



This is a repository copy of *Design rationale for symbiotically secure key management systems in IoT and beyond*.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/210385/>

Version: Published Version

Proceedings Paper:

Bartsch, W., Gope, P. orcid.org/0000-0003-2786-0273, Kavun, E. et al. (4 more authors) (2023) Design rationale for symbiotically secure key management systems in IoT and beyond. In: Proceedings of the 9th International Conference on Information Systems Security and Privacy. 9th International Conference on Information Systems Security and Privacy ICISSP, 22-24 Feb 2023, Lisbon, Portugal. SCITEPRESS - Science and Technology Publications , pp. 583-591. ISBN 9789897586248

<https://doi.org/10.5220/0011726900003405>

Reuse

This article is distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) licence. This licence only allows you to download this work and share it with others as long as you credit the authors, but you can't change the article in any way or use it commercially. More information and the full terms of the licence here: <https://creativecommons.org/licenses/>

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.



eprints@whiterose.ac.uk
<https://eprints.whiterose.ac.uk/>

Design Rationale for Symbiotically Secure Key Management Systems in IoT and Beyond

Witali Bartsch¹^a, Prosanta Gope²^b, Elif Bilge Kavun³^c, Owen Millwood²^d, Andriy Panchenko⁴,
Aryan M. Pasikhani²^e and Ilia Polian⁵^f

¹PointBlank Security, Steen Harbach AG, Leverkusen, Germany

²Department of Computer Science, The University of Sheffield, U.K.

³Faculty of Computer Science and Mathematics, University of Passau, Germany

⁴Chair of IT Security, Brandenburg University of Technology (BTU), Cottbus, Germany

⁵Institute of Computer Architecture and Computer Engineering, University of Stuttgart, Germany
fi

Keywords: Secure Processor Architecture, Secure IoT, Hardware Fingerprinting, PUFs, Attack-Resilient Hardware.

Abstract: The overwhelmingly widespread use of Internet of Things (IoT) in different application domains brought not only benefits, but, alas, security concerns as a result of the increased attack surface and vectors. One of the most critical mechanisms in IoT infrastructure is key management. This paper reflects on the problems and challenges of existing key management systems, starting with the discussion of a recent real-world attack. We identify and elaborate on the drawbacks of security primitives based purely on physical variations and – after highlighting the problems of such systems – continue on to deduce an effective and cost-efficient key management solution for IoT systems extending the symbiotic security approach in a previous work. The symbiotic architecture combines software, firmware, and hardware resources for secure IoT while avoiding the traditional scheme of static key storage and generating entropy for key material on-the-fly via a combination of a *Physical Unclonable Function* (PUF) and pseudo-random bits pre-populated in firmware.

1 INTRODUCTION


Internet of Things (IoT) has shown a spectacular growth over the last years and is steadily entering application domains that are considered security-critical. Security requirements in IoT systems must coexist with stringent cost limits and with an extended attack surface where adversaries can gain physical access to their components. A critical part of most secure systems is generation and management of key material, which includes not only secret keys for use in encryption and authentication, but also *critical security parameters* such as initialisation vectors (IVs), padding bits or randomness for masking schemes.


In this position paper, we discourse on a solution that provides an effective and yet cost-efficient key


management for IoT systems. It extends the concept of *symbiotic security* (Bartsch and Huebner, 2019b) that leverages a combination of software, firmware, and hardware for secure IoT, but not without upholding the principle of resource-constrained security. The latter propagates reduction of design complexity in favour of reduced attack surface (Bartsch and Huebner, 2019a). The derived scheme, consequently entitled **Symbiotic Key Management System (SKMS)**, avoids static key storage, which is vulnerable to readout and manipulation especially in IoT devices accessible to attackers. SKMS generates entropy for key material on-the-fly, using a combination of *Physical Unclonable Function* (PUF) and pseudo-random bits pre-stored in the system's firmware.


Keys thus derived and encapsulated in a *crypto engine API*, which offers services like encryption or authentication, but does not communicate actual key bits to applications. SKMS is designed to support zero-knowledge protocols for initial enrolment (ZKIE), for which it removes the need to keep the underlying secret in non-volatile memory whilst also allow-


^a  <https://orcid.org/0000-0002-3766-8130>

^b  <https://orcid.org/0000-0003-2786-0273>

^c  <https://orcid.org/0000-0003-3193-8440>

^d  <https://orcid.org/0000-0002-7250-8271>

^e  <https://orcid.org/0000-0003-3181-4026>

^f  <https://orcid.org/0000-0002-6563-2725>

ing to replace said secret when compromised. Traditional schemes based on One-Time Programmable (OTP) memory, possibly including eFuses or even PUFs are thus superseded. Moreover, it tightly integrates an Artificial Intelligence (AI) assisted metadata analysis procedure to detect and prevent intrusions. The composition of a hardware-based *Symbiotic Unclonable Function (SUF)* and software-level intrusion detection provides defence-in-depth: even when an adversary manages manipulating system hardware, software-level protections stay intact and can, upon an alert, request regeneration of entropy bits.

The remainder of this paper is organized as follows. We start with a case study where we discuss a recent vulnerability in a Trusted Execution Environment (TEE). For a number of observed aspects, we point out the precise challenges and discuss strategies how SKMS can address them. Additionally, we review PUFs and identify observations, challenges and strategies for them. The resultant security requirements in IoT systems are discussed in Section 3. Finally, Section 4 addresses the SKMS in detail followed by the conclusion in Section 5.

2 STATE OF AFFAIRS IN KEY MANAGEMENT SYSTEMS

This section motivates the need for key management approaches beyond today’s standard by dissecting a security flaw in a widely used product. It also provides background information on PUFs and shows their potential deficiencies that SKMS should overcome.

2.1 TrustZone Design Flaws

To support our deduction of the design rationale for SKMS in Section 4, we analyse a major vulnerability (Shakevsky et al., 2022) of a specific implementation of the original TEE design by ARM. Said Samsung implementation relies on proprietary and undocumented hardware which is a major criticism. We analyse all essential findings to compare them with the suggested SKMS idea depicted in Fig. 1.

↪ **Observation 2.1.1.** The overall complexity of ARM’s TrustZone concept, where a dedicated hardware-based keystore or Keymaster Trusted Application (TA) already follows the solid principle of operation of a Hardware Security Module (HSM) by performing all cryptographic operations within its confines. Still, Keymaster TA must also entrust user space applications with the housekeeping of the actual

keys albeit in a “wrapped” form and shape (utilising its hardware-embedded key).

↪ **Challenge 2.1.1.** Android is a full OS with hundreds of applications installed on a single device. Hence, there is the need for a key hierarchy based on key wrapping or similar protection as otherwise the TEE would need to provide considerable protected storage space for all such keys.

↪ **Strategy 2.1.1.** The first conclusion we draw from this observation is that in the typical IoT space where the “business logic” is mostly implemented by the server to minimise the computational load on the end devices, simplicity is the key both to efficiency and sustainability given that less complex software is naturally less error-prone and thus the attack surface is intuitively reduced. In other words, a dedicated firmware incorporating crucial functions such as cryptographic primitives and required security protocols (e.g., TLS) is unlikely to need hundreds of different cryptographic keys or a TEE-like key hierarchy. Even if a business logic does require numerous keys in such a monolithic and use-case specific firmware, then we would argue that a similarly monolithic HSM-like cryptographic co-processor with exclusive access to key material and encapsulation of all cryptographic operations within its cryptographic boundary would reduce the need for key wrapping. In Section 4, we indicate a possible design of such a module. Also, if implemented with separate data buses (Harvard architecture), specifically at the level of the suggested co-processor, it would diminish the need for shared memory or memory mapping. However, that alone would not remove the natural susceptibility of hardware to reverse-engineering given direct physical access – hence the next observation. ■

↪ **Observation 2.1.2.** The underlying vulnerability could only be uncovered and fully exploited through substantial reverse-engineering. However, the authors did not have to go so far as to target the secure hardware unit due to the design flaw of the related API as well as unnecessary exposure through backward-compatibility which in turn enabled the downgrade attack. The previous observation and conclusion suggests that this could be fixed by software which is not immutable (i.e., reverse-engineered software can be amended).

↪ **Challenge 2.1.2.** Present design focuses on a particular dimension only (e.g., software) and its interfaces while taking the other (here hardware) for granted or vice versa. Carefully crafted “security by design” to combine several realms is thus more difficult to implement later in production.

↔ **Strategy 2.1.2.** While software can be extracted or leaked such that it can then be dismantled separately, hardware reverse engineering is usually only possible in an offline state implying direct physical access and device theft to begin with. The latter is not too far fetched which is why ultimately a symbiotic design strategy could lead to construction of key material in a way that it neither exists statically inside a hardware enclosure, nor relies on software or firmware alone. Furthermore, design elements that do not require any subsequent modification like cryptographic primitives (but still need tamper-resistance) should be encapsulated by a hardware-based crypto engine including all critical algorithmic parameters, *e.g.*, IVs. Incidentally, application-controlled IVs were the root cause of the Samsung vulnerability. So, if software does not contain or control any secrets and if hardware does it exclusively, we conclude that to thwart both software and hardware reverse engineering, key material must not be stored persistently as opposed to the common root-of-trust approach or quite specifically here the device-unique, but permanent 256-bit AES key (Root Encryption Key) inside the crypto engine that the Keymaster TA relies on. ■

↔ **Observation 2.1.3.** The above-mentioned IV reuse threat was created by allowing app-layer code to handle IVs without any safety checks. Paired with the single static platform key, integrity could be compromised despite a solid choice of the underlying cryptographic primitive (AES-GCM). Consequently, all reliant device security schemes like FIDO2 (passwordless authentication) were completely exposed. Incidentally, FIDO2 among others also relies on pre-fabricated static Attestation Keys re-used across all devices of the same manufacturer. This shows how predominant the root-of-trust design still is in present security schemes.

↔ **Challenge 2.1.3.** It has long been known that solid random numbers are best generated in hardware relying on true sources of entropy like crystal oscillators (Ergun and Maden, 2021). However, those dedicated integrated circuits create an additional dependency. Additionally, software-based Pseudo Random Number Generators (PRNG) like in Barker and Kelsey's study (Barker and Kelsey, 2015) only use a fraction of 'true' entropy in their initial seeds. Selecting and correctly referencing a proper entropy source for seeds is usually exposed through an API creating a potential pitfall for high-level application developers. Correct initialisation and handling of PRNGs at runtime (*e.g.*, possible buffer overflows or memory corruption) is yet another software-induced caveat.

↔ **Strategy 2.1.3.** Beyond the beneficial isolation of critical functions through hardware as discussed in

the previous challenge, there is added value in relying on an independent cryptographic hardware unit: as will be discussed in the subsequent section dedicated to the design challenges of a PUF, noise generated by any PUF can be used beneficially for solid random numbers. Those can be harvested directly in hardware and subsequently used on-the-fly for IVs, SALT, or similar. If implemented as part of the crypto engine or PUF controller (see Fig. 1), this security-relevant function is completely out of reach of app-layer code and its programmers. Furthermore, PUF noise depending on the underlying hardware elements like DRAM cells (Miskelly and O'Neill, 2020) can reach considerable size simply because there will be most likely several GB of memory on any IoT module in use today. This ensures not only constant access to fresh entropy, but can also eliminate the need for software PRNGs given the volume of randomness. ■

↔ **Observation 2.1.4.** Another aggravating aspect of the Samsung design shortcoming is the (natural) fact that their API is "hookable", which means that software calls can be intercepted by a malicious handler to manipulate critical parameters at will. In this case, it led to an effective downgrade attack as legacy key BLOB version could be requested by the 'hook' (malicious call handler). This begs the exacerbating question as to why legacy key BLOBS were allowed to begin with which according to the authors did not seem to have any sensible reason.

↔ **Challenge 2.1.4.** Software interfaces need to rely on platform security mechanisms which are less likely to withstand privileged adversaries, malware, or major OS exploits. The latter have been shown (Brand, 2021) to work over a network, undiscovered, despite low-level protection like kernel/boot-loader integrity checks based on eFuses.

↔ **Strategy 2.1.4.** We have stipulated through the notion of an isolated crypto engine where security parameters must not be exclusively handled by high-level applications. The underlying HSM principle enforces isolation such that the client (app-layer code or firmware in general) either sends over cleartext to be encrypted or likewise provides ciphertext to be returned upon decryption. Key principle of operation is that all important keys are generated and kept within the cryptographic boundary. The symbiotic approach (see Section 4) extends this by incorporating the PUF controller into the crypto engine while also making the PUF dependent on the firmware to make the resulting entropy changeable if compromised. This creates a symbiotic link between software and hardware in the secret derivation process at runtime. ■

↔ **Observation 2.1.5.** The authors briefly discuss a supply-chain threat which is possible specifically with

root-of-trust security schemes. Another considerable attack vector is patching of the Keymaster Hardware Abstraction Layer through software.

↔ **Challenge 2.1.5.** Even recent specifications like Cooper et al.'s work (Cooper et al., 2020) stick to the root-of-trust principle which is highly convenient from the manufacturer's perspective. However, at the very least, they must introduce various sub-protocols (*e.g.*, Transfer Ownership Protocol 0 through 2), routing requirements, ownership vouchers, device attestation certificate chains, and other precautions all of which inflate the complexity. As noted, higher complexity may increase attack surface and is difficult to maintain both from the trust and technical perspectives. Patching is also hard to avoid as Android's history clearly shows. In this non-monolithic state, weakness of one critical system service is likely to compromise the others, if not the entire OS.

↔ **Strategy 2.1.5.** If complexity is a concern, then a 'clean slate' solution like (Bartsch and Huebner, 2019b) provides a much more transparent zero-trust like alternative, especially for an arbitrary supply chain. A security architecture incorporating zero knowledge-based bootstrap and over-the-air update mechanisms conveys provable trust given that end-devices can be deployed off-the-shelf with just a minimal security logic inside (*i.e.*, bootloader, see Fig. 1) that does not contain any embedded secrets. Consequently, every device creates its own unique key during enrolment which is then used in the zero knowledge protocol for mutual pre-authentication and forward secrecy-enabled traffic encryption in absence of any root of trust. This creates a secure out-of-band channel for want of (pre-deployed) digital certificates required in TLS communication mandated by every cloud-based IoT offering today.

However, end-devices in this scheme still rely on OTP memory with an alternative eFuse seal. Thus, a device whose secret has been extracted through a physical attack cannot be recovered anymore. A remedial option implies the use of PUFs to make secrets volatile, hence we explore their design in the following section. This is because PUFs have their own limits and challenges (*e.g.*, noise, temperature dependency, ageing) including the worst case that if a device-specific pattern has been discovered, such a PUF has essentially been broken too. Thus, we not only discuss strategies to integrate PUFs in a symbiotic fashion to make them dynamic when compromised, we also suggest in Section 4.2 how ML can be utilised to create an independent integrity control for end-devices and fulfill the realistic requirement for comprehensive analysis against ML-enabled adversaries in a symbiotic context. Ultimately, we advocate that

a zero knowledge like scheme can be created to protect the acquisition of the hardware fingerprint of any device during the ML training phase (initial enrolment) given its secret nature. ■

In summary, the idea of an SKMS that beneficially utilises all of its essential elements should be much better poised to withstand the discussed attacks.

2.2 Physical Unclonable Functions

We know from (Neshenko et al., 2019) that not only is it imperative to create new strategies (both technical and non-technical) to increase awareness of IoT threats to reduce the risk of exposure, but also to work out (external) capabilities to leverage data-driven measurements and methodologies for detecting exploited IoT devices. IoT and Cyber-Physical Systems (CPS) are usually part of a larger distributed infrastructure with scalable computational power for empirical or analytical needs. With respect to PUFs in particular, this means the entire distributed architecture could ideally be utilised to both define/control the unclonable function and constantly monitor its integrity/susceptibility to local or physical attacks. This is especially true in all critical infrastructure supported by IoT/CPS which may or may not be directly exposed to a powerful (internal) adversary. Thus, the "external view" and the spectrum of its added values can be manifold.

Traditional PUFs derive entropy from unique characteristics of hardware in order to generate reliable 'storage-less' key material and/or unique device identifiers (Pappu, 2002), (Gassend et al., 2002). In and of themselves, PUFs are naturally limited in regard to symbiotic architectural design as they collect such uniqueness from a single immutable dimension. Given ideal properties, a PUF would be suitable to shoulder the majority of the security burden with regard to authentication or key management, wherein the manifestation of reliable, unpredictable, just-in-time cryptographic data would reinforce what has the potential to be the weakest link in a secure system. Such a PUF could be deployed almost as a 'silver bullet' hardware unit to solve the problem of sufficiently secure ultra-lightweight systems. Much of the literature surrounding the application of PUFs take these ideal properties as an assumption at the hardware level, which is seldom the reality. Currently, PUFs continue to fall short of the mark in many aspects, often introducing promising features at the expense of bringing new vulnerabilities/performance issues to light. Resultantly, there is scepticism surrounding the idea that a true PUF is even at all possible. We identify issues regarding PUFs and argue

that while various PUFs are highly limited in their role as primary security element in a given system, they still may play an ideal role as a component in more comprehensive symbiotic security design, such that no one PUF feature is overly emphasised, yet each is utilised by its strength. We identify current challenges surrounding modern PUF design, and determine how each could be mitigated/exploited by symbiotic security design.

↔ **Observation 2.2.1.** A PUF would ideally be entirely reliable, where environmental conditions do not affect the PUF to output the exact same output for every measurement. With the primary means of entropy deriving from sub-atomic variations in the PUF material, noise is an inevitable feature that must be dealt with to reduce measurement instabilities.

↔ **Challenge 2.2.1.** Commonly, issues regarding PUF reliability are tackled with various forms of error correction as an accepted resource overhead, yet it has been shown that publicly known helper data required for error correction can provide adversaries with sufficient information to successfully compromise the security of PUFs (Delvaux and Verbauwhede, 2014), (Strieder et al., 2021). It is well known that environmental conditions, most notably temperature, have dominant effect on the physical characteristics, such as, *e.g.*, the clock skew of a processor. If one learns and knows this dependency, it is possible to filter it out and determine a stable value which can enable identification. A sensor, however, is required to measure a given environmental condition which requires additional equipment and is often unfeasible in the targeted scenarios. A method which does not require explicit knowledge of the environmental condition would solve this problem. This introduces new concerns on how best to manage the noisy nature of PUFs given a sufficiently strong (and very reasonable) threat model.

↔ **Strategy 2.2.1.** Common or comparable AI-assisted techniques can help overcome environmental variance issues in PUFs to improve reliability with regard to the resulting entropy (Wen and Lao, 2017). In addition, environmental effects on physical systems – hence what originally causes a reliability problem – can be exploited as a benefit, enabling anomaly/intrusion detection and device identification based on known individual behaviours in various operating conditions, provided training processes can be offloaded to server computation within the scheme. Therefore in essence, environmental effects on hardware characteristics can be eliminated by considering their dependency and adjusting the corresponding characteristic value based on the sensed condition. Since such sensors are not present in most off-

the-shelf devices, we propose a strategy which eliminates such dependencies without explicit knowledge of the sensor value. The idea is to learn legitimate combinations of hardware characteristics under different conditions (reflected by measurements at different times). This can be done by considering several devices at the same time, or alternatively, using different, statistically independent, physical parts of one device (*e.g.*, part of a PUF). The practicability of this idea was demonstrated by Lanze et al (Lanze et al., 2014), where the effect of temperature variations on clock skew enable physical device fingerprinting for identification of wireless access points. In that work, the authors applied one-class SVM to learn legitimate combinations of devices at a particular location. ■

↔ **Observation 2.2.2.** Arguably the most important feature to be provided by a PUF is unpredictability/unclonability, such that it is impossible for an adversary to generate a ‘copy’ of the target which exhibits identical behaviour. Failure to ensure this compromises PUF secrets, enabling an adversary to impersonate devices or decrypt collected encrypted network traffic. PUFs are popularly proposed in a limited variety of use cases, such as for authentication token generation or repeatable key storage. For generating unique authentication tokens, a *Strong PUF* is required (strong not indicating the security properties of the PUF, rather its ability to generate unique tokens exponentially with PUF size), where each token is discarded after use to prevent replay attacks.

↔ **Challenge 2.2.2.** Novel Strong PUF designs are commonly proven to be insecure against Machine Learning Modelling Attacks (ML-MA), creating rightful skepticism on whether a truly attack resistant Strong PUF implementation is possible given enough data (Rührmair et al., 2010). Once the PUF is compromised in common PUF-based security protocols, the entire scheme is compromised, leaving designers with little room to enhance threat models sufficiently for a realistic IoT threat environment.

↔ **Strategy 2.2.2.** We argue that the PUF entropy may be exploited as a single hardware entropy unit to be integrated tightly with – not on top of – a coinciding software-based entropy solution. Such integration creates a dependency between both sources, relieving some pressure for per-bit unpredictability as adversaries should be required to simultaneously model the software entropy, hardware entropy, and the dependency between both. ■

↔ **Observation 2.2.3.** The superposition of physical hardware component fingerprints can be used to assess the integrity of the system.

↔ **Challenge 2.2.3.** It is to design, if possible, a non-intrusive method for fingerprinting hardware components for detecting deviations, *e.g.*, when a hardware component is replaced.

↔ **Strategy 2.2.3.** We propose to use methods of physical device fingerprinting to learn characteristics of the devices and detect replacement of them, or, any anomalies in the environment. The emphasis is on the methods allowing stable fingerprints that, from one side, would allow for unique identification of the hardware in the environment and, from another side, would not interfere with regular functionality and communication. Hence, we propose to use passive remote hardware fingerprinting methods to achieve this goal. ■

↔ **Observation 2.2.4.** Even encrypted communication channels can be used for traffic analysis purposes. The goal of traffic analysis is to extract particular informative patterns from the traffic without breaking the encryption. The communication patterns, as a side-channel, can be further applied to detect attacks with the network traffic.

↔ **Challenge 2.2.4.** Perform passive traffic analysis and learn with the help of ML techniques a reliable representation of the communication in the system.

↔ **Strategy 2.2.4.** Similar to the suggested resolution of the temperature variations with the help of ML, AI-assisted meta data analysis can be utilised to observe the communication behaviour of IoT devices with the rest of the infrastructure (back-end) in a non-obtrusive way. Techniques like traffic analysis can be used to learn regular communication patterns and to detect any deviations from the learned patterns. Approaches for this kind of anomaly detection have been successfully applied in CPS (Schneider and Böttinger, 2018). The focus is hence shifted towards data in transit security and resilience by using the same resources and a similar technique. ■

↔ **Observation 2.2.5.** As many common PUF designs consist of custom arrangements of logic components (*e.g.*, Arbiter PUF variants (Gassend et al., 2002)), they must either be synthesised directly into the end device early in the supply chain, or, integrated onto the end device later in the supply chain.

↔ **Challenge 2.2.5.** Both approaches incur a manufacturing overhead, as custom integration must be included at some level. Additionally, this feature affects the flexibility of integration of a PUF solution, further affecting backward compatibility across multiple generations of devices, a key feature also identified for a coinciding software-based entropy module in Strategy 2.1.4.

↔ **Strategy 2.2.5.** We suggest designers consider the use of software PUFs which utilise already on-chip resources to generate physical entropy, such as memory-based or processor PUFs (Miskelly and O’Neill, 2020), (Tsiokanos et al., 2021). The implications of integrating such PUFs enable the hardware entropy unit to become more modular and thus independent of the supply chain. ■

↔ **Observation 2.2.6.** Due to the mentioned ML-MA vulnerabilities, there exists a requirement to include an ML-capable adversary in the threat model when including PUF-based primitives in a symbiotic security system in order to evaluate against such ML-based modelling attacks.

↔ **Challenge 2.2.6.** Most modelling attacks are performed using linear ML methods suitable for modelling (mostly) linear PUFs (Rührmair et al., 2010). Certain PUFs were proposed with enhanced non-linearity to counter modelling attacks, which were shortly after shown to be insecure to ML algorithms better suited for non-linear function approximation such as PAC learning and Evolutionary Algorithms (Ganji et al., 2016) (Delvaux, 2019). As a result, more exploratory ML methods are likely required to fully evaluate the PUF-based aspects of the symbiotic design such that it is proven sufficiently secure.

↔ **Strategy 2.2.6.** Tying PUF-based entropy closely with software mechanisms will likely provide strong non-linearity to the overall PUF function, yet not necessarily prevent modelling attacks given enough data. We propose system designers to consider ML attacks which act upon exploration and exploitation in their iterative learning such that PUF entropy can be ensured. State-of-the-art Adversarial Reinforcement Learning (ARL) based approaches will likely be useful/required to truly evaluate the modelling resilience of the SUF, such that given sufficient data, any information leakage (and thus correlative properties between different inputs and outputs) is detected. ■

3 REQUIREMENTS & DESIGN OBJECTIVES

Connected and physically exposed IoT systems operate at an intersection of physical and cyber domains. Consequently, they are more affected than, *e.g.*, servers accessible by authorised personnel only. Security requirements for IoT systems must guarantee their resistance against software-based attacks; attacks over network connections; and physical attacks against their hardware components. Moreover, the threat model must consider ML-MA against PUFs as

discussed in Strategies 2.2.2 and 2.2.6. Symbiotic security also requires defence-in-depth, *i.e.*, the ability of one range of protections to compensate for cases when an adversary has overcome other defences. In the context of key management, SKMS should be able to provide an acceptable residual security even if some key material has been compromised and replace that key within an adequate period of time. This gives rise to the following design considerations:

- The system should be distributed, rather than monolithic, to withstand a concentrated attack.
- Its unique information (such as IDs or cryptographic keys) must be adaptive, rather than immutably static, such that it can be replaced in the case of a partially successful attack.
- The secret handling should be as transparent and dynamic as possible, using zero-trust or even zero-knowledge principle of operation to avoid exposing key material. Examples are enhanced bootstrap-like mutual authentication schemes such as ZKIE.
- Protections must cover not only static data-at-rest (*e.g.*, stored in memories), but also data-in-transit during their processing.
- The system must support external monitoring and/or self-observation. AI-assisted metadata analysis can play a crucial role here, and an interesting open question is whether it can be applied to encrypted data in transit.

In the following section, we make a proposal for an approach that fulfills the listed requirements.

4 SKMS DESIGN RATIONALE

In this section, we summarise all essential strategies discussed thus far in Fig. 1. The depiction of SKMS covers its fundamental communication paths to outline the relationship between the secure IoT module incorporating such an SKMS and the core services hosted by a cloud back-end originally introduced and discussed in Bartsch and Huebner’s work (Bartsch and Huebner, 2019b).

4.1 Core Building Blocks

SKMS uses an unclonable function to eliminate the static key storage need whilst overcoming its natural limitation of being immutable as far as the hardware-based characteristics are concerned. This shall be achieved by an interplay of hardware and software such that a monolithic firmware implements only the

necessary device logic. Cryptographic functions shall be used by said firmware through a dedicated API exposed by the low-level HSM-like *Crypto Engine* as part of the SKMS forming an essential SSC.

Key material is produced through a resultant set of entropy within an isolated part of the volatile memory (*Isolated RAM*) which can only be read out by the *Crypto Engine* via a dedicated data bus. Alongside its *Crypto Engine API*, the *Crypto Engine* shall also expose a *Secrets/Entropy Selector* interface such that the high-level application can address certain areas of the overall entropy space (*i.e.*, byte array) for purpose-bound cryptographic keys (*e.g.*, 256 bits for AES encryption). The resulting entropy is created at run time by the *Entropy Multiplexer* (a dedicated hardware unit) to collect hardware-based entropy and combine it with the firmware-intrinsic entropy through a suitable unkeyed one-way function. The firmware itself shall only be the carrier of entropy without any hard-coding in source code. Thus, a *Firmware Dispenser* would process the firmware to add an ‘entropy layer’ to it. Following the principle of Zero Trust or Zero

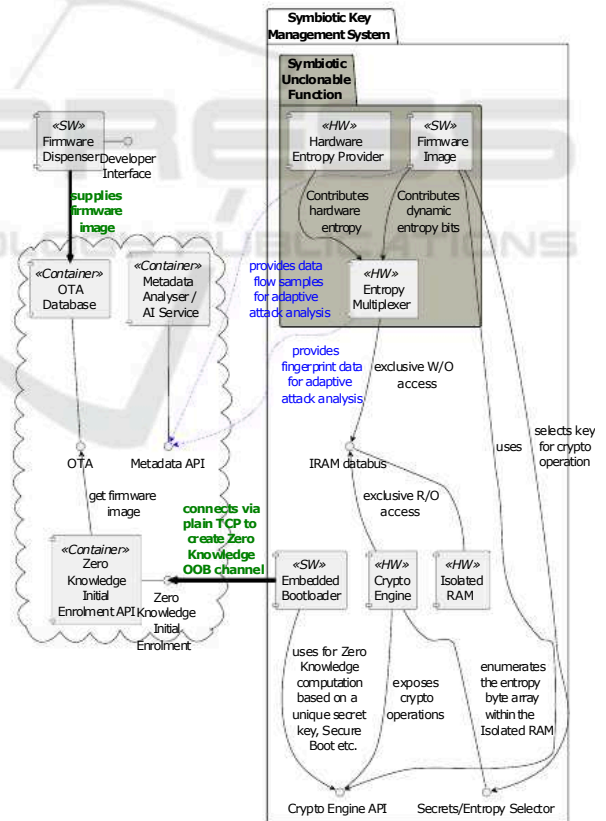


Figure 1: Symbiotic Key Management System.

Knowledge even, every SoC/IoT module shall be delivered to its designated environment with just a primitive *Embedded Bootloader* inside. On arrival, the IoT

module shall establish an encrypted and mutually authenticated Out-of-Band (OOB) channel through that bootloader. The latter leverages the *Crypto Engine* and a subset of entropy to generate the Zero Knowledge secret as well as execute the protocol as such. The *Zero Knowledge Initial Enrolment (ZKIE)* containerised logic unit shall then look up a use case specific firmware for the device in question and deliver it back to the requester via the secure OOB channel.

4.2 AI Services

To ensure continuous operational stability and to implement an external hybrid Intrusion Detection scheme (IDS), a *Metadata Analyser/AI Service* shall on the one hand receive the initial device helper data/fingerprint to be followed up by regular updates thereof (e.g., by means of Out-Of-Distribution or OOD data). On the other hand, this service shall also acquire the initial and regular samples of encrypted device traffic to be able to conduct (real-time) metadata analysis of data in transit past the AI training phase. The ultimate goal is to detect data leakage or malicious modification both at the device level (data at rest integrity) and at the network level (data in motion integrity). Ultimately, if an alert is issued, the device shall be instructed to self-erase and more importantly receive a fresh subset of entropy through a new firmware image to form a new resultant set of entropy and to go through yet another ZKIE life cycle. In other words, compromised modules must no longer be decommissioned thanks to the full wipe and key rollover mechanism implemented in this way. To also ensure the robustness of this hybrid IDS approach over time and make it generalised against OOD data (e.g., as a result of zero-day attacks), further research is needed to devise suitable error-based concept-drift detectors as part of the backend infrastructure which will also ideally suggest when both the anomaly-based and signature-based intrusion detector models need enhancement over time. In this regard and in this proposal, Adversarial Reinforcement Learning (ARL) frameworks could prove to be beneficial as they have been shown to enhance the performance of IDS for imbalanced and evolving data environments such as in 6LoWPAN (Pasikhani et al., 2022), which is similar to the suggested architectural constraints.

4.3 Discussion

The proposed scheme addresses the above-mentioned observations and challenges as follows. It supports simple firmware and does not require non-trivial

memory mapping (\leftrightarrow 2.1.1). It relies on software and hardware, thus complicating reverse engineering (\leftrightarrow 2.1.2). An independent hardware unit in charge of generating random bits isolates critical resources such as IVs (\leftrightarrow 2.1.3) from incorrect app-level usage; it avoids hookable APIs (\leftrightarrow 2.1.4), and enables ZKIE (\leftrightarrow 2.1.5). The SUF component incorporates AI-enabled temperature compensation (\leftrightarrow 2.2.1) and is protected against attacks on multiple levels: hardware-software integration overcomes modeling attacks (\leftrightarrow 2.2.2, 2.2.6) and supply-chain threats (\leftrightarrow 2.2.5); fingerprinting detects manipulation attempts (\leftrightarrow 2.2.3); AI-assisted meta-data analysis identifies anomalous usage. In summary, SKMS has the potential to provide defence-in-depth security in resource-restricted IoT environments.

5 CONCLUSION

In this paper, we analyse critical issues and threats in current KMS through a broad discussion of substantial vulnerability of the Samsung implementation of ARM's original TEE design. Following the observations and derived strategies around this attack, we postulate security requirements and deduce a novel SKMS for IoT. Our idea extends the symbiotic security concept introduced in an earlier study that uses software and hardware for designing efficient yet secure IoT. In this proposal, we avoid attack-prone static key storage and generate entropy for key material on-the-fly through a combination of functions based on physical variations and pseudo-random bits pre-deployed in the system firmware. Furthermore, the derived keys are encapsulated in a crypto engine API offering encryption or authentication, while the actual key bits remain completely invisible to applications. The resulting design of a hardware-based SUF and software-level protections ensures that the system remains intact even if one of the hardware- or software-level protection mechanisms is attacked. Lastly, we briefly discuss the notion and importance of an AI-based hybrid IDS to create an outside view of the end device regarding its susceptibility to relevant attacks which are best detected by external components.

REFERENCES

- Barker, E. and Kelsey, J. (2015). Recommendation for Random Number Generation Using Deterministic Random Bit Generators. In *NIST Special Publication*.
- Bartsch, W. and Huebner, M. (2019a). Efficient System Design of Scalable Ultra Low Power Architectures with Symbiotic Security. *IEEE VCaSL*, 5(4):4–11.

- Bartsch, W. and Huebner, M. (2019b). Reference Architecture for Secure Cloud Based Remote Automation – Zero-knowledge Initial Enrolment of Resource-constrained IoT with Symbiotic Security. *atp magazin*, 61(9):72–82.
- Brand, M. (2021). In-the-wild Series: Android Exploits / Project Zero.
- Cooper, G., Behm, B., Chakraborty, A., Kommalapati, H., Mandyam, G., Tschofenig, H., and Bartsch, W. (2020). FIDO Device Onboard Specification.
- Delvaux, J. (2019). Machine-Learning Attacks on Poly-PUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF-FSMs. *IEEE TIFS*, 14(8):2043–2058.
- Delvaux, J. and Verbauwhede, I. (2014). Attacking PUF-Based Pattern Matching Key Generators via Helper Data Manipulation. In *Proc. of CT-RSA'14*, volume 8366 of LNCS, pages 106–131. Springer.
- Ergun, S. and Maden, F. (2021). An ADC Based Random Number Generator from a Discrete Time Chaotic Map. In *Proc. of APCC'21*, pages 79–82. IEEE.
- Ganji, F., Tajik, S., Fäßler, F., and Seifert, J.-P. (2016). Strong Machine Learning Attack Against PUFs with No Mathematical Model. volume 9813, page 391–411. Springer-Verlag.
- Gassend, B., Clarke, D., van Dijk, M., and Devadas, S. (2002). Silicon Physical Random Functions. In *Proc. of CCS'02*, page 148–160. ACM.
- Lanze, F., Panchenko, A., Braatz, B., and Engel, T. (2014). Letting the Puss in Boots Sweat: Detecting Fake Access Points Using Dependency of Clock Skews on Temperature. In *ASIA CCS'14*, pages 3–14. ACM.
- Miskelly, J. and O'Neill, M. (2020). Fast DRAM PUFs on Commodity Devices. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 39(11):3566–3576.
- Neshenko, N. et al. (2019). Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-scale IoT Exploitations. *IEEE Communications Surveys & Tutorials*, 21(3):2702–2733.
- Pappu, R. (2002). Physical One-Way Functions. *Science*, 297.
- Pasikhani, A. M., Clark, A. J., and Gope, P. (2022). Adversarial RL-based IDS for Evolving Data Environment in 6LoWPAN. *IEEE TIFS*, 17:3831–3846.
- Rührmair, U. et al. (2010). Modeling Attacks on Physical Unclonable Functions. In *Proc. of the 17th ACM CCS Conference*, page 237–249. ACM.
- Schneider, P. and Böttinger, K. (2018). High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks. In *ACM CPS-SPC'18*, pages 1–12.
- Shakevsky, A., Ronen, E., and Wool, A. (2022). Trust Dies in Darkness: Shedding Light on Samsung's TrustZone Keymaster Design. In *Proc. of USENIX Security'22*.
- Strieder, E., Frisch, C., and Pehl, M. (2021). Machine Learning of Physical Unclonable Functions using Helper Data: Revealing a Pitfall in the Fuzzy Commitment Scheme. *IACR TCHES*, (2):1–36.
- Tsiokanos, I. et al. (2021). DTA-PUF: Dynamic Timing-Aware Physical Unclonable Function for Resource-Constrained Devices. *ACM JETC*, 17(3).
- Wen, Y. and Lao, Y. (2017). Enhancing PUF Reliability by Machine Learning. In *ISCAS'17*, pages 1–4. IEEE.

