3

SRAM Physically Unclonable Functions for Smart Home IoT Telehealth Environments

Fayez GEBALI[1] and Mohammad MAMUN[2]

[1]*Department of Electrical and Computer Engineering, University of Victoria, Canada*
[2]*National Research Council of Canada, Government of Canada, Canada*

One main application of smart home IoT networks is telehealth, which is timely given the current pandemic situation and increasing healthcare costs. To ensure security of IoT smart homes, it has been suggested that silicon-based physically unclonable functions (PUF) be incorporated in the IoT devices themselves. PUFs are used as the main technique for establishing device authentication and secure key exchange as well as any higher level security protocols. This chapter provides an analysis of the characteristics and performance of SRAM physically unclonable functions. The analysis takes into account several factors such as the static or slowly-varying random process variations as well as the dynamic CMOS noise sources. The main parameters affecting the performance are identified and techniques used to measure them at the fabricator and in the field are explained. Three algorithms are proposed for choosing the set of challenges and the corresponding responses. The three algorithms are: Algorithm #1: single challenge; Algorithm #2: repeated challenge; and Algorithm #3: repeated challenge with bit selection. The last algorithm manages to eliminate the bit errors in the response and hence will not require the use of error correction coding often used in secure sketch or fuzzy extractor methods that have previously been proposed. The use of physically unclonable functions, coupled

Cybersecurity in Smart Homes,
coordinated by Rida KHATOUN. © ISTE Ltd 2022.

Cybersecurity in Smart Homes: Architectures, Solutions and Technologies,
First Edition. Rida Khatoun.

with the proposed algorithms, provide a layer of protection against the common IoT attacks and the novel deep learning attacks that EW claimed to be a serious security threat to IoT devices in telehealth applications.

3.1. Introduction

An emerging application of smart homes is telehealth, where healthcare delivery is extended to serve stay-at-home patients and remote or isolated communities. Telehealth is motivated by the escalating healthcare costs and the fact that many patients can not afford long-term hospital stays and prefer staying in their homes or within their remote communities. Telehealth relies very heavily on equipping the home with smart IoT devices that can sense the patient's vitals and can also deliver medication in a secure environment that is immune to cyber attacks. This approach allows us to reach out to many disadvantaged communities, thereby democratizing healthcare, as well as leading to reduced costs and speedy patient recovery times (Ellenbecker *et al*. 2008; National Institute on Aging 2020).

IoT devices used in smart homes are considered the weakest link in the security protocols implemented. As a result, contemplating the implementation of critical telehealth services in a smart home is very risky due to the device limitations of the Internet of Things (IoT). Some of the limitations include:

1) Limited resources, such as computer processing capabilities, which often prevent the implementation of secure key exchange algorithms that use complex elliptic functions.

2) Storing secret keys in non-volatile memory (NVRAM) is considered a security gap since simple memory attacks can reveal those secret keys. Furthermore, these secret keys are are hard to update since the NVRAM must be reprogrammed.

3) Users often do not customize or update each IoT device password or operating system firmware and rely solely on factory-set defaults. This is what system attackers first look for to launch their attacks.

4) IoT devices are located in unsecured premises and can be subject to theft, counterfeiting and reverse engineering.

These limitations impact the effectiveness of security protocols for both authentication and secure key exchange. A very promising technique for endowing a simple IoT device with a unique identity and the ability to secure secret keys without using NVRAM is to use silicon-based physically unclonable functions (PUF).

There are many types of PUFs based on different physical phenomena such as optical, acoustical and electrical. However only silicon PUFs implemented as electric circuits are practical for inexpensive implementations on simple IoT devices. Silicon PUFs are practical means of adding unique, unclonable identities to IoT devices. This

is equivalent to biometrics in humans, such as iris, retina, voice, facial or fingerprint. PUFs not only help to authenticate IoT devices, but also aid in storing secret keys in the way a PUF is constructed. Traditionally, secret keys are stored in IoT devices using NVRAM. The disadvantage of NVRAM is the ability of an attacker to extract the secret keys, using many techniques such as memory persistence, reverse engineering, etc. A very attractive property of PUFs is their tamper-resistance which provides immunity from reverse engineering attacks that aim to extract the unique device response. The unique response of the PUF prevents the manufacturer, the user and the attacker from duplicating the PUF function, even when the PUF hardware design and structure are known.

Authentication using PUFs is based on establishing a challenge-response pair (CRP) where a set of challenges and their associated unique response is established by the device manufacturer. This dataset is then shared with a trusted certification authority (CA) for later use by administrators of the telehealth system to construct a secure and trusted system.

There are several criteria for CRP establishment:

1) Several CRP must be established so that each CRP is used only once to prevent attackers from forging a valid response by observing past CRP activities.

2) The number of bits for each response must be "large enough" to be able to establish enough Hamming distance (HD) separation between valid devices and counterfeit ones.

3) Techniques must be established to remove the inevitable dynamic noise from the PUF response to be able to match the noisy response to the one provided by the manufacturer and stored at a CA.

4) Algorithms must be provided to extract a high-entropy stable and repeatable secret key from a noisy low-entropy response.

The ability to construct inexpensive PUFs for IoT edge devices allows us to impart a unique device identity (ID), which is used for device authentication and developing stable and secure session keys. A very significant advantage is that the session key is obtained at the beginning of each session without the use of NVRAM. The key will be shared between the device and the authenticator through the use of publicly available helper data that will not compromise either the key or the device response.

Ensuring security of telehealth systems is hard, since many devices are distributed in insecure locations. Many types of attacks become feasible, such as eavesdropping, theft, tampering, man-in-the-middle, denial of service, etc. Central to ensuring security is authentication and key exchange. Cryptographic protocols are based on primitive operations such as block ciphers, stream ciphers and cryptographic hash functions. These primitive operations rely on storing a secret key stored in non-volatile memory, which proves to be their Achilles heel, especially for unsecured IoT devices (Delvaux 2017b).

The use of PUFs for mutual authentication in IoT devices has been the recognized solution to endowing IoT devices with a unique identity, akin to a fingerprint or retina image for human users. A PUF serves to authenticate a device and also provides a measure of tamper resistance (Gassend *et al*. 2002; Ravikanth *et al*. 2002; Guajardo *et al*. 2007; Suh and Devadas 2007; Maes *et al*. 2009, 2012; Maes 2013; Herder *et al*. 2014; Delvaux 2017b). Operation of the PUF relies on a challenge-response pair (CRP), where the server issues a challenge and the IoT device, or client, provides a response that is unique to the device. The problem with PUF response is it is noisy but has low entropy. Therefore, techniques have been developed to recover reliable and stable response from the noisy response using fuzzy extractors or secure sketch (Linnartz and Tuyls 2003; Boyen 2004; Dodis *et al*. 2004, 2008). The advantage of the fuzzy extractor is that it also serves to generate a secret key with high entropy from the low-entropy noisy response.

Contributions: The contributions of this chapter can be summarized as follows:

1) Novel statistical modeling and analysis of SRAM PUFs is presented. The model includes the effects of static random process variations and dynamic CMOS noise.

2) The main physical, device and system parameters affecting the PUF response are identified and techniques to estimate them are presented for both the IoT device manufacturer and for the IoT device user in the field.

3) A novel NOR-based SRAM PUF cell design is proposed that enables rapid device resetting at a speed matching the operating speed of the system and does not require the waste of too much delay or energy resetting the entire SRAM.

4) Three algorithms are proposed for generating the challenge response pairs. The techniques illustrate the impact of system parameters in uniquely identifying valid devices from counterfeit ones.

5) A discussion is provided on how to harden SRAM PUF against typical IoT attacks and deep learning attacks in particular.

Organization: The rest of this chapter is structured as follows. In section 3.2 we discuss the literature related to the use of PUFs for authentication and secure key exchange. In section 3.3 we review the architecture of a telehealth system where the smart home is the target for the healthcare delivery. In section 3.4 we discuss physically unclonable functions and using secure sketch and fuzzy extractors to remove the dynamic noise from the PUF response. In section 3.5 we discuss the use of convolutional coding as a means of generating the helper data without revealing the IoT device response when a challenge is issued. In section 3.6 the structure of SRAM PUFs is presented and a novel NOR-based SRAM is discussed. A statistical model of the SRAM PUF is also developed. In section 3.7 we propose three algorithms for issuing the PUF challenge-response pair (CRP) data and their effect on system design. In section 3.8 we discuss the attacks targeting smart homes, especially deep learning attacks.

3.2. Related literature

A high-level authentication and key exchange protocol for a smart home IoT system was recently proposed by Fakroon *et al*. (2020, 2021). The protocol used a two-factor authentication scheme that preserved user anonymity and untraceability. In the 2020 publication, the IoT edge devices were assumed to have secret keys stored in NVRAM. On the other hand, the 2021 publication assumed the secret keys could be derived from a built-in PUF that gave the IoT edge devices unique IDs. Security analysis of the scheme was conducted through formal analysis using the Burrows-Abadi-Needham logic (BAN), informal analysis and model check using the automated validation of Internet security protocols and applications (AVISPA) tool. A review of PUF-based security techniques can be found in Dodis *et al*. (2004, 2008). The authors discussed how to use a low-entropy PUF response to generate secure keys with high entropy. Secret key extraction techniques used fuzzy extractors to obtain session keys from the noisy PUF responses.

PUFs are classified as strong PUFs and weak PUFs as explained by Delvaux *et al*. (2014) and Delvaux (2017b). The discussion discussed the impact of strong and weak PUFs on device authentication and secure key exchange. A discussion was also provided about the helper data algorithm and how it can be used to obtain high-entropy stable keys from noisy, low-entropy PUF responses. At a different level, the abstraction of PUF operation as a one-way functions can be found in Ravikanth (2001) and Ravikanth *et al*. (2002). They compared algorithmic one-way functions (e.g. RSA encryption) with physical one-way functions (e.g. PUFs). A discussion of silicon PUFs can be found in Gassend *et al*. (2002). The discussion focused mainly on delay-based PUFs such as arbiter PUF and ring oscillator PUF. The authors also discussed helper data, which is used to generate session keys from PUF responses.

An initial attempt at analyzing delay-based PUFs can be found in Suh and Devadas (Suh and Devadas 2007). The analysis considered the need to use each CRP only once. Techniques were proposed to generate a sufficiently large number of responses through increasing the number of options to configure the circuit delays. The authors also discussed low-cost authentication techniques that do not require the use of the more expensive cryptographic primitives.

It is interesting to explore how PUFs can be incorporated using the popular FPGA technology. This is especially true for SRAM PUFs, which could make use of the built-in block RAM provided in many FPGA modules. However, resetting the SRAM might not be a simple matter, since this was not part of the design requirements of an FPGA block RAM (Xilinx 2021). Guajardo *et al*. (2007) studied SRAM PUF structures implemented in FPGA technology. To overcome noise associated with the

response of the PUF a fuzzy extractor was used. This also helped extract secure and stable session keys with high entropy.

In a series of publications, Maes *et al*. (2012, 2009) and Maes (2013) discussed seven silicon-based as well as non-silicon-based PUFs. The authors also discussed the secure sketch techniques used to generate session keys that were proposed by Dodis *et al*. (2004, 2008).

Reviewing the published literature, we can make several conclusions about the current state of the art in using PUFs for IoT authentication and secure exchange:

1) The literature discusses, sometimes implicitly, one algorithm for issuing the CRP pairs: a single challenge is issued and the device response is observed. This is a simple algorithm that does not utilize the IoT device statistical characteristics to its advantage. Perhaps the only advantage of this algorithm is that it maximizes the number of CRP pairs, which is critical, especially for weak PUFs.

2) The parameters that define the response of the IoT PUF device are not identified in most published works. General statements are typically stated such as: "a large number of response bits are needed to differentiate valid from counterfeit devices". At best a sketch is provided about the desired Hamming distance (HD) separation between valid devices and counterfeit devices.

3) Values of the PUF circuit parameters, the statistical parameters and the choice of overall system parameters are not studied to see how the response of the PUF can be controlled and optimized. A lack of accurate logical PUF models explains why this is the accepted view of using PUFs.

3.3. System design considerations

Telehealth Network Model: Figure 3.3 shows the architecture of the telehealth system. The main agents in the system include:

– **Network Server (*S*):** The network server is usually located in a hospital. We can consider the server to be a root-of-trust (RoT) since it contains tamper-resistant hardware like a trusted platform module (TPM).

– **Mobile User (*M*):** This can be thought of as the smart devices or telephones used by the healthcare professionals such as doctors and nurses.

– **IoT Edge Device (*D*):** The IoT edge devices include Internet-enabled sensors/actuators that could be located in a remote health care unit or could be located in a body area network (BAN) attached to a stay-at-home patient.

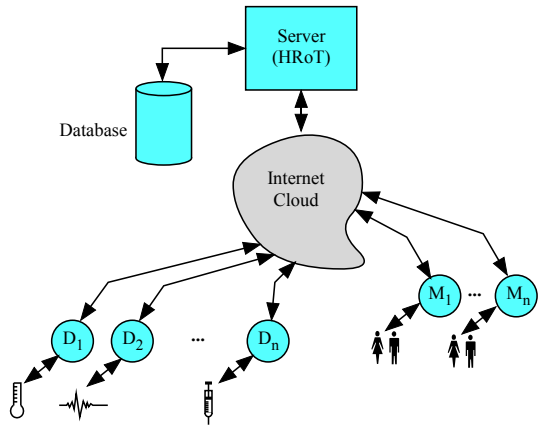

Figure 3.1. *Telehealth network model. For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip*

3.4. Silicon physically unclonable functions (PUF)

Silicon static random access memory (SRAM) used to construct a PUF is a practical technique to give a unique "fingerprint" or identity to a silicon device and the ability to generate a secret key without the need to store it in NVRAM. The main advantages of silicon SRAM PUF are several:

1) Silicon SRAM based on CMOS technology does not require any extra processing steps which makes them practical to implement at no additional costs or delays (Holcomb *et al*. 2009).

2) The area cost is less than that required by an identity stored in NVRAM since circuits often require extra hardware such as charge pump to program the NVRAM.

3) The identity can not be cloned or reverse-engineered without destroying the fingerprint itself and removing the possibility of any device recyling.

4) The number of CRP goes beyond the number of words of the memory. In fact, the number of challenge-response pairs (CRP) is given by equation [3.6] or equation [3.7] later in this chapter.

In addition, a PUF provides tamper resistance since any changes to the device physical parameters will lead to a corrupted identity (Maes *et al*. 2009). The concept of silicon PUF was first proposed by Gassend *et al*. (2002). Silicon PUF operation relies on the inevitable random variations that are introduced during the fabrication of semiconductor devices. This gives the means to uniquely identify the individual

devices. Furthermore, such a PUF can not be replicated through reverse engineering even by the device manufacturer. SRAM cells provide an compact way to create a silicon PUF through the unique startup values of the individual words in the memory (Guajardo *et al*. 2007; Boehm and Hofer 2009; Schrijen 2020). The SRAM content each time the SRAM PUF starts up is slightly different due to the inevitable dynamic noise (Su *et al*. 2008; Yu *et al*. 2011). Dodis was the first to propose using forward error correcting codes (FEC) to overcome the noisy inconsistent SRAM PUF output (Dodis *et al*. 2004, 2008). This was later improved upon by other authors (Boyen 2004; Bösch *et al.* 2008; Maes *et al.* 2009, 2012; van der Leest *et al.* 2012; Maes 2013; Delvaux *et al.* 2014; Hiller 2016; Delvaux 2017a, 2017b; Gao *et al.* 2018, 2019; Schrijen 2020).

3.4.1. *Mutual authentication and key exchange using PUF*

Figure 3.2 shows the basic structure of the secure sketch at the server and client. The server selects a challenge c and uses the database supplied by the manufacturer to extract the expected response r. The server also performs forward error correction coding (FEC) on the response to produce helper data w. The secure sketch also produces a hashed value h for the response. This value will serve to establish mutual authentication between the server (gateway provided by Internet service provider, in our case) and the client (IoT edge device, in our case).

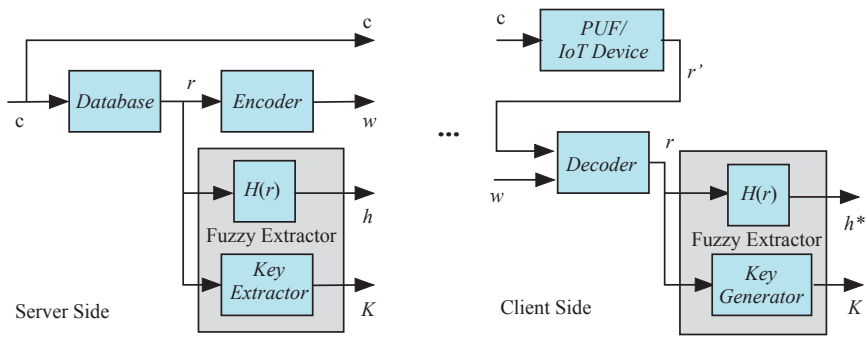

Figure 3.2. *Basic structure of the secure sketch at the server (on the left) and client (on the right). For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip*

The client receives the challenge c and helper data w and in response produces the actual noisy response r', and with the help of w it decodes r' to produce the error-free response r. The client then hashes this value and sends h^* to the server to be authenticated.

3.4.2. *Fuzzy extractor*

At the server side, the fuzzy extractor uses the expected response w to generate the secret key K and helper data w as shown on the left in Figure 3.3. The helper data r can be made public without divulging the secret key. On the right side of Figure 3.3, the IoT device with the PUF is the client which, upon receiving the challenge c and helper data P, generates the noisy response r'. As long as the Hamming distance between r and r' is less than a certain threshold, the fuzzy extractor uses the corrected response r and helper data w to generate the secret key K.

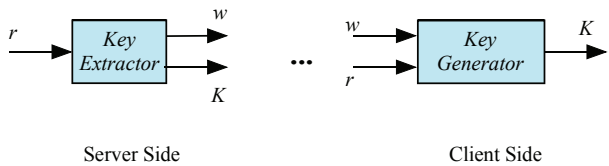

Figure 3.3. *Basic structure of the fuzzy extractor at the server and client. For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip*

It should be noted that the secret key changes each time a new challenge c is issued. In this chapter we will use this feature to generate a nonce which could be K or a hashed value of K to increase its entropy. This will serve to construct a secret key shared among the entities of our system: mobile device (M), server (S), and IoT edge device (D).

The key regeneration using the fuzzy extractor process can be expressed by the equation

$$(K_d,\ N_d) = \text{key\_regen}(c,\ w) \qquad [3.1]$$

where K_d is the secret key and N_d is the secret random number.

Some implementations were done in FPGA platforms (Herrewege *et al*. 2012; Maes *et al*. 2012) and some were implemented on microcontrollers (Aysu *et al*. 2015). Gao *et al*. (2019) proposed an SRAM-based PUF key generator on a microcontroller using RF energy harvesting.

3.5. Convolutional encoding and Viterbi decoding the SRAM words

As explained in section 3.4.1, the PUF response is inherently noisy due to the CMOS dynamic noise. Means have to be provided for removing this noise from the response. This is the job of the secure sketch which is derived from forward error correcting coding (FEC) theory. The helper data w in Figure 3.2 is used to remove the

dynamic noise. However, the system designer must ensure that w does not reveal any information about the device response since w will be sent across unsecured channels. Furthermore, the error correcting capability of the secure sketch must be limited to a certain number of bit errors. If it exceeds that limit, there is a danger of inadvertently converting the response from a counterfeit device to that of a valid device.

Convolutional codes are a powerful FEC technique that is the only FEC that can handle both random errors and bust errors. The error correcting capability can be increased or decreased by increasing or decreasing the code rate, respectively.

A rate k/n convolutional encoder accepts k message bits and adds redundant bits to produce n output bits for each message with $n > k$. A convolutional encoder is specified by the three-tuple (n, k, m) where:

1) n: number of bits of the message after encoding;

2) k: number of information bits of the message before encoding;

3) m: order of the code or number of storage registers.

The code rate is defined by the first two parameters k/n. We can write the convolutional encoder as

$$\mathbf{y}_i[n] = \sum_{j=0}^{k-1} \mathbf{h}_i[j]\, \mathbf{x}[n-j] \qquad [3.2]$$

where $\mathbf{x}[n]$ is the k-bit input symbol, $\mathbf{y}_i[n]$ is the ith k-bit output symbol with $0 \leq i < n$, and $\mathbf{h}_i[j]$ is the k-bit generator polynomial weight with $0 \leq j < m$ the number of delay elements. $\mathbf{x}[n]$ represents the input symbol stream and $\mathbf{y}[n]$ represents the output symbol stream.

Figure 3.4 shows a 1/2 convolutional encoder. The disadvantage of the structure in Figure 3.4 is the delay incurred to add m inputs using XOR gates. The constraint length for the k/n encoder with the structure in Figure 3.4 is km, which indicates the number of delay elements needed to generate the outputs from k inputs.

Figure 3.5 shows an alternative form for a 1/2 convolutional encoder. This form has the advantage of pipelining the partial output results at the expense of doubling the number of delay elements. The constraint length for the k/n encoder with the structure in Figure 3.5 is knm, which indicates the number of delay elements needed to generate the outputs from k inputs.

The generator polynomial for the structure in Figure 3.4 or 3.5 is defined as

$$p_1(x) = h_1[0]x^3 + h_1[1]x^2 + h_1[2]x^1 + h_1[3] \qquad [3.3]$$

$$p_2(x) = h_2[0]x^3 + h_2[1]x^2 + h_2[2]x^1 + h_2[3] \qquad [3.4]$$

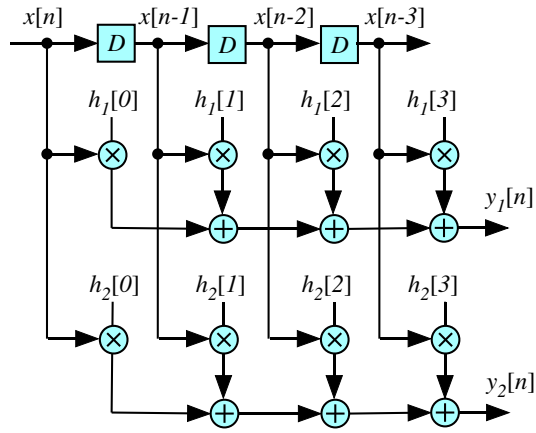

Figure 3.4. *Structure of a (2, 1, 3) convolutional encoder. For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip*

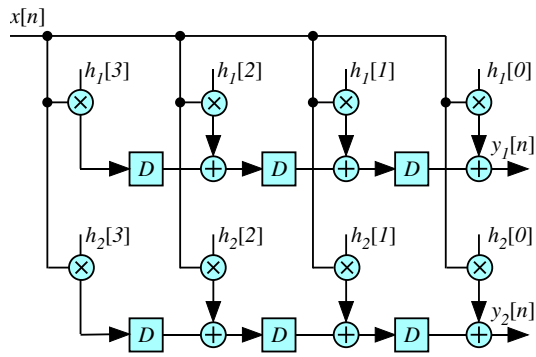

Figure 3.5. *Structure of an alternative form for a (2, 1, 3) convolutional encoder. For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip*

Typically the generator polynomial is represented in matrix form as an $n \times (m+1)$ matrix $\mathbf{G}$. For the (2, 1, 3) encoder we can write

$$\mathbf{G} = \begin{bmatrix} h_1[0] & h_1[1] & h_1[2] & h_1[3] \\ h_2[0] & h_2[1] & h_2[2] & h_2[3] \end{bmatrix} \quad [3.5]$$

For a degree 3 polynomial in GF(2) we can use several primitive polynomials such as

$$p(x) = \begin{cases} x^3 + x^2 + x + 1 \\ x^3 + x + 1 \end{cases}$$

The golden SRAM words are defined according to the rules in [3.18]. These criteria will determine which bits of a given SRAM word are to be used for encoding/decoding and which ones will be overpassed according to the algorithms discussed in section 3.7.

The manufacturer applies convolutional coding to the golden SRAM data word before transmission. The ICs in the field use Viterbi decoding on the actual PUF output to generate the corrected SRAM word. The decoder uses hard decision algorithm, where each bit is interpreted as either '0' or '1'.

3.6. CMOS SRAM PUF construction

The basic structure of an SRAM CMOS cell is shown in Figure 3.6, which is basically two cross-coupled CMOS inverters. An excellent discussion of the operation of the CMOS SRAM memory is found in (Prince 1991, section 5.5). As we shall see in section 3.6.2, part of the requirement for an SRAM PUF is to perform repeated resetting of the SRAM. An SRAM PUF might have to be reset over 1,000 or more times to obtain a dependable response free of dynamic noise. This is a basic feature of the proposed algorithm discussed in sections 3.7.2 and 3.7.3 later in this chapter. The basic SRAM can be reset in one of two ways

1) Disconnect then reconnect the power supply V_{DD}. This will force the initial state of the two outputs of the cell to be 0 simultaneously. However, this is a slow process since the power supply rails usually have very large parasitic capacitances.

2) Ground the bit lines $B = \overline{B} = 0$ and set the word line $W = 1$. This will ensure the initial state of the two outputs of the cell to be 0 simultaneously. This option requires modifying the word lines W and $\overline{W}$ for the entire SRAM module. This approach is not feasible if the SRAM block is used to store data in addition to the PUF function.

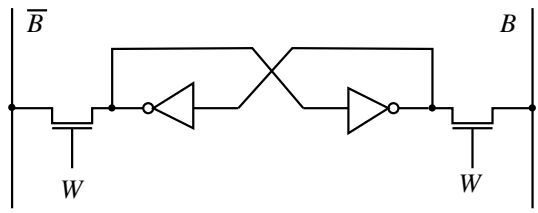

Figure 3.6. *Basic cell structure for a 6-transistor SRAM CMOS cell*

A third alternative is to modify the cell structure so that resetting the cell can be done at a speed matching the write speed of an SRAM. Figure 3.7 shows the basic cell

structure of a NOR gate-based SRAM PUF and Figure 3.8 shows the details of the cell structure. In Figures 3.7 and 3.8, the contents of the cell are obtained through the bit lines B and $\overline{B}$ for the bit value and its complement, respectively. Signal W is usually referred to as the word line and, when asserted, connects the outputs of the cell to the bit lines. Finally, signal R is the reset signal and when it is asserted to '1', both NOR gates' outputs will be 0. As soon as $R = 0$, the storage cell stores '1' or '0' depending on several factors such as:

1) Threshold voltage values for the n-MOS and p-MOS transistors of the NOR gates and the pass-gate controlled by signal W.

2) Delay between the signal R and the lower inputs to the NOR gates.

3) Parasitic capacitances seen by the outputs of the two NOR gates.

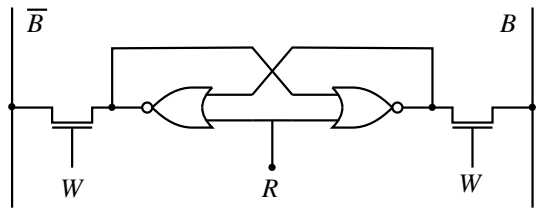

Figure 3.7. *Basic cell structure for NOR gate-based SRAM PUF*

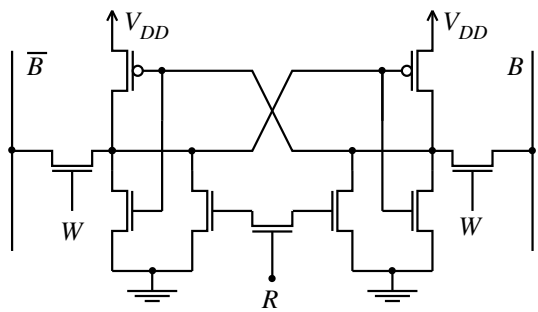

Figure 3.8. *Detail of the basic cell structure for NOR gate-based SRAM PUF*

The cell structure in Figure 3.8 was first simulated using the analog device simulator QUCS (Jahn and Borrás 2007). The simulator confirmed the basic operation of the SRAM cell under normal operation when $R = 0$ and $W = 1$. When the cell was reset ($R = 1$), both outputs B and $\overline{B}$ both reached the same reset value due to symmetry conditions. When the reset was not asserted $R = 0$, the SRAM cell assumed a random value 1 or 0.

It should be mentioned that the cell design can use two NOR gates or two inverters. The inverter-based design, also known as the 6-transistor design, must add enough pass-gates to allow for breaking up the feedback path and setting the inputs of the two inverters to equal values, whether 0 or 1. There is therefore no saving in terms of the MOS transistor count to using the 6-transistor cell design.

Assuming the number of words in the SRAM PUF to be N and that a challenge selects addresses of k words, the number of challenge-response pairs (CRP) is given by the permutation

$$\text{CRP} = N^k \gg N \qquad [3.6]$$

when repetitions are allowed. Alternatively we have

$$\text{CRP} = \frac{N!}{(N-k)!} \gg N \qquad [3.7]$$

when repetitions are not allowed. Adopting this strategy, one can construct strong PUF out of NOR-based SRAM PUF, especially if the order of the response bits is pre-arranged and can be securely varied at the start of each session.

3.6.1. *SRAM PUF statistical model*

The operation of SRAM PUF relies to two random physical phenomena: random processing variations and dynamic noise, which are analog processes. Both these phenomena control the digital binary value of the stored bits after SRAM initialization. Random process variation is static for a given device and facilitates creation of the device "biometric" or unique fignerprint. Random dynamic noise, on the other hand, is dynamic and introduces noise to the device identity (ID).

One way to analyze an SRAM-based PUF is to accurately model the devices and wire delays of the basic cell. However, this will not account for all the factors, such as doping variations, oxide thickness variations, random parasitic loading capacitances, etc. Instead we resort here to developing a logical model that encompasses all these physical phenomena. This approach is akin to the logical modeling of faults instead of modeling all possible physical faults in an integrated circuit.

The random variable we choose to model should be amenable to measurements under mass production settings by the device manufacturer. In the context of using an SRAM PUF, an appropriate random variable is the content of the SRAM memory cells. This is a binary random variable that is characterized by the two probabilities a and b denoting the probability that the SRAM cell is '1' or '0', respectively. Ideally random process variations and CMOS noise are absent and the structure of each

SRAM cell is completely symmetric making the ideal probabilities a_i and b_i satisfy the equality

$$a_i = b_i = 0.5$$

Due to the central limit theorem, the random process variation (RPV) effect on the pair $(a_i,\ b_i)$ follows the biased Gaussian distribution whose pdf is given by

$$f_{A_p} = \frac{1}{\sigma_p\sqrt{2\pi}} e^{-(a_p - a_i)^2/2\sigma_p^2} \quad [3.8]$$

where a_p is the adjusted value of a_i due to RPV and σ_p^2 is the variance of the RPV process. We should note that a_i and σ_p are identical for all SRAM bits within a device or among different devices.

The value of a_p is given by

$$a_p = \mathrm{G}(a_i, \sigma_p) \quad [3.9]$$

where $G(a_i, \sigma_p)$ is a Gaussian random process with mean a_i and variance σ_p^2. Figure 3.9 shows the different types of distributions due to the random processes involved in determining the bit value probabilities.

Figure 3.9(a) shows the pdf of the random variable a_p due to RPV which is a biased Gaussian process with mean a_i and variance σ_p^2.

There are several sources of dynamic or short-term noise in CMOS devices including:

1) Thermal noise as additive white Gaussian noise (AWGN), showing flat spectral distribution.

2) Shot noise due to charge carrier flow across p-n junctions.

3) Flicker noise due to charge trapping in the device, showing $1/f$ spectral distribution.

These noise sources introduce variations in the value of transition probability a each time the CMOS inverter undergoes a transition.

Figure 3.9(b) shows the pdf of the random dynamic noise n which is given by

$$f_{A_n}(a_n) = \frac{1}{\sigma_n\sqrt{2\pi}} e^{-a_n^2/2\sigma_n^2} \quad [3.10]$$

where σ_n^2 is the variance of the dynamic noise process. On the other hand, the pdf for the additive white Gaussian noise (AWGN) is common to all bits within a device and also for all devices.

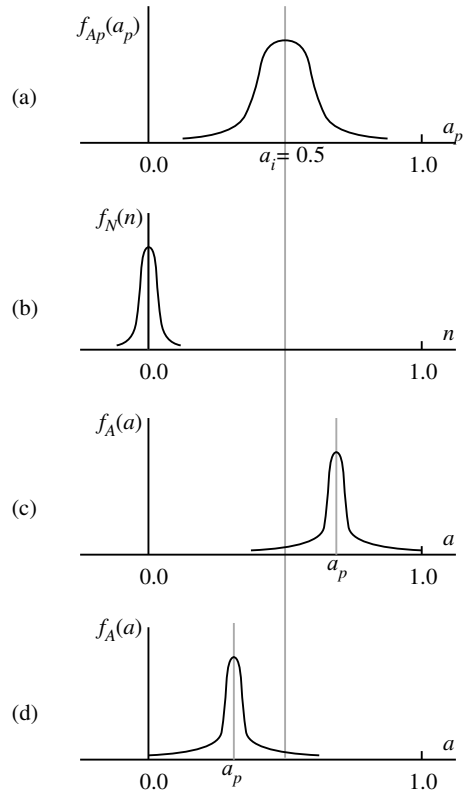

Figure 3.9. *The pdf distributions of transition probability a due to the different physical phenomena. (a) pdf of a_p due to random process variations (RPV). (b) pdf of n due to random dynamic noise. (c) pdf of a due to the combined effects of RPV and random dynamic noise when $a_p > a_i$. (d) pdf of a due to the combined effects of RPV and random dynamic noise when $a_p < a_i$*

The combined effects of RPV and dynamic CMOS noise generate a pdf given by

$$f_A(a) = \frac{1}{\sigma_n\sqrt{2\pi}} e^{-(a-a_p)^2/2\sigma_n^2} \qquad [3.11]$$

where a_p is the contribution of RPV and σ_n is the contribution of random dynamic noise.

Figure 3.9(c) shows the pdf of the transition probability a when both RPF and dynamic noise are present and the mean value $\mu_p > \mu_i$. Figure 3.9(d) shows the pdf

of the transition probability a when both RPF and dynamic noise are present and the mean value $\mu_p < \mu_i$. For either case, the probability a is given by:

$$a = \mathrm{G}(a_p, \sigma_n) \qquad [3.12]$$

3.6.2. *Extracting the SRAM cell statistical parameters*

The value of the a bit at location b in word w is denoted by $v(w, b)$ with w denoting the SRAM word and b denoting the location b in the word. The range of the indices w and b is given by

$$0 \leq w < W \qquad \text{and} \qquad 0 \leq b < B$$

where W is the total number of words in the SRAM and B is the word size.

The values of W and B are set during the fabrication phase of the device. The values of the probability a_p and variance σ_p^2 can be extracted by the manufacturer during the pre-deployment phase by following these steps:

1) The manufacturer performs N initializations and observes the stored values of $v_k(w, b)$ in the tagged bit at each step k.

2) The probability a_p is obtained as

$$a_p(w, b) = \frac{1}{N} \sum_{k=0}^{N-1} v_k(w, b) \qquad [3.13]$$

3) The variance σ_n^2 due to dynamic noise is obtained as

$$\sigma_n^2(w, b) = \frac{1}{N} \sum_{k=0}^{N-1} \left[v_k(w, b) - a_p(w, b) \right]^2 \qquad [3.14]$$

Alternatively, the overall σ_n^2 can be estimated as

$$\sigma_n^2 = \frac{1}{W\ B\ N} \sum_{w=0}^{W-1} \sum_{b=0}^{B-1} \sum_{k=0}^{N-1} \left[v_k(w, b) - a_p(w, b) \right]^2 \qquad [3.15]$$

In order to measure the RPV parameters a_i and σ_i the manufacturer now studies the contents of all the bits in the SRAM memory.

1) The values $a_p(w, b)$ for all bits in the SRAM memory are obtained previously.

2) The value a_i is obtained as:

$$a_i = \frac{1}{WB} \sum_{w=0}^{W-1} \sum_{b=0}^{B-1} a_p(w, b) \qquad [3.16]$$

3) The value σ_i is obtained as:

$$\sigma_i^2 = \frac{1}{WB} \sum_{w=0}^{W-1} \sum_{b=0}^{B-1} [a_p(w,b) - a_i]^2 \quad [3.17]$$

3.6.3. *Obtaining the golden SRAM PUF memory content*

The manufacturer of the SRAM PUF can run N initialization steps on each device to obtain the values of a_i, σ_i, a_p, and σ_n, as explained in section 3.6.2. The digitization, or analog-to-digital conversion, step gives the golden or reference stored value $v(w,b)$ of each bit in the SRAM PUF where $0 \leq w < W$ is the SRAM row index or word address and $0 \leq b < B$ is the bit index within a word. The assignment of golden or reference memory content is given by the rules:

$$v(w,b) = \begin{cases} 0, & 0 \leq a_p(w,b) \leq a_i \\ 1, & a_i < a_p(w,b) \leq 1 \end{cases} \quad [3.18]$$

The conditions in equation [3.18] indicate the cell is skewed toward 0 or 1, respectively, and the effect of dynamic noise is negligible. Such cells provide the desired randomness that make the PUF hard to clone or reverse engineer (Holcomb *et al*. 2009).

The manufacturer now prepares a dataset for each device's SRAM PUF. The dataset contains the following information:

1) W the number of words in the SRAM PUF.

2) B the number of bits in each word.

3) Golden value $v(w,b)$ associated with each bit in the SRAM PUF based on criteria in equation [3.18].

The user now has the ability to choose the challenge/response pairs to use.

3.6.4. *Bit error rate (BER)*

The bit error rate of an SRAM cell is due to two mutually exclusive events: the bit is measured as '0' when it should be '1' or it is measured as '1' when it should be '0'. We can write the BER as

$$p_e = A + B \quad [3.19]$$

where A is the probability that the SRAM bit is measured '0' when it should be '1' because $a_p > a_i$ and B is the probability that the SRAM bit is measured '1' when it should be '0' because $a_p < a_i$. The two probabilities are

$$A = \int_{a=0}^{a_i} \frac{1}{\sigma_n \sqrt{2\pi}} e^{-(a-a_p)^2/2\sigma_n^2} \, da \quad [3.20]$$

for the case when $a_p > a_i$, and

$$B = 1 - \int_{a=0}^{a_i} \frac{1}{\sigma_n \sqrt{2\pi}} e^{-(a-a_p)^2/2\sigma_n^2}\, da \qquad [3.21]$$

for the case when $a_p < a_i$.

3.6.5. *Signal-to-noise ratio (SNR) for SRAM PUF*

The term "signal" in the context of this work refers to the probability a_p. More specifically, we take the *absolute difference* $|a_p - a_i|$ as the definition of our signal for the following reasons:

1) When $a_p = a_i$ the SRAM cell value has equal probability of being 1 or 0 and this value totally depends on the effects of dynamic noise.

2) When $a_i < a_p \leq 1$ the SRAM cell value is biased to be 1 with little effects from dynamic noise especially when $a_p \to 1$.

3) When $0 \leq \mu_p < a_i$ the SRAM cell value is biased to be 0 with little effects from dynamic noise especially when $a_p \to 0$.

We can now define the system-level signal-to-noise ratio (SNR) of a tagged SRAM cell as the ratio of the energy due random process variations relative to dynamic noise energy:

$$SNR = 10 \log \left(\frac{(a_p - a_i)^2 + \sigma_p^2}{\sigma_n^2} \right) \qquad [3.22]$$

where the contribution of the random process variations (through a_p and σ_p) and dynamic noise (through σ_n) are beyond the control of the device manufacturer.

Bits in an SRAM word, and for that matter, all bits in the SRAM, do not have the same SNR. The mininum SNR is when $\mu_p = a_i$:

$$\begin{aligned} SNR_{min} &= 10 \log \left(\frac{\sigma_p^2}{\sigma_n^2} \right) \\ &= 20 \log \left(\frac{\sigma_p}{\sigma_n} \right) \end{aligned} \qquad [3.23]$$

On the other hand, maximum SNR occurs when either $a_p = 0$ or when $a_p = 1$. Since $a_i = 0.5$, we can write:

$$SNR_{max} = 10 \log \left(\frac{a_i^2 + \sigma_p^2}{\sigma_n^2} \right) \qquad [3.24]$$

When $SNR \approx SNR_{min}$, the response to the challenge is noisy. Similarly when $SNR \approx SNR_{max}$, the response to the challenge is more stable and less dependent on noise.

3.7. Algorithms for issuing CRP

In this section we propose and analyze several algorithms for issuing the CRP data and their effect on system design.

3.7.1. *Algorithm #1: single-challenge*

The single-challenge algorithm used to authenticate a device follows the steps depicted in Figure 3.10. Four steps are required for authenticating the device and generating the session key.

1: Server selects a single CRP (c, r)

2: Server generates w, K and h

3: Client uses (c, w) to generate r'_1, K, and h^*

4: Server authenticates device

Server	Channel	Client
# 1. Select CRP (c, r)		
# 2. Generate w, K, h	$\xrightarrow{(c,w)}$	# 3. Use (c, w) to generate r'_1, K, and h^*
# 4. Verify $h^* = h$	$\xleftarrow{h^*}$	

Figure 3.10. *Algorithm #1 for the authentication of an IoT edge device and secure key exchange*

Table 3.1 shows the maximum intra Hamming distance and inter Hamming distance for different word sizes B for the case when $W = 1$K words, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB.

We observe from Table 3.1 that the number of errors in the PUF response increases as B increases, as indicated by the intra Hamming distance. The errors are due to the effects of dynamic noise. We also observe from Table 3.1 that word lengths $B \geq 256$ are required to ensure clear separation between different device IDs.

B (bits)	32	64	128	256	512
Maximum Intra Hamming Distance (bits)	0	0	52	92	171
Inter-Intra Hamming Distance Separation (bits)	-12	-6	-3	10	49

Table 3.1. *The Algorithm #1 maximum intra Hamming distance and inter Hamming distance for the case when $W = 1K$ words, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB*

Algorithm #1 is vulnerable to effects of dynamic noise which leads to a large intra Hamming distance and a small, or even negative, inter Hamming distance. The former leads to developing error correction codes capable of correcting a large number of bits. The latter might lead to false positive that declares or accepts a device as being authentic while it is, in fact, fake.

To be able to mitigate the above effects, the system designer must be able to ensure that the distribution of the intra Hamming distance is sufficiently separated from the inter Hamming distance. This approach is expensive since it requires:

1) Using large SRAM word size.

2) Being able to correct a large number of error bits through using many redundancy bits.

Figure 3.11 shows the histograms for intra and inter Hamming distance distributions for the case when $W = 1$K words, $B = 128$ bits, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB. We notice that when $B = 128$ bits the inter and intra Hamming distance histograms are touching. It would be hard to distinguish between a valid device and a fake one.

Figure 3.12 shows the histograms for intra and inter Hamming distance distributions for the case when $W = 1$K words, $B = 512$ bits, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB. When $B = 256$ the the inter and intra Hamming distance histograms are well separated. It would be easy to distinguish between a valid device and a fake one.

Figure 3.13 shows the histograms for intra and inter Hamming distance distributions for the case when $W = 1$K words and $B = 512$ bits. When $B = 512$ the separation between inter and intra Hamming distances is increased compared to the case when $B = 256$.

It might prove expensive to implement an SRAM PUF with a word size of 512 bits. This problem can be simply solved by changing the challenge c to use multiple words that need not be consecutive. For example, if the SRAM PUF is a memory with word size $B = 64$, then generating a 512-bit response is feasible by simply having the challenge c correspond to addressing 8 words. This actually allows us to enrich the

space of possible challenges by being able to generate all possible permutations so that we have $8! = 40,032$ possible challenges that use the same 8 words of the SRAM.

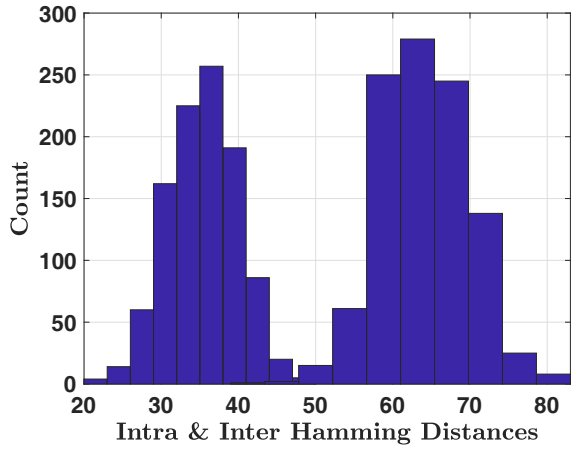

Figure 3.11. *The Algorithm #1 histogram on the left shows the intra Hamming distance distribution. The histogram on the right shows the inter Hamming distance distribution. The case when $W = 1$K words, $B = 128$ bits, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB*

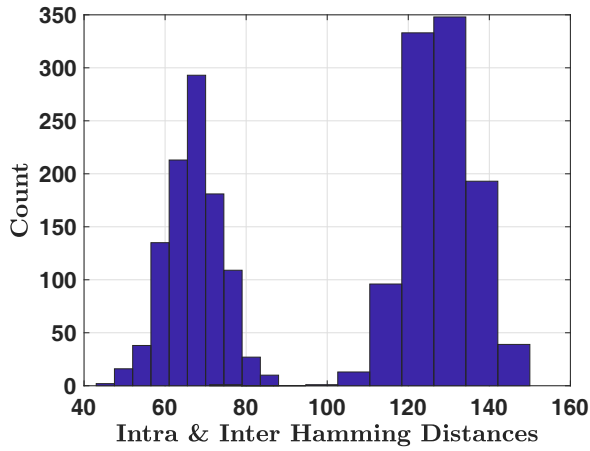

Figure 3.12. *The Algorithm #1 histogram on the left shows the intra Hamming distance distribution. The histogram on the right shows the inter Hamming distance distribution. The case when $W = 1$K words, $B = 256$ bits, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB*

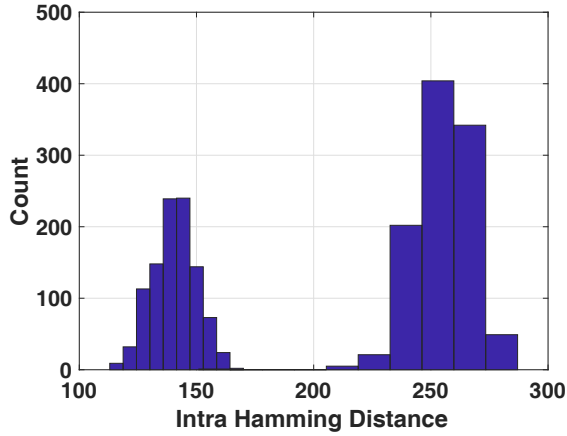

Figure 3.13. *The Algorithm #1 histogram on the left shows the intra Hamming distance distribution. The histogram on the right shows the inter Hamming distance distribution. The case when $W = 1$K words, $B = 0.5$K bits, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB*

3.7.2. *Algorithm #2: repeated challenge*

The basic idea behind Algorithm #2 is to eliminate dynamic noise by repeating the steps used by the manufacturer to obtain the golden reference SRAM as discussed in section 3.6.2.

Algorithm #2 performs N initializations of the SRAM PUF and prepares an $N \times B$ response matrix $\mathbf{R}'$ whose rows are the individual responses $r'[n]$ for the same challenge c. A row vector $\mathbf{x}$ is obtained as the sum of columns of $\mathbf{R}'$.

$$\mathbf{x} = \frac{1}{N} \times \text{SumColumns}(\mathbf{R}') \qquad [3.25]$$

where *SumColumns*$(\mathbf{R}')$ sums the individual B columns of matrix $\mathbf{R}'$ to produce a row B-vector $\mathbf{x}$. The sum operation effectively cancels out the random dynamic noise which effectively performs repetition coding or majority voting.

The response of the device being authenticated is estimated in bitwise fashion. The bit at location b of w_2' is obtained as:

$$v_2'[b] = \begin{cases} 0 \text{ when } 0 \leq \mathbf{x}[b] < a_i \\ 1 \text{ when } a_i \leq \mathbf{x}[b] < 1 \end{cases} \qquad [3.26]$$

Using the helper data w, the error-corrected response r_2 is obtained. The steps used by Algorithm #2 are shown in Figure 3.14. Four steps are required for authenticating the device and generating the session key.

1: Server selects a CRP (c, r_2, N)

2: Server generates w, K and h

3: Client uses (c, w, N) to generate r_2', K, and h^*

4: Server authenticates device

Server	Channel	Client
# 1. Select CRP (c, r_2, N)		
# 2. Generate w, K, h	$\xrightarrow{(c,w,N)}$	# 3. Use c, w to generate $\mathbf{R}'$, r_2', K, and h^*
# 4. Verify $h^* = h$	$\xleftarrow{h^*}$	

Figure 3.14. *Algorithm #2 for the authentication of an IoT edge device and secure key exchange.*

Table 3.2 shows the maximum intra Hamming distance and inter Hamming distance for different word sizes B for the case when $W = 1$K words, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB.

B (bits)	32	64	128	256	512
Maximum Intra Hamming Distance (bits)	0	2	4	7	12
Inter-Intra Hamming Distance Separation (bits)	7	21	42	211	207

Table 3.2. *The Algorithm #2 maximum intra Hamming distance and inter Hamming distance for the case when $W = 1$K words, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB*

We see from the table that the intra Hamming distance is at least an order of magnitude less that the case for Algorithm #1. The inter Hamming distance, of course, remained the same as in Algorithm #1.

From Table 3.2 we make a very interesting discovery which is the ability to reduce the word size B and yet be able to authenticate devices. Table 3.2 shows that we are able to authenticate IoT devices even when $B \approx 32$ bits. This would not be possible with Algorithm #1.

3.7.3. *Algorithm #3: repeated challenge with bit selection*

Algorithm #3 is derived from Algorithm #2. The main idea of this algorithm is to consider or select the response bits that have high SNR in a further attempt to reduce

the effects of dynamic noise. The criterion to select a response bit to be part of the filtered response is given by

$$w_3'[b] = w_2'[b] \qquad \text{when} \qquad \begin{cases} 0 \leq \mathbf{x}[b] < a_i - \Delta \\ \text{or} \\ a_i + \Delta < \mathbf{x}[b] \leq 1 \end{cases} \qquad [3.27]$$

The steps used by Algorithm #3 are shown in Figure 3.15 where A is the vector of bit addresses selected according to equation [3.27]. Four steps are required for authenticating the device and generating the session key.

1: Server selects a CRP (c, r_3, N, A, Δ)

2: Server generates w, K and h

3: Client uses (c, w, N, A, Δ) to generate r_3', K, and h^*

4: Server authenticates device

Server	Channel	Client
# 1. Select CRP (c, r_3, N, A, Δ)		
# 2. Generate w, K, h	$\xrightarrow{(c,w,N,A,\Delta)}$	# 3. Use c, w, N, A, Δ to generate $\mathbf{W}'$, r_3', K, and h^*
# 4. Verify $h^* = h$	$\xleftarrow{h^*}$	

Figure 3.15. *Algorithm #3 for the authentication of an IoT edge device and secure key exchange.*

Table 3.3 shows the maximum intra Hamming distance and inter Hamming distance for the case when $W = 1$K words, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB and $\Delta = 0.3$.

Table 3.3 shows that we are able to authenticate IoT devices even when $B \approx 32$ bits. This would not be possible with Algorithm #1.

B (bits)	32	64	128	256	512
Maximum Intra Hamming Distance (bits)	0	0	4	6	9
Inter-Intra Hamming Distance Separation (bits)	7	21	40	95	213

Table 3.3. *The Algorithm #3 maximum intra Hamming distance and inter Hamming distance for different word sizes B for the case when $W = 1$K words, $N = 1024$ initialization operations, $SNR_{max} = 20$ dB and $\Delta = 0.3$.*

3.8. Security of PUF-based IoT devices

The smart home is the target of several attacks such as (Fakroon *et al*. 2020):

1) replay;
2) eavesdropping;
3) device loss;
4) impersonation;
5) man-in-the-middle;
6) forward-backward secrecy;
7) user credentials;
8) session key guessing;
9) user identification and tracking;
10) side-channel;
11) over-production and counterfeiting;
12) deep learning and machine learning;
13) reverse engineering;
14) nonvolatile memory attacks.

We should note several general principles to ensure the security of the telehealth system, which includes smart home and IoT devices.

– The attacks mentioned above depend on getting the secret key associated with each IoT device through targeting the NVRAM content. Here this is prevented through storing the secret keys within the circuit structure of the PUF.

– Studying the CRP responses is thwarted by hiding the IoT device response r and never sending it between the communicating entities. This provides a level of protection against using deep learning to mimic the PUF function.

– Secret session keys K and hash values h are based on chaining and context such that a previous hash value or current device environment are used to generate a session K and h in addition to the response r (Fakroon *et al*. 2021).

– The use of PUFs in IoT devices constitutes an inexpensive means of providing tamper-proofing to a certain degree. It is not expected that each IoT device would be a root of trust (RoT) but at least it provides immunity to reverse engineering and tampering.

– Security measures must be layered starting from the physical layer (PUFs), then the communication layer and ending with the application layer. Multifactor authentication is also feasible here since each PUF can provide some of these factors.

3.9. Conclusions

This chapter developed novel statistical models for SRAM PUF performance. The main parameters affecting the SRAM PUF performance were identified and techniques to measure them were proposed. These parameters can be estimated by the manufacturer at the pre-deployment phase and can also be measured in the field. This chapter also proposed three algorithms for generating CRP and establishing device authentication and secure key exchange. Algorithm #1 is based on a single challenge. Algorithm #2 is based on repeated challenge. Algorithm #3 is based on repeated challenge with bit selection. We noted that Algorithm #1 can be used when the SRAM word size is $B > 256$ bits. Further, Algorithm #1 introduces a rather large number of bits in error in the response. Two new algorithms are proposed in this chapter: Algorithm #2 and Algorithm #3. These two algorithms solved the two main problems associated with noisy PUF responses: the need to use a large number of bits B and the large number of errors in the response.

3.10. Acknowledgements

This research was supported by a grant from the National Research Council of Canada (NRC) through the Collaborative R&D Initiative.

3.11. 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*, 556–576.

Boehm, C. and Hofer, M. (2009). Using SRAMs as physical unclonable functions. *17th Austrian Workshop on Microelectronics – Austrochip*, 117–122.

Bosch, C., Guajardo, J., Sadeghi, A.-R., Shokrollahi, J., Tuyls, P. (2008). Efficient helper data key extractor on FPGAs. In *Cryptographic Hardware and Embedded Systems (CHES)*, Oswald, E. and Rohatgi, P. (eds). Springer, Heidelberg.

Boyen, X. (2004). Reusable cryptographic fuzzy extractors. *11th ACM Conference on Computer and Communications Security – CCS*, October.

Delvaux, J. (2017a). Machine-learning attacks on PolyPUF, OB-PUF, RPUF, and PUF–FSM. *IACR Cryptology*, November.

Delvaux, J. (2017b). Security analysis of PUF-based key generation and entity authentication. PhD Thesis, University of KU Leuven and Shanghai Jiao Tong University.

Delvaux, J., Gu, D., Schellekens, D., Verbauwhede, I. (2014). Helper data algorithms for PUF-based key generation: Overview and analysis. *IEEE Transactions on Computers*, 34(6), 889–902.

Dodis, Y., Reyzin, L., Smith, A. (2004). Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, Cachin, C. and Camenisch, J. (eds). Springer, Heidelberg.

Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A. (2008). Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1), 97–139.

Ellenbecker, C.H., Samia, L., Cushman, M.J., Alster, K. (2008). Patient safety and quality in home health care. In *Patient Safety and Quality: An Evidence-Based Handbook for Nurses*, Hughes, R.G. (ed.). Agency for Healthcare Research and Quality, Rockville [Online]. Available at: https://www.ncbi.nlm.nih.gov/books/NBK2651/.

Fakroon, M., Alshahrani, M., Gebali, F., Traorè, I. (2020). Secure remote anonymous user authentication scheme for smart home environment. *Springer's Internet Things*, 9 [Online]. Available at: https://doi.org/10.1016/j.iot.2020.100343.

Fakroon, M., Gebali, F., Mamun, M. (2021). Multifactor authentication scheme using physically unclonable functions. *Springer's Internet Things*, 13 [Online]. Available at: https://doi.org/10.1016/j.iot.2020.100343.

Gao, Y., Ma, H., Al-Sarawi, S.F., Abbott, D., Ranasinghe, D.C. (2018). PUF-FSM: A controlled strong PUF. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(5), 1104–1108.

Gao, Y., Su, Y., Yang, W., Chen, S., Nepal, S., Ranasinghe, D.C. (2019). Building secure SRAM PUF key generators on resource constrained devices. *The Third Workshop on Security, Privacy and Trust in the Internet of Things*, 912–917.

Gassend, B., Clarke, D., Dijk, M.V., Devadas, S. (2002). Silicon physical random functions. *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 148–160.

Guajardo, J., Kumar, S., Schrijen, G., Tuyls, P. (2007). FPGA intrinsic PUFs and their use for IP protection. In *Cryptographic Hardware and Embedded Systems – CHES,* Paillier, P. and Verbauwhede, I. (eds). Springer, Heidelberg.

Herder, C., Yu, M.-D., Koushanfar, F., Devadas, S. (2014). Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8), 1126–1141.

Herrewege, A.V., 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. *International Conference on Financial Cryptography and Data Security*, 374–389.

Hiller, M. (2016). Key derivation with physical unclonable functions. PhD Thesis, Universität München, Munich.

Holcomb, D.E., Burleson, W.P., Fu, K. (2009). Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers*, 58(9), 1198–1210.

Jahn, S. and Borrás, J.C. (2007). Qucs: A tutorial getting started with Qucs [Online]. Available at: http://qucs.sourceforge.net/docs/tutorial/getstarted.pdf.

van der Leest, V., Preneel, B., van der Sluis, E. (2012). Soft decision error correction for compact memory-based PUFs using a single enrollment. In *Cryptographic Hardware and Embedded Systems (CHES)*, Prouff, E. and Schaumont, P. (eds). Springer, Heidelberg.

Linnartz, J.P. and Tuyls, P. (2003). New shielding functions to enhance privacy and prevent misuse of biometric templates. In *Audio- and Video-Based Biometric Person Authentication*, Kittler, J. and Nixon, M.S. (eds). Springer, Heidelberg.

Maes, R. (2013). *Physically Unclonable Functions: Constructions, Properties and Applications*. Springer, Heidelberg.

Maes, R., Tuyls, P., Verbauwhede, I. (2009). Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In *Cryptographic Hardware and Embedded Systems (CHES)*, Clavier, C. and Gaj, K. (eds). Springer, Heidelberg.

Maes, R., van Herrewege, A., Verbauwhede, I. (2012). PUFKY: A fully functional PUF-based cryptographic key generator. In *Cryptographic Hardware and Embedded Systems (CHES)*, Prouff, E. and Schaumont, P. (eds). Springer, Heidelberg.

National Institute on Aging (2020). Aging in place: Growing older at home [Online]. Available at: https://www.nia.nih.gov/health/aging-place-growing-older-home.

Prince, B. (1991). *Semiconductor Memories*, 2nd edition. John Wiley, New York.

Ravikanth, P. (2001). Physical one-way functions. PhD Thesis, Massachussetts Institute of Technology, MA.

Ravikanth, P., Recht, B., Taylor, J., Gershenfeld, N. (2002). Physical one-way functions. *Science*, 297(5589), 2026–2030.

Schrijen, G.-J. (2020). SRAM PUF: A closer look at the most reliable and most secure PUF [Online]. Available at: https://www.design-reuse.com/articles/47782/sram-puf-a-closer-look-at-the-most-reliable-and-most-secure-puf.html.

Su, Y., Holleman, J., Otis, B. (2008). A digital 1.6 pJ/bit chip identification circuit using process variations. *IEEE Journal of Solid-State Circuits*, 43(1), 69–77.

Suh, G.E. and Devadas, S. (2007). Physical unclonable functions for device authentication and secret key generation. *Design Automation Conference*, 9–14.

Xilinx, Inc. (2021). UltraScale architecture memory resources user guide, Xilinx [Online]. Available at: https://www.xilinx.com/support/documentation/user_guides/ug573-ultrascale-memory-resources.pdf.

Yu, M., M'Raihi, D., Sowell, R., Devadas, S. (2011). Lightweight and secure PUF key storage using limits of machine learning. In *Cryptographic Hardware and Embedded Systems (CHES)*, Preneel, B. and Takagi, T. (eds). Springer, Heidelberg.