Lightweight Authentication and Secure Communication Suitable for IoT Devices

Simona Buchovecká, Róbert Lórencz, Jiří Buček and Filip Kodýtek
Department of Information Security, Faculty of Information Technology,
Czech Technical University in Prague, Czech Republic

Keywords: Authentication, Secure Communication, PUF, TRNG, Key Generation, Key Management, IoT Security.

Abstract: In this paper we present the protocols for lightweight authentication and secure communication for IoT and embedded devices. The protocols are using a PUF/TRNG combined circuit as a basic building block. The goal is to show the possibilities of securing communication and authentication of the embedded systems, using PUF and TRNG for secure key generation, without requirement to store secrets on the device itself, thus allowing to significantly simplify the problem of key management on the simple hardware devices and microcontrollers, while allowing secure communication.

1 INTRODUCTION

Implementation of proper methods for authentication and secure communication, including secure management of key material in lightweight devices, is of significant importance, especially with rise of IoT devices. The use of cryptography and corresponding keys in proper manner is one of the leading problems, when talking about devices with limited computing resources and low power consumption.

Variety of communication protocols were proposed for secure authentication and commuunication in IoT world, such as machine-to-machine/Internet of Things connectivity protocol (MQTT), Constrained Application Protocol (CoAP), or Datagram Transport Layer Security (DTLS) that can be integrated with CoAP. However, those are still rather heavy-weight and computationally quite expensive protocols when considering simple and constrained devices and thus their variants are constantly being present, for instance Lithe (Raza et al., 2013) or E-Lithe (Haroon et al., 2007) as a lightweight variant of DTLS (Tschofenig et al., 2016). Moreover, these protocols do not deal with secure generation and storage of cryptographic keys, which is rather prerequisity for their usage.

Before introducing the principles of Physically Unclonable Functions (PUFs), we shall summarize the currently most widely used methods for key generation and storage. Nowadays, Random Number Generators (RNGs) are mostly used for key generation that are further used in cryptographic protocols for authentication and secure communication. For lightweight and embedded devices, the True Random Number Generators (TRNGs) are usually implemented, utilizing non-deterministic effects in analogue or digital circuits, since this is resource and power efficient way. Therefore, the quality of TRNG has a significant influence on security of whole system. Improperly implemented TRNG often leads to compromise of the whole system or reduces the complexity of the attack.

Moreover, once the key is generated, it needs to be stored securely in the device (Handschuh et al., 2010), e.g. utilizing storage with tamper-resistance techniques implemented. However, to implement such measures is a complex and cost-ineffective task,

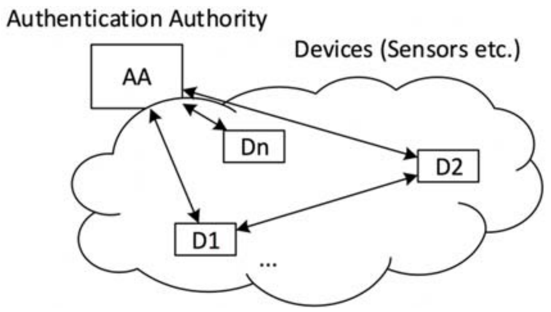

Figure 1: Interconnected systems with an Authentication Authority.

Buchovecká, S., Lórencz, R., Buček, J. and Kodýtek, F.
Lightweight Authentication and Secure Communication Suitable for IoT Devices.
DOI: 10.5220/0008959600750083
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 75-83
ISBN: 978-989-758-399-5; ISSN: 2184-4356

therefore often neglected in practical applications.

Thus, properly defined and implemented key management, including proper key generation, key storage and key usage for various applications (authentication, access control, encryption) in interconnected IoT and embedded systems, as depicted in Figure 1 is still a challenging task (Roman et al., 2013, Malina et al., 2016). A consistent way of handling various cryptographic keys, possibilities of reusing traditional security mechanisms and ensuring end-to-end integrity verification mechanisms is needed (Sicari et al., 2015). All security protocols require credentials, thus optimal key management systems must be implemented to store and distribute these credentials (Roman et al., 2013).

The systematic and formalized approach to key management in IoT devices and embedded systems with properly defined requirements, as well as efficient light-weight modules for key generation, storage and secure usage are missing. However, the need for proper key management in particular applications of embedded systems and IoT started to be raised in some research papers with regards to specific areas such as automotive context (Schleiffer et al., 2013), distributed sensor networks (Chan et al., 2005), or embedded systems in general (Sklavos et al., 2016).

Though TRNGs are mostly used nowadays for cryptographic key generation, in recent years, numerous works dealing with Physical Unclonable Functions (PUFs) for key generation had been published, proposing PUFs as another possible approach for key generation. The concept of PUF was originally introduced in (Pappu, 2002), showing that instead of relying on number theory, the mesoscopic physics of coherent transport through a disordered medium can be used to allocate and authenticate unique identifiers by physically reducing the medium's microstructure to a fixed-length string of binary digits. These physical one-way functions are inexpensive to fabricate, prohibitively difficult to duplicate, admit no compact mathematical representation, and are intrinsically tamper-resistant. This makes PUF as ideal candidate for providing tamper resistant design for cryptographic key generation and storage.

Therefore, PUFs usage is promising to solve the issue of secure storage of cryptographic keys. Instead of storing the key in memory, the key is generated at the time it is needed. A combined PUF/TRNG circuit used in our paper is therefore a suitable alternative for the purpose of key generation and authentication in lightweight cryptographic applications, such as IoT devices and other embedded platforms.

The structure of this paper is as follows. In Section 2, related work is summarized. In Section 3, our proposed approach to lightweight authentication and secure communication is presented. Section 4 presents a case study with a specific PUF/TRNG circuit. Section 5 concludes this paper.

2 RELATED WORK

Every protocol for authentication and secure communication requires cryptographic keys. The minimum common requirements for key generation and storage are summarized by Maes et al. (Maes et al, 2012): A source of true randomness that ensures unpredictable and unique fresh keys; and a protected memory which reliably stores the keys information while shielding it completely from unauthorized parties. As further discussed in (Fischer, 2012), the security of cryptographic systems is mainly linked to the protection of confidential keys. In high-end information security systems, when used in an uncontrolled environment, cryptographic keys should never be generated outside the system and they should never leave the system in clear. For the same reason, if the security system is implemented in a single chip (cryptographic system- on-chip), the keys should be generated inside the same chip.

As mentioned above, for secure key generation a source of true randomness is needed, and the generated keys should be unpredictable. Thus, for proper generation of cryptographic keys random bit stream is required. Therefore, traditional methods of generating cryptographic keys in hardware and embedded systems are mainly based on true random number generators (TRNGs). As stated by Schindler (Schindler, 2009), ideal random number generators are characterized by the property that the generated random numbers are independent and uniformly distributed on a finite range. Various TRNG designs suitable for cryptographic key generation include purely digital designs (Epstein et al. 2003, Fairfield et al. 1984), Phase-Locked Loops in designs targeting FPGAs (Fischer and Drutarovsky 2012, Deak et al. 2015), Random access memories (Gyorfi et al. 2009) or multiple designs (Kohlbrenner and Gaj, 2004, Bucci et al., 2003, Golic, 2006, Tkacik, 2003) based on Ring Oscillators as a source of entropy.

As discussed in (Maes et al, 2012), PUF-based key generators try to fulfill two requirements on secure key generation and storage at once. The randomness of the PUF response comes from the manufacturing process variation, and it is intrinsically present in the device. There is no need for a protected non-volatile memory since the randomness is measured only when needed. However, the PUF output may slightly vary in different measurements, and it is still challenging to get static PUF output as required by cryptographic schemes. Existing PUF designs proposed for cryptographic applications include PUFKY based on a ring oscillator PUF (Maes et al, 2012) providing low-failure rate, generation of read-once keys (Kirkpatrick et al. 2010), single-chip secure processor for embedded systems (Suh et al., 2007), arbiter PUF for device authentication and secret key generation (Suh and Devadas, 2007) and others.

Our goal is to propose a secure communication and authentication method using a combined PUF/TRNG circuit that will allow secure generation of keys using both PUF and TRNG at the same time, maximizing benefits of each one. There have been several similar works published recently, however, the first attempts in using PUF for the device authentication were rather simple. In (Suh and Devadas, 2007) simple authentication against authentication authority was discussed, using pre-generated challenge-response pairs stored centrally. At the authentication time, the challenge is sent to the device and response then compared with the output. Same challenge cannot be reused again due to possible replay attacks.

More sophisticated PUF-based authentication protocols were reviewed in (Delvaux et al., 2014). The work of (Ozturk et al., 2008) using reprogrammable non-volatile memory; Hammouri et al. (Hammouri et al. , 2008) using two arbiter PUFs; protocol based on logically reconfigurable PUFs (Katzenbeisser et al. 2011) which allows to recycle the challenge tokens; Reverse Fuzzy Extractor (Van Herrewege et al., 2012) allowing mutual authentication; Slender PUF protocol (Majzoobi et al., 2012) that does not expose the full PUF responses, only the random subset instead; and Converse authentication Protocol (Kocabas et al., 2012) which provides one-way authentication of the server.

All of the protocols discussed above deal with authentication only, leaving the need for key establishment and secure communication open, which is also one of the main conclusions of the PUF authentication usage review (Delvaux et al., 2014), discussing the caveats of the PUF responses being not perfectly reproducible, small output space of strong PUFs or need of secure TRNG, that is substantial for most of the protocols.

Another concern is privacy of the authenticated devices. As discussed in (Bolotnyy and Robins, 2007), an algorithm is privacy preserving if an adversary cannot distinguish between any pair of devices, and thus, the PUF must be able to generate long chains of unique IDs (i.e., without repetitions). Possible approach to privacy preserving authentication protocol is presented in (Aysu et al., 2015), based on fuzzy extractor with helper-data construction technique based on TRNG.

In this paper we discuss protocols for authentication and secure communication utilizing PUF and TRNG. We show, it is advantageous to have single module that will allow generation of both TRNG and PUF at the same time, since it minimizes implementation requirements and operational resource consumption. The aim is not to require storing secrets on the IoT or embedded system itself to simplify key management on the simple hardware devices and microcontrollers.

3 PROPOSED APPROACH

When designing the embedded module for secure authentication and communication, the main goal is to simplify the key management on the endpoint embedded device itself. Thus, we propose to utilize the single circuit for key generation using PUF and TRNG as a basic building block of the module so that

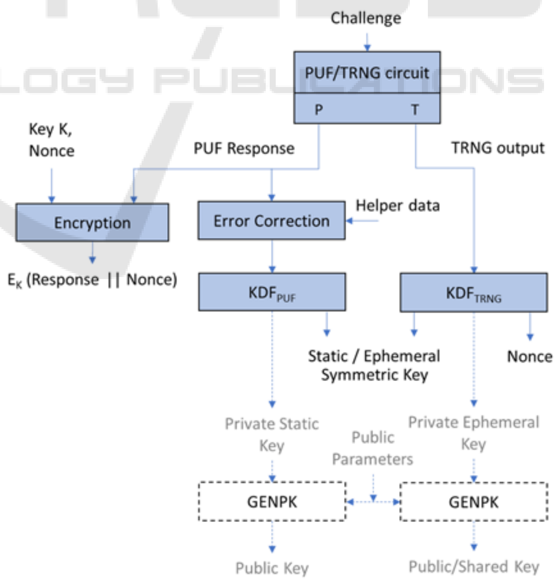

Figure 2: Embedded module for secure authentication and communication. KDF is a Key Derivation Function, GENPK generates a public key from a private key and public parameters. Error Correction is used to obtain a stable key material from the PUF Response (see Section 3.1). Functions covered in this paper are coloured in blue.

there is no need to store secrets on the hardware device.

The overall module depicted in Figure 2 provides PUF authentication and PUF/TRNG based key generation. For the authentication, the PUF is used, since it provides randomness intrinsically present in the device and utilizes the fact that the generated response is unique per device. Since there is a need for both static key, as well as ephemeral keys, combination of PUF and TRNG is used in this case – the PUF is used for generation of static (private) key that never leaves the device, thus utilizing all the advantages of PUF, while TRNG is used for generation of ephemeral, one-time keys, that are shared with other communicating parties.

Asymmetric schemes are suitable if the private key is easily generated from random sequence by a Key Derivation Function (KDF), such as PBKDF2 (RSA Laboratories, 2012). For example, ElGamal encryption (ElGamal, 1985) and DSA/ECDSA signature schemes can be used, if good quality public parameters are chosen (generation of the public key from a private key si denoted as GENPK in Figure 2). On the contrary, an RSA key requires more complex processing including secure prime generation. TRNG output is also used to generate random nonce and padding data. In this paper, we will focus only on symmetric schemes (with the exception of Algorithm 1).

The proposed protocols for authentication against a central authentication authority, and also mutual device-device authentication, are further discussed in detail in following sections.

3.1 Authentication against Central Authentication Authority

Before any communication is allowed the connected device must be properly authenticated. Since the PUF responses are unique per each device, and are intrinsically random, i makes PUF the ideal cryptographic primitive for device authentication. We propose simple and straightforward authentication protocol using pre-generated challenge-response pairs that can be easily implemented in hardware devices. This protocol does not require the PUF to have a large space of challenge-response pairs (it can be used even for one challenge-response pair). The authentication protocol consists of two phases – secure enrolment phase and authentication phase itself and is depicted in Algorithm 1.

The Enrolment phase is critical for the security of all protocols based on PUFs, and (analogous to biometric authentication methods) must be performed in a secure environment. We assume that a suitable environment can be created (for example, by separate physical access to the devices), but specific means are not elaborated in this paper.

During the Enrolment phase of Algorithm 1, the challenge/response pair(s) (C, R) are measured from the targeted device and securely stored at the central authenticating authority (AA), that can be either integrated into the gateway or be represented by separate device that the gateway is querying during authentication process. A database DB_{D_i} of the pairs (C, R) is created for each device D_i. Furthermore, the public key (PK_{AA}) of the authenticating authority is pre-set on the device, so as the authentication data can be securely transferred. For this purpose, an asymmetric scheme (ElGamal) can be used, as proposed in the section above. We assume that PK_{AA} is protected against unauthorized changes (by the tamper-evidence property of the PUF).

The first 4 steps of the Enrolment phase are common for all 3 algorithms presented in this paper. The database DB_{D_i} is used also in the Authentication phases of Algorithms 2 – 3.

```
Enrolment phase (Secure environment)
Common for Algorithms 1 – 3:
1. AA → D1: Challenges ($C_1$, $C_2$, ...)
2. D1:      $R_1$ = PUF($C_1$), $R_2$ = PUF($C_2$) ...
3. D1 → AA: Responses ($R_1$, $R_2$, ...)
4. AA:      Store ($C_i$, $R_i$) to $DB_{D1}$
Specific only for Algorithm 1:
5. AA → D1: Public key $PK_{AA}$
6. D1:      Store($PK_{AA}$)

Authentication phase for D1
1. AA:      Choose (C, R) from $DB_{D1}$
2. AA → D1: Challenge C, Nonce N
3. D1:      R' = PUF(C)
4. D1 → AA: CR = $E_{PK\_AA}$(R' || N)
5. AA:      (R', N') = $D_{SK\_AA}$(CR)
            Compare($R \cong R'$), Compare(N = N')
```

Algorithm 1: Enrolment and Authentication against central Authentication Authority.

The Authentication phase of Algorithm 1 is executed every time the device is connected to the network and needs to be authenticated. AA randomly chooses one of the challenges C and sends it together with the nonce value N to the device to be authenticated. The nonce value is used to prevent simple replay attacks and allows each challenge-response pair to be used repeatedly. On the device that is being authenticated the appropriate PUF response is generated, concatenated with nonce value and encrypted with public key of the Authenticating Authority. Authenticating Authority then compares (strictly) if the decrypted nonce value N' = N. Since the PUF response may slightly vary across various measurements, a predetermined number of faulty bits in R' is tolerated. If both match, the device is successfully authenticated. The disadvantage of Algorithm 1 is that it only performs authentication and does not provide a cryptographic context for future communication.

Authentication of a single device (D1) to the AA without asymmetric cryptography is depicted in Algorithm 2. (The Enrolment phase is the same as in Algorithm 1, steps 1. – 4.) This method includes generating a shared symmetric key K, which requires a stable error-free PUF output. This is achieved by using an error-correcting code (ECC), denoted in the algorithm by its functions Encode and Decode. This code must have enough redundancy and structure to correct the maximum amount of errors assumed in the PUF when operated under various conditions (voltage, temperature etc.).

Choosing a suitable ECC depends on the bit error rate and length of PUF response while meeting the required corrected output length. The computational power of the device is also a limiting factor. In the case of "lightweight" devices, simple codes (such as a repetition code) are preferable.

Authentication phase – using symmetric cipher

```
1.  D1 → AA: Call(D1)

2.  AA:    r = TRNG()
3.         Choose (C, R) from $DB_{D1}$
4.         H = R ⊕ Encode(r)
5.         K = KDF(r)

6.  AA → D1: Challenge C, Helper string H

7.  D1:    R' = PUF(C)
8.         r = Decode(R' ⊕ H)
9.         K = KDF(r)

10. D1 ↔ AA: Authentication + Encryption with K
```

Algorithm 2: Authentication of a device D1 to the AA.

The helper string H is a distance from the raw PUF response R to the random codeword Encode(r). It is computed by the AA (step 4 of Algorithm 2). The device then uses it to recover the key material (step 8), and subsequently derive the key K.

The shared key K can be used for authentication and encrypted communication, as opposed to Algorithm 1, which covers only authentication, limiting its usefulness. On the other hand, Algorithm 1 does not require the generation of a helper string, nor does it need any error correction codes.

3.2 Mutual Device Authentication

Not only the device needs to be authenticated to central authority when connected to the network, the devices must be mutually authenticated before they start to communicate, as well. Similarly, as in the previous case, central authenticating authority stores the pre-generated challenge-response pair(s), and acts as trusted 3rd party. This time though, a shared symmetric key is established between the two devices, and a conventional symmetric authenticated and encrypted session can follow afterwards. The goal is to use the PUFs in both devices D1 and D2, but not transmit any PUF response over the network. By using the one-wayness of the hash functions used, no device gets to know other device's PUF response, even if it monitors all communication. An error correcting code is used to ensure stable PUF outputs. The codewords are selected randomly from the code space by the AA. The overall process is described in Algorithm 3.

Mutual authentication of D1 and D2 using AA

```
1.  D1 → AA: Call(D1, D2)

2.  AA:    $r_{D1}$ = TRNG()
3.         $r_{D2}$ = TRNG()
4.         Choose ($C_{D1}$, $R_{D1}$) from $DB_{D1}$
5.         Choose ($C_{D2}$, $R_{D2}$) from $DB_{D2}$
6.         $H_{D1} = R_{D1} \oplus$ Encode($r_{D1}$)
7.         $H_{D2} = R_{D2} \oplus$ Encode($r_{D2}$)
8.         r = Hash($r_{D1}$) ⊕ Hash($r_{D2}$)

9.  AA → D1: ($C_{D1}$, $H_{D1}$, r)
10. AA → D2: Call(D1, D2) , ($C_{D2}$, $H_{D2}$, r)

11. D1:    $R'_{D1}$ = PUF($C_{D1}$)
12.        $r_{D1}$ = Decode($R'_{D1} \oplus H_{D1}$)
13.        Hash($r_{D2}$) = Hash($r_{D1}$) ⊕ r
14.        K = KDF(Hash($r_{D1}$) || Hash($r_{D2}$))

15. D2:    $R'_{D2}$ = PUF($C_{D2}$)
16.        $r_{D2}$ = Decode($R'_{D2} \oplus H_{D2}$)
17.        Hash($r_{D1}$) = Hash($r_{D2}$) ⊕ r
18.        K = KDF(Hash($r_{D1}$) || Hash($r_{D2}$))

19. D1 ↔ D2: Authentication + Encryption with K
```

Algorithm 3: Mutual device authentication and secure communication.

Let us assume that D1 wants to authenticate with D2 and set up a secure communication channel. D1 initiates the process by calling the AA with the identification of D1 and D2 (CALL(D1, D2)). AA contains the complete table of challenges and responses (C_{D1}, R_{D1} etc.). An error correcting code is chosen that can correct enough errors to make the PUF response stable, with the corresponding functions Encode and Decode. AA generates two random components r_{D1}, r_{D2} from the set of preimages, and encodes them, thereby forming randomly chosen codewords. The code length should correspond to the PUF response length. Helper strings H_{D1} and H_{D2} are created by XORing the expected PUF response (R_{D1}, R_{D2}) to the corresponding codeword. The two random components are hashed and the hashes XORed to form r.

To each of the devices, a triplet (C_{Di}, H_{Di}, r) with the challenge, helper string, and r is sent. Also, in step 10, AA relays the request for communication from D1 to D2. Each of the devices challenges its own PUF to get the response (R'_{D1}, R'_{D2}). By XORing the response with the corresponding helper string (H_{D1}, H_{D2}), resulting with a codeword with errors, which is then corrected by the Decode function. This way, each device recovers its component (r_{D1}, r_{D2}). D1 recovers the value Hash(r_{D2}) by XORing r with the hash of its r_{D1}, and vice versa. Moreover, both devices know the hashes of r_{D1} and r_{D2}, and can derive the shared key K by applying a key derivation function KDF on the concatenation of the hashes.

The hashing of r_{D1}, r_{D2} is done to hide the PUF responses from the other device. If D1 monitors the communication, it will know (C_{D1}, C_{D2}, H_{D1}, H_{D2}, r). It can recover r_{D1}, and if the hashing were not done, and r would be equal to $r_{D1} \oplus r_{D2}$ directly, D1 would compute r_{D2}, and using the helper string H_{D2}, it could discover the PUF response R_{D2}. We would have to either trust all devices in the network or use all challenges only once and discard them. In our case, because we do use hashing of r_{D1}, r_{D2}, D1 only gets Hash(r_{D2}), and the one-wayness of the hash function prevents it from discovering R_{D2}. Thus, we can reuse the challenges for future authentications.

PUF response correction code choice depends on the number of bit flips inherent in the PUF operation. The code length and codeword distance determine the number of information bits, thus the length of r_{D1}, r_{D2}, and limit the entropy contained in r. By using the same challenge with multiple random r_{Di}, we can extract more bits of entropy from the PUF. The entropy of the resulting shared key K is determined by the properties of used hash functions and KDF, and the inputs. If chosen correctly, it is as high as the entropies of r_{D1}, r_{D2}. The key K is always derived from randomly chosen codewords, and therefore for the same PUF challenges (C_{D1}, C_{D2}), a different K is obtained.

3.3 Secure Communication

After the authentication process described in the previous section, a shared key is established. At this point, a conventional symmetric authentication and session key derivation process can be performed using block ciphers such as AES. Several lightweight block ciphers suitable for embedded systems or sensor networks has been proposed, such as PRESENT (Bogdanov et al., 2007, McKay, 2017) with an 80-bit key. This allows generating the key in a single run of PUF circuit for most of the PUF designs and implementations, with no further stretching needed.

All presented algorithms in this Section utilized only PUF on the side of the devices and TRNG was used on AA. TRNG functionality on the devices is used after the secure channel establishment (steps 10 and 19) in dependence on the communication protocols. Random numbers are needed in many classical authentication protocols (Menezes et al., 1996, chapter 12), as well as modern internet standards such as DTLS (Tschofenig et al., 2016).

4 CASE STUDY

As arises from previous section, both TRNGs and PUFs have different characteristics that are advantageous in different applications. Thus, various implementations of cryptographic systems can take an advantage from a universal circuit for generation of PUF and TRNG at the same time, that allows secure generation of symmetric (session) keys (and potentially also asymmetric (private) keys). Such

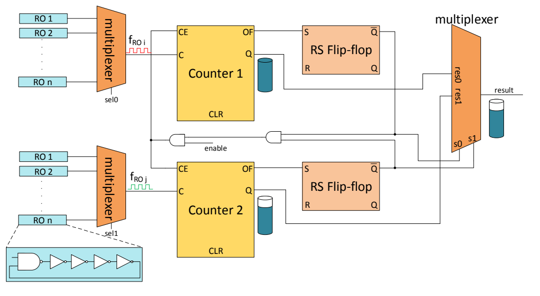

Figure 3: PUF/TRNG circuit based on Ring Oscillators, serving as basic building block for proposed authentication and secure communication scheme (Kodýtek et al., 2015, Buchovecká et al., 2017).

PUF/TRNG based on Ring Oscillators – ROPUF circuit was presented in our previous work (Kodýtek et al., 2015, Kodýtek et al., 2016, Buchovecká et al. 2016, Buchovecká et al., 2017), so the idea of the single RO circuit can be used both for PUF and TRNG generation was validated. This circuit is depicted in Figure 3.

In order to validate the proposed authentication process outlined in Section 3, we performed an experiment on one device containing the ROPUF design (Kodýtek et al., 2015, Buchovecká et al., 2017). For this purpose, we used a ROPUF design that consisted of 2 groups of ring oscillators (ROs), each group contained 150 ROs. Only ROs from different groups were selected to form a pair, which was then used to generate part of the PUF response. We extracted 3 bits from each RO pair and enhanced the stability of the PUF output by applying Gray code on these bits (Kodýtek et al., 2016). Finally, to create the PUF response, the selected bits from all of the RO pairs are concatenated.

In the first case, we generated the PUF responses from 150 pairs of ROs (each RO from each group was used only once), in the other, each RO was used five times (one RO from the first group is paired with 5 ROs from the other group) resulting in 750 RO pairs. These two setups achieved 450 and 2250 bits of PUF response respectively. In both cases, we performed 1000 measurements, from which we obtained a majority PUF response - R_{Di} (we determined the majority for each position of the PUF output).

In our experiment, the block length of 9 bits proved to be sufficient for the repetition code. In order to create the helper string H_{Di}, we need to generate 50 or 250 random bits (r_{Di}) that are then encoded by the repetition code and XORed with the major PUF output, forming the helper string H_{Di}. This process is related to steps 2 and 4 in Algorithm 2.

The example using a simple repetition code with 5-bit block length is depicted on Figure 4. On the device, the PUF generates a response R'_{Di} that is corrected by the helper string H_{Di}, corresponding to steps 7 and 8 in Algorithm 2. After correction, we obtained 50 and 250 bits respectively. These bits can be used to create a cryptographic key. For Algorithm 2, we can simply represent KDF as the selection of the first 128 bits (from r_{Di}) for symmetric cipher AES.

The same can be applied for Algorithm 3, where two devices are authenticating each other. However, this algorithm is more complex, since it requires implementation of suitable hash function. In case of Algorithm 1, no KDF is needed, since the AA's

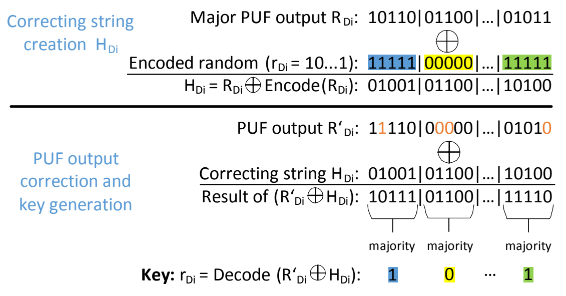
Correcting string creation H_{Di}
Major PUF output R_{Di}: 10110|01100|...|01011
Encoded random (r_{Di} = 10...1): 11111|00000|...|11111
$H_{Di} = R_{Di} \oplus \text{Encode}(R_{Di})$: 01001|01100|...|10100
PUF output correction and key generation
PUF output R'_{Di}: 11110|00000|...|01010
Correcting string H_{Di}: 01001|01100|...|10100
Result of $(R'_{Di} \oplus H_{Di})$: 10111|01100|...|11110
majority
majority
majority
Key: $r_{Di} = \text{Decode}(R'_{Di} \oplus H_{Di})$: 1 0 ... 1

Figure 4: Example of a simple repetition code with 5-bit groups.

public key is stored on the device and PUF is not used to derive any cryptographic key.

To increase the number of bits after correction, we can either use a more efficient error correcting code or we can reuse the same challenge multiple times with a new random codeword each time. The experiment showed and confirmed that it is possible to generate key material for the proposed protocols, using the state of the art PUF/TRNG designs, in sufficient length and quality.

5 CONCLUSIONS

In the paper, we discussed the need for the proper key management of cryptographic keys on the embedded devices and further proposed the design of the module for secure authentication and communication that fulfills the requirements for the secure generation and storage of the cryptographic keys, including proposal of basic authentication and secure communication protocols.

For the authentication, several protocols based on pre-generated PUF challenge/response values are proposed. Since the PUF responses are unique per each device and are intrinsically random, PUF is ideal cryptographic primitive for this purpose. Three variants of the protocol are discussed – authentication against central authority using PUF challenge and encrypted response, and two variants of authentication that use the PUF for key generation – single device authentication, and mutual device authentication.

After the authentication process, a shared key between the devices is established. At this point, a conventional symmetric authentication and session key derivation process can be performed using conventional or lightweight block ciphers as needed.

Further, we discussed the case study and suggested possible implementation of the module for secure communication and authentication, using

ROPUF/TRNG circuit. As it was presented and validated in previous work (Kodýtek et al., 2015, Kodýtek et al., 2016, Buchovecká et al., 2016, Buchovecká et al., 2017) a pair of RO circuits can be used both for PUF and TRNG generation, thus serve as basic building block for the module. Moreover, it is possible to generate the sequence long enough for the key generation in one run of ROPUF/TRNG circuit.

In the paper we have shown the possibilities of securing communication and authentication of the embedded systems, using PUF and TRNG for secure key generation, without requirement to store secrets on the device itself, thus allowing to significantly simplify the problem of key management on the simple hardware devices and microcontrollers.

Future work will be devoted to secure communication with a suitable asymmetric encryption scheme, using both PUF for generation of private key, as well as TRNG for generation of ephemeral keys, making use of the randomness already intrinsically present in the device. Thanks to PUF, the private key is generated when needed, thus there is no need for storing secrets on the device itself.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics”.

REFERENCES

Aysu, A., Gulcan, E., Moriyama, D., Schaumont, P., & Yung, M. (2015). End-to-end design of a PUF-based privacy preserving authentication protocol. *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg.

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., & Vikkelsoe, C. (2007). PRESENT: An ultra-lightweight block cipher. *International workshop on cryptographic hardware and embedded systems* (pp. 450-466). Springer, Berlin, Heidelberg.

Bolotnyy L., Robins G. (2007). Physically unclonable function-based security and privacy in RFID systems. *Pervasive Computing and Communications, 2007. PerCom'07*. Fifth Annual IEEE International Conference on. IEEE.

Bucci, M., Germani, L., Luzzi, R., Trifiletti, A., & Varanonuovo, M. (2003). A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC. *Computers, IEEE Transactions* on 52.4 (2003): 403-409.

Buchovecká, S., Kodýtek, F., Lórencz, R., Buček J. (2016) True Random Number Generator Based on ROPUF Circuit. *Digital System Design (DSD), 2016 Euromicro Conference on*. IEEE.

Buchovecká, S., Kodýtek, F., Lórencz, R., Buček J. (2017). True random number generator based on ring oscillator PUF circuit. *Microprocessors and Microsystems* 53 (2017): 33-41.

Chan H., Gligor V. D., Perrig A., Muralidharan G. (2005). On the distribution and revocation of cryptographic keys in sensor networks. *IEEE Transactions on Dependable and Secure Computing*, 2(3):233–247.

Deak N., Gyorfi T., Marton K., Vacariu L., Cret O. (2015). Highly efficcient true random number generator in FPGA devices using phase-locked loops. *20th International Conference on Control Systems and Computer Science*, pages 453–458. IEEE.

Delvaux, J., Gu, D., Schellekens, D., & Verbauwhede, I. (2014). Secure lightweight entity authentication with strong PUFs: Mission impossible? In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg. p. 451-475.

ElGamal T. (1985). A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472.

Epstein M., Hars L., Krasinski R., Rosner M., and Zheng H. (2003). Design and implementation of a true random number generator based on digital circuit artifacts. *In International Workshop on Cryptographic Hardware and Embedded Systems*, pages 152–165. Springer.

Fairfield R. C., Mortenson R. L., and Coulthart K. B. (1984). An LSI random number generator (rng). *Workshop on the Theory and Application of Cryptographic Techniques*, pages 203–230. Springer.

Fischer V. (2012). A closer look at security in random number generators design. *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 167–182. Springer.

Fischer V., Drutarovský M. (2002). True random number generator embedded in reconfigurable hardware. *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 415–430. Springer.

Golic J. D. J. (2006), New Methods for Digital Generation and Postprocessing of Random Data, *IEEE Transactions on Computers*, vol. 55, no. 10, pp. 1217-1229.

Gyorfi T., Cret O., and Suciu A. (2009). High performance true random number generator based on fpga block rams. *Parallel & Distributed Processing, 2009.* IPDPS 2009. IEEE International Symposium on, pages 1–8. IEEE.

Hammouri, G., Öztürk, E., Sunar, B. (2008). A tamper-proof and lightweight authentication scheme. *Journal Pervasive and Mobile Computing* 6(4).

Handschuh H., Schrijen G.-J., and Tuyls P. (2010). Hardware intrinsic security from physically unclonable functions. *Towards Hardware-Intrinsic Security*, pages 39–53. Springer.

Haroon, A., Akram, S., Shah, M. A., & Wahid, A. (2017). E-lithe: A lightweight secure dtls for iot. In 2017 *IEEE 86th Vehicular Technology Conference (VTC-Fall)* (pp. 1-5). IEEE.

Katzenbeisser, S., Kocabaş, Ü., Van Der Leest, V., Sadeghi, A. R., Schrijen, G. J., & Wachsmann, C. (2011). Recyclable PUFs: Logically reconfigurable PUFs. *Journal of Cryptographic Engineering*, 1(3), pp. 177–186.

Kirkpatrick M. S., Bertino E., and Kerr S. (2010). PUF ROKs: generating read-once keys from physically unclonable functions. *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*. ACM.

Kocabaş, Ü., Peter, A., Katzenbeisser, S., Sadeghi, A.-R. (2012). Converse PUF-Based Authentication. In: Katzenbeisser, S., Weippl, E., Camp, L.J., Volkamer, M., Reiter, M., Zhang, X. (eds.) *Trust 2012. LNCS*, vol. 7344, pp. 142–158. Springer, Heidelberg.

Kodýtek, F., Lórencz, R. (2015). A design of ring oscillator based PUF on FPGA. In *Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2015 IEEE 18th International Symposium on. IEEE.

Kodýtek, F., Lórencz, R., Buček J. (2016) Improved ring oscillator PUF on FPGA and its properties. *Microprocessors and Microsystems*.

Kohlbrenner P., Gaj K. (2004). An embedded true random number generator for FPGAs. *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*. ACM.

McKay K. A. (2017). Report on Lightweight Cryptography – NIST publication, available online https://doi.org/10.6028/NIST.IR.8114

Maes R., Van Herrewege A., and Verbauwhede I. (2012). PUFKY: a fully functional PUF-based cryptographic key generator. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 302–319. Springer.

Majzoobi, M., Rostami, M., Koushanfar, F., Wallach, D.S., Devadas, S. (2012). Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching. In: *IEEE Symposium on Security and Privacy* (SP), pp. 33–44.

Malina L., Hajny J., Fujdiak R., and Hosek J. (2016). On perspective of security and privacy-preserving solutions in the internet of things. *Computer Networks*, 102:83– 95.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of applied cryptography. CRC press.

Öztürk, E., Hammouri, G., Sunar, B. (2008). Towards Robust Low-Cost Authentication for Pervasive Devices. In: *IEEE Conference on Pervasive Computing and Communications*, PerCom.

Pappu, R., Recht, B., Taylor, J., and Gershenfeld, N. (2002). Physical one-way functions. *Science*, 297(5589):2026–2030.

Raza, S., Shafagh, H., Hewage, K., Hummen, R., & Voigt, T. (2013). Lithe: Lightweight secure CoAP for the internet of things. *IEEE Sensors Journal*, 13(10), 3711-3720.

Roman R., Zhou J., and Lopez J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279.

RSA Laboratories: PKCS #5 V2.1: Password Based Cryptography Standard (2012)

Schindler W. (2009). Random number generators for cryptographic applications. In *Cryptographic Engineering*, pages 5–23. Springer.

Schleiffer Ch., Wolf M., Weimerskirch A., and Wolleschensky L. (2013). Secure key management-a key feature for modern vehicle electronics. Technical report, SAE Technical Paper.

Sicari S., Rizzardi A., Grieco L. A., Coen-Porisini A. (2015). Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164.

Sklavos, N, Zaharakis I. D. (2016). Cryptography and Security in Internet of Things (IoTs): Models, Schemes, and Implementations, In *IEEE proceedings of the 8th IFIP International Conference on New Technologies, Mobility and Security* (NTMS'16), Larnaca, Cyprus

Suh E. G., Devadas S. (2007). Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, pages 9–14. ACM.

Suh E. G., O'Donnell Ch. W., and Devadas S. (2007). Aegis: A single-chip secure processor. *IEEE Design & Test of Computers* 24.6.

Tkacik T. E. (2003), A hardware random number generator, in Proceedings of the Cryptographic Hardware and Embedded Systems (CHES '02), B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, Eds., vol. 2523 of *Lecture Notes in Computer Science*, pp. 450–453, Springer, Redwood Shores, Calif, USA.

Tschofenig, H. and T. Fossati," Transport Layer Security (TLS)/Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things. RFC 7925, July 2016.

Van Herrewege, A., Katzenbeisser, S., Maes, R., Peeters, R., Sadeghi, A.-R., Verbauwhede, I., Wachsmann, C. (2012). Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-Enabled RFIDs. In: Keromytis, A.D. (ed.) *FC 2012. LNCS*, vol. 7397, pp. 374–389. Springer, Heidelberg.