Provably Secure Communication Protocols for Remote Attestation

Johannes Wilson
johannes.wilson@liu.se
Sectra Communications
Linköping, Sweden
Linköping University
Linköping, Sweden

Mikael Asplund
mikael.asplund@liu.se
Linköping University
Linköping, Sweden

Niklas Johansson
niklas.johansson@liu.se
Sectra Communications
Linköping, Sweden
Linköping University
Linköping, Sweden

Felipe Boeira
contact@felipeboeira.eu
Linköping University
Linköping, Sweden

ABSTRACT

Remote Attestation is emerging as a promising technique to ensure that some remote device is in a trustworthy state. This can for example be an IoT device that is attested by a cloud service before allowing the device to connect. However, flaws in the communication protocols associated with the remote attestation mechanism can introduce vulnerabilities into the system design and potentially nullify the added security. Formal verification of protocol security can help to prevent such flaws. In this work we provide a detailed analysis of the necessary security properties for remote attestation focusing on the authenticity of the involved agents. We extend beyond existing work by considering the possibility of an attestation server (making the attestation process involve three parties) as well as requiring verifier authentication. We demonstrate that some security properties are not met by a state-of-the-art commercial protocol for remote attestation for our strong adversary model. Moreover, we design two new communication protocols for remote attestation that we formally prove fulfil all of the considered authentication properties.

CCS CONCEPTS

• **Security and privacy → Security protocols**; *Logic and verification*; *Formal security models*; *Trusted computing*; *Mobile platform security*.

KEYWORDS

Remote Attestation, Formal Protocol Verification, Tamarin Prover, Authentication, Security Models, Protocol Attack

ARES 2024, July 30–August 02, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1718-5/24/07
https://doi.org/10.1145/3664476.3664485

ACM Reference Format:
Johannes Wilson, Mikael Asplund, Niklas Johansson, and Felipe Boeira. 2024. Provably Secure Communication Protocols for Remote Attestation. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30–August 02, 2024, Vienna, Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3664476.3664485

1 INTRODUCTION

Remote attestation is a process in which some target platform proves or attests its correct and trustworthy operation to some remote party. Remote attestation can be useful in order to show absence of malware, even on low-end devices with limited capabilities to run anti-malware software [14]. Additionally, TPM chips and Trusted Execution Environments (TEEs) have become commonplace in modern computing devices, many of which provide functionality for anonymous remote attestation. The usual procedure for performing remote attestation requires some hardware-protected environment able to attest the correct operation of the target platform. The local attestation procedure consists of some measurements on the target platform, and a cryptographic signature used to prove that measurements originated from a trusted source.

Existing works on remote attestation often focus on the measurements provided by the prover, and the strength of such evidence. Several sophisticated remote attestation mechanisms have been designed, allowing anonymous remote attestation [31, 33], scalable remote attestation for IoT [22], dynamic attestation of program run-time environment [1], including protection from time-of-check-time-of-use (TOCTOU) attacks [11, 37], and has been provided on verified architectures [19, 20]. We take a different perspective on remote attestation, focusing on the *attestation protocol*, meaning the network protocol necessary to perform attestation. Design of protocols, and especially design of multi-agent protocols, is difficult and error prone. Multi-party scenarios increase the design complexity of remote attestation protocols significantly over the simplest two-party case, and can introduce complex security requirements. Furthermore, remote attestation schemes often involve more than two-parties in practice. Remote servers provided by the Original

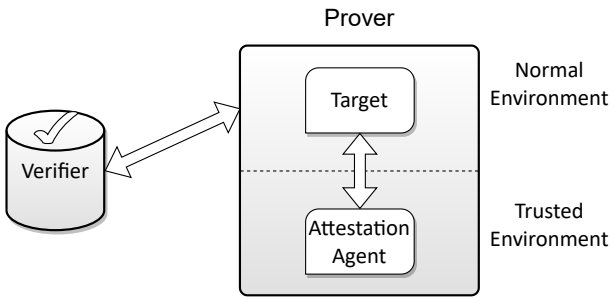

Figure 1: Simple Remote Attestation Scenario

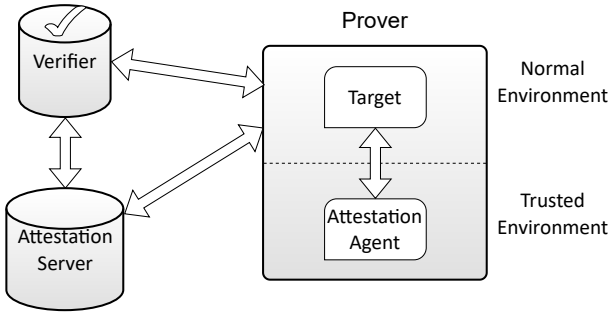

Figure 2: Remote Attestation Scenario with an Attestation Server

Equipment Manufacturer (OEM), or by third parties [28], are often needed to perform validation of the measurements, and perform the protocol in addition to both the device requesting the attestation, and the device performing attestation. At the same time, logical flaws in protocol design may jeopardise the security of remote attestation entirely.

Protocol design is a difficult problem in general. Many vulnerabilities in established protocols have been discovered after their specification, such as the original Needham-Schroeder public-key protocol [12, 16], or the SSL 3.0 specification [32]. Formal verification of protocols is important, since formal proofs ensure stronger guarantees of security than informal ones. Automated proofs using software tools have proven very useful in finding and fixing vulnerabilities, for example in the TLS 1.3 specifications [6, 10] and 5G specifications [4, 8]. One such tool is the Tamarin prover, which has been used during formal verification of a number of cryptographic protocols. While the usage of automated formal verification has shown great promise, it is not obvious for many protocols what properties need to be verified in order to ensure security. For remote attestation protocols, symbolic formal analysis has seen limited usage. Because of this, there is no common consensus on what security properties to verify.

1.1 Contributions

Our work defines a detailed list of protocol goals for remote attestation protocols, both for two-party and three-party scenarios involving an additional verdict-producing server. We translate the protocol goals into standard security properties which are verifiable using symbolic protocol analysis. Moreover, we showcase how our list of security properties can be useful in determining the security of a remote attestation protocol. First, we present formal verification of an existing state-of-the-art commercial protocol, revealing a weakness in the protocol design, requiring other security mechanisms to prevent it from being compromised. Next, we present and formally verify the security of two remote attestation protocols that we have developed, showing that our developed protocols fulfil our strict requirements of security. All of our models are openly available[1].

The contributions of this paper can be summarised as follows. We provide:

- A detailed specification of security properties for remote attestation protocols
- An analysis of an existing state-of-the-art commercial protocol, which revealed a potential weakness in the protocol design when considering a very strong adversary.
- Designs of two new protocols that fulfil our strong requirements of authenticity when considering a very strong adversary

1.2 Paper Outline

The remainder or this paper is organized as follows. We describe our remote attestation scenarios of interest, present a list of attacks that must be mitigated, and explain the basics of the Tamarin prover in section 2. In section 3, we extensively describe our selection process of security properties to verify. We first select a number of informal goals, then describe our system and attack model, next we turn our goals into security properties, and finally we list all of the lemmas we used during formal verification. We present the analysis of a state-of-the-art commercial protocol in section 5. In section 6 we describe the design and the verification of our own developed protocols. section 7 describes related work or verification of remote attestation protocols, while section 8 summarises our main conclusions from our work.

2 BACKGROUND

In this chapter we briefly describe remote attestation, as well as some attack scenarios. We additionally introduce the Tamarin prover which we use during formal verification of protocols.

2.1 Remote Attestation

We use the term *prover* to describe the platform that is proving its state. We refer to the party that is interested in the state of the prover as the *verifier*. The prover in turn consists of a *target*, an environment which we must attest the state of, and an *attestation agent*, an environment capable of measuring the state of the target and signing such measurements. An attestation scenario is shown in Figure 1. The verifier sends a challenge, requiring the prover to attest its state. The attestation agent on the prover performs some measurements, signs them, and sends them back to the verifier. Communication must pass through the target, since the target manages the network communication.

[1]All of our models are available at https://gitlab.liu.se/ida-rtslab/public-code/2024_ares_remote_attestation/

While this example covers the simplest scenario, in practice we often need to involve additional agents. Usually, how to evaluate the measurements or signatures made by the attestation agent is outside the knowledge of the verifier. Instead such knowledge is only available to the manufacturer of the device. Manufacturers may instead provide what we will refer to as an *attestation server*, which is able to give a verdict of the prover state based on the provided measurements. The usual procedure is that the attestation is sent to the attestation server for evaluation, and a verdict is produced. The verdict is sent to the verifier, and forms the basis for the trust evaluation of the prover.

Because using an attestation server is not always relevant, it is useful to handle the scenarios separately. We refer to a scenario involving only a prover and a verifier as a *two-party* attestation protocol. A scenario involving a prover, verifier, and an attestation server we refer to as a *three-party* attestation protocol. We show a three-party scenario in Figure 2.

2.2 Attacks on Remote Attestation

A remote attestation protocol must be resilient against many types of attacks. We mainly focus on four attacks that are very important to consider since they could endanger the integrity of attestation results, rendering remote attestation useless. These attacks are masquerading attacks, measurement tampering, replay attacks, and preplay attacks [23, 36].

Masquerading Attacks May allow a malicious device to masquerade as an honest device, so that the malicious device is believed to have the identity of the honest device.

Measurement Tampering Involves changing the attestation result, and if not protected against could allow an adversary to hide compromising evidence.

Replay Attacks May allow an attacker to reuse results from previous attestation requests in order to attest at a later time.

Preplay Attacks While similar to replay attacks, involve preparing attestation responses in advance, before any attestation request has been made. This may allow an adversary to attest at a later time, once the target is fully compromised.

We generally consider denial-of-service attacks to be out-of-scope, since there are many ways a network adversary can deny a protocol from progressing, with one notable exception. We consider cases where our target device is denied normal operation by superfluous requests to attest. Such attack may pose a threat to devices with limited capabilities.

Denial of Normal Target Operation An attestation protocol should protect the target device from performing superfluous attestations, so that an adversary cannot deny normal operation of the target.

Time-Of-Check-Time-Of-Use (TOCTOU) discrepancies could pose a danger to remote attestation. However, we consider such attacks out of scope since fully protecting against such discrepancies at a protocol level alone is not possible. Instead we refer to by Nunes et al. [11] on providing a mechanism for protecting against TOCTOU discrepancies during integrity measurements.

While there are many more types of attacks that could impact remote attestation than the ones covered here, we only consider attacks that are relevant at the protocol level, without considering implementation details. We are not aware of any other strictly protocol level attacks.

2.3 Formal Verification using Tamarin

The Tamarin prover is a state-of-the-art tool for formal verification of protocols in a symbolic model [18, 30]. Tamarin has been successfully used to find design flaws in protocols and verify their fixes. While Tamarin proves trace properties symbolically, assuming perfect cryptographic primitives, it is still capable of finding many logical flaws in protocol designs. Examples include flaws in the EMV standard for smartcards [5], weaknesses in the 5G key-agreement protocol [8], issues in early drafts of the TLS 1.3 specification [10], and many more.

Tamarin proves security against a strong Dolev-Yao adversary model with complete control over the network channel, meaning messages can be inserted, resent, modified or dropped [13]. Tamarin performs automated reasoning according to a heuristic, meaning the proof process is completely automated. While Tamarin cannot guarantee analysis will terminate, if it halts it will produce a sound and complete analysis of a protocol with respect to a given security property. Security properties typically specify authenticity or confidentiality requirements. Security properties are specified as formulas called lemmas, and Tamarin either produces a counter-example to the lemma in the form of an attack-trace, or asserts that it is not possible to violate the lemma in the given model. We use Tamarin to formally verify our remote attestation protocols with respect to a number of security properties.

We summarise some of the important features of the theory of Tamarin. In Tamarin, protocol steps and adversary capabilities are modelled using multiset rewrite rules. Rules operate on facts, where rules can specify facts as preconditions and postconditions. Simplified, facts in the precondition are required in order to perform the rule, and are consumed at invocation of the rule, while facts in the postcondition are generated. Additionally, facts may be persistent, meaning such a fact will not be consumed by the application of a rule. Two special types of terms exist, fresh terms and public terms. Fresh terms are only created using the fact Fr which may only appear as a precondition, and will always create a term which has not previously appeared. Public terms are known to the adversary by default. Cryptography is assumed to be ideal, where encryption and decryption is only possible upon knowing the corresponding key. A more comprehensive description of the theory of Tamarin can be found in [29].

3 SECURITY PROPERTIES FOR REMOTE ATTESTATION PROTOCOLS

We formalise security properties for remote attestation as a two-step process. First, we describe informal goals that we expect from any secure remote attestation protocol. We then propose a set of security properties that correspond to the informal goals and which can be formally verified.

3.1 Remote Attestation Protocol Goals

In this section, we specify eight informal protocol goals (G1 - G8) for remote attestation protocols. While some use-cases may put more emphasis on some goals than others, we expect that most protocols

Table 1: Mapping of Attestation Principles Presented by Coker et al. [7] and Attestation Goals

Principle	Attestation Goals
Fresh Information	G2 & G3 & G6
Comprehensive Information	out of scope
Constrained Disclosure	G4 & G5 & G8
Semantic Explicitness	out of scope
Trustworthy Mechanism	G1 & G4 & G5 & G7

will require each of these goals in some form. Goals G1-G6 are related to the authenticity of attestation, whereas goals G7 and G8 concern confidentiality of keys and attestation data, respectively. Goals G5 and G6 are only applicable for the three-party scenario when the attestation server must send a verdict to the verifier. We specifically focus on properties that show absence of the attacks listed in subsection 2.2, as well as the attestation goals given by Coker et al. [7]. The mapping between our goals and the goals given by Coker et al. is shown in Table 1.

G1: Authenticity of Attestation Agent For remote attestation to succeed from the point of view of the verifier, the verifier must be able to guarantee that the attestation was performed by the attestation agent on the desired prover. If the verifier cannot identify the origin of the attestation, then the attestation is of little use to the verifier. In particular, after a successful attestation the verifier should have learnt that a specific device can attest.

G2: Recentness of Attestation The verifier wants to ensure that any evidence of target state received reflects the recent state of the target. This is crucial especially for malware detection or trust evidence, since a compromise may have occurred after the attestation if a lot of time has passed. Having some way to ensure recency helps in defending against preplay attacks. This does not prevent TOCTOU attacks, but it can serve as a mitigation.

G3: Integrity of Attestation Data In order for the verifier to trust the prover, any evidence which the trust decision is based on must be authentic, and must accurately reflect the state of the target. A malicious entity on the target that can tamper with the measurements done by the attestation agent without detection could potentially disguise the state of a compromised target as benign. To prevent such attacks, we need to ensure that any data needed in making a verdict originated from the attestation agent.

G4: Authenticity of Verifier Coker et al. [7] lift the importance of letting the prover remain in control of the attestation procedure. In a setting of ensuring trustworthiness, a prover that is well-behaved should be allowed to reject challenges from a verifier that is ill-behaved. This is especially important in settings where attestation reveals information that should not be available to everyone, or where the target attestation mechanism is susceptible to DoS attacks. In order for the prover to only accept challenges from trusted parties, the verifier needs to authenticate to the prover.

G5: Authenticity of Attestation Server For the three-party scenario, we should also consider how the attestation server is safely made a part of the protocol. In order for the verifier to trust the attestation server, the attestation server should be authenticated to the verifier for this specific run of the protocol. The server should also be authenticated to the prover, so that the prover knows that the correct server received the data.

G6: Integrity of Attestation Verdict Because the verifier relies on the verdict from the attestation server in a three-party setting, it is important that such a verdict is delivered securely. An adversary should not be able to tamper with the attestation verdict, or be able to reuse verdicts from previous attestations.

G7: Confidentiality of Attestation Keys As a partial goal to ensuring authenticity of our attesting devices, we must also ensure that all attestation keys used remain secure during attestation.

G8: Confidentiality of Attestation Data It is often desirable for the data sent from the prover to the verifier or attestation server to remain confidential. The attestation may contain measurements of the target state, such as exact platform configuration or current memory contents, that are acceptable to send to a trusted verifier or attestation server, but not to the public internet.

3.2 Formalising Security Properties

While the goals serve as a high-level specification for remote attestation, we need to translate them into properties verifiable using Tamarin. We first map each of the informal goals from subsection 3.1 to a formal security property found in the literature, following Lowe's hierarchy [17], describing agreement properties for authentication. An agreement property states informally that the responding agent must have performed the protocol, with values that match the ones sent by the initiator. Following prior work [36], we differentiate between one-way and two-way authentication properties.

Some security properties can cover multiple goals if they are strictly stronger properties than ones given for other goals.

We show the chosen properties in Table 2. For authentication properties, the *Responder* column shows what agent should be authenticated, while the *Initiator* column shows which agent it should be authenticated to. For Goal 1, the attestation agent should authenticate to both the server and the verifier. Similarly, when using an attestation server, we verify for Goal 5 that the server is authenticated to both the verifier and the target. Table 3 shows how the attacks listed in subsection 2.2 are mitigated by the chosen security properties.

3.3 From Properties to Lemmas

Going from each security property to a Tamarin lemma is a straightforward process. We use standard formulations for authentication and secrecy properties where agents to be authenticated cannot have had their long-term keys leaked.

For some of the properties listed there is an overlap, in the sense that an authentication property is completely covered by a stronger

Table 2: Security Properties

Security Property	Informal Goal	Responder	Initiator
SP1 One-Way Injective Agreement	G1 Authenticity of Attestation Agent	Attestation Agent	Verifier
SP2 One-Way Recent Injective Agreement	G2 Recentness of Attestation	Attestation Agent	Verifier
SP3 One-Way Non-Injective Agreement	G3 Integrity of Attestation Data	Attestation Agent	Verifier OR Server
SP4 Injective Agreement	G4 Authenticity of Verifier	Verifier	Target
SP5 One-Way Injective Agreement	G5 Authenticity of Attestation Server G6 Integrity of Attestation Verdict	Server	Target AND Verifier
SP6 Secrecy	G7 Confidentiality of Attestation Keys	-	-
SP7 Strong Secrecy*	G8 Confidentiality of Attestation Data	-	-

*Verification of property was not performed in this work.

Table 3: Protocol Attacks and what Security Property they are Mitigated by

Attack	Security Property
Masquerading Attack	SP1
Measurement Tampering	SP3
Replay Attack	SP1
Preplay Attack	SP2
Denial of normal target operation	SP4

Table 4: List of Lemmas

Lemma	SP	
L1 O-W-R-I-Agree-Verif-AttAg	SP1 SP2 SP3	
L2 O-W-Non-I-Agree-Server-AttAg	SP3	*
L3 Inj-Agree-Target-Verif	SP4	
L4 O-W-I-Agree-Target-Server	SP5	*
L5 O-W-I-Agree-Verifier-Server	SP5	*
L6 Secrecy-Key	SP6	

* Three-agent only.

property in the hierarchy. As an example, one-way recent injective agreement is a strictly stronger property than both one-way injective agreement and one-way non-injective agreement. Since Properties SP1, SP2 and SP3 in the two-party scenario all describe authentication of the attestation agent to the verifier using authentication properties of varying strength, it is possible to simplify the verification effort by only verifying the strongest property: SP2. This way we can reduce the number of Tamarin lemmas required, and also simplify the presentation of our verification results. In total, we use six lemmas in the three-party case, and three lemmas in the two-party case in order to verify all security properties. These lemmas are listed in Table 4.

4 MODELLING ATTESTATION PROTOCOLS IN TAMARIN

Remote attestation aims to protect the target environment by detecting and reporting undesired behaviour to the verifier. As a consequence, appropriately modelling remote attestation requires not only modelling of the adversary behaviour over the network, but also modelling of potential adversarial presence on the target device. This creates new modelling challenges, which to the knowledge of the authors have not previously been addressed in other works. In particular, we want remote attestation to be secure even when the target of our prover is compromised. For this reason, we consider cases when the target of the prover is controlled by the adversary in our model. In this section, we detail our modelling choices for the verification of remote attestation protocols using Tamarin. We first present our system model and adversary model. We then detail how we model the described adversary in Tamarin.

4.1 Attestation Mechanism Assumptions

When making a model of a remote attestation protocol, we inevitably need to cover what proof the attestation mechanism is able to provide, since this will affect assumptions about the adversarial capabilities. A weaker attestation mechanism, which does not protect all parts of the prover, may allow an adversary to evade detection during attestation. As such, the target may in part remain vulnerable to the adversary even after successful attestation. On the other hand, a strong attestation mechanism which performs measurements of the entire run-time state of the target may be able

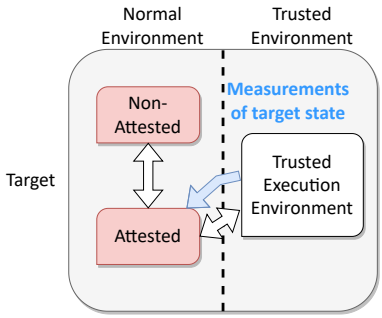

Figure 3: Attestation Mechanism on Prover System.

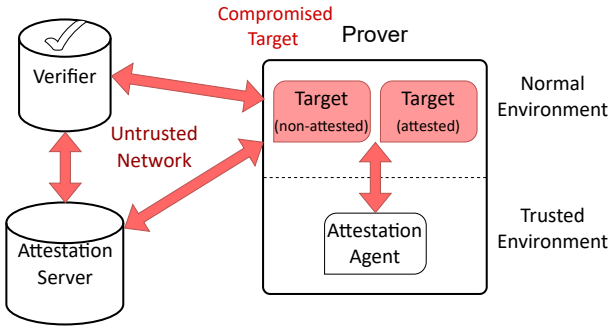

Figure 4: Attack Model.

to show absence of any adversary on the target upon completion of the protocol. However, such a mechanism would be very difficult to achieve in practice unless we are working with very small and constrained devices.

We have chosen to model that parts of the target may remain un-attested, meaning an adversary may remain undetected on such un-attested parts of the target device even after successful attestation. This is motivated by the idea that an OEM may be able to provide support to verify the firmware and operating system of their own devices, but providing support to also attest third-party software can be a lot more challenging. We have made the following assumptions regarding the attestation mechanism.

- The entire target is not attested during attestation, meaning there is a non-attested and an attested part of the target.
- The attestation mechanism is capable of finding any adversary presence on the attested part of the target, meaning any changes to or attacks on such parts can be detected.
- The attestation mechanism is NOT able to find all run-time attacks on software running on the device. In particular, we assume that the application initiating remote attestation on the target is outside the scope of the attestation mechanism, meaning an adversary may perform run-time attacks against the software performing attestation in order to initiate attestation without detection.

Figure 3 shows an overview of the described attestation mechanism. Although an adversary may potentially compromise both the attested and non-attested parts of the target (shown by boxes in red) it is possible to detect attacks on the attested part based on the measurements made (shown in blue) from the trusted execution environment. The trusted execution environment is assumed to be secure at all times.

4.2 Attack Model

We model our participating agents according to Figure 4. The verifier, attestation server, and prover all communicate remotely over an insecure network. We consider the network to be controlled by a Dolev-Yao adversary model in accordance with the standard attacker model of the Tamarin prover. The prover itself is divided into the potentially insecure target, containing both attestable and non-attestable parts, and the attestation agent, built into the trusted execution environment. We model that communication between the target and attestation agent is insecure in order to capture that an adversary with control of the target can initiate requests to the attestation agent and modify responses.

4.3 Tamarin Model

We now discuss how the modelling choices above translate into a Tamarin model for symbolic verification of attestation protocols.

4.3.1 Network. We model the network adversary according to standard practice [29] by using the built-in *In* and *Out* facts. This means that all messages sent on any of the insecure channels are under the control of the adversary, and that additionally the adversary can inject messages to any channel.

4.3.2 Modeling the Attestation Procedure. While the proof produced by the attestation agent may involve for example a hash of the measurements, we have to abstract away from the implementation-specific measurements in a symbolic model. We want to model that values can be either valid or invalid when taken as measurements. We model this using a rule which produces a persistent fact using a public term, which is additionally broadcast to the adversary. The attestation agent and the party verifying the measurements, require this persistent fact as part of performing a measurement or verifying one, respectively. The effect is that our model allows a virtually infinite set of valid measurements that can be made, which are all known to the adversary, where any measurement not in the set is considered invalid.

4.3.3 Compromised Targets. On a prover, the adversary may be present between the target and the attestation agent, yet may still evade detection during attestation according to our assumptions on the attestation mechanism made in subsection 4.1. We model this by considering the communication between the target and the attestation agent to be under adversary control. We model this in Tamarin using the same *In* and *Out* facts as for the network. This ensures that the adversary may intercept the communication, learn from the messages seen, as well as send their own messages.

A fully compromised prover (with also the attested part of the target compromised) should not be able to achieve valid measurements using its attestation agent according to our assumptions on the attestation mechanism (Section 4.1). We can assume that unless the requests to the attestation agent themselves reveal some information useful to the adversary, no progress can be made through requests to the attestation agent on a fully compromised prover.

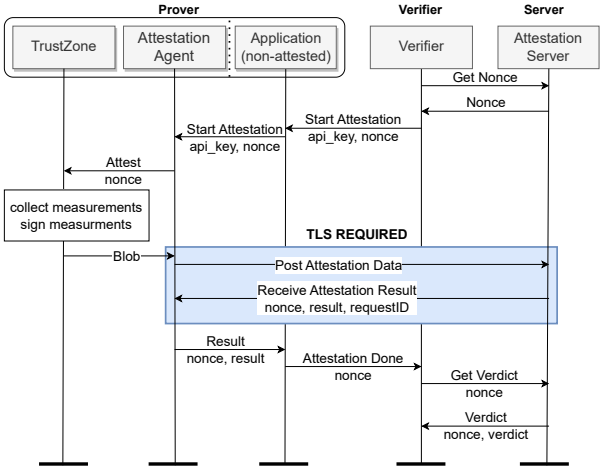

Figure 5: Enhanced Attestation v3 for Samsung KNOX Devices.

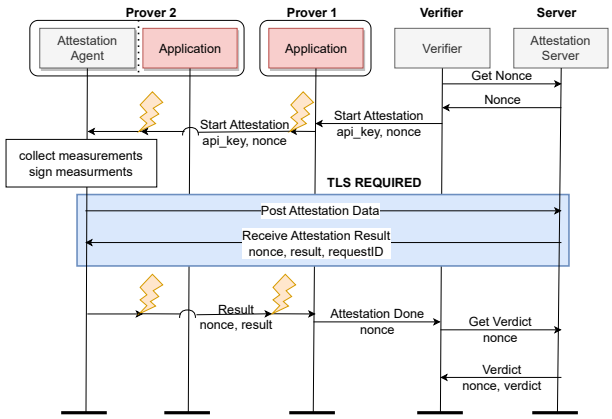

Figure 6: Attack Scenario Found for Enhanced Attestation v3 Protocol.

Under the assumption that no useful information is revealed, there is no reason to also model such a rule. What remains are the rules of the fully compromised target, but such rules are equivalent to the rules for the partially compromised prover.

Similarly, the un-compromised provers are already modelled by the behaviour of the partially compromised prover, since the behaviour is equivalent to a prover where the adversary faithfully forwards the messages sent from the target to the attestation agent and back. Therefore, we conclude that modelling only the partially compromised prover is sufficient for our attacker model.

5 ANALYSIS OF A STATE-OF-THE-ART PROTOCOL

We performed analysis of the remote attestation protocol provided on Samsung Knox devices. We analysed the Enhanced Attestation V3 [25], which is at time of time of writing the current version of the protocol. The Knox Enhanced Attestation protocol is a three-party protocol, making use of an attestation server, and is shown in Figure 5. The protocol is initiated by the verifier who requests a fresh nonce from Samsung's attestation server. The nonce is used to identify each attestation, and is sent to the device along with an API key. The device utilises the TrustZone architecture in order to perform some integrity measurements and sign them with a unique Samsung Attestation Key. The measurements are sent to the Attestation server over a TLS connection, and the server can from the attestation data produce a verdict. Next, the prover signals to the verifier that the attestation has concluded, after which the verifier can request the verdict from the attestation server. The server verdict contains, among other things like timestamp and version number, the following data.

- **verdict:** Yes or No response indicating whether device tampering is suspected based on the integrity checks.
- **warrantyFuseState:** Indicates if the device has ever been rooted.
- **trustBootState:** Indicates if the devices is using an official binary.
- **deviceIdState:** Shows if the device IMEI or serial number have been modified.
- **app:** Application package name and signature of the application which initiated attestation on the prover.

We note that there is no device identity visible from the verdict. The device id state is a Boolean showing if tampering was detected, and does not tie attestation to any particular device.

The assumptions made in subsection 4.1 regarding the attestation mechanism match the Knox protocol, as there are detection mechanisms for adversaries on system code, while user-level attestation is limited to the application name and signature for the application. As such, run-time attacks on the prover user-application would not be visible from the attestation verdict.

5.1 Protocol Specific Modeling Assumptions

Since we have not had access to any implementation details other than the high-level protocol overview and publicly available API-calls, we have had to make some assumptions about the protocol in order to model it in Tamarin. As a first step, we model the protocol in accordance with our generic system model by considering the prover application as our target, while we merge the TrustZone and the Attestation Agent into one trusted Attestation Agent. We motivate this by stating that this is the more generous assumption. We assume that system code, which would include the attestation agent, is secure and uncompromisable due to the kernel run-time protection provided by Samsung Knox [24]. For this protocol in particular, we assume the server and verifier are both static publicly known endpoints and will not change between runs. The reasoning being that the protocol is designed to be used by a Mobile Device Management service taking the role as the verifier. The implementation on the prover can thus assume it will always communicate with a given Mobile Device Management server. Similarly, the attestation server will always be a given endpoint provided by Samsung, and so we do not need to consider multiple attestation servers. Finally, to simplify the model, we assume that the connection over

Table 5: Verification Results Knox Protocol

Lemma	SP	Result
L1 O-W-R-I-Agree-Verif-AttAg	SP1 SP2 SP3	✗
L2 O-W-Non-I-Agree-Server-AttAg	SP3	✓
L3 Inj-Agree-Target-Verif	SP4	✗
L4 O-W-I-Agree-Target-Server	SP5	✓
L5 O-W-I-Agree-Verifier-Server	SP5	✓
L6 Secrecy-Key	SP6	✓

TLS works as a confidential and authenticated channel. Accurately modelling TLS is difficult and takes considerable effort [10] [9]. Furthermore, attempting to compose an authentic TLS model with another protocol would slow down verification considerably, and is not currently feasible to our knowledge.

5.2 Results

The result of the Tamarin analysis is shown in Table 5. Four out of the six lemmas are proven to hold for all possible runs of the protocol (marked as tic-boxes). For two out of the six lemmas Tamarin is able to find a counterexample, thus showing that the lemmas do not hold (marked with crosses).

Lemma L1 showing that the attestation agent is authenticated to the verifier fails for our model. This is mainly due to the absence of a unique device ID visible in the protocol. While the attestation server can identify the prover based on the attestation key used, no such information is conveyed back to the verifier in the verdict. The lemma is violated because there are traces possible in the model where the prover which the verifier communicated with was not the one which attested to the attestation server.

We show an example trace violating lemma L1 in Figure 6. In the trace, the adversary with control over the applications of both prover 1 and prover 2 may forward the request made to prover 1 to prover 2. Upon completion of the protocol, the verifier will believe that it was prover 1 which made the attestation.

The attack is possible since our model allows adversarial presence on the target of the prover. The adversary on the target may leak the unique nonce for the current run, and force an attestation on another trusted device using the same application. We can assume such attack may be possible using only user-level runtime attacks on the target. Such attacks are not detected by this attestation scheme. This would allow a rooted device to evade detection by delegating the attestation to a non-rooted device. We note that it would be incredibly difficult to perform such an attack in practice, since it would require stealthy run-time attacks of the specific prover application used during attestation. As such, our finding is mostly of theoretic interest. Making sure that the attestation application is written in a secure way, and also ensuring secure communication between the prover and verifier are sufficient preventative measures. For even more discussion regarding this scenario and preventative measures, we refer to the original master's thesis where the attack scenario was first found [35].

Lemma L3 showing that the verifier is authenticated to our target does not hold. We have assumed that the application receives the request from the verifier with an authenticated, but not replay-protected request. In order to protect the prover from repeated requests, it is necessary to use a secure channel such as TLS between the prover application and verifier in order to improve the security so that the authentication is also injective between the target and verifier.

Samsung have been made aware of our findings, but have decided that a change of the protocol is not necessary since our analysis did not reveal any realistic way to stage an attack in practice. We have no reason to dispute this view nor has it been our goal to find a practical attack against the protocol. Security can be attained in several ways, and our analysis is restricted to the security of the protocol itself under a strong adversary model, not considering additional mechanisms on the devices to prevent such attacks in practice.

6 PROTOCOLS FOR SECURE REMOTE ATTESTATION

We will now showcase two remote attestation protocols that fulfil our strict requirements for authentication. One is a two-party protocol, while the other is a three-party protocol. These protocols protect against the type of attack shown in Figure 6, where there is a mismatch of prover identity. This is provided by sending the prover identity as part of the verdict from the attestation server in our three-party protocol. In addition, we explicitly provide injective authentication of the verifier to the target, in order to prevent the prover from getting overloaded by inserted or replayed attestation requests. We provide this using an extra round-trip of messages where the verifier signs fresh data provided by the target, but in practice this could be implemented using a TLS connection where at the very least the verifier is authenticated to the target, since TLS ensures a recent session [9].

6.1 Protocol Design

We show our two-party remote attestation protocol in Figure 7. In the protocol, names of requests are shown above the arrow-lines, while message contents are shown below. Message names are included as part of the message, and are included in signatures (not shown in the figures). The two-party protocol only adds a single roundtrip for a handshake between the target and verifier over the simplest possible challenge-response protocol. The handshake lets both the verifier and target be sure that the protocol is injective by letting both agents establish a unique value for each run. The target confirms that the verifier can sign both its own and the prover's generated value before initiating the attestation. This prevents replayed attestation requests from an adversary to the prover, and prevents the adversary from reusing attestations from the prover to the verifier. A description of the protocol steps can be found in Table 6.

This same handshake is also part of the three-party protocol shown in Figure 8, for the same reason of avoiding superfluous attestations. In the three-party protocol, the attestation data is sent by the target to the attestation server. The verifier must then request a verdict from the attestation server in the same way as the Knox protocol. Compared to the Knox protocol, we include the prover identity in the attestation server verdict. Signing the prover identity

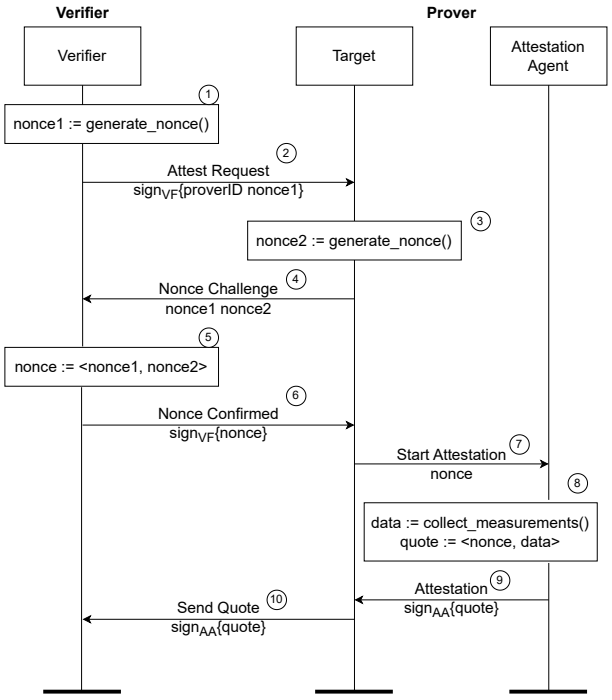

Figure 7: Two-party Remote Attestation Protocol

Table 6: Description of Steps in Two-Party RA Protocol

Step	Description
1	Verifier generates a random nonce
2	Verifier sends the nonce to the prover and signs the message
3	The target of the prover generates its own nonce
4	The target sends both nonces to the verifier
5	The verifier combines both nonces into one
6	The combined nonce is signed and sent back to the prover
7	The target of the prover receives the combined nonce, verifies that it matches the earlier request, and forwards the attestation request to the attestation agent
8	The attestation agent of the prover performs some integrity measurements and creates a quote from the measurements and the nonce
9	The quote is signed with the attestation key and sent back to the target
10	The target forwards the signed quote to the verifier

Table 7: Verification Results for Protocols

Lemma	Two-Agent	Three-Agent
L1 O-W-R-I-Agree-Verif-AttAg	✓	✓
L2 O-W-Non-I-Agree-Server-AttAg	N/A	✓
L3 Inj-Agree-Target-Verif	✓	✓
L4 O-W-I-Agree-Target-Server	N/A	✓
L5 O-W-I-Agree-Verifier-Server	N/A	✓
L6 Secrecy-Key	✓	✓

also from the side of the attestation server ensures that the run between the verifier and the target also matches the run performed between the attestation server and the attestation agent of the same prover. Ideally, such a device identity is tied to a serial number visible somewhere on the physical device, or otherwise provided by the OEM. A description of the protocol steps is found in Table 8.

6.2 Protocol Analysis

Verification results of the developed two- and three-party protocols are shown in Table 7. All the lemmas could be proven using Tamarin, showing that even for our strong adversary model, we are able to provide strict authenticity guarantees using our proposed protocols.

While the analysis performed by Tamarin is symbolic, and can only find logical flaws and not cryptographic ones, proving absence of logical flaws is still important. Notably, our protocols do not contain the flaw found in the studied commercial protocol, in that there is never any mismatch between provers.

7 RELATED WORK

Parts of this paper have previously been presented as part of a master's thesis [35]. Since then, we have revised our selection of security properties, clarified assumptions in our attacker model, and rewritten and redone analysis of all of our Tamarin models.

Attacks similar to the one presented for the Samsung Knox protocol have been discussed previously in the literature. An attack found by Asokan et al. showed how usage of tunnelling protocols could be made insecure if the secure tunnel was not tied properly to the endpoint using authentication [2]. The Trusted Attestation Protocol standard refers to this attack, stating that "Additionally, it is up to the protocol specifications to provide a means to mitigate against an Asokan attack" [15]. A similar attack scenario of abusing a remote 'oracle' device was discussed by Bryan Parno in the context of remote attestation using trusted platform modules [21]. The attack, which Parno named a 'cuckoo attack', is possible if the attestation cannot be tied to a given trusted platform module. Our work shows that these types of attacks may be present in real world remote attestation protocols, and may weaken security if not considered properly.

Formal verification of remote attestation protocols either verify the integrity proof mechanism, or verify protocol properties. Of the works on protocol verification, we have examples of verification for Direct Anonymous Attestation [3, 31, 33, 34], Intel SGX attestation [27], and Intel TDX attestation [26]. During formal verification of Direct Anonymous Attestation, focus has been on verifying the anonymity of the proposed cryptographic schemes.

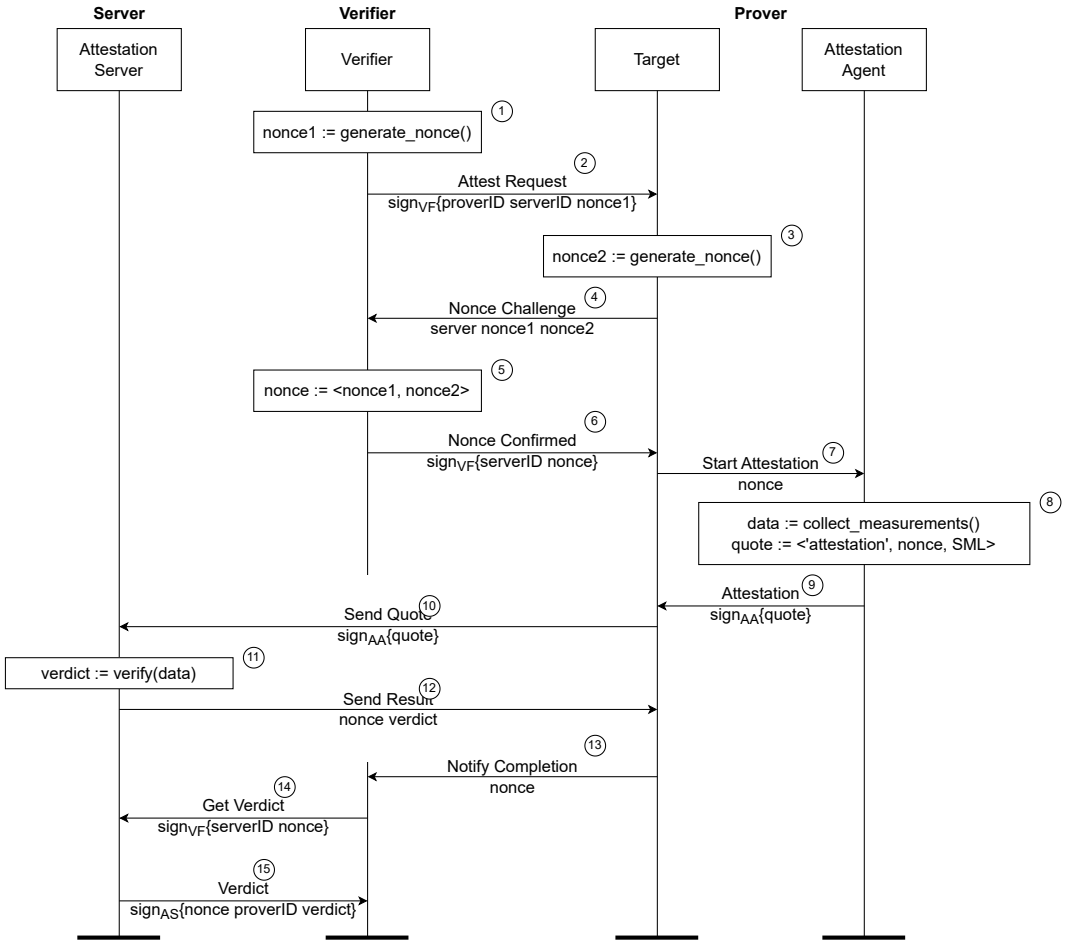

Figure 8: Three-party Remote Attestation Protocol

Sardar et al. formally verify remote attestation protocols for Intel SGX [27] and for Intel TDX [26] using ProVerif. During formal verification of Intel SGX, two confidentiality properties are verified. One property verifies that the report key cannot get leaked, the other verifies that the secret sent during attestation is never leaked.

ProVerif is also used during formal verification of Intel TDX by Sardar et al. [26]. The confidentiality properties verified consist of confidentiality of the established secret. Some authentication properties are also verified. One property verified was that if a quote was accepted, then such quote was produced by the CPU with all data fields unchanged. Another property verifies that any measurements accepted by the Quoting Enclave was sent by the TDX module. Compared to their work, our work provides a much more comprehensive list of authentication properties.

Formal verification of Direct Anonymous Attestation as implemented on Trusted Processing Modules was performed by Whitefield et al. [34] and Wesemeyer et al. [33] using Tamarin. They formally verified all authentication properties of Lowe's hierarchy when considering the issuer as the initiator and the platform as the responder. In addition, anonymity properties specific to the attestation scheme are verified. Compared to their work, we consider a

Table 8: Description of Steps in Three-Party RA Protocol

Step	Description
1–9	Same as for two-party protocol, except requests from verifier now contain a server ID
10	The target forwards the signed quote to the server
11	The server makes a verdict based on the received integrity measurements
12	The server notifies the target of the verdict made
13	The target notifies the verifier that the server has made a verdict
14	The verifier requests to see the verdict made by the server, and sends the nonce. The request is signed with the verifier public key
15	The server responds by sending a message containing the nonce, prover ID, and the verdict

stronger adversary model which allows for the target to be under adversary control.

8 CONCLUSIONS

In this work, we provide a comprehensive list of protocol-level security requirements for remote attestation. Extending from previous work we account for three-party attestation mechanisms (including an attestation server), as well as requiring verifier authentication. We design two new protocols (two-party attestation and three-party attestation) that both fulfil our requirements, requiring one additional handshake to avoid man-in-the-middle attacks. While confidentiality of attestation data was not formally analysed in this work, we believe that a secure tunnel such as a TLS connection between the target and the attestation server should suffice in order to provide this goal, but proper formal analysis is a potential avenue for future work. We also analysed version 3 of the Samsung Knox Enhanced Attestation protocol in this work, which we demonstrate does not meet all the listed properties at the protocol design level (thus requiring orthogonal security protection mechanisms).

9 DATA AVAILABILITY

All Tamarin models are available at https://gitlab.liu.se/ida-rtslab/public-code/2024_ares_remote_attestation/.

ACKNOWLEDGMENTS

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] Tigist Abera, N. Asokan, Lucas Davi, Jan-Erik Ekberg, Thomas Nyman, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. 2016. C-FLAT: Control-Flow Attestation for Embedded Systems Software. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 743–754. https://doi.org/10.1145/2976749.2978358

[2] Nadarajah Asokan, Valtteri Niemi, and Kaisa Nyberg. 2003. Man-in-the-middle in tunnelled authentication protocols. In *International Workshop on Security Protocols*. Springer, 28–41.

[3] Michael Backes, Matteo Maffei, and Dominique Unruh. 2008. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*. 202–215. https://doi.org/10.1109/SP.2008.23

[4] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A Formal Analysis of 5G Authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1383–1396. https://doi.org/10.1145/3243734.3243846

[5] David Basin, Ralf Sasse, and Jorge Toro-Pozo. 2021. The EMV Standard: Break, Fix, Verify. In *2021 IEEE Symposium on Security and Privacy (SP)*. 1766–1781. https://doi.org/10.1109/SP40001.2021.00037

[6] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. 2017. Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate. In *2017 IEEE Symposium on Security and Privacy (SP)*. 483–502. https://doi.org/10.1109/SP.2017.26

[7] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy, and Brian Sniffen. 2011. Principles of remote attestation. *International Journal of Information Security* 10, 2 (2011), 63–81.

[8] Cas Cremers and Martin Dehnel-Wild. 2019. Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society.

[9] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. 2017. A Comprehensive Symbolic Analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1773–1788. https://doi.org/10.1145/3133956.3134063

[10] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. 2016. Automated Analysis and Verification of TLS 1.3: 0-RTT, Resumption and Delayed Authentication. In *2016 IEEE Symposium on Security and Privacy (SP)*. 470–485. https://doi.org/10.1109/SP.2016.35

[11] Ivan De Oliveira Nunes, Sashidhar Jakkamsetti, Norrathep Rattanavipanon, and Gene Tsudik. 2021. On the TOCTOU Problem in Remote Attestation. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 2921–2936. https://doi.org/10.1145/3460120.3484532

[12] Dorothy E. Denning and Giovanni Maria Sacco. 1981. Timestamps in Key Distribution Protocols. *Commun. ACM* 24, 8 (aug 1981), 533–536. https://doi.org/10.1145/358722.358740

[13] D. Dolev and A. Yao. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory* 29, 2 (1983), 198–208. https://doi.org/10.1109/TIT.1983.1056650

[14] Aurélien Francillon, Quan Nguyen, Kasper B. Rasmussen, and Gene Tsudik. 2014. A minimalist approach to Remote Attestation. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1–6. https://doi.org/10.7873/DATE.2014.257

[15] Trusted Computing Group. 2019. TCG Trusted Attestation Protocol (TAP) Information Model for TPM Families 1.2 and 2.0 and DICE Family 1.0. https://trustedcomputinggroup.org/wp-content/uploads/TNC_TAP_Information_Model_v1.00_r0.36-FINAL.pdf Ver. 1.0, Rev. 0.36. Last accessed 27 February 2024.

[16] Gavin Lowe. 1996. Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 147–166.

[17] G. Lowe. 1997. A hierarchy of authentication specifications. In *Proceedings 10th Computer Security Foundations Workshop*. 31–43. https://doi.org/10.1109/CSFW.1997.596782

[18] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*, Natasha Sharygina and Helmut Veith (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 696–701.

[19] Ivan De Oliveira Nunes, Karim Eldefrawy, Norrathep Rattanavipanon, Michael Steiner, and Gene Tsudik. 2019. VRASED: A Verified Hardware/Software Co-Design for Remote Attestation. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1429–1446. https://www.

usenix.org/conference/usenixsecurity19/presentation/de-oliveira-nunes
[20] Ivan De Oliveira Nunes, Karim Eldefrawy, Norrathep Rattanavipanon, and Gene Tsudik. 2020. APEX: A Verified Architecture for Proofs of Execution on Remote Devices under Full Software Compromise. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 771–788. https://www.usenix.org/conference/usenixsecurity20/presentation/nunes
[21] Bryan Parno, Jonathan M. McCune, and Adrian Perrig. 2010. Bootstrapping Trust in Commodity Computers. In *2010 IEEE Symposium on Security and Privacy*. 414–429. https://doi.org/10.1109/SP.2010.32
[22] Lukas Petzi, Ala Eddine Ben Yahya, Alexandra Dmitrienko, Gene Tsudik, Thomas Prantl, and Samuel Kounev. 2022. SCRAPS: Scalable Collective Remote Attestation for Pub-Sub IoT Networks with Untrusted Proxy Verifier. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 3485–3501. https://www.usenix.org/conference/usenixsecurity22/presentation/petzi
[23] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert Van Doorn. 2004. Design and implementation of a TCG-based integrity measurement architecture.. In *USENIX Security symposium*, Vol. 13. 223–238.
[24] Samsung. 2023. Real-time Kernel Protection (RKP). https://docs.samsungknox.com/admin/fundamentals/whitepaper/samsung-knox-for-android/core-platform-security/real-time-kernel-protection/ Last accessed 27 February 2024.
[25] Samsung. n.d.. Welcome to Knox Attestation. (n.d.). https://docs.samsungknox.com/dev/knox-attestation/enhanced-attestation-v3/ Last accessed 27 February 2024.
[26] Muhammad Usama Sardar, Saidgani Musaev, and Christof Fetzer. 2021. Demystifying Attestation in Intel Trust Domain Extensions via Formal Verification. *IEEE Access* 9 (2021), 83067–83079. https://doi.org/10.1109/ACCESS.2021.3087421
[27] Muhammad Usama Sardar, Do Le Quoc, and Christof Fetzer. 2020. Towards Formalization of Enhanced Privacy ID (EPID)-based Remote Attestation in Intel SGX. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*. 604–607. https://doi.org/10.1109/DSD51259.2020.00099
[28] Vinnie Scarlata, Simon Johnson, James Beaney, and Piotr Zmijewski. 2018. Supporting third party attestation for Intel® SGX with Intel® data center attestation primitives. *White paper* (2018), 12.
[29] Benedikt Schmidt. 2012. *Formal analysis of key exchange protocols and physical protocols*. Ph. D. Dissertation. ETH Zurich.
[30] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. 2012. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *2012 IEEE 25th Computer Security Foundations Symposium*. 78–94. https://doi.org/10.1109/CSF.2012.25
[31] Ben Smyth, Mark D. Ryan, and Liqun Chen. 2015. Formal analysis of privacy in Direct Anonymous Attestation schemes. *Science of Computer Programming* 111 (2015), 300–317. https://doi.org/10.1016/j.scico.2015.04.004 Special Issue on Automated Verification of Critical Systems (AVoCS 2013).
[32] David Wagner, Bruce Schneier, et al. 1996. Analysis of the SSL 3.0 protocol. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, Vol. 1. 29–40.
[33] Stephan Wesemeyer, Christopher J.P. Newton, Helen Treharne, Liqun Chen, Ralf Sasse, and Jorden Whitefield. 2020. Formal Analysis and Implementation of a TPM 2.0-Based Direct Anonymous Attestation Scheme. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security* (Taipei, Taiwan) *(ASIA CCS '20)*. Association for Computing Machinery, New York, NY, USA, 784–798. https://doi.org/10.1145/3320269.3372197
[34] Jorden Whitefield, Liqun Chen, Ralf Sasse, Steve Schneider, Helen Treharne, and Stephan Wesemeyer. 2019. A symbolic analysis of ecc-based direct anonymous attestation. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 127–141.
[35] Johannes Wilson. 2023. Formally Verified Remote Attestation Protocols with Strong Authentication. *Linköping University* (2023). [Master's Thesis].
[36] Johannes Wilson, Mikael Asplund, and Niklas Johansson. 2023. Extending the Authentication Hierarchy with One-Way Agreement. In *IEEE 36th Computer Security Foundations Symposium (CSF)*. IEEE Computer Society, 377–391. https://doi.org/10.1109/CSF57540.2023.00025
[37] Shaza Zeitouni, Ghada Dessouky, Orlando Arias, Dean Sullivan, Ahmad Ibrahim, Yier Jin, and Ahmad-Reza Sadeghi. 2017. ATRIUM: Runtime attestation resilient under memory attacks. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 384–391. https://doi.org/10.1109/ICCAD.2017.8203803