Extending IPsec for Efficient Remote Attestation

Ahmad-Reza Sadeghi, Steffen Schulz

System Security Group, Ruhr-University Bochum
{ahmad.sadeghi,steffen.schulz}@trust.rub.de

Abstract. When establishing a VPN to connect different sites of a network, the integrity of the involved VPN endpoints is often a major security concern. Based on the Trusted Platform Module (TPM), available in many computing platforms today, remote attestation mechanisms can be used to evaluate the internal state of remote endpoints automatically. However, existing protocols and extensions are either unsuited for use with IPsec or impose considerable additional implementation complexity and protocol overhead.
In this work, we propose an extension to the IPsec key exchange protocol IKEv2. Our extension (i) allows for continuous exchange of attestation data while the IPsec connection is running, (ii) supports highly efficient exchange of attestation data and (iii) requires minimal changes to the IKEv2 protocol logic. The extension is fully backwards compatible and mostly independent of the employed low-level attestation protocol. Our solution has much less overhead than the TCG TNC design, however, we also discuss integration with TNC deployments.

1 Introduction

Secure communication between computer systems is typically established using secure channel technologies such as TLS [1] or IPsec [2]. While these protocols ensure secure transmission of data and the authenticity of the communication endpoints, they do not provide any guarantee on the integrity of the involved endpoints. In many cases however, it is highly desirable to ensure the *trustworthiness* of the involved remote endpoints, i.e., to have assurance that the remote system conform to a defined policy.

The secure remote assessment of a remote system's state is called *remote attestation*. It involves a mutually trusted attestor to assure that the possibly compromised system cannot lie about its current state. The attestor vouches for the correctness of the *attestation data* transmitted in one or more *attestation reports*. The Trusted Computing Group (TCG), a large consortium of hard- and software vendors, recently approached this problem by publishing several vendor-independent specifications to introduce Trusted Computing into the mainstream computer industry [3]. The core component of the TCG Trusted Computing Infrastructure [4] is the Trusted Platform Module (TPM) [5], a security module specifically designed to securely store and report a record of system events. Many computer vendors already ship the TPM in Laptops and PCs today. The TPM

is already used by some commercial applications such as Microsoft BitLocker [6] which is a full disk encryption software delivered with some versions of Windows Vista and "Sirrix.TrustedVPN" that is VPN infrastructure utilizing a broader set of TPM functionalities [7].

In the TCG approach to attestation, also called *binary attestation*, relevant system events are reported to the TPM in form of measurements. More specifically, SHA-1 hash values of binary code that is about to be executed are *extended* (stored) into Platform Control Registers (PCRs) of the TPM such that the order and value of all measurements can be verified. By requiring each software component to measure any other component before executing it, a chain of measurements is created from the initial bootstrapping phase to the start of individual user applications. For each running application, this chain of measurements can be followed back to the first component started in the system, which is typically part of the platform firmware (BIOS). For remote attestation of the current system state, the local TPM simply signs the current set of recorded measurements (*TPMQuote(PCRlist)*) such that a remote peer can verify an authenticated list of measurements. There have been several enhancements to this architecture: Examples are the Integrity Measurement Architecture (IMA) [8] that implements a TCG-style measurement architecture in the Linux kernel or the concept of a Dynamic Root of Trust for Measurement, where a CPU extension is used to initialize a trusted system state that can serve as a new root of the chain of measurements [9], or property-based attestation [10, 11]. Another enhancement is the concept of *Runtime Attestation*. While normal attestation typically only records the state of a program at startup, by measuring its program code and configuration, runtime attestation attempts to track or enforce the state transitions of running applications. Known approaches for such protocols either attest to a certain behavior that is enforced at runtime [12–14] or attempt to inspect the state of a running program to detect compromise [15–17]. Unfortunately, existing runtime attestation mechanisms are often tuned to specific use cases and only detect specific attacks. Several attack classes, for example using Return-Oriented Programming [18], are not yet reliably detected.

A major issue with the TCG approach to the concept of attestation is the large number of possible states that modern computer system can assume. Due to the complexity of todays operating systems and applications, it is very hard to create and maintain a list all valid states of a system. As a result, a lot of effort is invested into minimizing the Trusted Computing Base (TCB) of a system, i.e., the number and the size of components that must be trusted. Projects like NGSCB [19], EMSCB [20], OpenTC [21] and sHype [22] attempt to reduce complexity and enhance reliability and security of critical subsystems through modularization and isolation of the system components. In particular, an IPsec-based VPN service was recently presented in [23] that is optimized for security and low internal complexity. By using a microkernel-based operating system and by delegating all uncritical functionality like network card drivers and IP stack into isolated software modules, the so-called Secure VPN (sVPN) allows to create IPsec gateways with a small TCB. The obvious next step to enhance

the security of such deployments is to combine sVPN with remote attestation. *Trusted Channels*, secure (i.e., authentic, integral, confidential) channels with remote attestation, have been considered in [24–29]. However, no proposal exists that specifically targets IPsec VPNs, much less one that focuses on simplicity and allows efficient exchange of remote attestation reports at runtime, i.e., while the associated secure channel remains operational.

Contribution. We propose an extension to the IPsec key exchange protocol, the Internet Key Exchange version 2 (IKEv2) [30], to allow for continuous exchange of attestation data while the IPsec connection is running. As will be elaborated in Section 3, the IKEv2 protocol regularly establishes its own secure channel as a control channel for the actual IPsec communication channels. We propose an extension to IKEv2 to use this control channel for the exchange of remote attestation reports. This design allows for efficient exchange of attestation data during connection setup and during the whole lifetime of any associated communication channel. Thus, our solution can modularly and flexibly handle the underlying attestation protocol supporting various attestation protocols and architectures, an mentioned above (e.g., binary, property-based, IMA, etc.), and is highly suited for future developments in remote attestation. Our extension is fully backwards compatible to IKEv2 and need only minor changes to the IKEv2 protocol logic. Last but not least our solution can be implemented with significantly less components and protocol overhead than the TCG Trusted Network Connect (TNC) framework, nevertheless, we also discuss how our extension can be incorporated into TNC deployments.

Outline. We identify the requirements for our trusted channel in Section 2. Following a short introduction to the IKEv2 protocol flow and message format in Section 3, we then describe the details of our extension in Section 4. We demonstrate the security of our proposal in Section 5 and discuss the relation of our work to the TCG TNC framework in Section 6.

2 Requirements for Remote Attestation with IKEv2

The security requirements for remote attestation protocols are not difficult to identify and many solutions are known [24–29, 31]. However, as mentioned in Section 1, the practicability and scalability of available approaches is questionable. We feel that minimal complexity and modularization is the best available approach to achieve scalable trustworthy systems. By isolating critical functionality from the remaining software, the TCB of a system is expected to become less complex and thus more reliable and also more stable over time.

Our goal is thus to integrate existing and future solutions for binary, property-based or even runtime attestation protocols with system designs that feature TCBs with high modularity and low complexity, like the Secure VPN (sVPN) design presented in [23]. For successful integration, we thus identify the following technical requirements for our protocol extension:

R1 *Security.* The attestation reports must be cryptographically linked to the endpoints of the associated secure channel to prevent a compromised endpoint from relaying attestation reports of other parties (cf. [24]).

R2 *Privacy.* Confidentiality of transferred attestation messages can be a requirement depending on the usage scenario, e.g., to comply with a company's security policy.

R3 *Simplicity and Modularity.* As costs to validate and maintain software rise with its internal complexity, low software complexity is one of the main design goals of the sVPN architecture. To support this goal, the complexity added by our protocol extension should be minimal.

R4 *Efficiency.* For general usability and to limit server load, our extension must support the exchange of attestation data with minimal additional protocol overhead, message roundtrips and computational load.

R5 *Interoperability and Flexibility.* The protocol extension must be backwards compatible to IKEv2 and should support centralized management similar to TNC. As remote attestation is still a subject of research (cf. Section 1), the protocol must be extensible to support future developments in this field.

3 The Internet Key Exchange Protocol (IKEv2)

In this section, we briefly introduce the IKEv2 protocol specified in [30]. We focus on the general protocol flow and some details on the message format in order to give the reader a better understanding on the impact of the protocol extension presented in Section 4.

Overview. The Internet Key Exchange (IKE) protocol was designed as a general protocol for negotiation of Security Associations (SAs), i.e., of keys, algorithms and other attributes needed to establish a secure channel. Its most prominent application is the negotiation of *Child SAs* for IPsec, the Security Extension of the Internet Protocol. It is important for the reader to recognize that IKEv2 will always first negotiate the SA pair[1] for a secure control channel (*IKE SAs*). Within this control channel, SAs for the actual communication channels are be negotiated (*Child SAs*), refreshed or revoked without the need for further authentication. It is this control channel that we will use to transport the attestation reports.

Protocol Flow. Figure 1 depicts the basic message flow of IKEv2 and the required payloads in each exchange phase. The protocol works with pairs of messages, so-called exchanges. The first message of each exchange is sent by the *Initiator* and answered (possibly with an empty message) by the *Responder*. The standard IKEv2 protocol flow iterates through multiple phases, each of which consists of at least one message exchange with certain allowed payloads. The first phase, `INIT`, is used to exchange Diffie-Hellman public keys (K_i, K_r) and to

[1] Since SAs are unidirectional, they are typically created and managed in pairs.

negotiate attributes of the IKE SA pair ($SA1$). The resulting (unauthenticated) shared secret K_{ir} is used to generate a session key SK that protects subsequent exchanges under the IKE SA ($encap_{SK}()$). The `AUTH` exchange is started in the second phase to mutually authenticate the endpoints of the IKE SAs and to negotiate a first set of Child SAs ($SA2$) that can be used for actual data transfer. After the authentication phase succeeded, the peers may use the established IKE channel secured by the IKE SAs to transmit additional notifications or to negotiate additional Child SAs for secure communication channels (`INFO` phase).

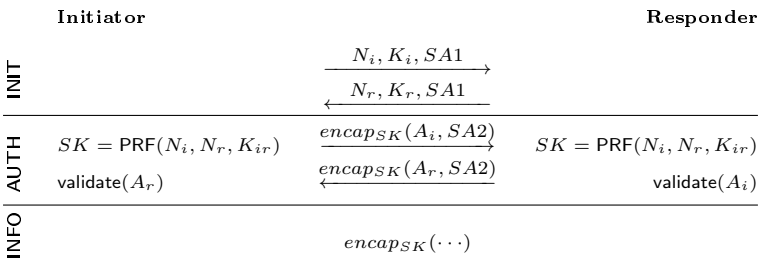

Fig. 1. Standard IKEv2 protocol flow with the IKEv2 Payloads for Diffie-Hellman key exchange (K), SA proposals for IKE SA ($SA1$) and first Child SA ($SA2$), nonces (N) and authentication of Initiator (A_i) and Responder (A_r).

Message Format. An IKEv2 message consists of the IKEv2 header followed by a list of payloads, each of which may contain several substructures. Each of the IKEv2 payloads start with a *Generic Payload Header* that specifies the type and offset of the next payload in the message. This allows Initiator and Responder to add optional or non-standard payloads to any message without interfering with the main handshake. If not supported or unexpected by the implementation of the receiver, payloads are simply ignored by jumping to the next available payload. However, the sender may also enforce processing of non-standard payloads by setting a flag in that payload's Generic Payload Header. In that case, the receiver must produce a corresponding error message if the payload could not be processed.

The *Security Association Payload* (SA Payload) used to negotiate attributes of an SA is the most complex payload in IKEv2. Each SA Payload contains a list of *SA Proposal Substructures* that represent alternative choices for the SA to be negotiated. Each SA Proposal in turn contains a list of *Transform Substructures* that correspond to the available algorithms that can be negotiated as part of the SA. The Transform Substructures are categorized according to the available types of algorithms, e.g., algorithms for encryption, employed pseudo-random functions or authentication. Finally, each Transform Substructure can contain a list of *Transform Attributes* to signal the allowed parameters for the respective

algorithm. To illustrate the recursive encoding of SA Proposals, Figure 2 (a) depicts an example SA Payload where the first SA Proposal structure proposes the use of Encapsulated Security Payload (ESP) with AES-CBC encryption and HMAC-SHA1-96 authentication. Note that the AES-CBC algorithm is supplied with a Transform Attribute specifying possible key lengths, while the key length of HMAC-SHA1-96 is implicit in the algorithm (96 bit [32]). Order and numbering of structures is used to efficiently encode preferences and available combinations algorithms. Also note that the type of an SA Proposal restricts its allowed Transform Substructures: While an SA Proposal for the Authenticated Header (AH) protocol only contains the authentication Transform, an SA of type IKE SA contains at least four different types of Transform Substructures, negotiating attributes for encryption, authentication, Diffie-Hellman group and Pseudo-Random Function (PRF). For a general introduction to IPsec we refer to [33, 34].

4 An IKEv2 Extension for Remote Attestation

Our extension is implemented in three steps. First, we define an additional SA Transformation type *Remote Attestation* as an optional component of the IKE SA. This allows a peer to propose and select remote attestation as part of the negotiated set of algorithms. Secondly, we define a new IKE payload *Attestation Data* to tunnel the actual remote attestation data. Finally, we show how the actual attestation is securely linked to the IKE SA.

4.1 Remote Attestation in the IKE SA

As explained in Section 3, the IKEv2 protocol negotiates algorithms, key lengths and other attributes of an SA by formulating them in an ordered list of SA Proposal Substructures. For each SA negotiation, such a list is sent in an SA Payload by the Initiator. The Responder parses the SA Payload, selects a set of SA parameters and returns them in a corresponding response SA Payload. Figure 2 (b) and (c) depict the format of the Transform and Transform Attribute Substructures that are encapsulated in the SA Proposals.

Since the message format of IKEv2 is extensible by design and contains large ranges of identifiers that are "reserved for private use", we can simply define a new Transform Substructure of type Remote Attestation and use its Transform ID field to identify up to 2^{16} specific remote attestation protocols. This makes the class of remote attestation algorithms available to the IKEv2 ciphersuite negotiation and, in case it is selected by both peers, allows us to define the additional semantics in Sections 4.2 and 4.3.

Unfortunately, since such protocols can be quite complex and are still subject to research, they may exist in multiple variations. While the exact version can be negotiated within the attestation protocol, merging it with the SA negotiation step is more efficient and consistent. In particular it prevents the case where

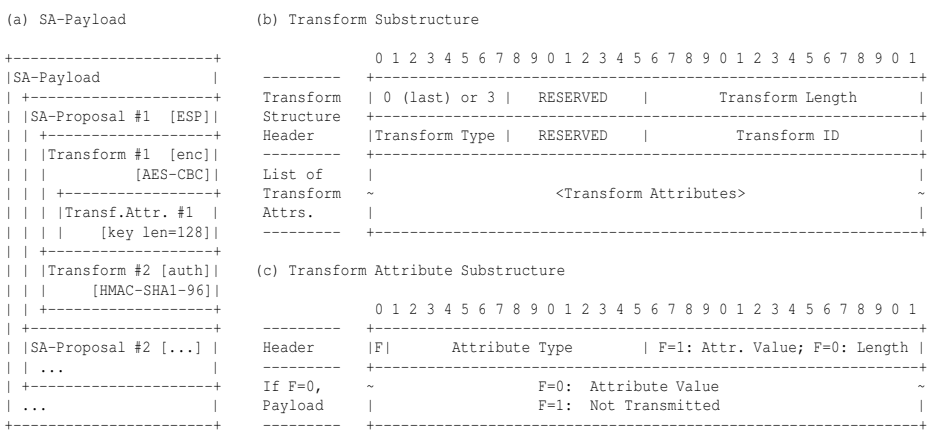

Fig. 2. Illustration of the recursive structure of an SA Payload (a). Details of the IKEv2 Transform (b) and Transform Attribute (c) structures as depicted in [30].

two parties agree on an attestation algorithm only to notice, multiple roundtrips later, that they do not support the same version of it.

We therefore also define a new Transform Attribute to encode protocol version numbers. Specifically, we use a simple Transform Attribute ($F = 1$ in Figure 2 (c)) and split the resulting 16 bit attribute value field into two 8 bit version numbers. The two numbers V_{min} and V_{max} are interpreted as an inclusive range of acceptable versions or, if V_{min} is higher than V_{max}, as a negated version range. Similar to the Key Length Attribute, multiple Version Attributes can be included in a single Transform to encode intersections of version ranges. As an example, a Remote Attestation Transform with a Transform ID set to 1 might identify the property-based attestation protocol presented in [10] and an attached Version Attribute with $V_{min} = V_{max} = 2$ might identify the revised version of that protocol from [11].

This design allows an Initiator to propose an IKE SA with a remote attestation protocol in the same way it proposes different encryption or authentication algorithms. It can suggest multiple alternative protocols at once or make remote attestation optional by also including SA Proposals without a Remote Attestation Transform. The Responder has to select one complete set of parameters and express this set in its reply, or report an error that none of the proposals is acceptable. Selecting an appropriate Remote Attestation Transform thus imposes minimal overhead for the peers and is fully backwards compatible.

4.2 The Attestation Data Payload

Once a remote attestation protocol is negotiated, the messages of this protocol must be transmitted through IKEv2. To send these messages within the IKEv2 exchange, we have to define the layout and semantics of a payload structure

that transports these messages. As creation and verification of attestation messages is a separate task that can be useful to many different applications besides IPsec, we assume that actual attestation messages are handled by some external *Attestation Service*, however, such a component could also be included into the IKEv2 server directly.

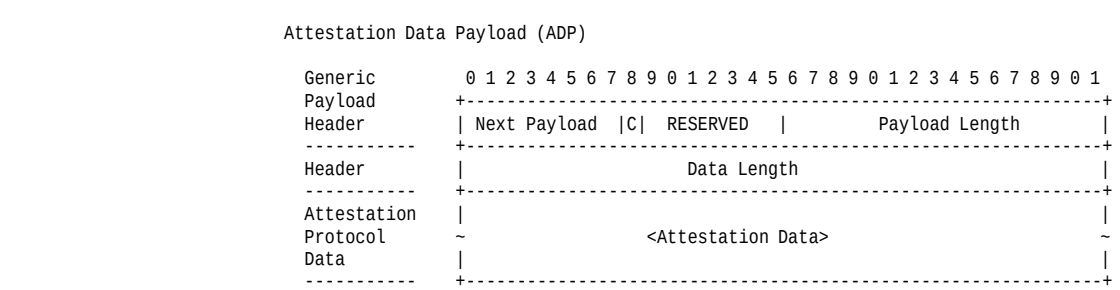

Fig. 3. New payload to transport attestation messages through IKEv2.

As shown in Figure 3, the Attestation Data Payload (ADP) consists of the Generic Payload Header, a Data Length field and an opaque Attestation Data field. To rule out possible problems with duplicated or maliciously manipulated attestation requests as well as privacy concerns, ADPs must only be transmitted protected by the IKE SA, after the last IKE_AUTH exchange succeeded. The opaque content of the payload may consist of multiple subsequent messages or logical channels, as for example supported by the TNC Client Server (TNCCS) protocol specified in [35]. The ADP defined here thus does not itself implement aggregation of multiple messages into a single payload but delegates this functionality to the Attestation Service (AS). However, to also support simple attestation protocols in an efficient manner the IKEv2 server may include multiple ADPs within a single IKEv2 message and thus transmit multiple queued attestation messages at once. In this case, the IKEv2 server is responsible for maintaining the order of Attestation Data messages. This order is already well-defined through the order of IKEv2 messages and the order of payloads within a message.

More sophisticated attestation mechanisms like property-based attestation may require the exchange of larger attestation messages than the maximum message size of a UDP datagram, $2^{16} - 1$ bytes or 64 KB, allows [10]. Following the example of [29], we thus include the separate Data Length field to allow an overall attestation message of up to $2^{32} - 1$ bytes or 4 GB to be fragmented over multiple ADPs. Since the order of IKEv2 messages and payloads within a message is well-defined and the secure channel provided by the IKE SA addresses packet loss and Denial of Service (DoS) attacks, reassembling such fragments is straightforward. Since IKEv2 messages are always exchanged in pairs, fragments are acknowledged with an empty IKEv2 message as defined in [30]. Defragmentation errors can be handled as IKEv2 payload parsing errors.

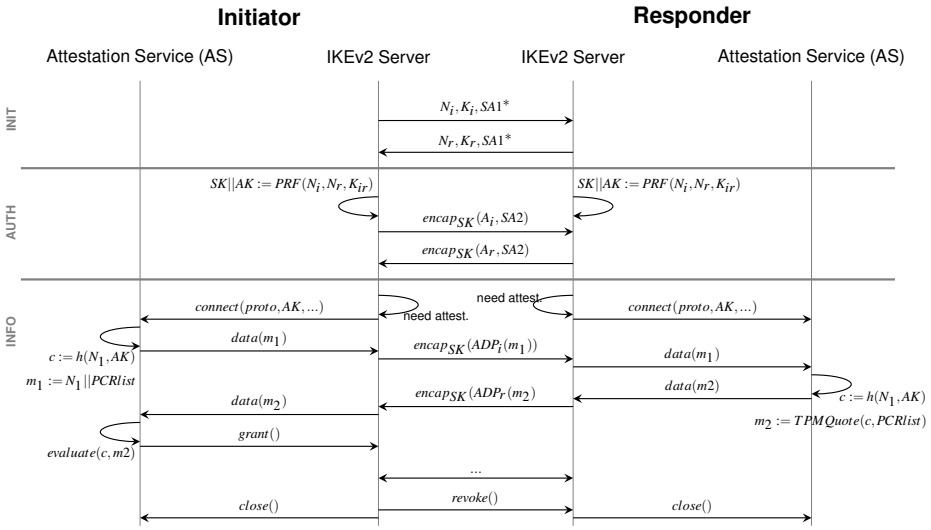

Fig. 4. Modified IKEv2 protocol flow from Figure 1 using new Transformation Structures in the $SA1^*$ payloads (cf. Section 4.1) and additional ADP payloads that carry attestation messages m_1, m_2 in the third protocol phase (cf. Section 4.2). The involved IKEv2 servers are ignorant of the attestation protocol details. They relay the messages of their responsible ASs and act upon any received policy decisions (cf. Section 4.4). A simple unilateral attestation protocol is used in phase three to clarify the distinct roles of AS and IKEv2 server.

4.3 The Shared Attestation Key

As specified in [30], the peers involved in the IKEv2 exchange initialize an internal PRF for each of the two negotiated IKE SAs. Based on the exchanged nonces and the shared Diffie-Hellman key K_{ir}, the PRF is used to extract shared fresh symmetric keys for each algorithm of the two IKE SAs. The length of the keys depend on the respective negotiated algorithms and their attributes and is computed accordingly.

We extend this definition to create a shared Attestation Key AK if a remote attestation algorithm is selected as part of the IKE SA negotiation. As shown in Figure 4, we define the extraction process as $SK||AK := \mathsf{PRF}(N_i, N_r, K_{ir})$. This is a simplified version of the extraction process defined in [30] which includes additional data into the PRF input and defines how to generate several keys for encryption, authentication etc. that we represent with SK here. Note that all the keys extracted in this manner are statistically independent from each other as long as the PRF is secure. Therefore, the order in which they are extracted is not relevant for their security. More importantly, this allows us to use the Attestation Keys (AKs) as input for attestation protocols that potentially disclose these values, e.g., when used as nonces in the TCG `TPMQuote()` operation.

4.4 Attestation Service (AS) Interface

As flexibility is one of our main goals, we do not intend to restrict our protocol extension to one or more remote attestation protocols. Instead, we delegate interpretation, verification and creation of attestation data to an external generic Attestation Service (AS). In the following, we present the semantics of the Inter-Process Communication (IPC) interface between the IKEv2 server and its (implicitly trusted) AS. A sample communication flow for a unilateral attestation of the Responder is illustrated in Figure 4.

Connect(): After the last `AUTH` exchange succeeds, the IKEv2 server uses the *connect*() call to inform the AS that an attestation of the local platform or evaluation of a remote platform's attestation report is needed. The call to the AS contains (1) the negotiated remote attestation protocol and attributes (*proto*), (2) the symmetric key AK of respective IKE SA, (3) the public key certificates, if used to authenticate the IKE SA and (4) an identifier of the corresponding network channel. Note that in case of mutual attestation, the IKEv2 server will receive attestation requests and responses under the same IKE SA, and will thus also provide them to the AS under the same channel identifier.

Data(): This call implements the exchange of attestation data between the local AS and IKEv2 server. It contains the channel identifier and the opaque attestation message m that was received or is to be sent by the IKEv2 server.

Grant()/deny(): The *deny*() call can be used by the Attestation Service (AS) at any time to revoke all Child and IKE SAs associated with the connection. The *grant*() call informs the IKEv2 server that the attestation succeeded and the associated Child SAs can be disclosed to the respective subsystems. The signaling of error messages or alternative attestation exchanges is the responsibility of the involved ASs.

Close(): This call can be issued by both, AS and IKEv2 server to signal that the respective IPC connection and its associated IKE SA shall be closed. Any associated Child SAs are revoked (*revoke*()).

5 Security Considerations

In this section, we discuss the security of the trusted channel that can be established using the extension proposed above. Since our proposal is not restricted to a particular remote attestation protocol, we will use the unilateral challenge-response attestation shown in the third protocol phase of Figure 4 to show by example that our design achieves the following security goals.

G1 Based on the security of the IKEv2 secure channel and careful choice of the attestation protocol, the IKEv2 extension allows to establish a trusted channel that meets our security requirements **R1** and **R2** of Section 2.

G2 A compromise of the attested platform can be recognized in subsequent attestation exchanges if it is detected by the employed attestation protocol.

Assumptions. To show that **G1** and **G2** can be met with the protocol shown in Figure 4, we need the following additional assumptions.

A1 *IKEv2 Security.* As specified in [30], the IKEv2 protocol establishes a secure channel based on the fresh shared keys stream that can be extracted from the PRF in the second phase. After the second phase succeeded, this channel provides an ordered exchange of authenticated and encrypted messages secure against packet loss, replay and downgrade or version rollback attacks.

A2 *TCG PKI.* A Public Key Infrastructure (PKI) exists that allows the Initiator to validate the result of the attestation report of the Responder (e.g., result of `TPMQuote()` operation plus measurement log) to gain assurance that the report is authentic and executed by a mutually trusted attestor component.

A3 *Attestation.* The attestation mechanism (e.g., the `TPMQuote()` operation) on the Responder is secure in the sense that a *compromise* of the platform's system state is reflected in subsequent attestation reports. In this context, *compromise* denotes any change to a platform's state that violates the security policy of the Initiator.[2]

A4 *IPsec Security* The security of the IKEv2 channel (**A1**) extends to the associated Child SAs and how they are used within IPsec. More precisely, the communication channels that are associated with a secure IKEv2 channel provide a secure channel with the properties negotiated in the Child SA negotiation, according to the security policies of Initiator and Responder (e.g., authentication, confidentiality, partial sequence integrity as specified in [36].

Adversary. The adversary considered here is provided with two major attack vectors. Firstly, we assume that the network channel used by Initiator and Responder to communicate is fully under control of the attacker (**V1**). Secondly, we assume that the attacker can take control the platform of the Responder at any time, even while the trusted channel is already established (**V2**), with the exception that the mutually trusted attestor component of the platform remains integral.

V1 The control of the network channel allows the attacker to launch downgrade, version rollback, replay, injection and many more attacks on IKEv2 and associated communication channels. Due to assumptions **A1** and **A4** however, these attacks are all prevented by the employed secure channel protocols. The adversary is still capable of launching a DoS attack, however, this is always possible with the given level of control on the network and thus trivial. In a more selected DoS attack, the adversary may attempt to either prevent the exchange of attestation messages after the secure channel is already established. However, as specified in Section 4.1, the use of attestation is negotiated in IKEv2 phase one and thus known and confirmed at both peers

[2] Our goal is to show that the proposed system design is sound *if* a sufficiently secure attestation protocol is used, i.e., the attestation protocol is out of scope. We thus use this definition to obviate any discussion of the attestation scheme, including the problem of runtime attestation.

as soon as authentication succeeded. As specified in Section 4.4, the IKEv2 servers will thus wait for the decision of the Attestation Service to grant or deny the use of the actual associated communication channels. In face of selective DoS in later IKEv2 exchanges, the involved ASs can simply deny all further communication due to attestation timeout. As the attestation messages are only exchanged once the IKEv2 authentication phase succeeded, requirement **R2** of goal **G1** is met in case of **V1**.

V2 When taking over control of a platform (in our example, that of the Responder) the adversary is free to modify or inspect its state, including, e.g., long term authentication keys and session keys for the secure channel established via IKEv2. However, by assumption **A3** any such action is detected by the employed attestation protocol and reflected in subsequent attestation reports if it is relevant to the Initiator[3]. Based on the example attestation protocol illustrated in Figure 4, one can easily see how the Initiator can assure in this case that any subsequent attestation reports either report the compromise or fail the authentication:

- **P1** When requesting an attestation report (m_1 with requested property list `PCRlist`), the freshness of the response is assured by including a fresh nonce (N_1) in the request that must be used in combination with a one-way function when computing the response.
- **P2** The attestation protocol must be designed such that attestation reports cannot be spoofed. In our example, this is achieved by combining the TCG `TPMQuote` operation with assumption **A2**.
- **P3** The remaining option for the adversary is to reflect the request of the Initiator to an uncompromised third party to receive a fresh and valid attestation report to answer the original request. This is prevented in our example by including the shared Attestation Key (AK) into the attestation report using a one-way function $h()$. More precisely, the Responder hashes the nonce together with the shared Attestation Key of the associated connection and uses the result $c = h(N_1, AK)$ as additional input to the digital signature computed in the `TPMQuote()` command. With growing bit length of AK, the adversary has exponentially decreasing probability that the AK' used by the third party is the same as the AK used in the connection between Initiator and Responder, so that the attestation report is linked to the respective channel endpoint as required by **R1**. Alternatively, we also provide the certificate data used to authenticate the peers of the IKEv2 channel, thus providing additional ways to meet **R1**.

Finally, the adversary may choose not to send a response at all. However, the Initiator may simply signal the IKEv2 server to close the associated connections after some timeout to address this issue. Unfortunately, the violation of **R2** is always possible (and thus trivial) if the peer that validates an attestation report is compromised and the attestation protocol discloses the state

[3] In practical systems, the platform configuration may be divided into isolated compartments that prevent the instant compromise of the whole system, thus allowing the relevant components to detect the compromise.

it attests to. This can be solved using privacy-preserving remote attestation protocols like [10, 11, 37].

With appropriate choice of the employed attestation protocol, our design thus achieves the security goals **G1** and **G2**. The argument is easily extended to multilateral attestation and repeated attestation exchanges at runtime. In fact, any attestation protocol that follows the requirements in **P1** to **P3** meets our security goals under aforementioned assumptions **A1** to **A4** if only communicating using our extension.

6 Related Work - TCG Trusted Network Connect

The TCG work group for Trusted Network Connect (TNC) published several specifications on the integration of remote attestation into existing secure channel protocols. Their proposed TNC architecture [38] is a general framework for request, transmission and validation of attestation reports: Attestation data is exchanged between multiple *Agents* on the involved network endpoints. The messages are collected from the Agents [39, 40], and encapsulated in the TNC Client Server (TNCCS) signaling protocol [35]. Two alternative protocols are specified to transport these TNCCS messages to the peer, one using the Extensible Authentication Protocol (EAP) framework [41] and one using a separate dedicated Transport Layer Security (TLS) [42] channel.

Several modern secure channel protocols support EAP, a protocol framework that supports many different authentication mechanisms as sub-protocols (*EAP methods*). The TCG thus defined the *IF-T Binding to EAP* [29] to describe a way to tunnel TNCCS messages within EAP methods (inner EAP method). Alternatively, if EAP is not available, the *IF-T Binding to TLS* [42] specifies how the TNCCS messages can be transmitted through a separate dedicated TLS [1] connection.

While the TNC framework is highly flexible and integrates well with EAP-based centralized network access control management, it fails to meet our requirements for simplicity and efficiency: The use of EAP imposes a significant protocol overhead in terms of roundtrips and relies on the secure configuration and implementation of multiple additional protocol layers. The additional layers introduced by the TNC framework aggravate this problem. Further, the design requires to repeat the EAP handshake and possibly reset the channel when additional attestation exchanges are desired after the channel is established (e.g., to report changes to the local policy of a peer at runtime). The IF-T Binding to TLS on the other hand requires a dedicated TLS channel for the exchange of attestation messages. This allows to exchange additional attestation reports at runtime, however, the cost of implementing and negotiating TLS as well as the associated certificate management is considerable. The approach is complicated by the requirement to cryptographically link the remote attestation reports to the secure channel. As we have shown, an extension of the IKEv2 protocol is the less cumbersome and more flexible solution.

6.1 TNC Compatibility

As explained in Section 2, the primary goal of our proposal is the efficient and flexible transport attestation messages over the IKEv2 protocol. From perspective of the TNC architecture, our proposal can thus be seen as a new *IF-T Binding to IKEv2* which leverages the existing secure channel.

In fact, our extension meets the requirements of the TNCCS protocol. Specifically, the requirements for *Chunking*, *Transport* and *Security* are met through transparent in-order transfer of messages of up to $2^{32} - 1$ bytes and the secure channel provided by the IKE SA[4]. Our protocol extension can thus be used to transport TNCCS messages transparently, however, with one major caveat: As our design leverages the secure channel provided by IKEv2, exchanged attestation messages are only protected during transmission between the two involved IKEv2 servers. Our protocol does not explicitly support the case where (part of) the Attestation Service is on a remote system. However, where such a design is desired, the existing IPsec implementation can be used to configure additional secure tunnels towards the AS.

7 Conclusion

In this work we proposed an extension to the IKEv2 key exchange protocol used in IPsec VPNs. We leverage the high flexibility of IKEv2 to implement the transport of remote attestation messages within the IKEv2 channel, resulting in a highly efficient and simple design. The result is particularly interesting for use with resource constrained devices or if formal verification is desired. As IKEv2 is designed as a generic key exchange server, our solution is also more versatile than previous TLS-based trusted channels. We are currently working to integrate our extension into the Turaya Secure VPN service [23], together with a simple attestation protocol to continuously report changes to the low-level IPC access control. The result can be used to build highly reliable VPN appliances based on the Turaya Secure OS, featuring a minimal TCB with a small set of security services on top of a microkernel [43, 44].

References

1. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (2008)
2. Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC 4301 (2005)
3. Trusted Computing Group (TCG): Tcg homepage. https://www.trustedcomputing.org (2009)
4. Trusted Computing Group: TCG Architecture Overview, v1.4. (2007)
5. Trusted Computing Group: TPM Main Specification, v1.2. (2005)

[4] If an insecure IKE SA is negotiated, by design the lack of security also extends to its Child SAs and thus to all communication channels.

6. Microsoft TechNet: Bitlocker drive encryption technical overview. http://technet.microsoft.com/en-us/library/cc732774.aspx (2008)
7. Sirrix AG security technologies: Homepage. http://www.sirrix.com (2009)
8. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. Research Report RC23064, IBM Research (2004)
9. McCune, J.M., Parno, B., Perrig, A., Reiter, M.K., Seshadri, A.: Minimal TCB code execution. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press (2007)
10. Chen, L., Landfermann, R., Loehr, H., Rohe, M., Sadeghi, A.R., Stüble, C.: A protocol for property-based attestation. [45]
11. Korthaus, R., Sadeghi, A.R., Stüble, C., Zhan, J.: A practical property-based bootstrap architecture. In: STC '09: Proceedings of the 2009 ACM workshop on Scalable trusted computing, New York, NY, USA, ACM (2009) 29–38
12. Alam, M., Zhang, X., Nauman, M., Ali, T., Seifert, J.P.: Model-based behavioral attestation. In: SACMAT '08: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, New York, NY, USA, ACM (2008) 175–184
13. Peng, G., Pan, X., Zhang, H., Fu, J.: Dynamic trustiness authentication framework based on software's behavior integrity. In: 9th International Conference for Young Computer Scientists, Los Alamitos, CA, USA, IEEE Computer Society (2008) 2283–2288
14. Nauman, M., Alam, M., Zhang, X., Ali, T.: Remote attestation of attribute updates and information flows in a ucon system. [46] 63–80
15. Loscocco, P.A., Wilson, P.W., Pendergrass, J.A., McDonell, C.D.: Linux kernel integrity measurement using contextual inspection. [47] 21–29
16. Petroni, Jr., N.L., Hicks, M.: Automated detection of persistent kernel control-flow attacks. In: CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security, New York, NY, USA, ACM (2007) 103–115
17. Baiardi, F., Cilea, D., Sgandurra, D., Ceccarelli, F.: Measuring semantic integrity for remote attestation. [46] 81–100
18. Buchanan, E., Roemer, R., Shacham, H., Savage, S.: When good instructions go bad: generalizing return-oriented programming to RISC. In: CCS '08: Proceedings of the 15th ACM conference on Computer and communications security, ACM (2008) 27–38
19. England, P., Lampson, B., Manferdelli, J., Peinado, M., Willman, B.: A trusted open platform. IEEE Computer **36** (2003) 55–63
20. EMSCB Project Consortium: The European Multilaterally Secure Computing Base (EMSCB) project. http://www.emscb.org (2004)
21. The OpenTC Project Consortium: The Open Trusted Computing (OpenTC) project. http://www.opentc.net (2005)
22. Sailer, R., Valdez, E., Jaeger, T., Perez, R., van Doorn, L., Griffin, J.L., Berger, S.: sHype: Secure hypervisor approach to trusted virtualized systems. Technical Report RC23511, IBM Research Division (2005)
23. Schulz, S., Sadeghi, A.R.: Secure VPNs for trusted computing environments. [46] 197–216
24. Goldman, K., Perez, R., Sailer, R.: Linking remote attestation to secure tunnel endpoints. [45] 21–24
25. Asokan, N., Ekberg, J.E., Sadeghi, A.R., Stüble, C., Wolf, M.: Enabling Fairer Digital Rights Management with Trusted Computing. Research Report HGI-TR-2007-002, Horst-Görtz-Institute for IT-Security (2007)

26. Stumpf, F., Tafreschi, O., Röder, P., Eckert, C.: A robust integrity reporting protocol for remote attestation. Revised version (2006)
27. Gasmi, Y., Sadeghi, A.R., Stewin, P., Unger, M., Asokan, N.: Beyond secure channels. [47] 30–40
28. Armknecht, F., Gasmi, Y., Sadeghi, A.R., Stewin, P., Unger, M., Ramunno, G., Vernizzi, D.: An efficient implementation of trusted channels based on OpenSSL. In Xu, S., Nita-Rotaru, C., Seifert, J.P., eds.: STC, ACM (2008) 41–50
29. Trusted Computing Group: TNC IF-T: Protocol Bindings for Tunneled EAP Methods, v1.1. (2007)
30. Kaufman, C.: Internet Key Exchange (IKEv2) Protocol. RFC 4306 (2005)
31. Trusted Computing Group: Subject Key Attestation Evidence Extension, v1.0. (2005)
32. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (1997)
33. Paterson, K.G.: A Cryptographic Tour of the IPsec Standards. citeseer.ist.psu.edu/737404.html (2006)
34. Doraswamy, N., Harkins, D.: IPsec: The new Security Standard for the Internet, Intranets and Virtual Private Networks (second edition). Prentice Hall (2003)
35. Trusted Computing Group: TNC IF-TNCCS: Trusted Network Connect Client-Server, v1.2. (2009)
36. Kent, S.: IP Encapsulating Security Payload (ESP). RFC 4303 (2005)
37. Chen, L., Löhr, H., Manulis, M., Sadeghi, A.R.: Property-based attestation without a trusted third party. In Tzong-Chen, Lei, W.C.L., Rijmen, V., Lee, D.T., eds.: Information Security — 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008, Proceedings. Volume 5222 of LNCS., Springer-Verlag (2008) 31–46
38. Trusted Computing Group: TNC Architecture for Interoperability, v1.3. (2008)
39. Trusted Computing Group: TNC TNC IF-IMC Specification, v1.2. (2007)
40. Trusted Computing Group: TNC TNC IF-IMV Specification, v1.2. (2007)
41. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP). RFC 3748 (2004) Updated by RFC 5247.
42. Trusted Computing Group: TNC IF-T: Binding to TLS, v1.0. (2009)
43. Pfitzmann, B., Riordan, J., Stüble, C., Waidner, M., Weber, A.: The PERSEUS system architecture. In Fox, D., Köhntopp, M., Pfitzmann, A., eds.: VIS 2001, Sicherheit in komplexen IT-Infrastrukturen. DuD Fachbeiträge, Vieweg Verlag (2001) 1–18
44. Alkassar, A., Stüble, C. In: Die Sicherheitsplattform Turaya. Vieweg+Teubner (2008) 86–96 (German).
45. Juels, A., Tsudik, G., Xu, S., Yung, M., eds.: Proceedings of the 1st ACM Workshop on Scalable Trusted Computing (STC'06). In Juels, A., Tsudik, G., Xu, S., Yung, M., eds.: Proceedings of the 1st ACM Workshop on Scalable Trusted Computing (STC'06), New York, NY, USA, ACM Press (2006)
46. Chen, L., Mitchell, C.J., Martin, A., eds.: Trusted Computing, Second International Conference, Trust 2009, Oxford, UK, April 6-8, 2009, Proceedings. In Chen, L., Mitchell, C.J., Martin, A., eds.: Trusted Computing, Second International Conference, Trust 2009, Oxford, UK, April 6-8, 2009, Proceedings. Volume 5471 of Lecture Notes in Computer Science., Springer-Verlag, Berlin Germany (2009)
47. Ning, P., Atluri, V., Xu, S., Yung, M., eds.: Proceedings of the 1st ACM Workshop on Scalable Trusted Computing (STC'07). In Ning, P., Atluri, V., Xu, S., Yung, M., eds.: Proceedings of the 1st ACM Workshop on Scalable Trusted Computing (STC'07), New York, NY, USA, ACM Press (2007)