Extend MCU Security Capabilities Beyond Trusted Execution with Hardware Crypto Acceleration and Protection

Saurin Choksi, Paul Kimelman, Durgesh Pattamatta
NXP Semiconductors, San José, CA, USA

Abstract

This paper discusses our approach to crypto acceleration and asset protection using novel techniques that help bring high levels of security to low-cost microcontrollers with minimal power and area penalty. CASPER, our asymmetric cryptography acceleration engine, aims to optimize crypto algorithm execution (e.g., RSA, ECC). It is built on a hardware-software partitioning scheme where software functions map asymmetric crypto functions to the hardware modules of the accelerator, delivering sufficient flexibility to software routines to enable mapping of new algorithms. Further efficiency is achieved by making use of the co-processor interface on the Arm® Cortex®-M33 core. Important assets such as keys, proprietary and/or licensed application software are protected against side-channel analysis or cloning using SRAM PUF and PRINCE. SRAM PUF technology enables secure storage of root-of-trust keys and user keys by exploiting the deep sub-micron process technology variations. PRINCE is a low-latency lightweight cryptography algorithm implementation in hardware that allows encrypted non-volatile storage and real-time, latency-free decryption of the execution code.

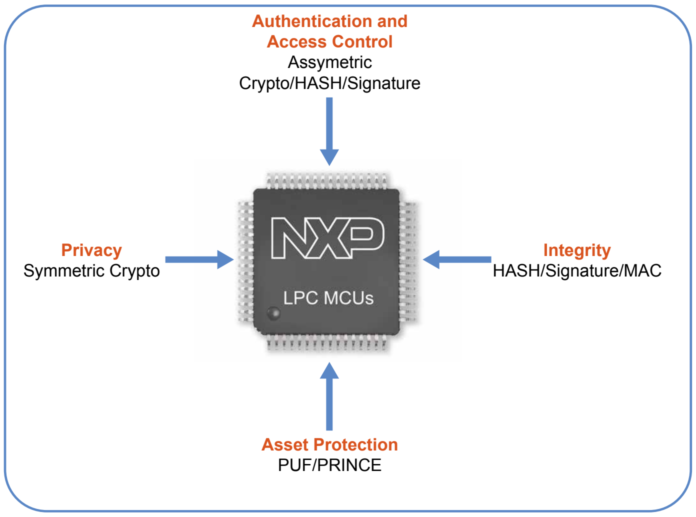

Figure 1: Security Matrix

I. INTRODUCTION

With the advent of the IoT, connected devices are proliferating and billions of them are expected to exist in the next few years. This trend of a connected world through IoT is bringing the need for security and asset protection to the smallest microcontrollers. Even simple edge-node IoT devices need higher security features, as shown in Figure 1, to support privacy, integrity, asset protection and access control. However, cost and power are critical considerations for low- to mid-level microcontrollers operating within a tight power budget. Processors based on Armv8-M architecture with TrustZone® technology provide the fundamental building blocks with secure trusted execution. However, to build a secure system, additional features as described in Arm's Platform Security Architecture (PSA) framework [6] are necessary.

NXP offers products that use the Armv8-M based Cortex-M33 core and fulfill these PSA requirements. Let's explore some elements critical to securing an IoT edge node.

II. CASPER

Signature generation and signature verification are mandatory steps for device certificate provisioning, secure boot and update. SA2048 is a more commonly used algorithm for the signature function. However, some applications are moving to newer algorithms such as ECDSA for more robustness against malicious attacks. Secure cloud connectivity makes ECC-based asymmetric key generation support a necessity; Elliptic Prime Curve 256b (EC_prime256v1), 384b (EC_secp384r1) and 521b (EC_secp521r1) are examples of the most in-demand curves.

RSA- and ECC-based asymmetric cryptographic algorithms are compute-intensive (high operation count), inherently slow (for large and very large math operations) and power hungry if done purely in software. Since asymmetric crypto algorithms are primarily used in start-up routines or session set-up phase, slow execution can have a negative impact on device boot time, wakeup time, or pairing time. Pure hardware implementation is an alternative; however, these algorithms keep evolving and new curves are introduced on a regular basis to address the latest security threats and contain more intense compute operations. Hence, a fixed hardware solution can quickly become obsolete.

The CASPER crypto accelerator solves this issue by offering an efficient hardware-software partitioning approach. It is designed to be a very generic engine, which can be applied to all manner of cryptographic algorithms in combination with software, including asymmetric public-key (e.g., RSA and various flavors of ECC) and the related Diffie-Hellman key exchange methods, exponentials generating functions, non-standard large number algorithms, and in some cases, symmetric block ciphers, stream ciphers, and hashes.

The CASPER cryptographic engine is normally used in conjunction with other standard hardware blocks for hashing (SHA) and symmetric cryptography (AES). However, it offers flexibility to implement newer non-supported crypto algorithms, thereby providing performance and energy efficiency for a range of cryptographic uses.

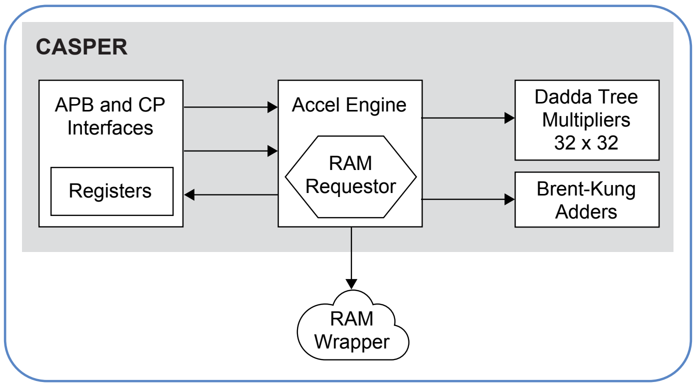

Figure 2: CASPER feature diagram

The CASPER architecture has the following key features to optimize performance and energy efficiency.

1. **Coprocessor interface:** The Cortex-M33 core provides a 64-bit wide coprocessor (CP) interface. This interface allows adding an accelerator directly on the main CPU. This eliminates extra cycles of submitting jobs through the bus. CASPER can be accessed from AHB bus but also provides interface to the Cortex-M33 using the CP bus. Arm can kickoff CASPER using MCRR instructions and retrieve results using MRRC instructions.

   This approach offloads the CPU from compute-intensive tasks, letting it sleep, allowing the system to run at a lower clock speed or letting the CPU do other operations in parallel, e.g., setup the next hardware operation while the previous one is being processed.
2. **RAM access:** CASPER can access system-RAM using a dedicated 64-bit interface (2 x 32-bit interleaved RAMs in parallel) without going through the system bus. Since these RAMs are shared with system space, this interface can be used for other purposes when CASPER is not using it.
3. **Multipliers and a bank of adders and registers:** CASPER has Brent-Kung style adders for maximum efficiency (faster carry-out, lower power compared to CSLA) and 2 x 32-bit MUL-ADD type (extra rows for add) multipliers that are configured to allow for fast 64-bit x 64-bit multiplication in place, but using far fewer resources.
4. **State-machine:** CASPER implements a state-machine that performs operations such as multiply, modular multiply, reduce (Montgomery), add, sub, rsub, double, compare, compare-early-out, fill, zero, copy, re-mask-copy, modular add and subtract, etc.
5. **Masking for side-channel countermeasure:** CASPER provides countermeasure against side-channel analysis by masking. Data is never stored in plain text when using masking with a random value. Masking is also used to flood adders and multipliers to obscure actual data processing.

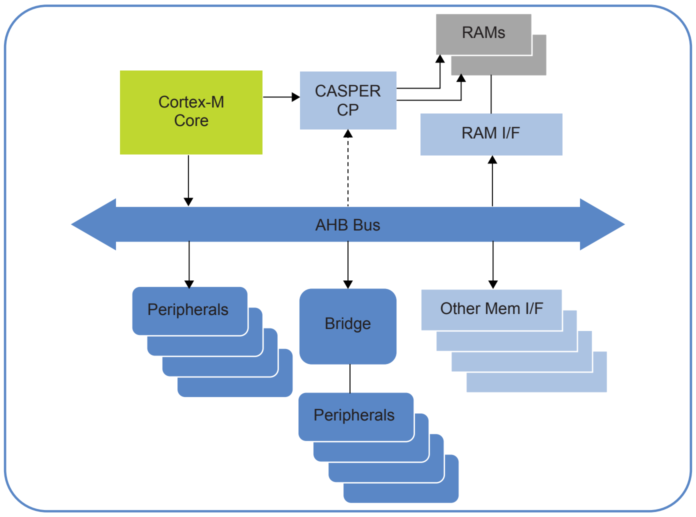

Figure 3: CASPER's view in the system

CASPER's hardware/software partitioning approach allows easier adaptation to new crypto algorithms. By replacing the underlying math code of library APIs with direct hardware invocation, code porting is easy, and gives near-maximum performance gains. The aim of CASPER is to optimize crypto algorithm execution. Although simple use of a library is sufficient to utilize CASPER, code can be optimized to use the memory (64-bit RAM access) and parallelization to the full extent and achieve the best performance and the lowest energy use.

To enable this, a special API library is developed to port mbed™ TLS functions to CASPER acceleration features. CASPER functionality is made available to the user in three layers:

1. Casper_Driver: The base layer has init() and deinit() functions. It exports macros to generate coprocessor instructions.
2. Casper_drive_pkha: This layer exposes Public/Private Key HW Acceleration (PKHA) routines. It accelerates a selected set of routines that could be used to optimize asymmetric cryptography operations. At present, NXP MCUs support the following:
   a. RSA signature verification
   b. ECC operations - ECDsa (sign, verify), ECDH and EDCHE for following curves:
      – Elliptic Prime Curve 256-bit (EC_prime256v1)
      – Elliptic Prime Curve 384-bit (EC_secp384r1)
      – Elliptic Prime Curve 521-bit (EC_secp521r1)
3. mbed TLS library port: Port for mbed TLS library that uses casper_driver_pkha layer functions

The user can develop additional crypto routines support using available API functions.

Benchmarking between the NXP API for mbed TLS running on CASPER versus pure software running on the CPU shows up to six times performance improvement for certain crypto operations as shown in Table I.

Operation	Algorithm	Software Only	With CASPER	Performance Improvement
Signing	ECDSA-secp256r1	83.3 ms	24.5 ms	3.4 x
Verification	ECDSA-secp256r1	158 ms	25.8 ms	6.1 x
Key exchange	ECDHE-secp256r1	150 ms	41.7 ms	3.6 x
Key exchange	ECDH-secp256r1	79 ms	20.9 ms	3.8 x
Verification	RSA-1024	2.4 ms	0.6 ms	4.0 x
Verification	RSA-2048	8.9 ms	1.9 ms	4.7 x

Table 1: Casper performance benchmarking on silicon

On the same set-up, current consumption measurements are as shown in TABLE II. Thanks to performance improvements delivered with the use of CASPER, overall execution time is reduced and an up to 12x energy improvement is observed for certain crypto operations.

Operation	Algorithm	Software Only	With CASPER	Energy Eff. Improvement
Signing	ECDSA-secp256r1	77.5 mA	76.6 mA	3.4 x
Verification	ECDSA-secp256r1	77.5 mA	76.2 mA	6.2 x
Key exchange	ECDHE-secp256r1	77.4 mA	75.7 mA	3.7 x
Key exchange	ECDH-secp256r1	77.4 mA	75.8 mA	3.9 x
Verification	RSA-1024	76.3 mA	76.3 mA	4.0 x
Verification	RSA-2048	76.3 mA	76.3 mA	4.7 x

Table 2: Casper power benchmarking on silicon

Setup:

Hardware: MIMXRT685 EVK
Board Example: MBEDTLS_BENCHMARK.EWW (Release)
Example Located: SDK_2.5.0_EVK-MIMXRT685\BOARDS\EVKMIMXRT685\MBEDTLS_EXAMPLES\MBEDTLS_BENCHMARK\IAR\

III. SRAM PUF

Secret keys are fundamental to the security of microcontrollers. Traditionally keys are stored in OTP eFuses or alternative non-volatile storage. However, on the latest generation of NXP MCUs, a new type of key storage called physically unclonable function (PUF) is offered. PUF provides secure key storage without storing the key.

Due to deep sub-micron process variations that are inherent in the production process, every transistor in SRAM cells has a random electric property variation. This randomness is expressed in the startup values of uninitialized SRAM memory, forming a unique chip fingerprint that is impossible to clone, but is highly consistent for every powerup on a given die [4]. During operation, a small percentage of cells are inconsistent, creating randomness (<15%) with the use of error correction techniques such as helper data algorithms [2] or fuzzy extractors [3]. The PUF engine can generate a reliable root key from this unique but noisy digital fingerprint of the device derived from a dedicated SRAM.

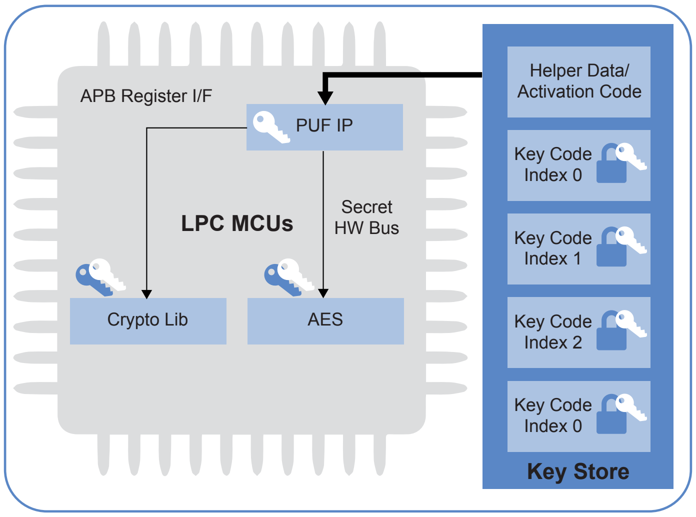

Fig. 4 SRAM PUF Key Storage

PUF comes with the following features:

1. Key strength of 256-bits
2. Generation, storage, and reconstruction of intrinsic keys
3. Storage and reconstruction of user keys
4. Key sizes from 64- to 4096-bits
5. Key output via 1 dedicated secret key interfaces, or key register accessible via the bus

The SRAM PUF algorithm has two modes: enrollment and key reconstruction. Enrollment mode is typically executed once over the lifetime of the chip, SRAM PUF response is read out and processed to generate the activation code (AC). The AC doesn't reveal any information about the key and hence can be stored in any non-volatile memory. Key reconstruction mode is used to retrieve the root-key. This mode is also used to store any private information as a user key and retrieve it using the relevant key code (KC) as shown in Figure 4. The AC and KC are specific to an individual chip, and any change to codes prevents key reconstruction.

User keys can be used as the roots of trust for a set of other uses such as for protection of code IP (PRINCE), secure boot, secure update, data protection (AES), etc. The PUF engine provides support for generating unique symmetric cryptographic keys for local usage (root key and session keys), or a seed for DICE unique device secret key. Key reference is unique per die for each key. Secret-key output from PUF is fed into symmetric cryptographic engines via a dedicated secret hardware bus. The PUF output is masked, offering a side-channel countermeasure.

Keys that are derived from the SRAM PUF are not stored "on the chip" but they are derived 'from the chip,' only when they are needed. In that way they are only present in the chip during a very short window of time. When the SRAM is not powered, there is no key present on the chip. This makes the solution very secure. Fig. 5 below shows the SRAM-PUF advantage compared to traditional methods for key-storage. SRAM PUF has shown robustness against various invasive and non-invasive physical attacks in security labs. Attacks with scanning electron microscopes, lasers, FIBs and probes have not been successful. In addition, side-channel attacks have not lead to any leakage of sensitive information. A special anti-aging technique implementation assures that PUF behavior is robust over the product lifecycle. [4]

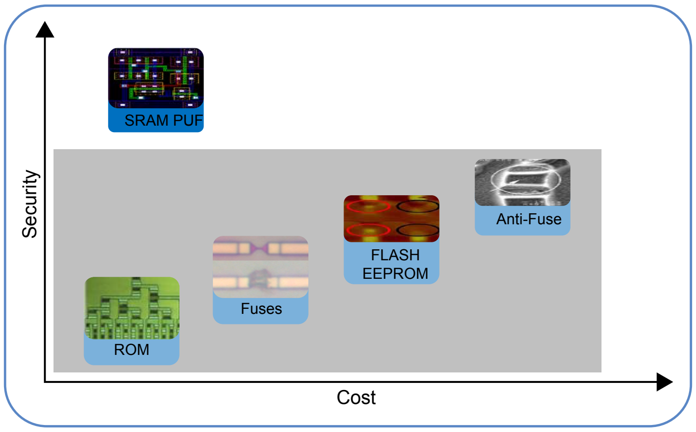

Figure 5: SRAM PUF compared to traditional key storage

The use of PUF also simplifies device provisioning. With traditional OTP eFuse methods, these devices could not be used in any other product once provisioned, and must be scrapped if there was an issue with the provisioning process. With the introduction of PUF, the secret key is already in the part and must only be activated in the field without the possibility of leaking the secret key. This helps make inventory management simpler and helps save cost by eliminating an additional key programming step during the manufacturing process.

IV. PRINCE

On-chip non-volatile storage contains many important assets. It is used to store secret keys, proprietary software from the silicon vendor and OEM, licensed application software and other sensitive information. Traditionally, this storage is in plain-text format. This makes it easy for hackers to steal keys, read the code for cloning, or tamper with it to alter the execution sequence or programming value. This challenge can be overcome by encrypting the flash. However, traditional encryption engines, such as AES, are slow, require RAM and would compromise system performance by adding latency.

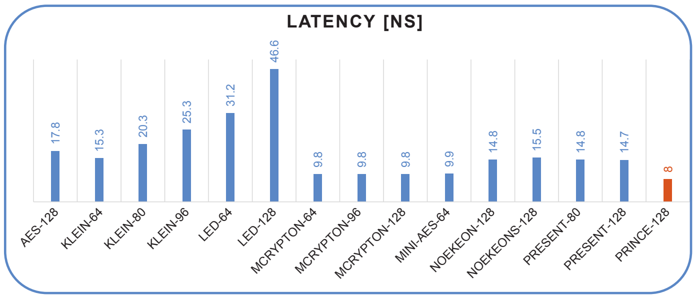

Figure 6: Latency benchmark: PRINCE vs. various other encryption engines

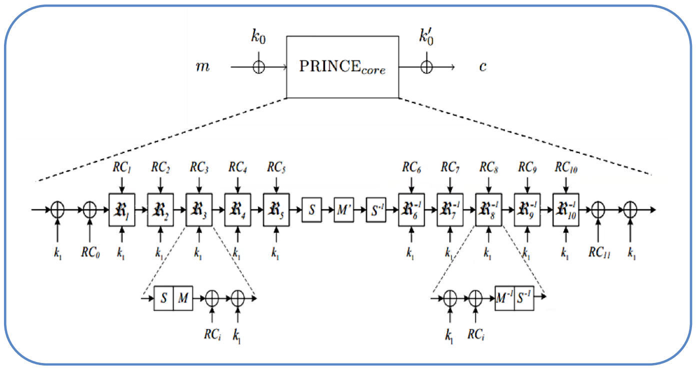
k_0
k_0'
m
PRINCE_{core}
c
RC_1 RC_2 RC_3 RC_4 RC_5
RC_6 RC_7 RC_8 RC_9 RC_{10}
k_1 RC_0 k_1 k_1 k_1 k_1 k_1
k_1 k_1 k_1 k_1 k_1 RC_{11} k_1
S M' S⁻¹
S M
RC_i k_1
M^{-1} S^{-1}
k_1 RC_i

Figure 7: PRINCE with 12-round block cipher PRINCE core

PRINCE is a lightweight symmetric block cryptography algorithm that can solve these challenges. As shown in Figure 6, PRINCE is one of the lowest latency algorithms. The PRINCE is a 64-bit cypher with a 128-bit key. The PRINCE encryption process concept is depicted below in Figure 7. The number of rounds can be configured based on security vs. performance trade-off. RC is 64-bit round constants, K1 is upper half of key, S is Sbox transform (S, S-1), M is matrix multiply (M, M`, M- 1).

NXP's approach builds on the core PRINCE engine with 12 rounds, providing a robust implementation against various attacks [5 7]. Since the algorithm became public in 2012, PRINCE paper has received more than 500 citations, most of which are related to its security analysis. Neither theoretical nor practical attacks have been found so far, which proves the security claims of the algorithm. The security level of PRINCE logarithmically decreases from 126 to 94 bits as a number of collected plaintext-ciphertext pairs increases from zero to 2^32 (32 GB). The PRINCE encryption key is updated before the 32 GB of data have been encrypted, hence PRINCE-based memory encryption is seen to be practically as competitive as the industry-standard encryption algorithms.

Since the address is always available in advance compared to data, the address is used as an input vector to PRINCE together with 128-bit key and 64-bit NONCE (Initialization vector, used only once for a given firmware version and region, unique to device). This output cipher is XORed with the data for encrypted storage during a write operation. The same is done for decryption during a read operation.

(nonce^address)->encrypt->(data^cipher)

This CTR-based implementation provides 0-latency on-the-fly decryption capability, allowing direct instruction execution from encrypted on-chip flash region and without the need for RAM buffers. Since PRINCE is used primarily for on-chip non-volatile store of firmware image and the nonce is changed with every image update, use of CTR does not create the vulnerability that is seen with the external nonvolatile memory. XEX mode would be used if encryption is for the storage on the external memory. Additionally, code/data from NVM to CPU is masked when in cache, providing a side-channel countermeasure.

Sbox transform can be selected from many available variants and is updated from one product to another, providing additional obscurity and uniqueness to guard against attacks. 128-bit secret keys are fed to the PRINCE engine from the SRAM PUF via a hardwired secret path, and in combination with masking, they provide a side-channel counter measure.

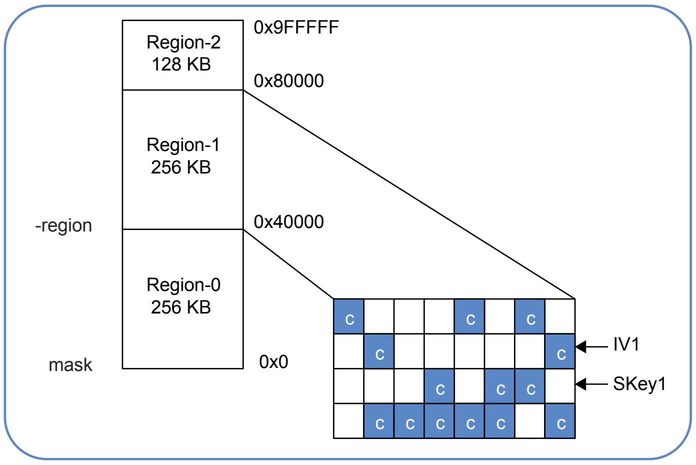

Figure 8: PRINCE secure memory regions

NXP's implementation can support multiple unique encrypted memory regions with individual secret keys. This isolation between regions allows system partners to separately store their proprietary software IPs in non-volatile memory and guard against tampering or cloning, without sharing key and IV. The start address for each encrypted storage region is programmable. Depending on the application, only certain parts of the region might need encrypted storage. To support that, further granularity with programmable control is defined within each region.

There is no observable difference between benchmarking CoreMark® performance on silicon with PRINCE on-the-fly decryption enabled vs. performance without PRINCE on-the-fly decryption, as shown in TABLE III. This proves that the PRINCE implementation offers 0-latency on-the-fly decrypt and does not incur any performance penalty. Also, power consumption is minimal.

CoreMark® @ 100 MHz	Score	Current Consumption
PRINCE enabled	220.25	4.12 mA
PRINCE disabled	220.25	3.25 mA

Table III: PRINCE performance and power benchmarking on silicon

V. SUMMARY

This paper introduced three unique features (CASPER, SRAM-PUF, PRINCE) on the the latest generation of MCUs from NXP to help enable next-generation security at minimal power and area penalty. With CASPER and PUF combined, NXP's latest generation of MCUs offer robust support for symmetric keys and asymmetric key-pair generation and management. PRINCE enables asset protection to an OEM's proprietary software stored in non-volatile memory.

CASPER enables low-cost, high-efficiency hardware acceleration for ECC and RSA, and the hardware-software partitioning approach offers sufficient flexibility to accommodate new algorithms on existing devices.

SRAM-PUF provides a way to authenticate devices and generate and store cryptographic keys from an unclonable silicon fingerprint. It offers a scalable solution that also simplifies the provisioning step.

PRINCE is a lightweight cipher that enables flash encryption and on-the-fly (0-latency) decryption, with minimal area and power cost and without compromising the system performance.

VI. REFERENCES

[1] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)

[2] J.-P. Linnartz and P. Tuyls, "New Shielding Functions To Enhance Privacy And Prevent Misuse Of Biometric Templates," in International Conference on Audio and Video-based Biometric Person Authentication (AVBPA'03), ser. LNCS, J. Kittler and M. S. Nixon, Eds., vol. 2688. Heidelberg: Springer-Verlag, 2003, pp. 393–402.

[3] X. Boyen, "Reusable Cryptographic Fuzzy Extractors," in ACM Conference on Computer and Communications Security (CCS'04). New York, NY, USA: ACM, 2004, pp. 82–91. AND Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in EUROCRYPT'04, ser. LNCS, C. Cachin and J. Camenisch, Eds., vol. 3027. Heidelberg: Springer-Verlag, 2004, pp. 523–540.

A. Smith, "Fuzzy Extractors: How To Generate Strong Keys From Biometrics And Other Noisy Data," in EUROCRYPT'04, ser. LNCS, C. Cachin and J. Camenisch, Eds., vol. 3027. Heidelberg: Springer-Verlag, 2004, pp. 523– 540.

[4] Intrinsic-ID EENews "SRAM PUF: The Secure Silicon Fingerprint". June 30, 2016 https://www.eenewsautomotive.com/Learning-center/intrinsic- id-sram-puf-secure-silicon-fingerprint

[5] https://www.emsec.ruhr-uni- bochum.de/research/research_startseite/prince-challenge/

[6] https://www.arm.com/why-arm/architecture/platform-security- architecture

[7] S. Rasoolzadeh and H. Raddum "Cryptanalysis of PRINCE with Minimal Data". Progress in Cryptography - AFRICACRYPT 2016

How to Reach Us:

Home Page: **www.nxp.com**
Web Support: **www.nxp.com/support**

USA/Europe or Locations Not Listed:
NXP Semiconductors USA, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.nxp.com/support

Europe, Middle East, and Africa:
NXP Semiconductors Germany GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.nxp.com/support

Japan:
NXP Japan Ltd.
Yebisu Garden Place Tower 24F,
4-20-3, Ebisu, Shibuya-ku,
Tokyo 150-6024, Japan
0120 950 032 (Domestic Toll Free)
www.nxp.com/jp/support/

Asia/Pacific:
NXP Semiconductors Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@nxp.com

www.nxp.com

Document Number: IOTSECWP REV 0