Trusted Platform Architecture Hardware Requirements for a Device Identifier Composition Engine

Family “2.0”
Level 00 Revision 69
December 16, 2016

Committee Draft

Contact: admin@trustedcomputinggroup.org

- *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

Public Review

TCG

Disclaimers, Notices, and License Terms

Acknowledgements

The TCG would like to acknowledge the contribution of those individuals (listed below) and the companies who allowed them to volunteer their time to the development of this specification.

Special thanks are due to Dr. Ronald Aigner and Dr. Nicolai Kuntze who served as Co-Chairs of the group that produced this specification.

The TCG would also like to give special thanks to David Wooten, Graeme Proudler, and Ronald Aigner who were the editors of this specification.

Contributors

American Express - Wael Ibrahim
AMOSSYS - Dimitri Kirchner
ANSII - Pierre Chifflien
Atmel - Todd Slack
BSI - Dietmar Wippig
CESG - Paul Waller
Fraunhofer AISEC - Carsten Rolfes, Steffen Wagner
Fraunhofer Institute for Secure Information Technology (SIT) - Andreas Fuchs
Freescale Semiconductor - Carlin Covey, Lawrence Case
Fujitsu Limited - Seigo Kotani, Yoshitaka Hiyama
Google Inc. - Darren Krahn
Graeme Proudler
High North Inc - Ira McDonald
HP Inc. - Jim Mann
Huawei Technologies Co., Ltd. - Carsten Rudolph, Kenny Li, Nicolai Kuntze
IBM - Guerney Hunt
Infineon - Ga-Wai Chin, Georg Rankl, Johann Schoetz, Steve Hanna
Intel Corporation - Giroyuki Koike, Monty Wiseman, Will Arthur, Thomas Bowen
Juniper Networks, Inc. - Guy Fedorkow
Microsoft Corp. - David Wooten, Paul England, Rob Spiger, Ronald Aigner, Stefan Thom
Nuvoton - Dana Cohen
NXP Semiconductors - Lawrence Case
Security Innovation - Brenda Baggaley, Michael Cox
STMicroelectronics - Andrew Marsh, Benoit Houyere, Enrico Gregoratto, Charly Villette, Serge Fruhauf
Thales Communications & Security - Ben Thomas, Joan Mazenc, Nicolas Waroquier
United States Government - Apostol Vassilev, Andrew Regensheid, Eugene Myers, Jessica Fitzgerald-McKay, Daren Bennett, Jonathan Hersack, Tom Brostrom, Mike Boyle, Stanley Potter
Wave Systems - Andrew Tarbox
Winmagic - Garry McCracken, Rob Decarux, Derek Tsang

Contents

Trusted Platform Architecture

Hardware Requirements for a Device Identifier Composition Engine

1 Scope and Audience

This specification describes the hardware requirements and process for creating an identity value that is derived from a Unique Device Secret and the identity (a condensed cryptographic representation) of the first mutable code. This specification calls the derived value the Compound Device Identifier. The composition of the Compound Device Identifier may include hardware state or configuration that influences the execution of the first mutable code.

One of the possible uses of the Compound Device Identifier is to attest to the trustworthiness of an embedded device.

The intended audience for this document is designers of programmable components when they do not have access to a TPM.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] ISO/IEC 10118-3, Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash functions

[2] TPM Library Specification; Family 2.0; Level 00; Revision 01.16 or later

3 Terms and definitions

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

3.1
CDI

the Compound Device Identifier is a set of data used to identify the software running on a system that was used to generate this data

3.2
digest

result of a hash operation

3.3
device

a platform that integrates a programmable component with other optional programmable components and peripherals

3.4
DICE

the Device Identifier Composition Engine is immutable and creates the CDI

3.5
measurement

a cryptographic hash (or equivalent) of code or data

3.6
UDS

the Unique Device Secret is known only to the manufacturer and the DICE, and is used in the creation of the CDI by the DICE

4 Symbols and Abbreviated Terms

4.1 Symbols

For the purposes of this document, the following symbol definitions.

A || B	concatenation of B to A
F()	denotes a function **F**
H()	denotes the hash function
HMAC(k, m)	denotes the HMAC function over message *m* using key *k*

4.2 Abbreviations

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

Abbreviation	Description
TCG	Trusted Computing Group
TPM	Trusted Platform Module

5 Introduction

The Compound Device Identifier (CDI) is derived using both the Unique Device Secret (UDS) and the measurement of the first mutable code that runs on the platform. It can optionally include measurements of hardware state information and configuration data that influences the execution of the first mutable code. The CDI is generated by the Device Identifier Composition Engine (DICE), which has exclusive access to the UDS after reset and before transferring control to the measured mutable code. The UDS is provisioned by the manufacturer in any way that is consistent with this specification. The general process is shown in Figure **1** with an illustration of the computation of the CDI. Any revision or change in the UDS or any of the measured components results in a different value for the CDI.

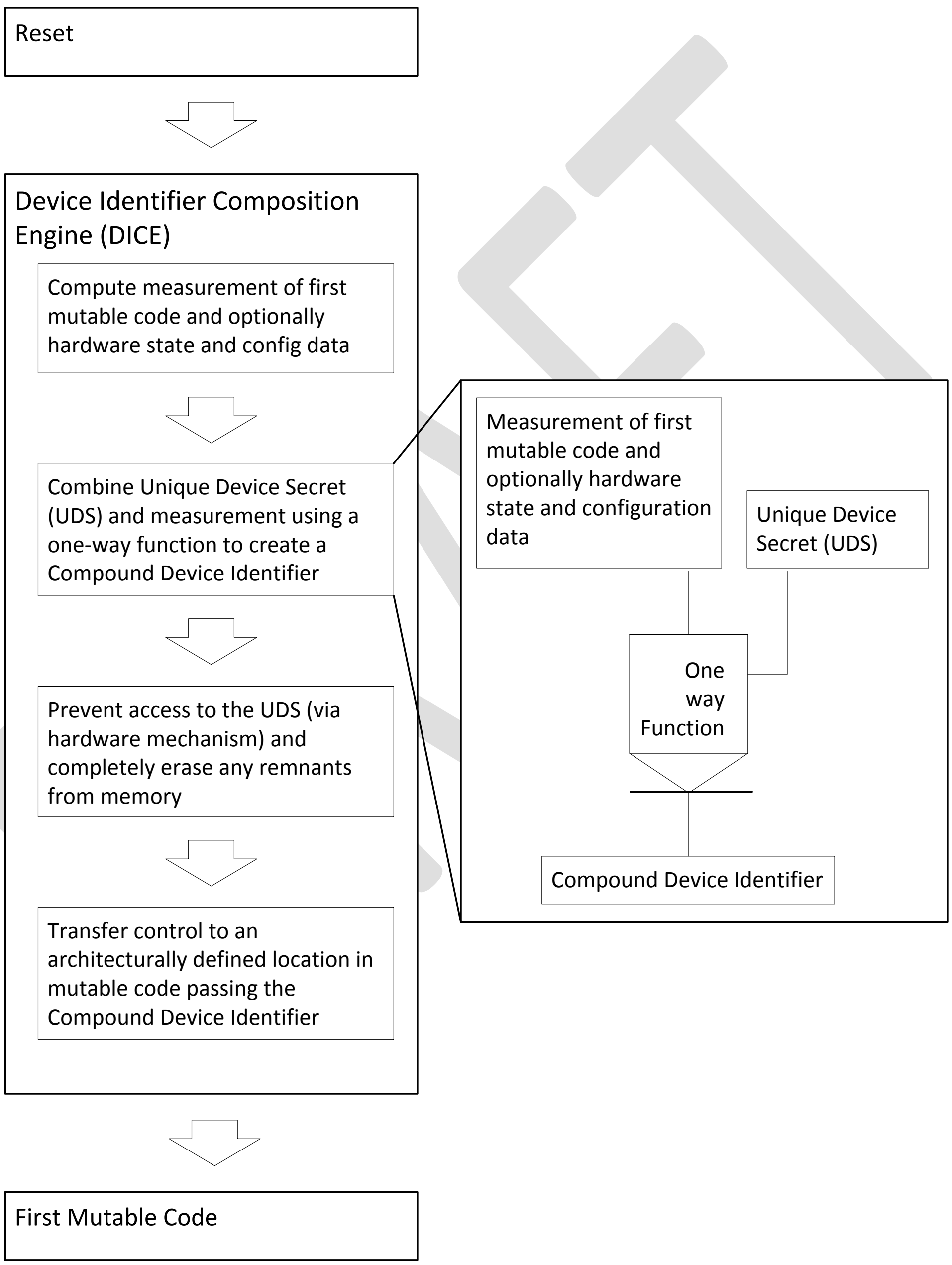

Figure 1: Compound Device Identifier Derivation Process

The UDS and the measurement of mutable code must be cryptographically mixed in a way that it is infeasible to use the CDI and the code measurement to recover the UDS. This may be accomplished by the DICE using a secure hash algorithm to hash the concatenation of the two values. Alternatively, the two values could be used in an HMAC with the UDS as the HMAC key. An HMAC would provide a higher level of protection for the UDS than would a simple hash. The specific method to combine the values is the manufacturer's choice, because it does not affect interoperability.

The secure hash function is:

$$\mathbf{H}(UDS \,||\, \mathbf{H}(Mutable\ Code)) \tag{1}$$

The secure HMAC function is:

$$\mathbf{HMAC}(UDS,\ \mathbf{H}(Mutable\ Code)) \tag{2}$$

Where:

UDS	the Unique Device Secret
$Mutable\ Code$	is code not in ROM that will be executed after all DICE operations (see Section 6.4) are finished (no mutable code is executed prior to DICE execution)

The HMAC operation takes a little more time but provides the UDS with twice the level of protection of the simple hash in (1), as described in NIST SP800-57.

The device is responsible, where required, to protect access (read, write, and modify) to the CDI. It may not be possible for the mutable code to protect the CDI. How protection of the CDI is achieved is outside the scope of this specification.

6 Requirements

6.1 Unique Device Secret properties

UDS values MUST be uncorrelated and statistically unique.

The UDS MUST NOT be used as an identity value by any other entity.

The device MUST have a UDS that has at least the same security strength as used in the attestation process of the device. The attestation process reports the software state and identity of the device.

When the attestation process is determined by software that is not under control of the device manufacturer, the size of the UDS SHOULD be at least 256 bits.

NOTE 1 The value of 256 bits is suggested, because the use of SHA1 hashing algorithm has been deprecated. Using more bits for the UDS increase chances of longevity of the implementation.

The UDS SHOULD NOT be rewritable.

NOTE 2 Change of the UDS will change the identity of the device. In most cases, changes to the UDS will prevent proper device attestation and access to previously stored device secrets.

6.2 Device Identifier Composition Engine properties

The DICE implemented on the device SHALL be immutable (unchangeable).

NOTE 3 The DICE becomes immutable at the end of the manufacturing process of the device.

The DICE MUST have exclusive read access to a UDS.

NOTE 4 This means that the packaging of the programmable component that implements DICE will normally preclude use, reading, and observation of the UDS by an entity other than DICE.

NOTE 5 Typically, read access to the storage location containing the UDS will be enabled when a hardware event, such as a reset, causes the DICE to begin execution. Then read access of the storage location would be disabled by a software command. Other implementations are possible.

If the device has a debug port or debug mode:

- The debug port or debug mode SHALL only be enabled at reset or when explicitly enabled by software that executes after the DICE.
- When the debug port or debug mode is enabled, any attempt to read the UDS (including from the DICE) SHALL be denied or produce a value that is uncorrelated with the UDS.

NOTE 6 Any constant value such as all 0's is an uncorrelated value.

6.3 Compound Device Identifier properties

Specification of normative requirements for the CDI is outside the scope of this document.

NOTE 7 The device may need dedicated hardware to protect access (read, write, and modify) to the CDI.

NOTE 8 If hardware is unavailable to protect the CDI, then mutable code provided by the manufacturer is responsible for reducing opportunities for exposure of the CDI to unauthorized entities. Devices that leak a CDI produced from measurement of authorized mutable code may be vulnerable to a replay attack and impersonation.

NOTE 9 Device manufacturers are encouraged to use best practices (for example: ISO/IEC 27034) to prevent leakage of the CDI. Measures taken may include:

- Avoid design, coding, and logic errors.
- Erasure of the CDI immediately after its use (e.g. in RAM, registers, cache).

NOTE 10 A CDI that has been leaked by mutable code should be made obsolete in part by updating manufacturer mutable code. This will cause the DICE to produce a new CDI and in addition should remove the cause of the leak.

6.4 DICE Operation

The DICE SHALL execute without interference or alteration each time the device is reset, prior to the execution of any mutable code on the device.

Before execution of mutable code, the DICE SHALL combine the UDS with the measurement of the first mutable code to be executed in such a way that the UDS cannot be deduced from the CDI, even if the measurement is known.

The DICE SHALL create this CDI using a one-way function with at least the same cryptographic strength as the UDS.

NOTE 11 According to NIST SP800-57, Part 1; using a hash algorithm in an HMAC provides as many bits of security as the number of bits in a digest but the same algorithm used in a hash would only provide about half as many bits. Using the UDS as a HMAC key would make the security strength as strong as the UDS.

Before execution of mutable code access to the UDS SHALL be disabled until the next reset.

NOTE 12 Disablement can be achieved, for example, by placing the UDS into read-once memory, by an explicit software instruction, by hardware that recognizes whether the instruction pointer is inside the range of DICE instructions and only allows access to the UDS from that range, by executing only DICE on a secure coprocessor and allowing access to the UDS only from the secure coprocessor. Other implementations are possible.

Before execution of mutable code, the DICE SHALL securely erase any values that could be used to determine the UDS.

The DICE SHALL write the CDI to a location to which the measured mutable code has exclusive access as long as mutable code requires exclusive access.

NOTE 13 The mutable code is expected to use and erase the CDI and any values that could be used to determine the CDI. While mutable code uses the CDI, it needs the ability to prevent access to the CDI and disclosure of the value.

NOTE 14 Access includes read, write, and modify.

NOTE 15 Use of the CDI by the mutable code is outside the scope of this document.

The DICE SHALL cause the device to execute mutable code starting at an architecturally defined address in the range of the mutable code that was measured.