PUF based Lightweight Hardware Trust Anchor for Secure Embedded Systems

Kai Fischer¹, Andreas Mucha¹, Erwin Hess¹, and Fabian Riess¹
¹Corporate Technology, Siemens AG, Munich, Germany

Abstract—Storage of cryptographic keys on embedded systems is often one of the weakest points of their security architecture, especially if they do not provide a separate security chip, e.g., for cost reasons. But even devices with a security IC in addition to the main controller often lack in mechanisms to securely pair the insecure system and the security IC and are vulnerable for local attackers. We propose the application of Physical Unclonable Functions (PUFs) to realize a lightweight and secure universal key provision token (PUF-KT) to store different keys on devices without a secure non volatile memory (NVM). For embedded devices with a security IC, PUF-KT is used to cryptographically bind the main controller with the security chip without a need to store authentication credentials on the insecure part of an embedded system. As PUF implementation for these mechanisms, we improved the concept of bistable ring PUFs and developed a flexible delay based PUF named Criss-Cross PUF.

Keywords: PUF Key Provisioning Token, Trust Anchor, Criss Cross PUF, Physical Unclonable Function, Embedded Security

1. Introduction

In recent years the amount and professionalism of security attacks on embedded systems has increased significantly. One reason is that Internet-of-Things and cyber physical system (CPS) scenarios are driving the interconnection/communication between embedded devices. Another reason is that embedded devices are getting smarter and gain more attraction for attacks. A broad set of security mechanisms, cryptographic protocols and algorithms are required in open scenarios to protect the confidentiality, integrity and authenticity of communication and data at rest.

Cryptographic protocols and algorithms require secret/private keys and, therewith, it is needed to generate, store and manage the necessary keys on embedded devices. Typically, keys are generated based on physical or seeded pseudo-random number generators and are then permanently stored in a protected memory area (non volatile memory (NVM) or a battery powered RAM). Unfortunately, embedded systems often do not provide a secure memory. Separate security ICs that would offer secure key management and memory are typically not available, commonly due to cost limitation requirements of constrained commercial products. But even devices providing a security IC in addition to the main controller lack in mechanisms to control the access to the security IC as no cryptographic binding between the insecure system and the security IC exists.

In this paper, we propose the application of a lightweight and secure hardware trust anchor for embedded devices based on Physical Unclonable Functions (PUFs). This universal key provision token (PUF-KT) covers three important requirements simultaneously. Firstly, as a PUF – ideally – produces device specific random-looking and unpredictable data it relieves from implementing a physical random source for key generation. Secondly, it addresses scenarios in which a secure NVM for key-storage is not available, as the PUF will generate the desired device specific key every time it is activated. The key provision token allows to generate symmetric as well as asymmetric keys, e.g., for encryption of memory resources, device authentication or secure communication. Thirdly, for the class of embedded devices that already have a security IC on board, we apply the PUF-KT to achieve a cryptographic pairing between the micro controller of an embedded system and the security IC and mitigate attacks in which the security IC is used in unintended contexts.

The rest of the paper is organized as follows: In Section 2, we describe our key provision token approach, followed by Section 3 where we outline possible PUF-KT based usage scenarios like a secure pairing mechanism. In Section 4, we introduce the concept of the Criss-Cross PUF as basis of our PUF-KT implementation. We improved the concept of delay based bistable ring PUFs and provide first evaluation results. At the end, we give a conclusion and outlook of further work.

2. A Universal PUF based Key Provision Token

2.1 Related Work

The direct use of PUFs for device authentication has been intensively discussed in recent years, but the outcome of these investigations is rather negative. The main problem of all the known concepts is that the proposed schemes are vulnerable to machine learning attacks. These attacks yield a software clone that is indistinguishable in its challenge-response-behavior from that of the original PUF. For a comprehensive overview on the status of direct PUF based authentication we refer to [1], [2]. One may conclude that
based on the currently available PUFs – these concepts are not yet mature enough for deployment. On the other hand, these problems are not relevant in the key generation scenario in which the PUF output is never transferred to the “outer world” for direct use. It is used device internal only to generate secret/private keys for usage in cryptographic schemes.

Currently known PUFs cannot be used directly for key provision as they do not show the ideal behavior to produce in a stable way data that are device specific, unpredictable and random. In the literature one can find various approaches to derive a secret key from the output of a PUF; we refer especially to [3]–[5] for detailed descriptions of different concepts. Roughly speaking, all the suggested PUF key provision methods are based on the components of a raw PUF, an entropy extractor (EE) and a post-processing function (PP) (for implementation variants of EE and PP we refer also to [6]–[8]).

2.2 Our Approach

Our idea of a universal PUF based key provision token was not to develop a PUF based alternative to a secure key storage able to output one fixed secret key in a reproducible way. Instead, our goal was to provide a storage able to output one fixed secret key in a reproducible way. We consider the two keys \( K_{M,sym} \) and \( K_{M,priv} \) as the symmetric and the asymmetric PUF-KT internal master keys intrinsically related to the specific PUF implementation under the fixed start configuration.

We modify and extend this process to produce the four desired key types. Generally, the PUF-KT structure offers two modes to generate symmetric as well as asymmetric keys. The mode controls the number of PUF bits to be generated, the output lengths of PP and whether the desired key shall be a permanent or a session key. A configuration vector, which is stored in the internal ROM or in fuses, acts as initial value for a configurable feedback shift register to generate the challenge \( C_{sym} \) or \( C_{priv} \) dependent on the selected mode. Based on the challenge, PP produces a binary master vector, either \( K_{M,sym} \) or \( K_{M,priv} \) of the appropriate lengths for secret and public key cryptography in a reproducible way. We consider the two keys \( K_{M,sym} \) and \( K_{M,priv} \) as master keys and some key derivation data (KD-Data) as additional input. These data may be generated internally or be fed in from outside and allows us to generate different application specific keys. For permanent keys, static data is required as input for key derivation, whereas for session keys volatile data shall be applied (e.g. counter values, random numbers).

For derivation of the four desired key types \( K_{P,sym}, K_{P,priv}, K_{S,sym} \) and \( K_{S,priv} \) we use a key derivation function (Key-Der) taking – respectively – \( K_{M,sym} \) or \( K_{M,priv} \) as master keys and some key derivation data (KD-Data) as additional input. These data may be generated internally or be fed in from outside and allows us to generate different application specific keys. For permanent keys, static data is required as input for key derivation, whereas for session keys volatile data shall be applied (e.g. counter values, random numbers).

The asymmetric cryptographic schemes we have in mind to work with the keys \( K_{P,priv} \) and \( K_{S,priv} \) are elliptic curve based schemes like ECDSA and ECDH. From a cryptographic point of view elliptic curve cryptography (ECC) is currently the most attractive public-key system regarding performance and “security per bits”. For a detailed reference on elliptic curves and ECC we refer to [9]. We mention here only some important properties that simplify essentially the use of ECC as PUF-KT asymmetric scheme compared to...
RSA:

- ECC works with fixed system parameters. These are the underlying finite field \( \mathbb{F} \), the equation defining the elliptic curve \( E \) over \( \mathbb{F} \) and a fixed chosen point \( P \) on \( E \) – the base point of the system.
- There is an operation – point multiplication – that adjoining to any integer \([k]\) and the point \( P \) a new point \( Q := [k] \cdot P \) on \( E \).
- The public keys are derived from the private keys via point multiplication. Given \( K_{M,\text{priv}} \) and a point \( P \), we derive the asymmetric permanent and session keys \( K_{P|S,\text{priv}} \) by applying the key derivation function \( K_{\text{ey}} - \text{Der} \). The associated public keys are generated by a point multiplication \( K_{P|S,\text{pub}} = [K_{P|S,\text{priv}}] \cdot P \) where the binary private keys have to be applied as positive integers.

In contrast to that, the RSA scheme is not well suited for interplay with the PUF-KT generated master key \( K_{M,\text{priv}} \). Actually, it is rather difficult and inefficient to establish a RSA public key system on the PUF-KT approach. \( K_{M,\text{priv}} \) may be used to derive two secret, randomly looking and fixed seeds for the generation of the two primes \( p \) and \( q \) whose product constitute the RSA module \( N = p \cdot q \). To complete the RSA setup one chooses in the next step the intended public key \( e \). To obtain the adjoined private key \( d \) for the RSA system one has to solve the equation

\[ e \cdot d \equiv 1 \mod N. \]

Hence, in the case of the RSA scheme the secret information \( K_{M,\text{priv}} \) derived from the intrinsic PUF secret is only a starting point for the complete RSA parameter construction procedure in contrast to ECC.

2.3 Lifetime properties of the PUF-KT generated keys

The keys generated by the universal PUF-KT have different lifetime properties. All keys get lost after power-off of the embedded system as no secure NVM is claimed to be available. The keys \( K_{P,\text{sym}} \) and \( K_{P,\text{priv}} \) are reproduced based on static (unprotected) NVM data by PUF-KT after system restart. This does not hold for the session specific keys \( K_{S,\text{sym}} \) and \( K_{S,\text{priv}} \) that depend on volatile data.

These different lifetime properties of the keys yield in specific security implications: Compromise of a permanent key does not affect the PUF-KT internal master keys and the derived session keys (under the assumption a strong key derivation functions is used). The same holds for compromise of session keys vice versa. If one the two master keys \( K_{M,\text{sym}} \) and \( K_{M,\text{priv}} \) is broken, a security application using the permanent keys gets insecure. This is different for the session specific keys. If the volatile data used for key derivation are lost, it is impossible to regain these keys. Hence, usage of session keys derived from the master keys supports forward security. However, for future use of the master keys it is required to re-configure the feedback shift register to generate different PUF challenges \( C_{\text{sym}} \) or \( C_{\text{priv}} \) and therewith fresh master keys \( K_{M,\text{sym}} \) and \( K_{M,\text{priv}} \) if one the two master keys is broken.

3. Applications of the PUF-KT

This section gives two important applications that make use of PUF key token.

3.1 External Memory Protection

Crucial security problems in any embedded device scenario are the authenticity and confidentiality of data or programs stored in the external EEPROM/Flash and RAM memories. If the embedded device is not physically protected – this is the usual case – an attacker can easily access and/or change these data. Confidentiality of data can be protected using symmetric cryptographic algorithms \( ENC \) with the key \( K_{P,\text{sym}} \). The data \( DAT \) to be stored in external memory is encrypted to \( C := ENC(DAT, K_{P,\text{sym}}) \) and only \( C \) is stored externally. To regain \( DAT \) again in clear the micro controller calls the decryption algorithm \( DEC \) adjoined to \( ENC \) in connection with the key \( K_{P,\text{sym}} \).

In a similar way authenticity of \( DAT \) is protected by a message authentication code \( MAC(DAT, K_{P,\text{sym}}) \), whereas the message authentication code is stored together with the data in the external memory. For verification whether the data is unmodified the micro controller recalculates the output \( MAC(DAT, K_{P,\text{sym}}) \) and compares the calculated value with the stored one.

3.2 Secure Component Pairing

Up to now, we discussed a PUF based lightweight trust anchor mainly in the role of an alternative to a key storage in a security NVM. We will now consider an application scenario where such a token offers additional security benefit, even in the presence of security μC providing a secure NVM.

A popular architecture for secure embedded systems is often based on the approach to combine a first industrial micro controller μC1 with a security controller μC2, e.g., a smartcard IC or a TPM, to execute the necessary cryptographic algorithms and to hold the keys. If some data \( DAT \) generated by the first controller μC1 has to be protected cryptographically, μC1 sends \( DAT \) to μC2 for execution of the desired cryptographic operation using the appropriate key stored in the security NVM. This may be, e.g., the generation of a digital signature \( sign(DAT) \) to \( DAT \) with the device specific private key \( k_{\text{priv}} \). The result is transferred back from μC2 to μC1, and dependent on the application scenario, μC1 may send the cryptographically protected data in this example, the signed message \([DAT, sign(DAT)]\), to an external recipient.
It is often assumed that this "two-chip approach" provides an overall solution to the security problems of embedded systems. However, this solution does not fit to attacker models in which the attacker has physical access to the embedded system and security IC. It is possible to manipulate the data stream from μC1 to μC2 on physically unprotected embedded devices. Even worse, it is possible to remove μC1 completely and to replace it by a tampered IC or to use μC2 on a different board. In any case, the security IC will execute the requested cryptographic operation without any additional access control.

Typically, access to a security IC is protected by knowledge and proof-of-possession of some kind of credential like a numerical pin or a password. Often an embedded system operates in an unattended way without a possibility to enter a pin or password, so that the credential again needs to be stored in a secure way on the insecure embedded system. To encounter these attack scenarios a fixed pairing between μC1 and μC2 is needed and, hence, the establishment of a secure channel connecting both. This fixed and secure pairing can be achieved with our universal key provision token at the beginning of the device life-cycle and could be realized in the following way:

- The device is inserted into a secure environment.
- The PUF-KT in μC1 is activated and the resulting symmetric key $K_{P,sym}$ is transferred via the device bus to μC2. The key $K_{P,sym}$ is permanently stored in the security NVM of μC2.
- The device is removed from the secure environment and ready for operational use.

Fig. 2 depicts the just described pairing.

In operational service of the embedded device, at any power-on, PUF-KT in μC1 is automatically activated and produces again the key $K_{P,sym}$. Using a challenge-response protocol based on a symmetric authentication algorithm $AUTH$ the security controller μC2 verifies whether or not μC1 indeed possesses $K_{P,sym}$:

1) μC2 generates a random challenge $C$, computes $R = AUTH(C, K_{P,sym})$ and sends $C$ to μC1.
2) On receipt of $C$ the standard controller μC2 computes $R' = AUTH(C, K_{P,sym})$ a sends $R'$ back to μC2.
3) If the received response $R'$ coincides with $R$ the security IC μC2 accepts μC1 as authentic.

With the acceptance of the received response by μC2 the secure pairing of the two controllers is established for the actual session.

In the sequel, additional keys might be derived to support encryption and data authenticity on the device internal link connecting both components. It is also possible to extend the described unilateral authentication to mutual authentication.

4. Criss-Cross PUF

A PUF’s responses should be random and independent, but from an efficiency standpoint, as many response bits as possible should be extractable from a small area footprint. These two goals are conflicting, and depending on which of them is given more weight, different PUF constructions are preferable: PUFs with a response space that scales linearly with their area (also called "weak" PUFs) are usually better at providing independent responses, provided that circuit design and operation do not introduce systematic effects [10], [11], since an independent circuit element is the dedicated source of each response bit. The most well-known representatives of this class are ring oscillator and SRAM PUFs [4]. On the other hand, so-called “strong” PUFs, such as the Arbiter PUF and its derivatives, offer a challenge-response space which scales exponentially with their area, but may be susceptible to machine learning attacks if their true entropy content is much smaller than the theoretical response space [12].

For dynamically generating different sets of keys $K_{M,priv}$ and $K_{M,sym}$, which can be changed over the lifetime of a device, a PUF with a large response space is needed. In addition, as the entropy per bit of a PUF is less compared to a physical random generator, more bits and a larger response space are necessary to achieve the same security level. To maintain its advantage over secured memory areas as key store, the manufacturing costs of the PUF should be minimal, meaning no additional process steps and minimal area. While ring oscillator and SRAM based PUFs fit the first criterion, the flexibility to repeatedly generate different sets of keys makes a strong PUF preferable.

4.1 Analysis of bistable ring based PUFs

The Bistable Ring PUF (BR PUF) [13] was a promising delay-based strong PUF with seemingly complex behavior, but it, as well as the improved derivative TBR PUF, has been attacked successfully with machine learning techniques [14], [15]. Based on the following discussion of the behavior of these PUFs on an analog level, we now will highlight the effects leading to this vulnerability, and subsequently propose the Criss-Cross PUF (CC PUF) as a new PUF primitive which in our view can overcome these problems.

The basic principle of the BR PUF consists of an even number of inverters connected in a ring, resulting in a
system with two stable states. Via multiplexers controlled by challenge bits \( c[i], 0 \leq i < L \), one of two possible signal paths per stage is selected, as indicated in Fig. 3. The circuit is made resettable by forcing it into a defined unstable state where all nodes are at the same value, e.g. by replacing the inverters by NOR gates with one input connected to a global reset signal. After leaving the reset state, the entire ring starts to oscillate in a traveling wave pattern as the inverters try to reconcile their input-output inconsistencies. But each stage has a slightly different driving strength and output load (e.g. routing capacitance), resulting in a different frequency response. The oscillation at different nodes thus gets out of phase, allowing inverters to achieve consistent input-output states one by one. When all inconsistencies in the ring are resolved, the ring settles into its stable state, which is determined by the interplay of the timing characteristics of the ring elements.

Looking at the schematic in Fig. 3, it is clear that a challenge bit \( c[i] \) typically has very little influence on the timing behavior, as only one element of the ring is exchanged. If there is an element in the ring whose timing diverges strongly from the mean, it will dominate the entire ring’s behavior, regardless of the other challenge bits. Even worse, the diverging timing may arise from an element which is not affected by the challenge, such as inter-stage routing capacitance, which is often neglected in simulations, but can be relatively large, especially in FPGAs. This can result in a settling state completely independent of the challenge. In practice, it is the position of the slowest element in the ring which has the highest influence on the final state, as arriving fast oscillations get damped there and the influence of other elements can be smoothed over.

This problem was correctly identified by the creators of the TBR PUF, in which each challenge bit swaps the position of two corresponding inverter elements in the ring [14]. However, the interconnection segments between stages remain unaffected by the challenge.

### 4.2 Benefits of the Criss-Cross PUF

Based on the above analysis, we propose the Criss-Cross PUF (CC PUF) as a new PUF primitive which overcomes this limitation. As shown in Fig. 4, the CC PUF consists of an even number of stages, each stage having two input signals and two inverters, and a corresponding challenge bit controls which input is applied to which inverter. The two inverter outputs are applied to the next stage, and so the two signals can “criss-cross” between the upper and lower tracks, with the challenge determining the order of the inverters as well as the interconnection segments traversed. Responses are generated as in the (T)BR PUF from the two possible settling states of a bistable ring after a reset signal (not shown in Fig. 4) is briefly applied to enforce an initial unstable state.

Defining that a ‘1’ bit in the challenge means that the tracks are crossed, an interesting property of the CC PUF becomes apparent: If the number of ‘1’ bits in the challenge (equivalently: its Hamming weight) is even, two independent bistable rings with length equal to the number of stages result, from whose settling states two theoretically independent response bits can be extracted. If the Hamming weight is odd, a single bistable ring with twice the length results, yielding one bit of settling state information, but generated from a more complex (and longer) stabilization process. In both cases, all ring elements remain active in determining the PUF’s response, like in the TBR PUF. The CC PUF stage however requires only two input-side multiplexers and is thus easier to implement with balanced routing than the TBR stage. In FPGA implementations, it fits in two 3-input LUTs, and is also more area efficient as an ASIC implementation. The key difference however is that in the CC PUF, not only the traversal order of the inverter elements but also the order of the interconnection segments between stages is permuted by the challenge, which improves the behavior in the presence of significant and variable routing capacitance.

This is confirmed by transistor-level SPICE simulations, from which data is presented in Fig. 5. The histograms show inter-device Hamming distances from 100 Monte Carlo runs of TBR (top) and CC PUFs (bottom) under various conditions of the inter-stage capacitance: Without capacitance (left), and with a nominal capacitance of 100 fF whose random variability was set to 0, 10%, and 20%. The results show that the TBR is affected negatively by increasing influence of inter-stage capacitance, with its responses becoming more and more biased. Simulated BR PUFs (data not shown) were affected even more. The reverse seems to be the case...
for the CC PUF, where the responses are in fact biased in the ideal case, but become more evenly distributed with higher influence of the inter-stage capacitance. This confirms our notion that the CC PUF can make use of inter-stage routing variability to improve response entropy.

### 4.3 FPGA Evaluation of the Criss-Cross PUF

A first evaluation of the CC PUF based on FPGA implementations was also performed with ten Xilinx Zynq Z-7020 SoC devices. Implementations of length 16 were placed four times on the chip, length 32 twice and length 64 once. In the following, different placements on the same chip are referred to as subinstances. Sets of 4096 challenges chosen randomly from each challenge space were split into “even” and “odd” sets according to their Hamming weights to obtain a good coverage of the challenge space while accounting for the different behaviour of the CC PUF depending on the challenge type. Although theoretically two response bits could be extracted per “even” challenge, only one will be used in the following for practical reasons.

Fig. 6 shows the obtained inter-device Hamming distances. The distributions for “even” challenges are more compact than those for “odd” challenges. The latter generated a relatively high rate of response timeouts (defined as the ring not stabilizing within 327.68 μs or $2^{15}$ clock cycles): On average, timeouts occurred with 8%, 15%, and 22% of “odd” challenges for 16, 32, and 64 stages, respectively, and virtually never with “even” challenges. The response was considered invalid in this case. The correlation between timeout rate and ring length is expected, as longer distances between any remaining inconsistent states traveling around the ring translate into a longer time until they can meet and neutralize each other, but it is not yet clear why the two types of challenges behave so differently in this regard. In summary, the inter-device distributions are tighter than in the simulation results, indicating that inter-stage influences have higher influence and variability than accounted for in the simulations. The median value of the inter-device distance is below 0.5 in all cases, however. This suggests that some systematic component remains in the response behaviour of all implemented PUFs.

As PUF responses need to be stable under varying environmental conditions, the CC PUF implementations were also tested in a temperature controlled chamber between -40 °C and 100 °C. In Fig. 6, the intra-device distances between all temperature pairs from this experiment are plotted next to the corresponding inter-device distances. For the responses from the “even” challenge sets, there is a safe margin to reliably reconstruct a nominal response through error correction and identify devices under large temperature variations. To give an accurate impression of the PUF’s reliability, for the calculation of the intra-distance, any comparison involving an invalid response bit arising from a timeout was evaluated as different. Therefore, the intra-distances for the “odd” challenge sets are higher. If invalid responses are ignored, the intra-distance is in fact on the same level with both types of challenges.

Systematic influences could not be eliminated completely.
in the FPGA implementations, resulting in some response bias and an average Shannon entropy of 0.6 bit per response bit. Two PUF response bits per key bit could thus be sufficient for key generation with the current implementation, but a much higher factor is both advisable for security and unproblematic due to the large challenge space.

5. Conclusion

PUFs offer an approach as lightweight trust anchors that provide cryptographic keys in a secure way and without a need to store them on external non-volatile memory. Additionally, the PUF based key can be applied for requirements like application specific key derivation, memory and firmware encryption or pairing of security ICs with a specific instance of a main controller.

As basis for these PUF applications, we implemented the Criss-Cross PUF, which improves BR PUF limitations regarding uniformity of response bits. First measurements of the FPGA implementation showed reduced bias values compared to the BRPUF, but does not reach the ideal value of 0.5 due to non-controllable side effects of the FPGA. Therefore, we will implement the Criss-Cross PUF as ASIC in a second step and expect further improved bias values. Finally, the Criss-Cross PUF, the PUF key provision token and some of the outlined application scenarios will be evaluated in a demonstrator of an industrial scenario.

Acknowledgment

This work was partially supported by the German Ministry for Education and Research (BMBF) project SIBASE under grant number 01IS13020.

References