TRUSTED® COMPUTING GROUP

SPECIFICATION

Symmetric Identity Based Device Attestation

Version 1.0
Revision 0.94
July 24, 2019

Contact: admin@trustedcomputinggroup.org

PUBLIC REVIEW

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

© TCG 2019

DISCLAIMERS, NOTICES, AND LICENSE TERMS

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

This document is copyrighted by Trusted Computing Group (TCG), and no license, express or implied, is granted herein other than as follows: You may not copy or reproduce the document or distribute it to others without written permission from TCG, except that you may freely do so for the purposes of (a) examining or implementing TCG specifications or (b) developing, testing, or promoting information technology standards and best practices, so long as you distribute the document with these disclaimers, notices, and license terms.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

DRAFT

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
1.00/0.94	July 24, 2019	• Initial Release of Version 1.00.

DRAFT

 © TCG 2019

CONTENTS

DRAFT

1 DOCUMENT SCOPE

This specification describes the foundational elements for remote identification and implicit attestation with DICE using symmetric cryptography. In addition to providing a strong Device Identity rooted in hardware, Device Attestation is an extension to typical attestation schemes in that it also relies, implicitly, on a device's statistically unique, cryptographically strong, identity. This solution is intended for devices containing a Device Identifier Composition Engine. The approach described in this specification builds on TCG's Device Identifier Composition Engine specification [1].

This Symmetric Identity Based Device Attestation architecture describes keys and operations for a cryptographic Device Attestation scheme. In addition to strong Device Identity and Device Attestation, one possible use for this architecture is as a foundation for a secure storage (sealing) implementation in resource constrained devices.

1.1 Key Words

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this document normative statements are to be interpreted as described in RFC-2119, *Key words for use in RFCs to Indicate Requirement Levels*.

1.2 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statement.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST requires no action).

End of informative comment

 © TCG 2019

2 TERMS AND DEFINITIONS

For the purposes of this specification, the following terms and definitions apply.

TERM	DEFINITION
Digest	The result of a hash operation
Device	highly integrated platform containing a programmable component with other optional programmable components and peripherals
Measurement	The cryptographic hash (or equivalent) of code and/or data. Note that it is implementation-specific, and out of scope for this specification, whether a measurement is over a region of memory, a firmware or software image, or some combination thereof.

DRAFT

 © TCG 2019

3 ACRONYMS

For the purposes of this document, the following abbreviations apply.

ABBREVIATION	DESCRIPTION
CDI	Compound Device Identifier
DICE	Device Identifier Composition Engine
HMAC	Hash-based Message Authentication Code
IP	Internet Protocol
MAC	Message Authentication Code
MCU	Microcontroller Unit
PAN	Personal Area Network
PSK	Pre-Shared Key
TLS	Transport Layer Security
UDS	Unique Device Secret

DRAFT

 © TCG 2019

4 INTRODUCTION

This specification describes a Device Identifier Composition Engine (DICE) [1] architecture that provides hardware-based Device Identity and Device Attestation using symmetric key cryptography. The approach taken in this specification is to provide a description of each architectural element of this solution and then to enumerate the benefits and consequences of design decisions around these elements. The solution uses a DICE Compound Device Identifier (CDI) as a basis for Device Identity with some basic assumptions. The assumptions impose constraints on the solution. For example, Symmetric Identity Based Device Attestation assumes the Device Identity will be represented cryptographically as a symmetric key. The benefit of limiting the number of assumptions is that it maximizes the set of Device Identity and Attestation scenarios this specification supports.

Even though this document is intended to enable as many different scenarios as possible, it is still necessary to make some assumptions about the environment in which this specification may be implemented. This document assumes:

1. Devices implementing this specification are connected to, and capable of communication over, some form of network. For devices that are not IP-capable, it is assumed network connectivity is achieved via a Personal Area Network (PAN) and gateway. Further, it is assumed that networked devices are differentiated in a way that allows a verifier to retrieve information about a device necessary to carry out attestation.
2. Individual devices are associated with a single infrastructure or cloud provider with knowledge of the device architecture. This assumption simplifies the end-to-end key and secret derivations but involves disclosing information about a device's identity that could be used to track the device. For scenarios in which this is not a desirable tradeoff, this document discusses options for ensuring privacy sufficient for enabling devices to safely communicate with multiple service providers during their operational lifetime.
3. Devices implementing this specification use symmetric key cryptographic algorithms like hashes or block ciphers. They may not be capable of using asymmetric algorithms. Implementation requirements for specific cryptographic algorithms are outside the scope of this document.

This specification presumes DICE support in hardware. How the Unique Device Secret (UDS) is provisioned within a device is not in scope; only that it has been provisioned.

4.1 Symmetric Cryptography

DICE was created as a solution for trusted computing on resource-constrained devices. Other TCG publications explain how to use DICE with asymmetric cryptography but do not address using symmetric cryptography. This is a problem for the most resource-constrained devices. Many 8-bit microcontroller units (MCUs) exist that provide enough computing power to support DICE but not asymmetric cryptography. To enable DICE identification and attestation on such devices, a more lightweight approach is necessary, based on symmetric cryptography. This approach uses message Authentication Codes (MACs) instead of asymmetric digital signatures. Devices that are powerful enough to feature a MAC-based DICE are powerful enough to use a MAC-based attestation and identification scheme.

DRAFT

5 ARCHITECTURE

5.1 Basic Attestation Protocol

DICE is the Root of Trust for Measurement for this architecture. It must be inherently trusted because its misbehavior cannot be detected. This architecture relies on DICE unconditionally generating the correct CDI for layer 0. Layer 0, which is also referred to as First Mutable Code, is the next link in the chain of trust. The purpose of DICE in this architecture is to establish that the device booted the First Mutable Code provided by the manufacturer. This enables detection of persistent modification of Layer 0 and above. The architecture is illustrated in Figure 1.

Start of informative comment

Note that, generally, device secrets need to be shared with the verifier before device deployment. It is not essential for secrets to be transmitted via a network. They could be communicated via standard methods for sharing and storing secret keys.

End of informative comment

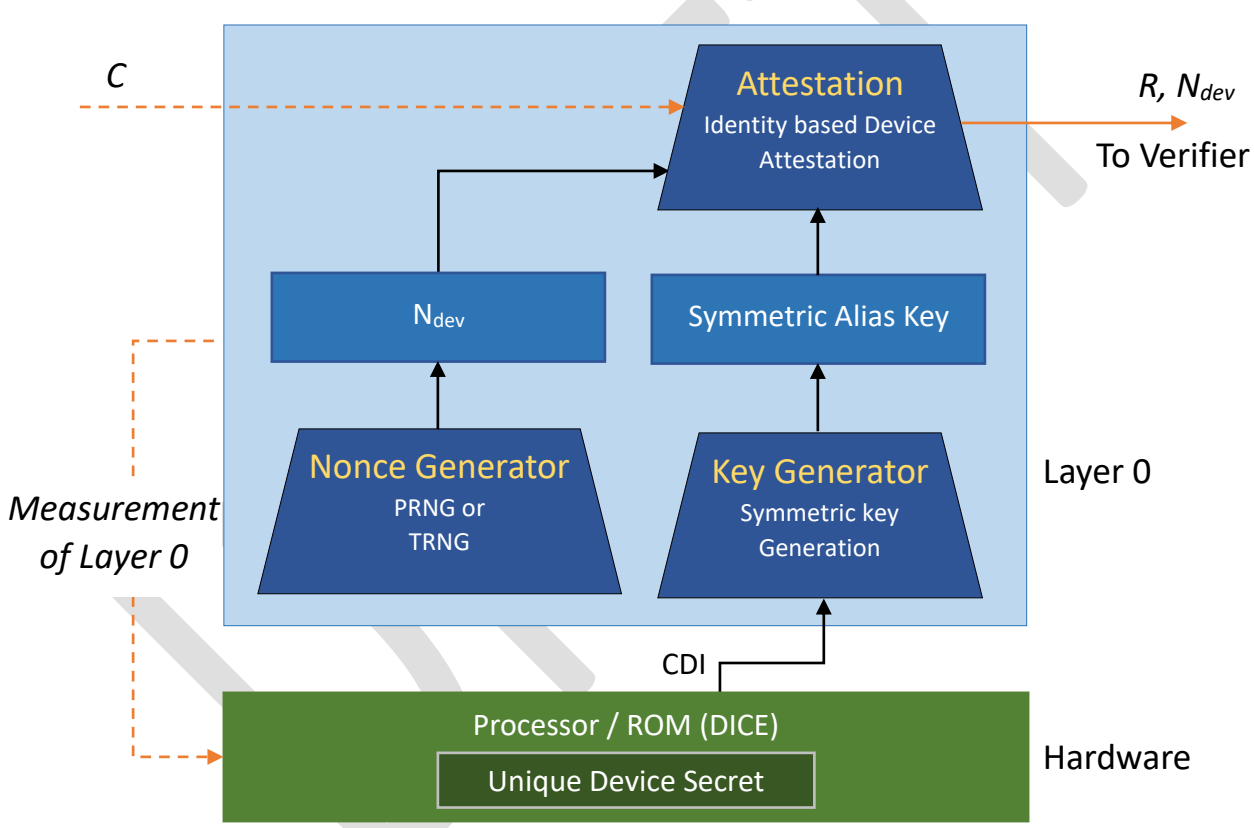

Figure 1: DICE architecture with symmetric attestation

First Mutable Code MUST use the CDI to derive an identification and attestation key. Using additional inputs for key derivation is OPTIONAL. This functionality closely resembles the Alias key found in the Implicit Identity Based Device Attestation reference [2]. Due to this similarity, and to make naming consistent across architectures, it is referred to as the Symmetric Alias Key. A Key Derivation Function (KDF) that generates a symmetric key is REQUIRED. The function MUST be a one-way function to guarantee the CDI's confidentiality

The Symmetric Alias Key is used to calculate the MACs of provided challenges. The protocol SHALL proceed as follows. The verifier generates a random challenge C that is sent to the device. The device generates a nonce N_{dev}.

 © TCG 2019

The device then calculates the response R by concatenating the server's challenge C with N_{dev} and computing the corresponding MAC using the Symmetric Alias Key.

The response R and the nonce Ndev are sent to the verifier. The verifier has a stored copy of the device's Symmetric Alias Key and uses it to calculate the expected response R'. If R and R' are equal, the device's identity and the integrity of first mutable code are verified. Figure 2 illustrates this scheme.

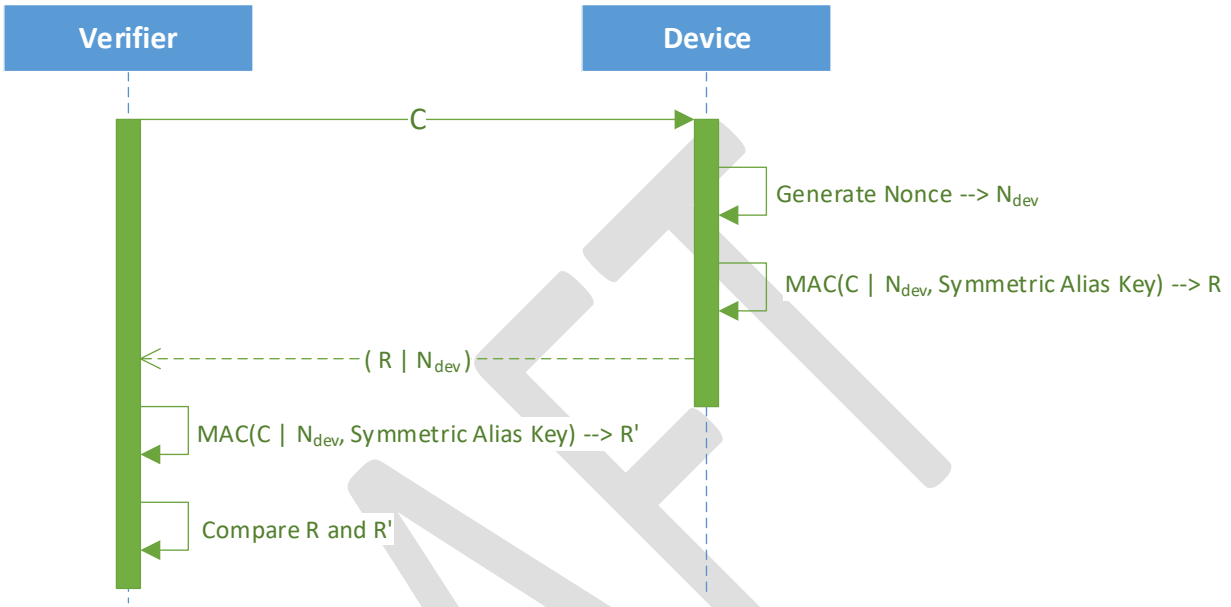

Figure 2:The basic attestation protocol

Using a challenge nonce mitigates replay attacks and using a device nonce mitigates correlation attacks. Therefore, values for C and N_{dev} SHOULD NOT be used more than once.

The device SHOULD generate its nonce in a cryptographically secure way. A hardware RNG SHOULD be used if it is available. Otherwise, a cryptographically secure PRNG MUST be used with a secret seed and a monotonic counter stored in nonvolatile memory. [3] is one example of a specification for cryptographically secure PRNGs.

The protocol illustrated in Figure 2 does not mitigate Denial of Service attacks (DoS) in this basic form. If this mitigation is required, the device SHOULD use a timer to implement an interval after successful attestation in which all incoming attestation requests are dropped. The length of this time window is to be adjusted to the application of the device.

Start of informative comment

The one-way function implemented by DICE can be used in upper layers of the software stack as well. This is especially useful in a resource constrained device. It is more efficient to use the same one-way function in DICE, the firmware layers, the key derivation, and the attestation scheme.

To derive a key, one possible implementation could apply the one-way function to a string that defines the purpose of the key (e.g. "Attestation") using the CDI or an equivalent level key.

The MAC is to be selected carefully. The most secure solution is the HMAC, which is proven to be secure, even if the underlying hash function is no longer strongly collision resistant. This is desirable for the long-term security of the device. CBC-based MACs are a more lightweight option, but with inferior security due to the lower security level of the CDI (128 bits when using AES). If DICE itself uses a hash function as described in section 5 of [1], the same hash function can be used in the attestation protocol as well, as long as the security guarantees of the hash algorithm are sufficient for attestation.

 © TCG 2019

This specification assumes the client is resource constrained and does not possess a TPM. However, no such assumption is made about the verifier. The verifier could use a TPM to store the HMAC keys and perform the verification. A verifier's use of a TPM adds more security to the overall process. Further, if the verifier possesses a TPM then the HMACs of the DICE devices it is responsible for verifying could even be included in a TPM-based attestation of the verifier. These TPM-equipped verifiers could then provide "all-up" attestation statements to, for example, a centralized authority or relying party.

End of informative comment

5.2 TLS-PSK Based Attestation Protocol

DICE-based attestation using TLS-PSK can also be deployed on devices. A pre-shared symmetric key (PSK) is used for attestation. To use TLS-PSK for attestation, this PSK is derived from the CDI using the same deterministic symmetric key generation method that is applied to derive the Symmetric Alias Key. The verifier is also in possession of the CDI (either calculated based on the UDS or provided by the device manufacturer) and derives the same symmetric PSK using one of the provided PSK ID hints. The key generation method is applied using that hint and the CDI or an equivalent level key to generate the PSK for a TLS-PSK handshake. If the TLS-PSK based protocol is used alongside the basic protocol, the CDI-derived keys MUST NOT be the same. Figure 3 shows the architecture for the key generation when using the TLS-PSK based Attestation protocol.

For attestation, the device and the verifier execute a TLS-PSK handshake initiated by the server. Due to the dependency of the PSK on the CDI, a successful handshake is implicit attestation of the device's identity and firmware integrity.

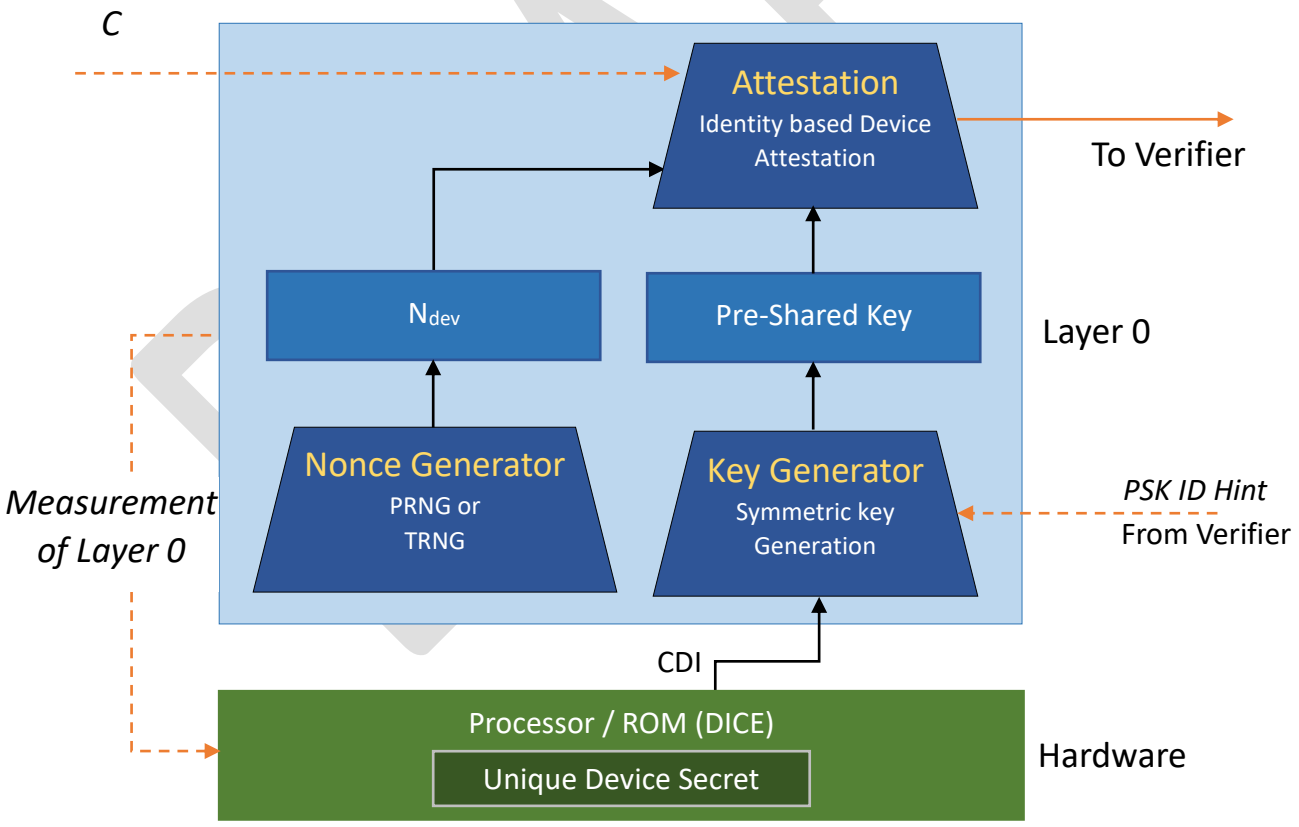

Figure 3: Key derivation for TLS-PSK based attestation

All messages and objects described in the following section are specified in [4] if not stated otherwise. The DICE device in this specification corresponds to the client in the TLS-PSK specification whereas the verifier corresponds to the TLS-PSK server. The protocol SHALL proceed as follows. The verifier requests that the client initiates attestation

 © TCG 2019

by sending an attestation request to the device. The device generates the device nonce and sends a Client Hello message as specified in [4] to the verifier. This message contains ID hints for each potential PSK. The verifier generates the server nonce, selects one of the offered PSK ID hints, includes both in a Server Hello message and transmits this to the device. Both parties derive the Pre-Shared key (PSK) using the CDI of the device and the PSK ID hint. Both the verifier and the device derive a key schedule for all subsequent messages from the PSK, the Server Hello and the Client Hello message. This terminates the key exchange phase of the handshake.

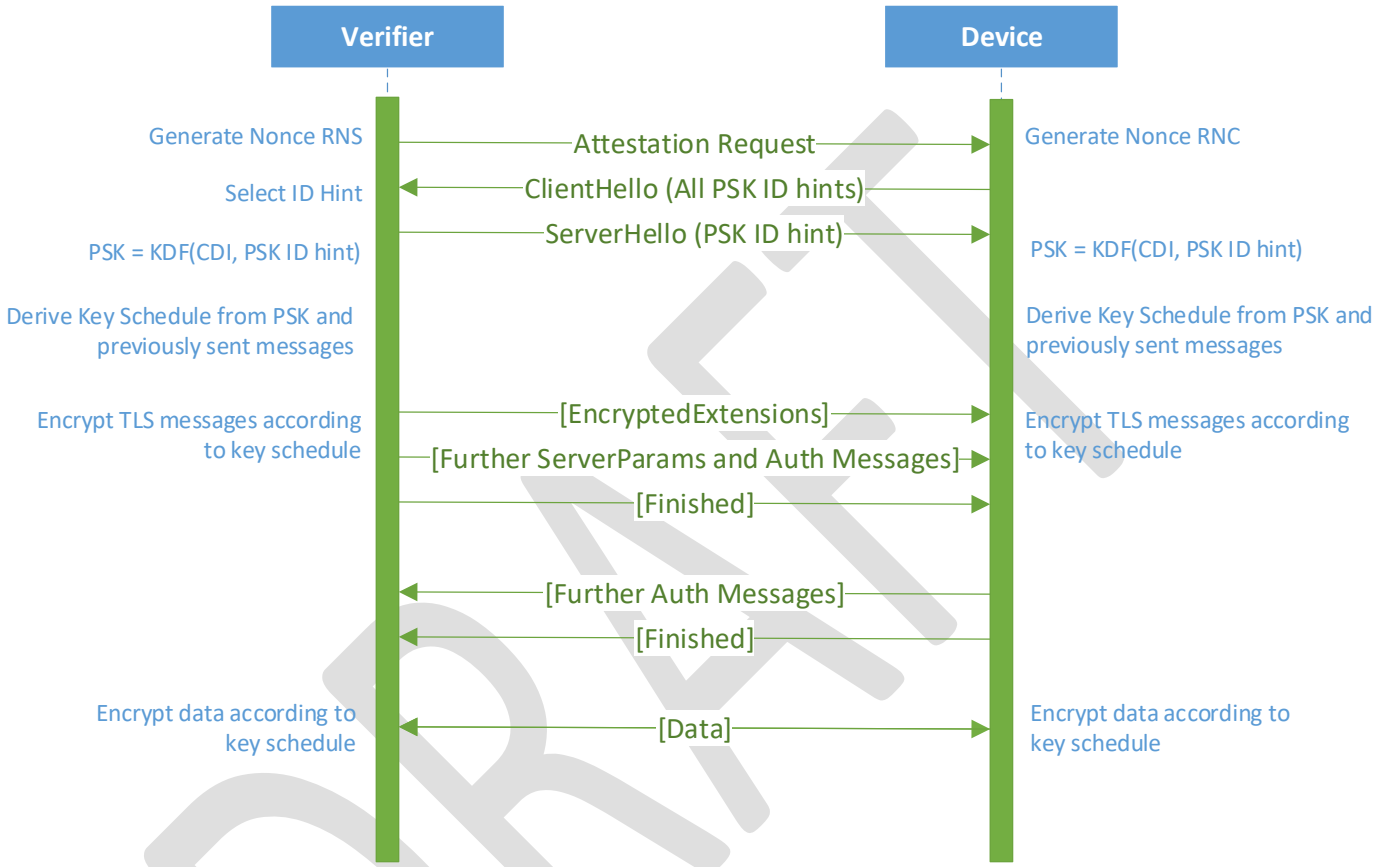

Figure 4: The TLS-PSK handshake with a CDI derived PSK

The generated key schedule is used to encrypt subsequent TLS messages. Server parameters and authentication are sent according to the TLS specification. Since the PSK is based on the CDI and all further keys are derived from the PSK, a successful TLS-PSK handshake is an implicit attestation of the device's identity and firmware integrity. The TLS channel that has been set up MAY now be used for application data. Figure 4 shows the TLS-PSK handshake.

5.3 Provisioning

Provisioning is less flexible when using a symmetric identity scheme as compared to an asymmetric identity scheme. The verifier MUST be in possession of the attestation key or PSK, depending on the protocol in use. Since knowledge of the attestation key is sufficient to fake attestations, it is necessary to provision the DICE under conditions of physical security in a trusted environment, for example, during some stage of device manufacturing. Critically, to compute the CDIs for future updates of the device, the verifier needs to have knowledge of the device's UDS.

Start of informative comment

 © TCG 2019

Device UDS values are obviously sensitive information that a verifier needs to handle with greatest care. Device secrets should never be stored decrypted outside of an area with restricted access.

It is recommended that verifiers use a Hardware Security Module, like a TPM, to securely contain the UDS of devices during computations. The TPM can also be used for the computation of CDIs for firmware updates. For this, the use of a TPM for UDS storage and the HMAC function as the key derivation function is highly recommended. On a verifier, for the secure derivation of a device's CDI, the firmware is measured, and the measurement is passed to the TPM. The TPM then computes the HMAC of the measurement with the UDS as the key. The resulting HMAC can now be exported from the TPM or kept within the TPM for the verification of attestation. The UDS can be secured by the TPM with any combination of Enhanced Authorization Policies. The TPM provides a secure and trustworthy environment for all required crypto operations on the verifier.

This specification assumes client devices are resource constrained and do not possess a TPM.

End of informative comment

5.4 Delegation of Verification

If the device owner delegates verification of attestation information to a third party, key material MUST be transferred securely to the third party. The transferred key depends on the privileges the third party is granted. Table 1 lists the third party's privileges and the corresponding key that is to be transferred.

THIRD PARTY'S PRIVILEGES	KEY TO BE TRANSFERRED
Attestation with fixed Symmetric Alias Key / PSK ID hint	Symmetric Alias Key / PSK
Attestation with dynamic Symmetric Alias Key / PSK ID hint	CDI
Attestation and Firmware Updates	UDS

Table 1: Keys needed by third-party verifier.

Like the provisioning of any key material, any transfer of secret values is to be handled with appropriate security measures. A communication channel that forces the third party to authenticate itself SHOULD be chosen. The keys SHOULD be encrypted using end-to-end protocols that provide a state-of-the-art security level and integrity protection. An example of such a secure communication channel is an up-to-date version of the TLS-protocol with sufficiently secure cipher suites.

Start of informative comment

If the third party uses a TPM-based system for key storage and attestation verification, the TPM can also be used to secure the communication channel for the transfer of the attestation keys.

Note that revocation of Symmetric Alias Keys and PSKs is not addressed in this specification. It is up to a verifier to know, at any point in time, what keys and/or client devices are valid.

End of informative comment

5.5 Using Attestation Protocols in Upper Layers of DICE Hierarchy

Although the attestation protocols are designed to work with the First Mutable Code in a DICE-enabled device, they are not limited to that. As described in [5], DICE can be the fundamental building block for a chain of trust where each layer of software is provided with a key, similar to a CDI, and uses it to calculate another key for the next layer of code that is executed. In that case, the attestation protocol SHOULD be adjusted in a way that the CDI is mapped to the

 © TCG 2019

respective level key. If the attestation takes place in layer N, the key of layer N is to be used for key derivation instead of the CDI. Figure 5 illustrates this.

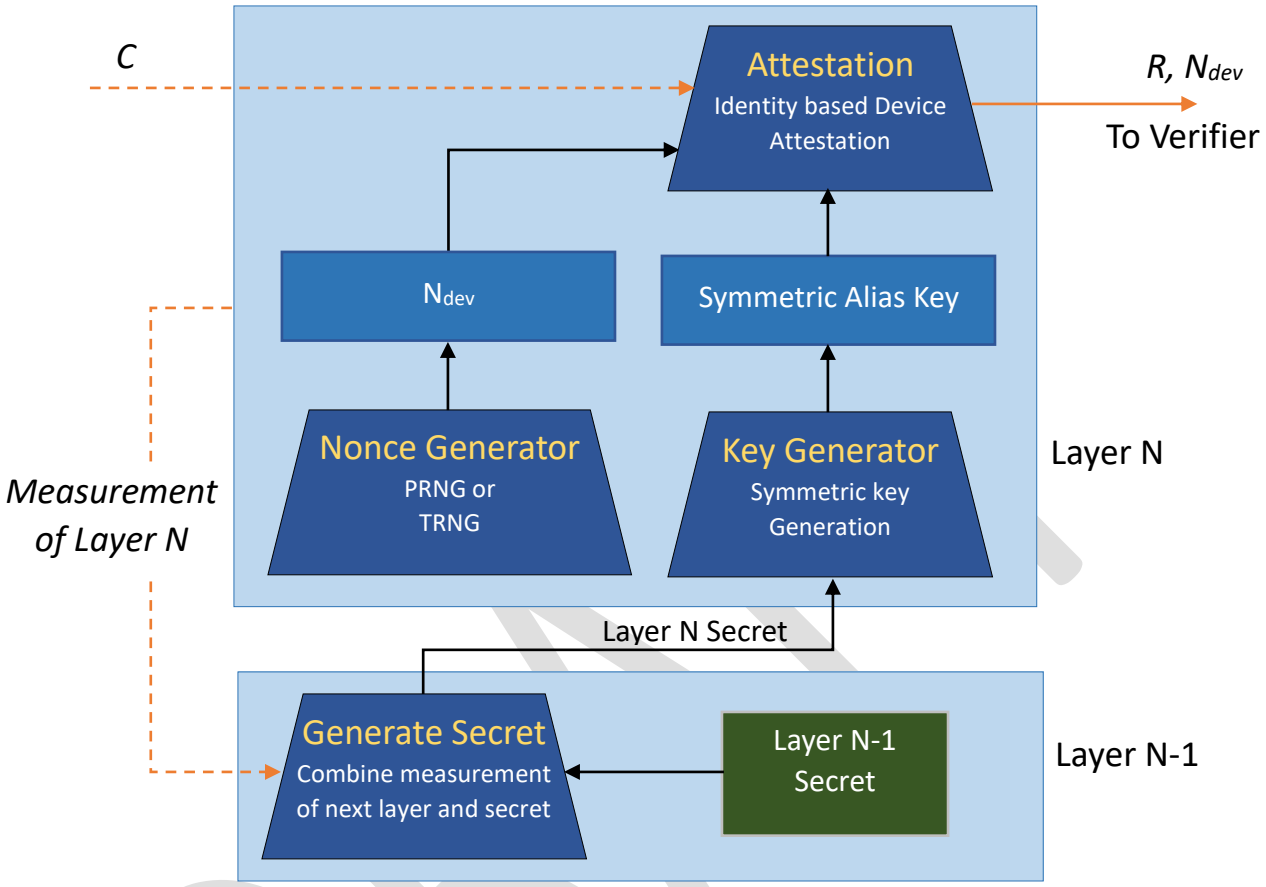

Figure 5: Key derivation for the basic attestation protocol when applied to a higher level of software

 © TCG 2019

6 References

[1] TCG, "Hardware Requirements for a Device Identifier Composition Engine," [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf. [Accessed 15 August 2018].

[2] TCG, "Implicit Identity Based Device Attestation," [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Arch-Implicit-Identity-Based-Device-Attestation-v1-rev93.pdf.

[3] National Institute of Standards and Technology, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators," 15 August 2018. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final.

[4] IETF, "The Transport Layer Security (TLS) Protocol Version 1.3," [Online]. Available: https://tools.ietf.org/html/rfc8446. [Accessed 15 August 2018].

[5] P. England, A. Marochko, D. Mattoon, R. Spiger, D. Wooten and S. Thom, "RIoT - A Foundation for Trust in the Internet of Things," [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/06/RIoT20Paper-1.1-1.pdf. [Accessed 15 August 2018].

DRAFT

7 Background

7.1 Device Identifier Composition Engine

The Device Identifier Composition Engine, or DICE, is a hardware/firmware capability that generates a cryptographically unique value, called the Compound Device Identifier, based on a Unique Device Secret and the measurement of the First Mutable Code that runs on a device. This section provides a summary of the DICE specification. Readers familiar with DICE may ignore this appendix.

7.1.1 Purpose

The Device Identifier Composition Engine provides a way to know what mutable code is running on a specific device. This is essential for strong Device Identity. This strong Device Identity and the DICE approach to protecting secrets and keys, also provides the foundation for Attestation and Data Protection (Sealing).

7.1.2 Unique Device Secret (UDS)

The Unique Device Secret is a statistically unique, device-specific, secret value. The UDS may be generated externally and installed during manufacture or generated internally during device provisioning. The UDS must be stored in non-volatile memory on the device, e.g., eFuses, or any other suitably protected NV storage to which the DICE can restrict access. See the DICE specification [1] for details.

7.1.3 Compound Device Identifier (CDI)

The DICE's immutable code measures the device's First Mutable Code. This measurement is combined with the device specific UDS to form the Compound Device Identifier. The CDI is unique not only to the device, but to the combination of the device, the cryptographic identity of the device's First Mutable Code and, optionally, configuration data.

7.1.4 Implementation

This Device Identity and Attestation solution is intended to support any DICE implementation that is compliant with the Device Identifier Composition Engine specification.

DRAFT

 © TCG 2019