $\mathcal{A}$ corrupts the user or platform of some party, $\mathcal{S}$ corrupts the corresponding user or platform in the ideal world, and provides the obtained information to $\mathcal{A}$. This has no effect on the simulation, as the real world protocol maintains no secret state.

It can be readily seen that the view of of $\mathcal{Z}$ in an interaction of $\mathcal{A}$ and SKAE-EX protocol is distributed identically to the view of $\mathcal{Z}$ in an interaction with and $\mathcal{S}$ and $\mathcal{F}_{\text{D-Cert}}$ in the ideal world.

3.3 Dual-authentication Secure Channel Functionality

We describe our dual-authentication secure channel functionality $\mathcal{F}_{\text{D-SC}}$ in Figure 3. $\mathcal{F}_{\text{D-SC}}$ enables the responder of the secure channel to authenticate both the user and the platform identify of the initiator, and guarantees that the adversary gains no more information than some side channel information about the transmitted plaintext m, such as the length of m. The leakage information is expressed by a leakage function $l(m)$.

We argue that the IF-T protocol binding to TLS using keys with SKAE extension for client authentication (IF-TLS-SKAE) securely realizes $\mathcal{F}_{\text{D-SC}}$. Gajek et al. [6] has presented the security analysis of the complete TLS protocol, combining Handshake and Record Layer, in the UC framework. They first showed that the master key generation subroutines in Handshake protocol UC-realize the key exchange functionality $\mathcal{F}_{\text{KE}}$ given the traditional certification functionality $\mathcal{F}_{\text{Cert}}$, then that the complete TLS protocol framework securely realizes secure channel functionality $\mathcal{F}_{\text{SC}}$ in the $\mathcal{F}_{\text{KE}}$-hybrid model. Since 1) we have proved that user keys with SKAE extension UC-realizes the dual-authentication functionality, and 2) the IF-TLS-SKAE protocol is a complete TLS protocol using keys with SKAE extension for client authentication, we directly get theorem 2. The proof can be easily got following the proof of [6].

Theorem 2. *The IF-TLS-SKAE protocol securely realizes functionality $\mathcal{F}_{\text{D-SC}}$.*

Protocol SKAE-EX

Signature Protocol: When activated with input (Sign, sid, m), party S does:

1. S verifies that $sid = (U, P, s)$ for some identifier s; if not then the input is ignored. (That means that S verifies that it's the legitimate user and platform for this sid)
2. If this is the first activation then S does:
   a) Generates a user key using its TPM. We model the generation by utilizing $\mathcal{F}_{\mathrm{SIG}}$, i.e., sends $(\mathsf{KeyGen}, (U, s))$ to $\mathcal{F}_{\mathrm{SIG}}$.
   b) Once S obtains a key from the TPM, i.e., receiving $(\mathsf{Verification\ key}, (U, s), v)$ from $\mathcal{F}_{\mathrm{SIG}}$, it invokes the AIK of the TPM, which is bound to the platform identity P, to certify that v comes from a genuine TPM identified as P. We model the AIK certification by sending $(\mathsf{Sign}, (P, s), v)$ to $\mathcal{F}_{\mathrm{Cert}}$.
   c) After obtaining a certification from AIK, i.e., receiving $(\mathsf{Signature}, (P, s), v, \sigma_{\mathrm{AIK}})$ from $\mathcal{F}_{\mathrm{Cert}}$, S sends $(\mathsf{Register}, U, v, P, \sigma_{\mathrm{AIK}})$ to $\mathcal{F}_{\mathrm{CA}}$, who doesn't perform any check and just record $(U, v, P, \sigma_{\mathrm{AIK}})$ if this is the first request from S. From then on, the user key is bound to the user identity U.
3. S sends $(\mathsf{Sign}, (U, s), m)$ to $\mathcal{F}_{\mathrm{SIG}}$. Upon receiving $(\mathsf{Signature}, (U, s), m, \sigma)$ from $\mathcal{F}_{\mathrm{SIG}}$, S outputs $(\mathsf{Signature}, sid, m, \sigma)$.

Verification Protocol: When activated with input $(\mathsf{Verify}, sid, m, \sigma)$, where $sid = (U, P, s)$, party S' does:

1. S' check whether it has a tuple $(U, v, P, \sigma_{\mathrm{AIK}})$ recorded. If not, then S' sends $(\mathsf{Retrieve}, U, P)$ to $\mathcal{F}_{\mathrm{CA}}$, and obtains a response $(\mathsf{Retrieve}, U, v, P, \sigma_{\mathrm{AIK}})$. If $v = \perp$ then S' rejects the signature, i.e., it outputs $(\mathsf{Verified}, sid, m, 0)$. Else it records $(U, v, P, \sigma_{\mathrm{AIK}})$.
2. S' uses its user key to verify σ, i.e., sends $(\mathsf{Verify}, (U, s), m, \sigma, v)$ to $\mathcal{F}_{\mathrm{SIG}}$, and obtains a response $(\mathsf{Verified}, (U, s), m, f)$. If $f = 0$, S' outputs $(\mathsf{Verified}, sid, m, 0)$.
3. S' uses its AIK to verify the SKAE extension, i.e., sends $(\mathsf{Verify}, (P, s), v, \sigma_{\mathrm{AIK}})$ to $\mathcal{F}_{\mathrm{Cert}}$, and obtains a response $(\mathsf{Verified}, (P, s), v, f)$. If $f = 0$, S' outputs $(\mathsf{Verified}, sid, m, 0)$. Else, S' outputs $(\mathsf{Verified}, sid, m, 1)$.

Fig. 2. The SKAE-EX protocol for realizing $\mathcal{F}_{\text{D-Cert}}$

Functionality $\mathcal{F}_{\text{D-SC}}$

$\mathcal{F}_{\text{D-SC}}$ proceeds as follows, running with some initiators $(U_1, P_1), \ldots, (U_n, P_n)$ and some responders $S_1, \ldots, S_n$, and parameterized by a leakage function $l : \{0,1\}^* \to \{0,1\}^*$.

1. Upon receiving an input (Establish-session, sid, S_j, initiator) from some initiator (U_i, P_i), send $(sid, (U_i, P_i), S_j)$ to the adversary, and wait to receive an input (Establish-session, sid, (U_i, P_i), responder) from S_j. Once receiving this input, set a boolean variable active. Say that (U_i, P_i) and S_j are the partners of this session.
2. Upon receiving an input (Send, sid, m) from one of the partners of this session, and if active is set, send (Receive, sid, m) to the other partner and (Sent, sid, $l(m)$) to the adversary.

Fig. 3. The Dual-authentication Secure Channel Functionality, $\mathcal{F}_{\text{D-SC}}$

4 Analysis of Platform Attestation

In this section we first introduce the ideal platform attestation functionality $\mathcal{F}_{\text{P-A}}$ which formalizes the PCR-based platform attestation proposed by TCG, and then show that the TCG platform attestation protocol securely realizes $\mathcal{F}_{\text{P-A}}$ given an incorruptible $\mathcal{F}_{\text{Cert}}$ functionality. The incorruptibility models the protection capability provided by TPM, a tamper-resistant hardware token.

4.1 Platform Attestation Functionality

We first describe the ideal platform attestation functionality $\mathcal{F}_{\text{P-A}}$ modeling a prover to attest its PCR information to a verifier informally. See Figure 4 for a precise definition.

Functionality $\mathcal{F}_{\text{P-A}}$

$\mathcal{F}_{\text{D-SC}}$ proceeds as follows, running with some provers $P_1, \ldots, P_n$ and some verifiers $V_1, \ldots, V_n$, and every prover P_i is initialized with a acceptable PCR status PCR_i.

1. Upon receiving an input (Challenge, *sid*, P, verifier) from some verifier V, do:
   a) Send (sid, P, V) to the adversary, and receive an response (Ok, $nonce'$) from the adversary. If V is corrupted, then record $nonce'$, else generate a fresh *nonce* and record it. Then output (sid, n) to the adversary, V and P (n stands for the recorded nonce).
   b) Wait to receive an input (Attest, *sid*, V, prover) from P. Once receiving this input, output (SentPCR, *sid*, P, V) to the adversary.
2. Upon receiving (SendPCR, *sid*, V, (n, PCR')) from the adversary, check whether (n, PCR') is recorded. If it's recorded, output (SentPCR, *sid*, P, (n, PCR')) to V. Else, record the pair (n, PCR), and output (SentPCR, *sid*, P, (n, PCR)) to V.
3. Upon receiving an input (Corrupt, *sid*, V, Verifier) from the adversary, mark V as corrupted.
4. Upon receiving an input (Corrupt, *sid*, P, Prover) from the adversary, change PCR to $\overline{PCR}$.

Fig. 4. The Platform Attestation Functionality, $\mathcal{F}_{\text{P-A}}$

At beginning, each honest prover is initialized with a PCR status which is acceptable for a honest verifier. Through the Challenge interface, a verifier declares that it's willing to verify the PCR status of a prover. This request is forwarded to the prover and the adversary. If the verifier is corrupted, $\mathcal{F}_{\text{P-A}}$ receive a nonce from the adversary and record it. If the verifier is not corrupted, $\mathcal{F}_{\text{P-A}}$ generates a fresh random nonce itself and record it. The nonce is then output to the adversary, the verifier and the prover. Through the Attest interface, the requested prover provides its received nonce and declares that it wants to attest its PCR status to the verifier. After receiving the challenge and attest messages, $\mathcal{F}_{\text{P-A}}$ record the nonce and the current PCR status, then: 1) if the verifier is not corrupted, $\mathcal{F}_{\text{P-A}}$ returns the nonce and the current PCR status to the verifier, and 2) if the verifier is corrupted, and there exists a nonce and a PCR status in the record, $\mathcal{F}_{\text{P-A}}$ asks the adversary whether to return the previous or the current PCR status. Through the corruption interface, the adversary can corrupt any prover or verifier. If the adversary decides to corrupt the prover, the PCR of the prover will be changed to a unacceptable status. $\mathcal{F}_{\text{P-A}}$ captures the intuitive notion of TNC:

1. Once the platform is corrupted, which means that the adversary runs some malicious code on the platform, the PCR will record this code and change to a status which is not acceptable to a honest

verifier. This property captures the integrity measurement and storage capability of a platform equipped with a TPM.
2. Against the replay attack for honest parties. $\mathcal{F}_{\text{P-A}}$ sends not only the PCR status but also a fresh nonce aiming to prove that the PCR status is fresh.
3. If the verifier is corrupted, the platform attestation will not be reliable. That is, if the nonce is not fresh, then the adversary can mount a replay attack.

4.2 Analysis of TCG Platform Attestation Protocol

We show that the TCG platform attestation protocol (we call P-Attest for short), depicted in Figure 5, securely realizes $\mathcal{F}_{\text{P-A}}$ given $\mathcal{F}_{\text{Cert}}$ which models an AIK. In order to model the protection capability of TPM, corruption of a prover only changes the PCR status of the prover and the certification party in $\mathcal{F}_{\text{Cert}}$, which means that the AIK is incorruptible.

TCG Platform Attestation Protocol

Each prover is initialized a PCR status which is valid to the verifier.

1. Upon receiving an input (Challenge, sid, P, verifier), verifier V generates a fresh random nonce n, sends (sid, V, n) to P.
2. Upon receiving an input (Attest, sid, V, prover), prover P waits for a nonce n from verifier V. Upon receiving of (sid, V, n), P sets $sid' = (P, sid)$, sets $m = (n, PCR)$, sends (Sign, sid', m) to $\mathcal{F}_{\text{Cert}}$, obtains the response $(\mathsf{Signature}, sid', m, \sigma)$, and sends (sid, P, m, σ) to V.
3. Upon receiving (sid, P, m, σ), V sets $sid' = (P, sid)$, checks whether the nonce in m equals n, sets $m = (n, PCR)$, sends $(\mathsf{Verify}, sid', m, \sigma)$ to $\mathcal{F}_{\text{Cert}}$, and obtains a response $(\mathsf{Verified}, sid', m, \sigma, f)$. If $f = 1$ then V outputs $(\mathsf{SentPCR}, sid, \text{P}, \text{V}, (n, PCR))$ and halts, else B halts without output.

Fig. 5. The TCG Platform Attestation Protocol

Theorem 3. *The P-Attest protocol securely realizes functionality $\mathcal{F}_{\text{P-A}}$ in the $\mathcal{F}_{\text{Cert}}$-hybrid model.*

Proof. Let $\mathcal{A}$ be an adversary that interacts with parties running P-Attest in the $\mathcal{F}_{\text{Cert}}$-hybrid model. We construct an ideal world adversary (or simulator) $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish whether it's interacting with $\mathcal{A}$ and P-Attest in the real world or it's interacting with $\mathcal{S}$ and $\mathcal{F}_{\text{P-A}}$ in the ideal world. $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$ and simulates for this copy an interaction with parties running P-Attest. All messages from $\mathcal{Z}$ to $\mathcal{A}$ and back are forwarded. The operations of $\mathcal{S}$ are shown below:

Simulating the verifier. When an uncorrupted verifier V is activated with input (Challenge, sid, P, verifier), $\mathcal{S}$ obtains this input from $\mathcal{F}_{\text{P-A}}$ and waits for a message (sid, n) from $\mathcal{F}_{\text{P-A}}$. Then, $\mathcal{S}$ simulates a run of V and the interaction with $\mathcal{F}_{\text{Cert}}$ for $\mathcal{A}$.

1. $\mathcal{S}$ delivers $\mathcal{A}$ the message (sid, V, n) sent from P to V.
2. When receiving a message (sid, P, m, σ) where $m = (n, PCR)$ from $\mathcal{A}$, $\mathcal{S}$ simulates for $\mathcal{A}$ the interaction with $\mathcal{F}_{\text{Cert}}$. That's, send $(\mathsf{Verify}, sid' = (P, sid), m, \sigma)$ to $\mathcal{A}$ and record the response. If there exists a recorded signature (m, σ) (simulating $\mathcal{F}_{\text{Cert}}$), then $\mathcal{S}$ deliver the message $(\mathsf{SentPCR}, sid, P, V, (n, PCR))$ to V. Otherwise, deliver nothing.

Simulating the prover. When an uncorrupted prover P is activated with inputs (Attest, sid, V, prover), $\mathcal{S}$ obtains this input from $\mathcal{F}_{\text{P-A}}$ and waits for a message (sid, V, n) from $\mathcal{F}_{\text{P-A}}$. Then, $\mathcal{S}$ simulates a run of P and the interaction with $\mathcal{F}_{\text{Cert}}$ for $\mathcal{A}$: $\mathcal{S}$ sends to $\mathcal{A}$ the message (Sign, $sid' = (P, sid), m = (n, PCR)$) in the name of $\mathcal{F}_{\text{Cert}}$, and obtains a signature σ from $\mathcal{A}$. Next, $\mathcal{S}$ hands $\mathcal{A}$ the message (sid, P, m, σ) sent from P to V.
If the prover is corrupted, which means the PCR status of P is invalid, the simulation is similar to a uncorrupted party.

Simulating party corruptions. When $\mathcal{A}$ corrupts a party, $\mathcal{S}$ corrupts the corresponding party in the ideal world, and provides the obtained information to $\mathcal{A}$. This has no effect on the simulation, as the real world protocol maintains no secret state.

The only potential case in which the view of $\mathcal{Z}$ in the real world and in the ideal world may differ is if the verifier receives a message (Verified, $sid', m, \sigma, f = 1$) from $\mathcal{F}_{\text{Cert}}$ in the real world, while $\mathcal{A}$ never delivered the message $(sid, P, m = (n, PCR), \sigma)$. But if $\mathcal{A}$ never delivered $(sid, P, m = (n, PCR), \sigma)$, then m was never signed by $\mathcal{F}_{\text{Cert}}$. So in this case, V would always receive (Verified, $sid', m, \sigma, f = 0$). In a conclusion, the simulation is perfect.

5 Analysis of Complete TNC

We first introduce our general ideal TNC functionality $\mathcal{F}_{\text{TNC}}$. We then analyze the complete TNC protocol, combining TCG attestation protocol and IF-TLS-SKAE protocol, and show that it realizes the ideal TNC functionality in the $(\mathcal{F}_{\text{D-SC}}, \mathcal{F}_{\text{P-A}})$-hybrid model. To show the usefulness of the dual-authentication feature provided by $\mathcal{F}_{\text{D-SC}}$, we show that the above simulation will not hold if $\mathcal{F}_{\text{D-SC}}$ is replaced with an usual secure channel functionality without dual-authentication feature.

5.1 A General TNC functionality

Our ideal TNC functionality is depicted in Figure 6. Our functionality $\mathcal{F}_{\text{TNC}}$ is much more general than the TNC functionality presented in [21] as our functionality models the IF-T protocol and the above IF-TNCCS protocol as secure channel and platform authentication functionality respectively, while the TNC functionality in [21] only models the IF-T binding to EAP protocol. Our TNC functionality captures the following TNC features:

1. User Authentication and Platform Authentication. $\mathcal{F}_{\text{D-SC}}$ authenticates both user and platform identities of connecting endpoints.
2. Platform Attestation. $\mathcal{F}_{\text{P-A}}$ abstracts the main goal of the IF-TNCCS protocol carried in the secure channel established by IF-T, that is, the platform integrity attestation.
3. Corrupt Platforms. The $\mathcal{F}_{\text{P-A}}$ models the corruption of a platform in TNC by marking the PCR status as invalid.
4. Corrupt Users. We model a relaxed network policy which mainly validates the integrity status reported by a genuine TPM and the platform identity, that is, "integrity information reported by the platform and by the proof-of-identity supplied by the platform". Even the user of an endpoint is corrupted, the endpoint is able to connect to the network if it has valid PCR status. However, a constrained policy can be easily modeled by changing our functionality.

Functionality $\mathcal{F}_{\text{TNC}}$

$\mathcal{F}_{\text{TNC}}$ proceeds as follows, running with some NARs and some NAAs. Each NAR is composed of a user U and a platform P, and is initialized by a valid PCR status PCR.

1. Upon receiving an input (TNC Request, *sid*, NAR=(U, P), NAA) from some NAR, send (sid, (U, P), NAA) to the adversary, and wait to receive an input (TNC Response, *sid*, NAR=(U, P), NAA) from NAA. Once receiving this input, set a boolean variable active.
2. Upon receiving (TNC Establish, *sid*, NAR=(U, P), NAA, f) from the adversary, and if active is set:
   a) If neither P nor NAA is corrupted, output (TNC Established, *sid*, NAR=(U, P), NAA, 1) to NAA.
   b) Else, if NAA is corrupted, output (TNC Established, *sid*, NAR=(U, P), NAA, f) to NAA.
   c) Else, if P is corrupted, output (TNC Established, *sid*, NAR=(U, P), NAA, 0) to NAA.
3. Upon receiving an input (Corrupt Platform, *sid*, P), change the PCR of P to $\overline{PCR}$ and mark P as corrupted.
4. Upon receiving an input (Corrupt NAA, *sid*, NAA), mark NAA as corrupted.

Fig. 6. The TNC Functionality, $\mathcal{F}_{\text{TNC}}$

5.2 Realizing TNC functionality

Figure 7 shows the complete TNC protocol. We prove it in the following theorem.

Theorem 4. *The complete TNC protocol securely realizes functionality $\mathcal{F}_{\text{TNC}}$ in the $(\mathcal{F}_{\text{D-SC}}, \mathcal{F}_{\text{P-A}})$-hybrid model.*

Protocol TNC

1. When activated with input (TNC Request, *sid*, NAR=(U, P), NAA) by $\mathcal{Z}$, NAR sends an input (Establish-session, *sid*, (U, P), initiator) to $\mathcal{F}_{\text{D-SC}}$, then waits for a challenge from NAA.
2. When activated with input (TNC Response, *sid*, NAR=(U, P), NAA)by $\mathcal{Z}$, NAA do:
   a) Send an input (Establish-session, *sid*, (U, P), responder) to $\mathcal{F}_{\text{D-SC}}$.
   b) Send an input (Challenge, *sid*, P, verifier) to $\mathcal{F}_{\text{P-A}}$, receive (sid, n), and record n.
   c) Send (Send, *sid*, n) to $\mathcal{F}_{\text{D-SC}}$.
3. Upon receiving the challenge n from $\mathcal{F}_{\text{D-SC}}$, NAR sends (Attest, *sid*, NAA, prover) to $\mathcal{F}_{\text{P-A}}$.
4. Upon receiving (SentPCR, *sid*, P, $m = (n', PCR')$) from $\mathcal{F}_{\text{P-A}}$, NAA checks whether $n' = n$ and PCR' is valid. If m passes the two checks, NAA sets $f = 1$, else sets $f = 0$. Then NAA outputs(TNC Established, *sid*, NAR=(U, P), NAA, f).

Fig. 7. The TNC protocol for realizing $\mathcal{F}_{\text{TNC}}$

Proof. Let $\mathcal{A}$ be an adversary that interacts with parties running TNC in the $(\mathcal{F}_{\text{D-SC}}, \mathcal{F}_{\text{P-A}})$-hybrid model. We construct an ideal world adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish whether it's interacting with $\mathcal{A}$ and TNC in the real world or it's interacting with $\mathcal{S}$ and $\mathcal{F}_{\text{TNC}}$ in the ideal world. As usual, $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$ and simulates for this copy an interaction with parties running TNC. All messages from $\mathcal{Z}$ to $\mathcal{A}$ and back are forwarded. The operations of $\mathcal{S}$ are shown below:

Simulating the NAR. When an uncorrupted NAR is activated with input (TNC Request, *sid*, NAR=(U, P), NAA) by $\mathcal{Z}$, $\mathcal{S}$ obtains this input from $\mathcal{F}_{\text{TNC}}$. Then $\mathcal{S}$ simulates a run of NAR and the interaction with $(\mathcal{F}_{\text{D-SC}}, \mathcal{F}_{\text{P-A}})$ for $\mathcal{A}$. For the seek of simplicity, we omit the simulation of NAR, which directly follows the flow of Figure 7. We focus on the interaction with $\mathcal{F}_{\text{D-SC}}$ and $\mathcal{F}_{\text{P-A}}$:
- When the logic of NAR instructs it to send (Attest, *sid*, V, prover) to $\mathcal{F}_{\text{P-A}}$, send (SentPCR, *sid*, P, NAA) to $\mathcal{A}$.

Simulating the NAA. When an uncorrupted NAA is activated with input (TNC Response, *sid*, NAR=(U, P), NAA) by $\mathcal{Z}$, $\mathcal{S}$ obtains this input from $\mathcal{F}_{\text{TNC}}$. Then $\mathcal{S}$ simulates a run of NAA and the interaction with $(\mathcal{F}_{\text{D-SC}}, \mathcal{F}_{\text{P-A}})$ for $\mathcal{A}$. Similarly, we omit the simulation of NAA, and focus on the interaction with $\mathcal{F}_{\text{D-SC}}$ and $\mathcal{F}_{\text{P-A}}$:
- When NAA sends (Challenge, *sid*, P, verifier) to $\mathcal{F}_{\text{P-A}}$, send (sid, P, V) to $\mathcal{A}$.
- When NAA sends a nonce n to $\mathcal{F}_{\text{D-SC}}$, send (Send, *sid*, $l(n)$) to $\mathcal{A}$ in the name of $\mathcal{F}_{\text{D-SC}}$.

Simulating party corruptions. When $\mathcal{A}$ corrupts NAR or NAA, $\mathcal{S}$ corrupts the corresponding party in the ideal world, and provides the obtained information to $\mathcal{A}$.

It is easy to verify that the above simulation is perfect, i.e., for any environment $\mathcal{Z}$ and $\mathcal{A}$ the view of $\mathcal{Z}$ in an interaction of $\mathcal{A}$ and TNC is distributed identically to the view in an interaction with $\mathcal{S}$ and the ideal TNC functionality.

5.3 Realizing TNC without Dual Authentication

To demonstrate the usefulness of the dual-authentication feature we proposed in this paper, we show that the complete TNC protocol depicted in Figure 7 cannot realize $\mathcal{F}_{\text{TNC}}$ if the dual-authentication feature is removed, i.e., replacing $\mathcal{F}_{\text{D-SC}}$ with $\mathcal{F}_{\text{SC}}$.

Theorem 5. *The complete TNC protocol doesn't UC-realize functionality $\mathcal{F}_{\text{TNC}}$ in the $(\mathcal{F}_{\text{SC}}, \mathcal{F}_{\text{P-A}})$-hybrid model.*

Proof. Consider the following environment $\mathcal{Z}$ and adversary $\mathcal{A}$. $\mathcal{Z}$ first activates a NAR=(U, P_1) with input (TNC Request, *sid*, NAR=(U, P_1), NAA) and a NAA with input (TNC Response, *sid*, NAR$'$=(U, P), NAA). Then $\mathcal{Z}$ instructs $\mathcal{A}$ to:

1. Corrupt both the user U and platform P_1 of NAR.
2. Instruct NAR to send an input (Establish-session, *sid*, U, responder) to $\mathcal{F}_{\text{SC}}$.
3. After receiving challenge nonce n from $\mathcal{F}_{\text{SC}}$, notify $\mathcal{Z}$.

Finally, $\mathcal{Z}$ instructs P to send (Attest, *sid*, V, prover) to $\mathcal{F}_{\text{P-A}}$ (or activates a NAA$''$=(U', P)). If NAA outputs (TNC Established, *sid*, NAR$'$=(U, P), NAA, 1), then $\mathcal{Z}$ outputs REAL. Otherwise $\mathcal{Z}$ outputs IDEAL.

In the real execution of TNC, NAA authenticates the user U and checks that the PCR status of platform P is not corrupted, so it would output that NAR is allowed to access the requested network. However, in the simulation of ideal world, $\mathcal{F}_{\text{TNC}}$ never receives the TNC request from NAR=(U, P), so it would always output that NAR is not allowed to access the requested network.

6 Conclusion

We analyze the complete TNC protocol, combining the IF-T binding to TLS and TCG platform attestation protocol, in the UC framework. We show that the SKAE extension introduced by TCG plays an important part in preventing the MitM attack presented in [1]. Our roadmap for the analysis of the complete TNC protocol adopts the modular feature of UC framework. We partition the complete TNC into a secure channel functionality and a platform attestation functionality. Then show that the IF-TLS-SKAE protocol and the basic TCG platform attestation protocol securely realizes the two primitive functionalities respectively. Finally, we use the composition theorem to argue that the complete TNC protocol realizes TNC. Besides, the dual-authentication certification functionality $\mathcal{F}_{\text{D-Cert}}$ and the platform attestation functionality $\mathcal{F}_{\text{P-A}}$ enable us to analyze protocols that use SKAE extension or TCG platform attestation in a modular way.

References

1. N. Asokan, V. Niemi, and K. Nyberg. Man-in-the-middle in tunnelled authentication protocols. In *Security Protocols*, pages 28–41. Springer, 2005.
2. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.
3. R. Canetti. Universally composable signature, certification, and authentication. In *Computer Security Foundations Workshop, 2004. Proceedings. 17th IEEE*, pages 219–233. IEEE, 2004.
4. R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology-EUROCRYPT 2002*, pages 337–351. Springer, 2002.
5. L. Chen and B. Warinschi. Security of the TCG Privacy-CA solution. In *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on*, pages 609–616. IEEE, 2010.
6. S. Gajek, M. Manulis, O. Pereira, A.-R. Sadeghi, and J. Schwenk. Universally composable security analysis of tls. In *Provable Security*, pages 313–327. Springer, 2008.
7. Institute for Electrical and Electronics Engineers (IEEE). IEEE802, Port-Based Network Access Control, IEEE Std 802.1X-2004, December 2004.
8. R. Küsters and M. Tuengerthal. Joint state theorems for public-key encryption and digital signature functionalities with local computation. In *Computer Security Foundations Symposium, 2008. CSF'08. IEEE 21st*, pages 270–284. IEEE, 2008.
9. J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: An execution infrastructure for tcb minimization. In *ACM SIGOPS Operating Systems Review*, volume 42, pages 315–328. ACM, 2008.
10. A. Melnikov and K. Zeilenga. Simple Authentication and Security Layer (SASL). Technical report, RFC 4422, June, 2006.
11. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 184–200. IEEE, 2001.
12. R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn. Design and implementation of a tcg-based integrity measurement architecture. In *USENIX Security Symposium*, volume 13, pages 16–16, 2004.
13. Trusted Computing Group. Subject Key Attestation Evidence Extension Version 1.0, Revision 7., 16 June 2005.
14. Trusted Computing Group. TNC IF-T: Protocol Bindings for Tunneled EAP Methods Specification Version 1.1, Revision 10., 21 May 2007.
15. Trusted Computing Group. Trusted Platform Module Library Part 1: Architecture, Family "2.0" Level 00, Revision 00.99., 22 August 2013.

16. Trusted Computing Group. Trusted Platform Module Library Part 3: Commands, Family "2.0" Level 00, Revision 00.99., 22 August 2013.
17. Trusted Computing Group. TNC IF-TNCCS: TLV Binding Specification Version 2.0, Revision 16., 22 January 2010.
18. Trusted Computing Group. TNC IF-T: Binding to TLS Specification Version 2.0, Revision 7., 27 February 2013.
19. Trusted Computing Group. TNC Architecture for Interoperability Specification Version 1.5, Revision 3., 7 May 2012.
20. Y. Xiao, Y. Wang, and L. Pang. Security analysis and improvement of TNC IF-T Protocol Binding to TLS. *Communications, China*, 10(7):85–92, 2013.
21. J. Zhang, J. Ma, and S. Moon. Universally composable secure TNC model and EAP-TNC protocol in IF-T. *Science China Information Sciences*, 53(3):465–482, 2010.
22. Z. Zhang, L. Zhu, F. Wang, L. Liao, C. Guo, and H. Wang. Computationally sound symbolic analysis of EAP-TNC protocol. In *Trusted Systems*, pages 113–128. Springer, 2012.