A Novel Approach to PUF-based Hardware Security: Noise-aware Authentication and Key Exchange Protocol

by

Hamza Al Far
B.Sc., Jordan University of Science and Technology, 2009

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master in Applied Science

in the Department of Electrical and Computer Engineering

A Novel Approach to PUF-based Hardware Security: Noise-aware Authentication and Key Exchange Protocol

by

Hamza Al Far
B.Sc., Jordan University of Science and Technology, 2009

Supervisory Committee

Dr. Fayez Gebali, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Haytham El Miligi, Co-Supervisor
(Department of Electrical and Computer Engineering)

ABSTRACT

This thesis proposes a novel approach to PUF-based IoT security. PUFs are used to provide a unique identity to the IoT device using it. However, the authentication process can be undermined by the noisy responses of PUFs. Traditional error correction codes can address this issue but increase system complexity, overhead, and security risks. To overcome these limitations, the study proposes an innovative approach that leverages a statistical analysis technique to extract the relevant information from the PUF response bits, resulting in a strengthened authentication key. The proposed method achieves an innovative authentication and key exchange protocol with enhanced security without traditional error correction codes. The results of evaluating the approach with synthetic PUF data demonstrate a significant improvement in PUF-based security. This study represents a significant step towards enhancing the security of IoT devices, demonstrating the potential of a unique combination of PUF and the statistical analysis approach in addressing the challenges of hardware-based security.

Contents

List of Tables

List of Figures

Abbreviations

AVISPA	: Automated Validation of Internet Security Protocols and Applications
BAN	: Burrows–Abadi–Needham
CI	: Confidence Interval
CL-AtSe	: CL-based Attack Searcher
CMOS	: Complementary Metal-Oxide Semiconductor
CRP	: Challenge Response Pair
HLPSL	: High-Level Protocol Specification Language
HMAC	: Hash-Based Message Authentication
HRoT	: Hardware Root of Trust
IF	: Intermediate Format
ID	: Identification
IoT	: Internet of Things
MSC	: Message Sequence Chart
NVM	: Non Volatile Memory
OFMC	: On-the-fly Model-Checker
PMKG	: Pattern Matching Key Generator
PKI	: Public Key Infrastructure
PUF	: Physical Unclonable Function
RA	: Registration Authority
RPV	: Random Process Variation
SATMC	: SAT-based Model-Checker
SNR	: Signal to Noise Ratio
SPAN	: Secuirty Protocol Animator
SRAM	: Static Random Access Memory
TA4SP	: Tree Automata-based Protocol Analyser
WSN	: Wireless Sensor Network
Wi-Fi	: Wireless Fidelity
XOR	: Exclusive OR
5G	: Fifth Generation

ACKNOWLEDGEMENTS

I want to take this opportunity to express my gratitude to the people who supported me in preparation for my master's degree:

- **My Supervisor:** I want to thank Dr. Fayez Gebali for his continuous guidance, support, and mentorship throughout my journey. I have gained considerable experience in terms of research methodology, writing style, and systematic approach that were beneficial for both academic and personal lives.
- **My Co-Supervisor:** I want to thank Dr. Haytham El Miligi for his co-supervision, support, and advice throughout my degree preparation.
- **My Family:** I am profoundly grateful to my beloved mother, brothers, and sister, and special thanks to my lovely wife and children who provided me with full support and encouragement while studying in Canada.

Lastly, I express my appreciation to all those persons who may not be mentioned here but have played a significant role in my academic journey.

DEDICATION

To my mother **Baheye Matar**, for her unwavering support and encouragement.

To my wife **Banan Mahmoud**, for her support, love, and sacrifice.

To my lovely children **Faris** and **Noureddine**, for their boundless love and for being the reason for joy and inspiration in my life.

To my sister **Nisrin Al Far**, for her continuous support and love.

Chapter 1

Introduction

1.1 Context

The Internet of Things (IoT) represents a modern model shift that has transformed the conventional way of life into a technologically advanced lifestyle. This transformation has given rise to innovations such as smart cities, intelligent homes, smart healthcare, pollution management, energy conservation, efficient transportation systems, and advanced industrial processes.

The Internet of Things (IoT) is an emerging concept that allows electronic devices and sensors to communicate via the Internet, enhancing various aspects of our lives.

IoT is becoming widely spread, with a huge collection of smart systems, frameworks, intelligent devices, and sensors working together. As of 2023, there are 15.14 billion IoT-connected devices worldwide [3], and this figure is expected to double by 2030 [3].

IoT devices are usually resource-limited and deployed in unmonitored, physically unsecured infrastructure, often spanning significant geographical distances. However, due to previously mentioned constraints, the security of IoT edge devices becomes a major challenge to the security and integrity of the IoT systems as a whole [4].

1.2 Research Problem

The problem addressed in this dissertation is the need to develop an enhanced authentication protocol for IoT devices that utilizes physically unclonable functions (PUFs).

Edge devices that are vulnerable to attacks [5]. To ensure the security of IoT systems, it has been proposed that silicon-based physically unclonable functions (PUF) be implemented in the IoT devices themselves. PUFs are used as a primary method for establishing device authentication and secure key exchange as well as any advanced security protocols.

The use of PUFs for device authentication is a promising technique. Still, it presents several challenges, including the thermal noise associated with each challenge response, which requires error correction codes to eliminate the noise and provide a secure authentication process. Additionally, error correction codes generate helper data to be exchanged between the server and the client. Sending helper data information in the clear can reveal partial information about the secret message, considered a vulnerability in the current PUF-based authentication protocol [6].

1.3 Approach

To overcome these challenges, we propose a new approach to leverage the noise associated with PUF response to extract a unique feature of stable and unstable (noisy) bits. This enhanced model generates a featured authentication key between the server and the client. The unique features of each PUF are extracted by processing a large dataset for each PUF representing a large number of PUF responses against a single challenge. By processing PUF data, the unique features are extracted from each PUF using statistical analysis methodology.

1.4 Research Contributions

1. Developed an enhanced PUF authentication and key exchange protocol.
2. Simulated the proposed protocol using a computer-aided tool.
3. Studied the threat models against the proposed protocol and verified its immunity towards cyber security attacks.
4. Demonstrated the benefit of our proposed protocol in authentication and key exchange over other state-of-the-art methods.

1.5 Thesis Outline

Chapter 1 This chapter presents the problem, context, research contributions, and thesis outline.

Chapter 2 presents the key concepts of a preliminary literature review of the state-of-the-art PUF-based authentication systems. It also presents the related work in this area of research.

Chapter 3 explains the proposed methodology used in this research study

Chapter 4 presents the experimental setup, implementation, major protocol entities, and phases of the system under consideration.

Chapter 5 discusses the results of the proposed model's informal and formal security analysis and evaluates the model's immunity against cyber security attacks.

Chapter 6 presents the conclusion and future works.

Chapter 2

Related Work

Internet of Things (IoT) devices are widely deployed in various applications, from smart homes to telehealth systems. However, the security of these devices is a significant concern due to their physical exposure, unsupervised deployment, physically distributed nature, and limited processing capabilities [7]. Hardware attacks, particularly those that exploit vulnerabilities in the device's hardware, can have severe consequences since they are often undetectable and can rapidly transmit throughout the entire network [7]. Physical Unclonable Functions (PUFs) are widely used as a promising solution for lightweight security solutions for limited-resource IoT devices[8], PUFs exploit random process variations in semiconductor manufacturing to generate unique fingerprints, and those fingerprints are not reproducible for silicon chips that are built with the same manufacturing process.

2.1 PUF Based Authentication and Key Exchange Protocols

The concept of using PUF as a hardware security primitive was introduced by [9]. The authors implemented delay-based PUF that utilizes the silicon random process variation to identify and authenticate a given integrated circuit. The authors discussed using error correction codes "Improved PUF" to overcome the problems related to incorrect authentication rejection caused by environmental variation.

In [10], the authors discussed an authentication scheme using delay-based PUF called "Noisy PUF". The authors utilized the inherited noise in PUF responses to increase the precision of the device features and its sensitivity toward physical attacks.

The authors claimed that the inherited noise in PUF responses would increase the number of challenge-response pairs needed to model the system. In addition, to prevent modeling attacks against the proposed system, the author proposed additional PUF to be added to the noisy PUF circuit, this additional PUF generates a vector X which is XOR-ed with the challenge to be applied to the noisy PUF. The proposed modifications for the authentication protocol shall not expose the challenge-response pair to attackers and protect the system against modeling attacks.

In [11], the authors used the PUF for light-weight authentication and key generation applications, the publication identified two types of PUFs: "Strong PUF" and "Weak PUF". This classification is based on the available ranges of challenge-response pairs that PUFs can generate and the necessity to associate the PUFs with crypto hardware supporting Hash-Based Message Authentication Code (HMAC) or related authentication processes.

The authors in [11] indicated several methods to mitigate the effect of noise in PUF responses and improve error-correction performance, such as using differential design techniques and soft decision coding, which is based reliability information of given response bits.

A high-level authentication and key exchange protocol for smart homes and IoT telehealth system was recently discussed by [1, 12], The authors protocol proposed used a two-factor authentication scheme that preserved user anonymity and untraceability. In [12] publication, the IoT devices are assumed to be equipped with non-volatile memory to store the secret keys, while in [1], the proposed protocol assumes that IoT devices (client) equipped with PUF that can generate a secret key and provide unique IDs for its edge devices, the server securely authenticates the client via a given challenge, and the corresponding response by the client, traditional error correction technique was implemented to eliminate the effect of noise on PUF response and provide successful authentication for IoT devices.

The authors in [13] introduced a new lightweight mutual authentication and key exchange protocol, leveraging Physically Unclonable Functions (PUF). This protocol enables two endpoint devices with constrained resources equipped with PUF to authenticate each other without relying on local storage for storing challenge-response pairs or private keys. Additionally, it prevents the need for an intermediary verifier node (server) during the authentication procedure.

The proposed protocol in [13] effectively mitigated the security risk of storing secret data in non-volatile memories. Consequently, the secure exchange of keys between the two endpoints is directly established following successful mutual authentication.

In [14], the authors proposed a PUF-based authentication protocol tailored for wireless medical sensor networks, the authors employed fuzzy extractors, a one-way hash function, and bitwise XOR operation. The proposed scheme facilitates the dynamic update of password and biometric information, seamless deployment of additional medical sensors, and the collection and transmission of sensor data to the central monitoring facility.

2.2 Error-Correction Techniques for PUF Based Protocols

PUF-based authentication protocols rely on challenge-response pairs, where the server sends challenges, and the IoT device (client) provides a unique response to the given challenge. PUFs are sensitive to dynamic thermal noise, so the PUF response is normally noisy. Therefore, techniques have been implemented to eliminate the noise from PUF response and regenerate a reliable noise-free response from noisy responses using error correction codes normally referred to by fuzzy extractors [15].

Fig 2.1 shows a PUF-based authentication protocol proposed by [1], the proposed protocol employing an error-correction technique using a fuzzy extractor. The server selects a challenge-response pair from the stored dataset and issues a challenge c to the fuzzy extractor to generate helper data w. Simultaneously, using the response r, the server generates a key k via a key generator and subsequently hashes the response h.

Following this, the server dispatches the challenge c and helper data w to the client, typically endowed with a built-in PUF. It is worth noting that the PUF response typically comprises noise, as represented by r^*. Therefore, a decoder is employed to recover a noise-free response r from the combination of r^* and helper data w. This noise-free response is then hashed as h^* and transmitted back to the server for verification. The authentication process entails a match between h and $h*$. In the event of a match, authentication is deemed successful. Otherwise, the authentication fails.

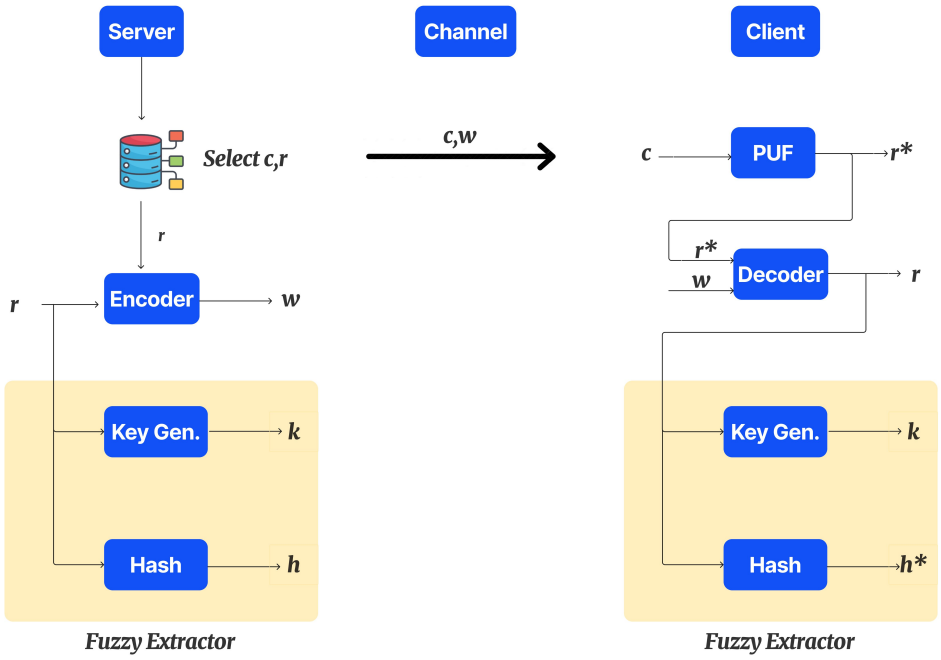

Figure 2.1: Traditional structure of PUF-based authentication protocol using a fuzzy extractor [1].

2.3 Observations on the Current PUF Based Authentication and Key Exchange Protocols

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

1. Sending helper data information in the clear can reveal partial information about the secret message. Sharing this data in public is considered a vulnerability in the current PUF-based authentication protocol [6].

2. In the security protocol that is implemented using weak PUFs, there are limitations at this type of PUF in using of CRPs since they are linearly related to the number of PUF circuit components, and the same challenge can't be repeated [11].

3. In [15], The authors presented a comprehensive framework to ensure the secure correction of errors in biometric data. To achieve robustness, the number of corrected errors must match or surpass the highest potential number of bit

errors resulting from a broad spectrum of environmental variations. The proposed scheme requires a heavyweight error correction logic, hence this requires additional overhead and increases the system's complexity.

4. In [16], the authors proposed a Key Generation technique called Pattern Matching Key Generator (PMKG) using that Non-volatile memory (NVM) to store the PUF response, storing secret information in NVM is considered a security risk since simple memory-based attacks expose these secret keys. Furthermore, the challenge arises in updating these secret keys as the NVM must be reprogrammed [2].

These observations affect the effectiveness of PUF-based security protocols for authentication and secure key exchange. In this work, we propose an enhanced approach that could be a perfect solution to the above-mentioned observations.

Chapter 3

Background and System Analysis

This section presents a comprehensive overview of the authentication protocol's roadmap. As explained earlier, contemporary authentication and key exchange protocols based on Physical Unclonable Functions (PUFs) pursue to eliminate the noise from PUF responses, trying to produce noise-free outputs applicable to authentication protocols.

The proposed protocol's underpinnings rest upon exploiting the nature of noisy PUF responses, aiming to leverage the inherent noise in PUF responses toward a specific challenge applied in several instances. Statistical analysis is employed to extract dependable features from these noisy PUF responses, which are subsequently harnessed within the framework of the proposed authentication protocol.

3.1 Internet of Things and Authentication

The Internet of Things (IoT) is a cutting-edge computing model that emerged in the 21st century. It connects both living and non-living entities, forming interconnected ecosystems. IoT encompasses everyday objects capable of sensing their environment and autonomously sharing data with other objects and services via the Internet, without human intervention [17].

Wireless sensor networks (WSN), dating back to the 1980s, are integral to IoT. WSNs consist of sensor nodes linked wirelessly, providing digital interfaces for real-world objects [18]. However, IoT differs from WSN in several ways. A key distinction is that WSN comprises numerous interconnected sensor nodes primarily for sensing and data collection, whereas IoT systems involve a wide array of interconnected objects, sensors, devices, etc. These IoT entities offer value-added services like location

tracking and analytics through intelligent data processing and management for various applications.

Furthermore, IoT necessitates Internet connectivity for its nodes, whereas WSNs do not require such connectivity.

In addition, IoT devices have several limitations including limited computer processing capabilities, which may prevent the implementation of complex algorithms for secure key exchange [2].

Authentication typically involves verifying an individual's identity using a unique identifier, but this can lead to privacy concerns because transactions associated with the same identity can be traced over time. This traceability, especially in an Internet of Things (IoT) environment like a smart home, can potentially reveal sensitive information about a household's lifestyle, such as health and credit history, living patterns, and more [19].

Anonymity and traceability serve as crucial privacy protection measures, making it considerably more challenging to trace user transactions. Despite various authentication schemes suggested for the Internet of Things (IoT), few of them have addressed extensively the issue of achieving complete anonymity for IoT sensor nodes in the context of authentication and access control protocol [20]

3.2 Physical Unclonable Functions (PUFs)

A Physically Unclonable Function (PUF) is a hardware primitive that leverages the inherent randomness arising from the manufacturing process of silicon semiconductors. This randomness is harnessed to offer cryptographic capabilities [21]. Consequently, PUFs are cost-effective to produce, exceptionally difficult to replicate, lack a concise mathematical representation, and possess inherent resistance to tampering[22].

In the context of mathematical analysis, a Physical Unclonable Function (PUF) is not adequately represented as a deterministic function [22]. This is because a single input (challenge) can yield multiple different outputs (responses) due to the effect of the physical environment and random noise that influence the response generation. Hence, a more appropriate mathematical approach to PUF involves treating it as a probabilistic function. In this regard, an intrinsic element of uncertainty becomes inherent in the input to the function [22]. Conversely, the outcome of a probabilistic function can be regarded as a random variable, described by a specific probability density function, rather than a deterministic value [23].

3.3 SRAM PUF

SRAM cells facilitate the construction of silicon PUF through the unique startup values of each word line in the memory [24]. The SRAM binary values each time the SRAM PUF initialization is slightly different due to the dynamic noise [25].

The SRAM PUF offers a great approach to analyze and simulate the effects of random process variation (RPV) and thermal noise in relation to PUF operations.

The working principle of the SRAM PUF is based on two aspects: random process variation and dynamic noise. These events together determine how the SRAM PUF behaves during its operations[2].

The functionality of the SRAM PUF heavily depends on utilizing the said event, which controls the digital binary values stored within the SRAM bits after initialization[2].

Once the fabrication process is completed, random process variation becomes static, creating a fingerprint for each SRAM instance. On the hand, dynamic noise varies across initialization of the SRAM PUF, resulting in diverse outcomes for each instance.

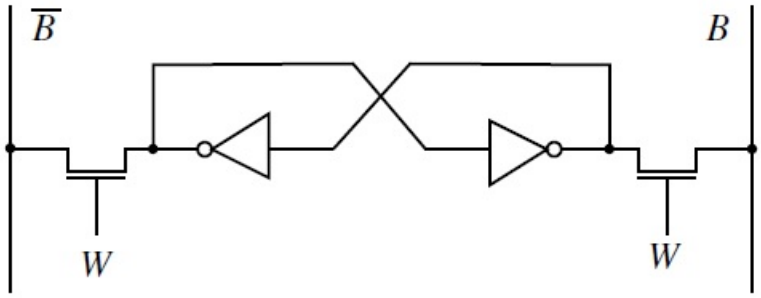

Figure 3.1: Basic cell structure for a 6-transistor SRAM CMOS cell

Fig. 5.1 shows the basic structure of 6-transistor SRAM PUF. The logic memory in SRAM cells generated by the two cross-coupled inverter, the SRAM circuits contain a positive feedback loop which makes it keep its current state. The SRAM circuit has two stable (bi-stable) states, and the cells store the binary value at one of its stable states.

There is another temporary operational state for SRAM cells called the unstable state. The SRAM never stays unstable as it moves quickly to any of its two stable states.

The SRAM PUF operation principle is based on the random transition of unstable

SRAM cells to their preferred stable state. After the transition, the value stored in the SRAM cell depends on the random process variation (strengths) between the two cross-coupled inverter[2].

The difference in inverters' strength is called device mismatch, caused by random process variations of silicon semiconductors intended to be produced identically[2].

To create an unstable state on SRAM cells:

1. Disconnect and re-connect the power supply V_{DD}, this will force both outputs B and $\overline{B}$ to be '0'.

2. Ground both bit lines, $B = \overline{B} = 0$, and set word line $W = 0$. This will force both outputs to equal '0'.

3. [2] proposed a new modified model of SRAM PUF shown in Fig.5.2. This model facilitates the rest operation of SRAM PUF by setting the value of R = 1. This will force both output lines B and $\overline{B}$ equal '0'. As soon as R = 0, the output changes to '0' or '1' randomly due to random process variation and other factors.

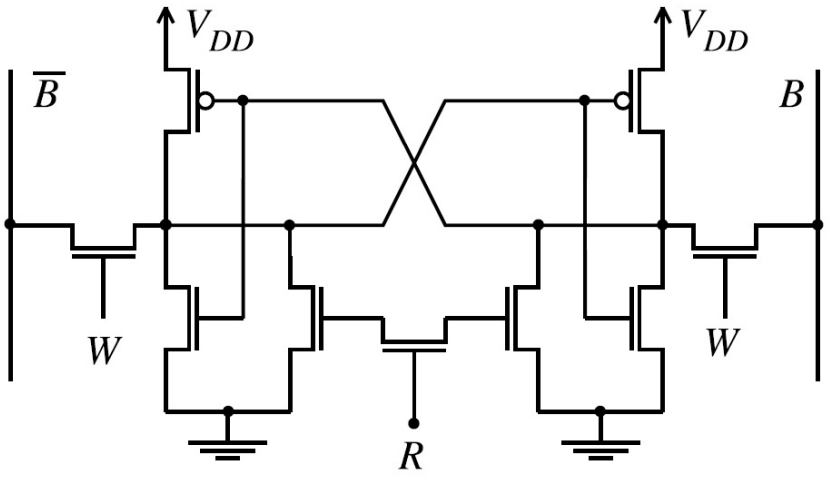

Figure 3.2: Detail of the basic architecture of NOR gate-based SRAM PUF [2]

3.4 SRAM PUF Statistical Model

One approach to study the PUF performance is the statistical model of SRAM PUF [23], a random variable is chosen to represent the content of SRAM memory cells, the

binary random variable that is characterized by two probabilities, a_i and b_i, denoting the probability of the SRAM cell is "1" or "0", respectively.

Normally, the random process variations (RPVs) and thermal noise are absent, and the structure of each SRAM cell is completely symmetric, which implies the equal probability of "1" and "0" for a and b [23]

$$a_i = b_i = 0.5 \tag{3.1}$$

Based on the central limit theorem, random process variations (RPVs) affect the equal probability of (a_i and b_i), which follows a biased Gaussian distribution with a probability density function represented by the following equation [23]:

$$f_{Ap} = \frac{1}{\sigma_p\sqrt{2\pi}} \exp \frac{-(a_p - a_i)^2}{2\sigma_p^2} \tag{3.2}$$

where a_p is the biased value of a_i due to random process variations, and σ_p^2 is the variance of random process variations process. The value of a_p is given by:

$$a_p = G(a_i, \sigma_p) \tag{3.3}$$

Where G(a_i, σ_p) is the Gaussian distribution with a mean equal to $a_i = 0.5$ and variance σ_p^2.

[23] represented random process variation of SRAM PUF by taking a random value from Gaussian distribution with mean equal to $a_i = 0.5$ and variance σ_p^2.

3.5 Synthetic Model of SRAM PUF

To provide a clear picture of the statistical analysis approach employed in developing the proposed protocol,a synthetic model was constructed using the Python programming language [26]. This model enables the user to define specific Physical Unclonable Function (PUF) parameters, such as the random process variation (RPV) denoted by a_p and σ_p as previously mentioned in Section 3.2.

Furthermore, the implemented code allows the user to regulate the thermal noise affecting the PUF response, allowing the generation of PUFs with a wide spectrum of noise levels, ranging from highly noisy to nearly noise-free responses. The extent of noise in our model can be modulated by manipulating the Signal-to-Noise Ra-

tio (SNR), where the SNR for the Static Random-Access Memory (SRAM) PUF is characterized by the following equation [23]:

$$\text{SNR} = 10 \log \left(\frac{(a_p - a_i)^2 + \sigma_p^2}{\sigma_n^2} \right) \tag{3.4}$$

where a_p and σ_p represents the random process, and σ_n represents the dynamic noise.

[23] mentioned that bits within the same SRAM word line have the same SNR, while all bits in the SRAM, do not have the same SNR.

[23] stated inducated that the minimum SNR occurs when $a_p = a_i$, and the response is totally dependent on thermal noise. The following equation represents SNR_{min}:

$$\text{SNR}_{min} = 10 \log \left(\frac{\sigma_p^2}{\sigma_n^2} \right) \tag{3.5}$$

On the other hand, the maximum SNR occurs when either $a_p = 0$ or $a_p = 1$, since $a_i = 0.5$. Therefore the equation for SNR_{max} is written as follows:

$$\text{SNR}_{max} = 10 \log \left(\frac{a_i^2 + \sigma_p^2}{\sigma_n^2} \right) \tag{3.6}$$

When SNR = SNR_{max}, the PUF response is more stable and less dependent on noise [23].

3.6 Estimating the appropriate value of (n)

Comparing the probability of PUF response to the probability of tossing a fair coin, a fair coin with two equally possible outcomes, Head or Tail, represents a PUF without random process variation, with an equal probability of “1” and “0”.

On the other hand, tossing an unfair coin with a biased probability towards the head or tail represents the actual PUF response with random process variations (RPVs). So, the question is how many trials are required to check whether a coin is fair.

Verifying if a coin is unbiased might seem like a straightforward task by conducting a significant number of trials. However, statistics and probability theory offer guidance on two critical issues:

1. determining the appropriate number of trials to perform.

2. estimating the accuracy of the probability of obtaining a head based on a given sample of trials.

3.7 The Estimator of True Probability (Frequentist Approach)

In this method, the experimenter can choose the number of times to toss the coin. To begin, the experimenter determines the desired level of confidence and the acceptable margin of error. These parameters dictate the minimum number of tosses necessary to complete the experiment [27]. To determine the number of times the challenge c shall be repeated on a specific PUF to determine the mean that a response bit is biased to, we should specify three parameters:

1. The *confidence interval* (CI), which is a range of estimates for an unknown parameter.

2. The *confidence level* (Z), which represents the long-run proportion of CIs (at the given confidence level) that theoretically contain the true value of the parameter.

3. The *maximum (acceptable) error* (E), which is the difference between estimated probability and true probability.

The confidence level is donated by Z and is given by the Z-value of standard normal distribution. This value can be examined in a standard score statistical table for normal distribution which is described in table 3.1 [27].

The maximum error E is defined by: $\mid a_i - a_p \mid < E$. The estimated same proportion a_i is given by:

$$a_i = \frac{\text{number of success}}{\text{number of trails}} = \frac{\text{number of 1's}}{\text{number of responses}} \tag{3.7}$$

$$\sigma_p = \sqrt{n a_p (1 - a_p)} \tag{3.8}$$

$$\sigma_i = \frac{\sigma_p}{n} \tag{3.9}$$

So, the estimate of a proportion of a sample (denoted by a_i) has a standard error (uncertainty) given by:

$$\sigma_i = \sqrt{\frac{a_i(1 - a_i}{n}} \tag{3.10}$$

Table 3.1: Z values vs. confidence level for normal distribution function

Z value	**Confidence level**
0.6745	50.000%
1.0000	68.269%
1.6449	90.000%
1.9599	95.000%
2.0000	95.450%
2.5759	99.000%
3.0000	99.730%
3.2905	99.900%
3.8905	99.990%
4.0000	99.993%
4.4172	99.999%

Where n is the number of repeated PUF challenges.

The standard error σ_i a function of a_i, it has a maximum value at a_i= (1-a_i) = 0.5. Additionally, in the case of PUF response, it is most likely that a_i will not be far from 0.5, so it is logical to take $a_i = 0.5$ in the following equation:

$$\sigma_i = \sqrt{\frac{a_i(1-a_i)}{n}} \leq \sqrt{\frac{0.5 \cdot 0.5}{n}} = \frac{1}{2\sqrt{n}} \tag{3.11}$$

Hence, the value of maximum error (E) is given by:

$$E = Z \times \sigma_i = \frac{Z}{2\sqrt{n}} \tag{3.12}$$

Solving for the required number of repeated challenges n:

$$n = \frac{Z^2}{4 \cdot E^2} \tag{3.13}$$

Based on the above, if we are seeking for 95.000 % confidence level with maximum error equals to 10^{-2} to determine the appropriate number of repeated challenges to obtain the value of a_p.

By substituting the values of $Z = 1.9599$, and $E = 0.01$ in equation (3.10):

$$n = \frac{1.9599^2}{4 \cdot 0.01^2} = 9{,}604 \text{ trials}$$

However, if we seek a higher level of confidence i.e., 99.90 % with the same maximum error value, the number of trials shall be increased to 27,069 trials.

3.8 SRAM PUF Model with SNR = SNR$_{min}$

In this section, we shall proceed to construct a synthetic model of SRAM PUF using Python programming language. This endeavour aims to investigate and conduct a comprehensive statistical analysis of the PUF responses against a specific challenge repeated n times.

For simplicity, we shall assume the proposed SRAM model consists of 10-word lines (W) and 16-bit lines. To incorporate the effect of random process variations, we adopt a standard deviation $\sigma_p = 0.1$ for the RPV effect. We shall specify a minimum value for the signal-to-ratio, SNR$_{min}$ = 1 dB, which implies that the PUF has a noisy response and most of its bits are unstable. The repeated challenge is applied at word line (W) = 5. Given a confidence level of 95.000 % and maximum error equals 10^{-2}, the number of repetitive challenges n is set to 10,000.

Model 1

The model parameters are as per the following:

- SRAM PUF word lines (W) = 10.
- SRAM PUF bit lines (B) = 16.
- RPV standard deviation, $\sigma_p = 0.1$.
- SNR$_{min}$ = 1 dB.
- Number of repeated challenges n = 10,000.
- Word line address that represents the challenge $c = 5$.

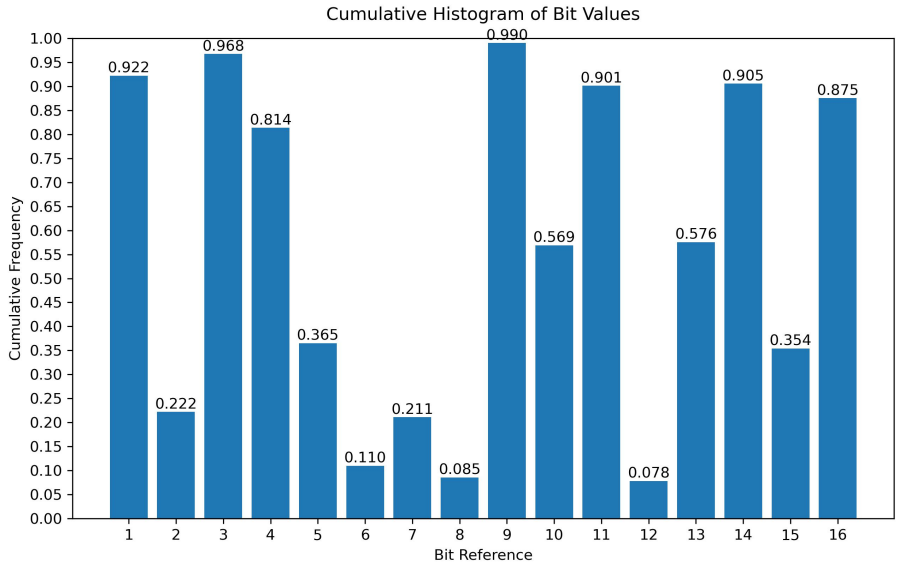

Figure 3.3: Commutative histogram of SRAM PUF bits response against n repeated challenge (Model 1)

3.9 SRAM PUF Model with SNR = SNR$_{max}$

Similar to Sec. 3.7, in this section we shall proceed to construct a synthetic model of SRAM PUF using Python programming language. This endeavor aims to investigate and conduct a comprehensive statistical analysis of the PUF responses against a specific challenge repeated n times.

For simplicity, we shall assume the proposed SRAM model consists of 10-word lines (W) and 16-bit lines. To incorporate the effect of random process variations, we adopt a standard deviation $\sigma_p = 0.1$ for the RPV effect. We shall specify a maximum value for the signal-to-ratio, SNR$_{max}$ = 10 dB, which implies that the PUF has a stable response and most of its bits are stable bits. The repeated challenge is applied at word line (W) = 5. Given a confidence level of 95.000 % and maximum error equals 10^{-2}, the number of repetitive challenges n is set to 10,000

Model 2

The model parameters are as per the following:

- SRAM PUF word lines (W) = 10.
- SRAM PUF bit lines (B) = 16.

- RPV standard deviation, $\sigma_p = 0.1$.
- $\text{SNR}_{max} = 10$ dB.
- Number of repeated challenges $n = 10{,}000$.
- Word line address that represents the challenge $c = 5$.

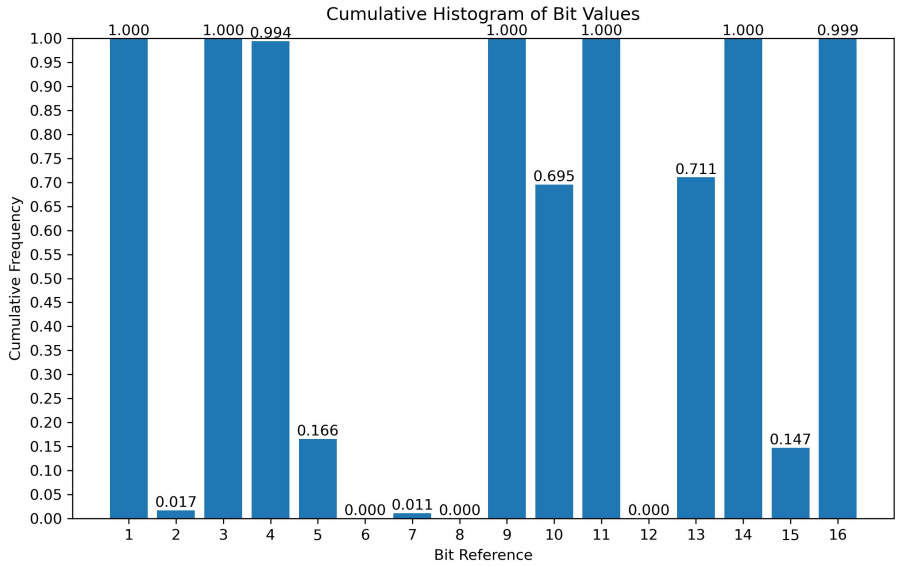

Figure 3.4: Commutative histogram of SRAM PUF bits response against n repeated challenge (Model 2)

3.10 SRAM PUF Statistical Analysis

Fig. 3.3 and Fig. 3.4 represent the histogram of two SRAM models with the same RPV standard deviation σ_p but with different values of signal-to-noise ratio SNR.

Model 1 histogram represents 16 SRAM indices that are primarily unstable due to significant noise contribution in PUF response, while model 2 represents 16 SRAM PUF indices that are mostly stable due to vegetable noise contribution in the PUF response.

We can apply statistical analysis of both PUFs responses against specific challenge c that is applied n times to extract unique features for the proposed protocol.

3.10.1 Statistical Analysis Methodology

1. Create a histogram for each response bit to determine the bit value frequency (0 or 1) over n trials. The histogram indicates some bits' average value is close to 1, some bits' average value is close to 0, and other bits' average value is close to 0.5 within a range predefined thresholds.

   The bits with mean values close to 1 or 0 with predefined thresholds are identified as *Stable bits*, and the remaining bits are identified as *Unstable bits*, the upper and lower thresholds range values for stable bits are described in Eq. (3.14) and Eq. (3.15) respectively.

$$T_{upper} = (1 - T) \rightarrow 1 \tag{3.14}$$

$$T_{lower} = 0 \rightarrow T \tag{3.15}$$

2. We identify the indices of the entire PUF response as vector $\boldsymbol{i}$, stable bits indices as vector $\boldsymbol{v}$, and unstable bit indices as $\boldsymbol{u}$.

3. The number of elements in $\mathbf{s}$ depends on selection threshold value T, T represents two values identified previously in Eq.(3.14) and Eq. (3.15) for which if the average bit values fall within the range of T the bit will be selected, and its index will be included in $\mathbf{s}$.

4. Based on a set of threshold values: T_1, T_2, T_3, a three selection vectors $\boldsymbol{v_0}, \boldsymbol{v_1}, \boldsymbol{v_2}$ are formed. where $T_3 > T_2 > T_1$.

   These threshold values are chosen based on the behaviour of each PUF to ensure the generation of multiple selection vectors with distinct indices, maximizing the entropy of the selection matrix values.

5. $a \times b$ matrix called selection matrix $\mathbf{M}$ is formed, where a represents the number of selection vectors $\boldsymbol{v}$, and b represents the elements of each selection vector which are exactly equal to the number of response bit indices. A zero padding is applied for unselected bit indices to generate a consistent number of elements in matrix columns. Also, bit index numbers shall start from index 1 to avoid any confusion with 0 padding values.

6. We determine the unstable bit vector $\boldsymbol{u}$ based on threshold value T_1 based on

the following equation:

$$\boldsymbol{u} = \{\boldsymbol{i}/\boldsymbol{v_0}\} \tag{3.16}$$

Where $\boldsymbol{u}$ is a vector that represents the **values** of unstable bits with 0 padding for unselected indices, $\boldsymbol{i}$ is a vector that represents the entire response indices, and $\boldsymbol{v_0}$ is a vector that represents stable bit indices at threshold T_1 with 0 padding for unselected indices.

$\boldsymbol{u}$ is a vector with random values that changes with each challenge c applied to the PUF, even if it's a repeated challenge. $\boldsymbol{u}$ will be used later to generate a *nonce* that is used to establish the communication session between the server and client as described in Sec. 4.2.4.

3.10.2 Statistical Analysis of the Proposed PUF Models

Considering the previous SRAM PUF models (model 1 and 2), We can apply the statistical analysis steps described in Sec. 3.9.1 as follows:

1. Histograms for both SRAM models are plotted in Fig. 3.3 and Fig. 3.4.

2. Setting $T1 = 0.05$ and applying Eq. 3.14 and Eq. 3.15 to calculate the upper and lower threshold limits:
   $T1_{upper} = 0.95 \rightarrow 1$
   $T1_{lower} = 0 \rightarrow 0.05$

3. Setting $T2 = 0.1$ and $T3 = 0.15$, we get the following threshold limits:

   $T2_{upper} = 0.9 \rightarrow 1$
   $T2_{lower} = 0 \rightarrow 0.1$
   $T3_{upper} = 0.85 \rightarrow 1$
   $T3_{lower} = 0 \rightarrow 0.15$

4. Applying the previous threshold limits on model 1, we can get the following selection vectors:
   $\boldsymbol{v0} = \{3,9\}$
   $\boldsymbol{v1} = \{1,3,8,9,11,12,14\}$
   $\boldsymbol{v2} = \{1,3,6,8,9,11,12,14,16\}$

5. Applying the previous threshold limits on model 2, we can get the following selection vectors:

$\boldsymbol{v0}$ = {1,2,3,4,6,7,8,9,11,12,14,16}
$\boldsymbol{v1}$ = {1,2,3,4,6,7,8,9,11,12,14,16}
$\boldsymbol{v2}$ = {1,2,3,4,5,6,7,8,9,11,12,14,15,16}

6. Forming selection matrices for model 1 and model 2:

$$M_1 = \begin{bmatrix} 3 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 8 & 9 & 11 & 12 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 6 & 8 & 9 & 11 & 12 & 14 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 1 & 2 & 3 & 4 & 6 & 7 & 8 & 9 & 11 & 12 & 14 & 16 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 6 & 7 & 8 & 9 & 11 & 12 & 14 & 16 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 11 & 12 & 14 & 15 & 16 & 0 & 0 \end{bmatrix}$$

7. Generating unstable bit vectors for model 1 and model 2:

$$\boldsymbol{u}_1 = \begin{bmatrix} 1 & 2 & 4 & 5 & 6 & 7 & 8 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{u}_2 = \begin{bmatrix} 5 & 10 & 13 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As previously mentioned, the employed statistical analytical methodology on models 1 and 2 yielded the extraction of distinct fingerprints corresponding to individual SRAM PUFs. These fingerprints hold potential for utilization within an upcoming chapter's discussion on authentication protocols.

Chapter 4

System Implementation

The current PUF-based authentication and key exchange protocols are considered vulnerable due to sharing helper data in the clear [6]. However, the proposed protocol in this study addresses this vulnerability and enhances the security of IoT systems that use Physical Unclonable Functions (PUF). The proposed protocol utilizes the noisy nature of PUF responses to a given challenge. The protocol's basic concept is to exploit the noisy PUF responses to specific challenge, and perform a statistical analysis to extract reliable features to be used in the propsoed authentication protocol.

4.1 Preliminaries

Fig 4.1 shows the basic structure of the IoT system under consideration. The main components in the proposed protocol are:

1. Fabrication house (Fab. house)
2. Registration Authority (RA)
3. Server (HRoT)
4. Client (IoT edge devices)
5. Communication network (channel)

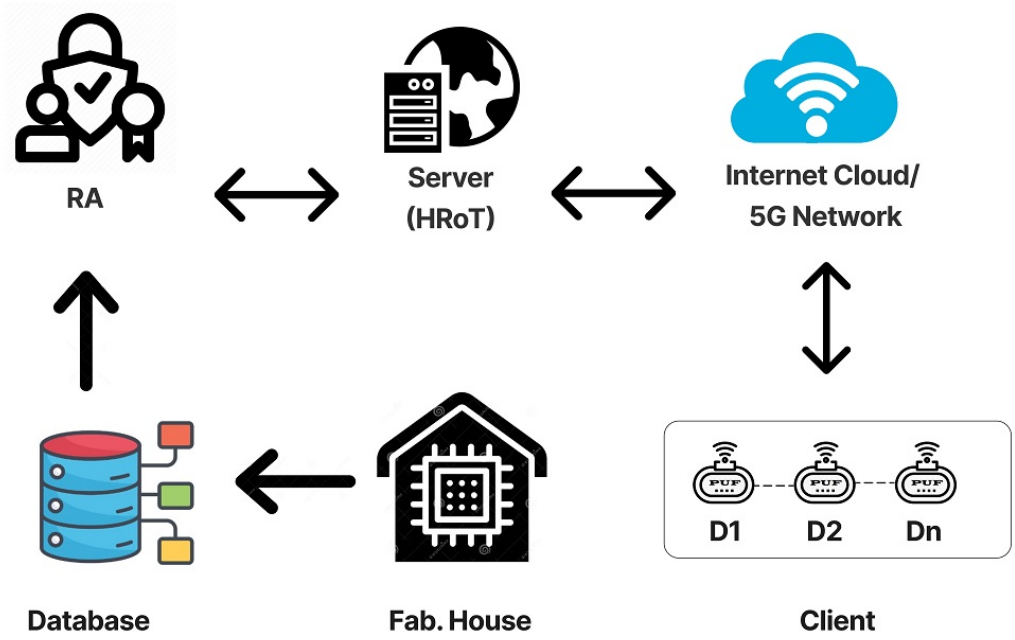

Figure 4.1: Basic structure of proposed IoT system

4.1.1 Fabrication house (Fab house)

The Fab. house is the entity responsible for manufacturing the edge devices that contain PUF, and its primary role is identified in the pre-deployment phase of the proposed protocol that is described in Sec. 3.2.1

4.1.2 Registration Authority (RA)

Registration Authority (RA) ensures secure communication between the Fab. house and the server at the registration phase through public key infrastructure.

4.1.3 Server

The server forms an essential part of the authentication protocol. It is considered hardware root-of-trust (HRoT) as the security is maintained from the application level down to the hardware level. This is considered mandatory to ensure the security of data required for authentication and key exchange between server and client.

4.1.4 Edge device (Client)

The client is the IoT edge devices, which we aim to provide a high level of security with our proposed protocol. The edge device will be equipped with PUF that is responsible for authentication and key exchange with the server.

4.1.5 Communication network (channel)

Internet cloud, which could rely on 5G and Wi-Fi 6 technologies for increased throughput and reduced latency.

4.2 PUF authentication and key Exchange protocol under consideration

Fig. 4.2 shows the block diagram of the proposed protocol, and it indicates briefly the basic steps of the authentication process between server and client.

The keystone of the proposed protocol is to limit sharing of secret data in clear between communicating entities, as we can see from the block diagram that only challenge c is being sent from server to client. On the other hand, hashed value $h*$ will be sent back from the client to the server to verify and authenticate the edge device in question.

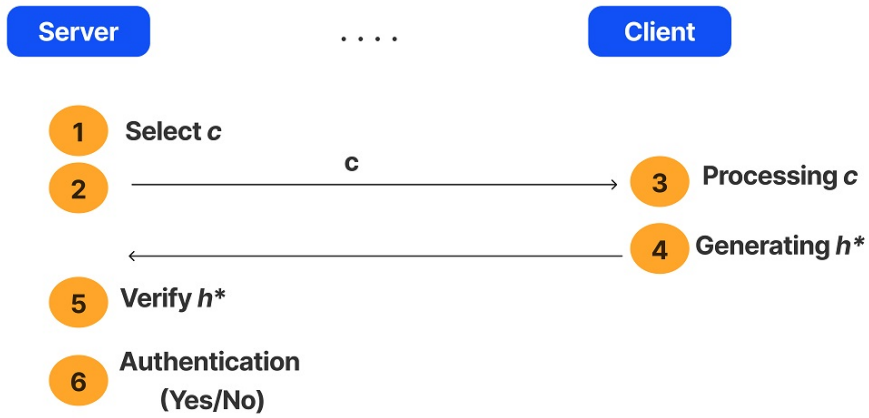

Figure 4.2: Block diagram of proposed protocol

The authentication and key exchange process is divided into four phases:

1. Pre-deployment
2. Registration
3. Pre-authentication
4. Authentication

The phases are classified in chronological order according to the authentication process.

4.2.1 Notation of Proposed Scheme

Table 4.1 illustrates the notification used in the different phases of the proposed protocol.

Table 4.1: Notations of the Proposed Scheme

Symbol	Description
S	Server
D	Edge Device
RA	Registration Authority
ID_s	Identity of server S
ID_d	Identity of device D
DID_d	Dynamic identity of D
c	Challenge
M	Selection matrix
$E_s(K, m)$	Symmetric encryption of message m with secret key K
$S \rightarrow \mathrm{D} : m$	S sends a message m to D through a communication channel
$m_1 \| \| m_2$	Concatenating two messages m_1 and m_2
$h(m)$	one-way hash function of message m
$m_1 \oplus m_2$	XOR operation between m_1 and m_2
N_s	Nonce
$A == B$	Check equality of A and B
cMp_d	Challenge and selection matrix pairs of D

4.2.2 Pre-deployment

Fig.4.3 shows the pre-deployment phase. The main entities involved in the pre-deployment phase are the fabrication house, registration authority, and server.

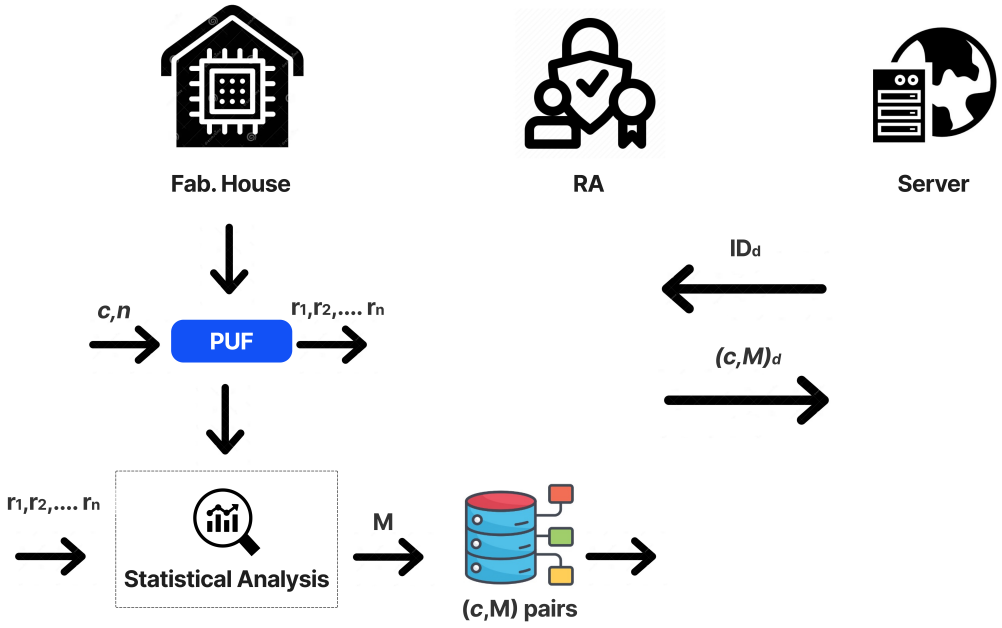

Figure 4.3: Proposed protocol (pre-deployment phase)

Fabrication house

The fabrication house applies the same challenge n times into the PUF and observes the n responses. Then, it performs *statistical analysis* on the generated responses, *statistical analysis methodology* was described in Sec. 3.9.

Server

During the pre-deployment phase, the server establishes a secure communication channel with the Registration Authority (RA) using a symmetric secret key K_{sr}. Moreover, the server is assigned a unique identifier ID_s. The server implements multiple layers of security protocols. Hence, it's considered a HRoT. Its primary objective is to communicate with the RA to acquire the credentials for edge devices connected to the IoT system.

We establish predefined values for the number of repeated challenges n and threshold values T, which are associated with each PUF and linked to its unique ID. So, These parameters are predetermined prior to the deployment of IoT devices and are accessible to both the server and the client.

Edge Device

In the manufacturing process, the edge device is assigned a unique identifier ID_d. The unique device ID encapsulates the device number, number of repeated challenges n, and threshold values T. Subsequent fabrication, the fabrication house generates the device data set that includes the challenges corresponding selection matrix (c,$\mathbf{M}$ pairs). This information must be kept secret and will be shared at a later stage with the registration authority (RA).

4.2.3 Registration

The main entities involved in the registration phase are the registration authority (RA), the server, and the Fab. house. The registration process starts by establishing secure communication between the server and Fab. house with the registration authority (RA) through symmetric key infrastructure.

Edge devices registration

This step is done offline. Considering the nature of IoT systems, they are server and ith edge devices connected to a server. The RA selects a unique ID for the server S ID_s and each edge device ID_d.

The registration process involves the following steps:

1. Upon the completion of the pre-deployment process, the fabrication house provides the registration authority (RA) with the data sets (c,$\mathbf{M}$ pairs) for the edge devices deployed at the field.

2. The server requests the registration authority (RA) for the data related to the edge device that requires authentication. The server provides the unique device ID_d that identifies the edge device (client).

3. The registration authority (RA) sends the data sets associated with the edge device to the server. (c,$\mathbf{M}$ pairs) to the server.

The following equations summarize the registration process explained in sec. 3.2.2:

$$S : m_1 = E_s(K_{sr}, (ID_s||ID_d||N_{s1}) \tag{4.1}$$

$$S \rightarrow RA : Request(ID_s, ID_{ra}, m_1) \tag{4.2}$$

$$RA : m_2 = (cMp_d||N_{s1}) \tag{4.3}$$

$$RA \rightarrow S : E_s(K_{sr}, m_2) \tag{4.4}$$

where N_{s1} is a nonce to ensure the freshness of the communication between RA and server from one side and the server and edge devices from another.

Operation in Eq. (4.1): the server prepares an encrypted message m_1 for RA using shared symmetric key K_{sr}, and the message m_1 includes the server ID, edge device ID, and the nonce N_{s1}.

Operation in Eq.(4.2): the server sends RA a request to communicate that includes the message m_1.

Operation in Eq. (4.3): the RA prepares a message m_2 that includes the challenge/selection matrix pairs for the same edge device, along with the received nonce N_{s1}.

Operation in Eq. (4.4): the RA responds to the server request with encrypted messages that include the nonce to confirm the freshness of the session.

4.2.4 Pre-authentication

They are two communicating entities involved in the pre-authentication process: server and client (edge device). As indicated previously in Fig. 4.2, which shows a block diagram of the authentication entities of the proposed protocol.

Server

The server initiates the pre-authentication process by selecting a challenge c and selection matrix $\mathbf{M}$.

The pre-authentication process overlaps with the registration and authentication process.

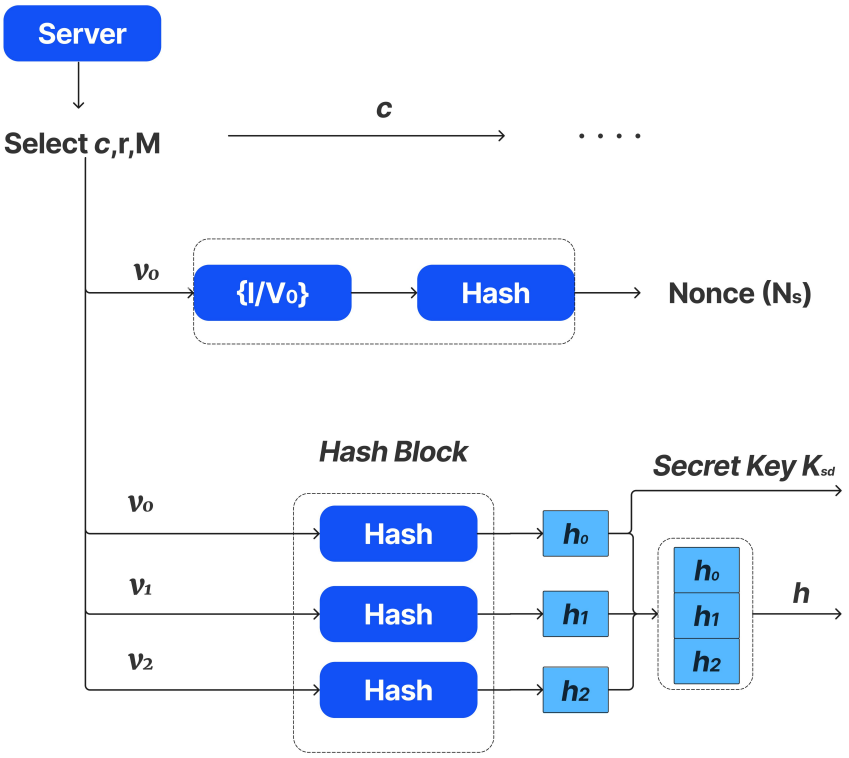

Figure 4.4: Proposed protocol (Pre-authentication phase/Server)

Fig 4.4 shows the pre-authentication process on the server side. The server processes the data sets associated with the edge device to establish a series of hash values through a sequence of hashing functions called "**Hash Block**" applied on the selection vectors $\boldsymbol{v_0}, \boldsymbol{v_1}, \boldsymbol{v_2}$ to generate the hash values h_0, h_1, h_2 respectively.

The server sends the challenge c in the clear to the edge devices along with an encrypted message that includes the server ID and a nonce.

After generating the hash values, the value h_0 will be assigned as a secret key K_{sd}. Also, it combines the values of h_0, h_1, h_2 by **XOR operation** to form authentication hash **h** between server and client.

Eqs. (4.5 - 4.10) summarise the pre-authentication process on the server side:

$$S : h_0 = h(v_0), h_1 = h(v_1), h_2 = h(v_2) \tag{4.5}$$

$$S : K_{sd} = h_0 \tag{4.6}$$

$$S : h = h_0 \oplus h_1 \oplus h_2 \tag{4.7}$$

$$S : m_3 = c \tag{4.8}$$

$$S : m_4 = E_s(K_{sd}, (ID_s||N_{s2})) \tag{4.9}$$

$$S \rightarrow D : Request(ID_s, ID_d, m_3||m_4) \tag{4.10}$$

Operation in Eq. (4.5): the server picks a challenge c, and the corresponding selection matrix process $\mathbf{M}$, then the server processes the selection matrix data by creating three hash values h_0,h_1, and h_2, that correspond to the selection vectors $\boldsymbol{v_0}, \boldsymbol{v_1}, \boldsymbol{v_2}$ respectively.

Operation in Eq. (4.6): the servers generate a symmetric encryption key K_{sd} between the server and edge device, using the hash value h_0.

Operation in Eq. (4.7): the server generates the authentication hash value h by XOR-ing the three hash values h_0,h_1, and h_2.

Operation in Eq. (4.8): the server generates an unencrypted message m_3.

Operation in Eq. (4.9): the server generates an encrypted message m_4 using the generated key in Eq. (4.6) K_{sd}. This cipher includes the server's unique ID and newly generated nonce N_{s2}.

Operation in Eq. (4.10): the server sends the edge device a request to communicate that includes both messages m_3 and m_4.

Edge device (Client)

The client, equipped with PUF, applies the challenge c received from the server to the PUF n times, n is a predefined defined value by the registration authority (RA) for a predefined *confidence level* and *maximum error value* for the authentication process. The client generates n responses for the same challenge received from the server. Then it performs *statistical analysis* on the n responses based on the number of predefined threshold values T_1, T_2, T_3, the statistical analysis is the same process described in pre-deployment phase.

The statistical analysis allows the client to generate the selection matrix $\mathbf{M}$, generate hash values h_0^*, h_1^*, h_2^* for the selection vectors $\boldsymbol{v_0^*}, \boldsymbol{v_1^*}, \boldsymbol{v_2^*}$ respectively.

The client assigns the value h_0^* as a shared secret key K_{sd}^* between the server and client.

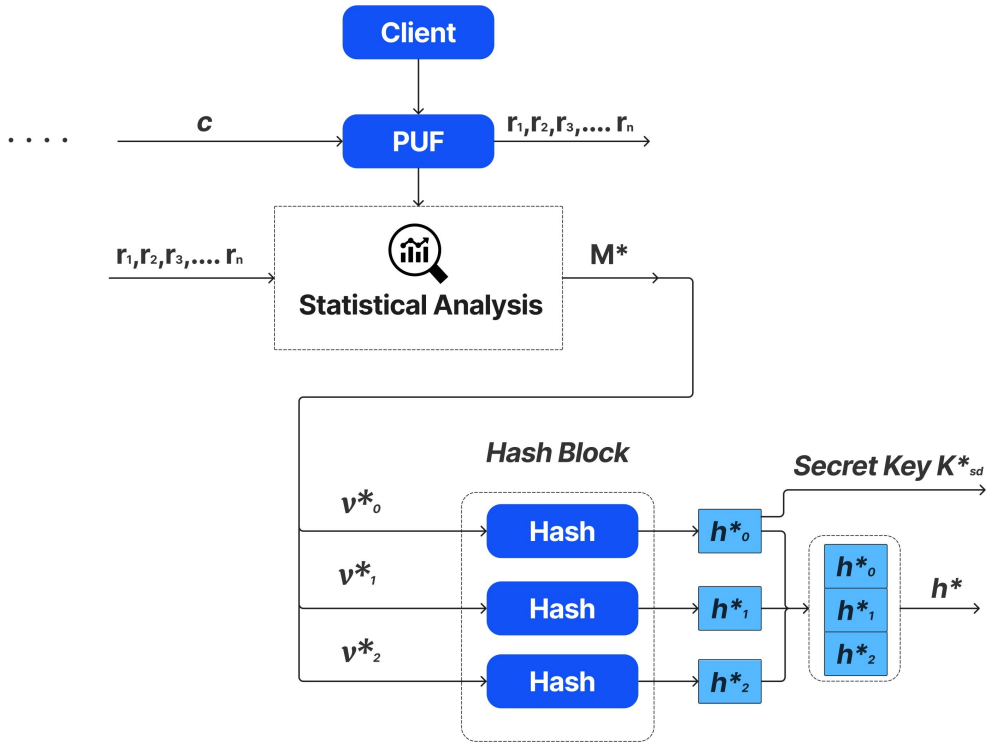

Figure 4.5: Proposed protocol (Pre-authentication phase/client)

Fig.4.5 describes the pre-authentication phase of the proposed protocol on the server side.

The server combines the generated hash blocks h_0^*, h_1^*, h_2^* by **XOR operation** to generate authentication hash value h^*.

Eqs.(4.11 - 4.15) summarises the pre-authentication process at the edge device (client) side:

$$D : M^* = PUF(c, n) \tag{4.11}$$

$$D : h_0^* = h(v_0^*), h_1^* = h(v_1^*), h_2^* = h(v_2^*) \tag{4.12}$$

$$D : K_{sd}^* = h_0^* \tag{4.13}$$

$$D : h^* = h_0^* \oplus h_1^* \oplus h_2^* \tag{4.14}$$

$$D \rightarrow S : E_s(K_{sd}, N_{s2}) \tag{4.15}$$

Operation in Eqs.(4.11): the edge device receives the challenge c, and applies it n times on the PUF to generate its own selection matrix $\mathbf{M}^*$.

Operation in Eq. (4.12): the edge device repeats the same operation explained in Eq.(4.5) to generate another three hash values h_0^*,h_1^*, and h_2^*.

Operation in Eq. (4.13): the edge device generates the symmetric key K_{sd}^*, similar to Eq. (4.6).

Operation in Eq. (4.14): the edge device generates the authentication hash value h^*, similar to Eq. (4.7).

Operation in Eq. (4.15): the edge device responds to the server request by sending an encrypted message that includes the nonce N_{s2} to ensure the freshness of the session.

4.2.5 Authentication

To establish a secure communication protocol between the server and edge device, the implementation of client authentication becomes imperative to verify the identities of edge devices.

Sever and edge device authentication

The following steps outline The authentication process between the server and edge devices:

1. The edge device (client) calculates its dynamic identity DID_d by XOR-ing the client's unique ID with newly generated nonce N_{s3}. This is an important step to mask the client ID and maintain anonymity and intractability:

$$D : DID_d = ID_d \oplus N_{s3} \tag{4.16}$$

2. The client prepares a message DS that includes DID_d concatenated with N_{s3}, this concatenated message is XOR-ed with the authentication hash h^*:

$$D : DS = (DID_d||N_{s3}) \oplus h^* \tag{4.17}$$

3. The client hashes $(DID_d||N_{s3})$ to creates a hash value H_{DS}:

$$D : H_{DS} = h(DID_s||N_{s3}) \tag{4.18}$$

4. The client sends the server a message that includes DS and H_{DS}:

$$D \rightarrow S : (DS, H_{DS}) \tag{4.19}$$

5. The server receives DS^*, then XORs it with h to generates its own values of DID^* and N^*_{s3}:

$$S : (DID_s||N_{s3})^* = DS^* \oplus h \tag{4.20}$$

6. The server calculates the edge device unique identity ID^*_d:

$$S : ID^*_d = DID^*_d \oplus N_{s3} \tag{4.21}$$

7. The server hashes the concatenated message that includes DID^*_d and N^*_{s3} to generate H^*_{DS}:

$$S : H^*_{DS} = h(DID_d||N_{s3})^* \tag{4.22}$$

Then, the server compares between the values ID_d and ID_d^*, H_{DS} and H_{DS}^*, as indicated in Eqs. (3.26 - 3.27):

$$S : ID_d == ID_d^* \tag{4.23}$$

$$S : H_{DS} == H_{DS}^* \tag{4.24}$$

If the said values match, the authentication process is deemed successful.

Fig. 4.6 shows the overall phases of the proposed authentication protocol between the server and edge device (client).

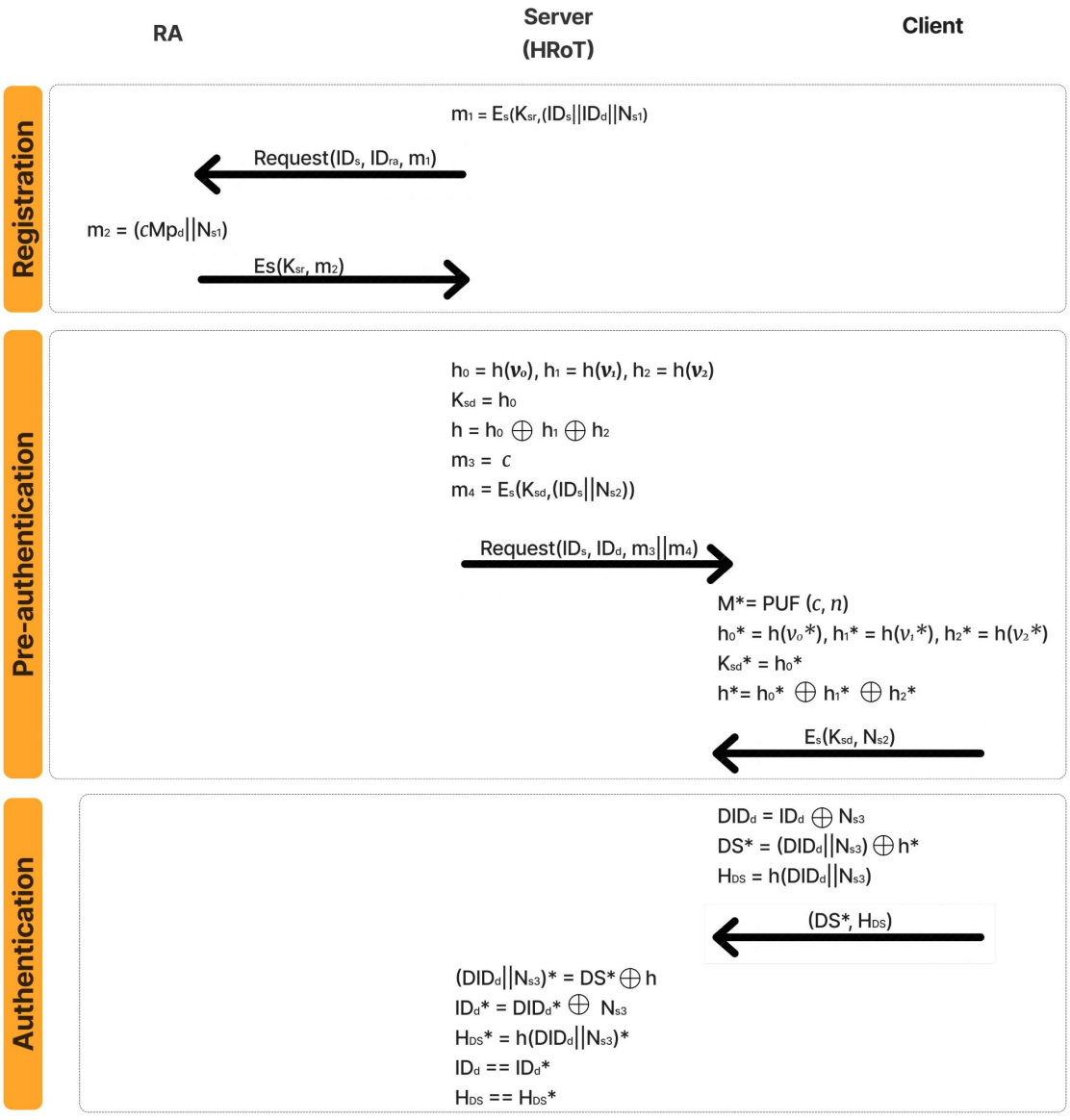

Figure 4.6: Proposed authentication protocol (overview)

Chapter 5

Results and Discussion

This section represents the analysis and assessment of the authentication scheme under consideration. A thorough informal system analysis is conducted to assess the system's robustness against commonly known attacks. Furthermore, The proposed protocol was formally validated by AVISPA which is a widely adopted tool for security system validation security protocol assessments.

5.1 Informal Security Analysis

An informal analysis is conducted to demonstrate that the proposed protocol is secure against various well-known attacks.

5.1.1 Replay attack

The assumed adversary model of the proposed scheme can intercept the transmitted messages over a public channel. However, To provide robustness against reply attacks, a random number is generated which is called nonce upon the initialization of each session, nonce generation ensures the freshness for each communication session and eliminates the risk of reply attacks.

5.1.2 Eavesdropping attack

According to the proposed protocol, the adversary can intercept the messages between the registration authority, server, and client. since all messages are encrypted using the public key infrastructure. The adversary can not access secret information from

any messages because the private keys of each entity secure the sensitive information. Furthermore, a one-way hash function is used to provide additional security for the information being exchanged between the entities. As a result, the attacker would not be able to unfold the sent information, and thus can not obtain any useful data.

5.1.3 Impersonation attack

Impersonation attack Assume Eve (adversary) attempts to impersonate a device. EVE would not succeed because she does not know the unique device ID ID_d, needed for the verification of client identity.

5.1.4 Man-in-the-middle attack

The proposed authentication protocol offers robustness against the Man-in-the-Middle attacks, the transmitted messages are either encrypted by the symmetrical keys or a one-way hash function. Although challenge c is sent in clearly without encryption or hashing, it does not reveal any useful information that the adversary can exploit to launch an attack.

5.1.5 Forward/backward secrecy

Session keys between communicating entities are built using random number generators, session keys used for each session are different. Furthermore, the challenge and symmetric key between server and client K_{sd} will never be repeated in future authentication sessions. Thus if a session key or challenge is compromised by the adversary, the secret information of the past and future authentication sessions can not be compromised. Hence, the proposed protocol is considered immune against forward/backward secrecy.

5.1.6 Session key guessing attack

The session key ensures the freshness of each communication initiated between communicating entities, the session keys are donated by the nonce (N_{s1}, N_{s2}, N_{s3}), they are constructed using random number generators. As a result, the probability of predicting any session key value is exceedingly low, effectively mitigating the risks associated with adversary guessing of the session key.

5.1.7 User anonymity and untraceability

Since device IDs between communicating parties are encrypted using symmetric key infrastructure, the anonymity of all edge devices is protected. Moreover, the session key that is randomly generated doesn't link any sensitive information to a particular edge device. Furthermore, the utilization of dynamic identity during the authentication stage introduces an additional layer of security that enhances User anonymity and untraceability. Thus, the adversary won't be able to trace any authentication session to a specific edge device. anonymity and untraceability are essential security features that shall be maintained in our proposed protocol.

5.1.8 Cloning attack

A cloning attack involves the attacker creating a copy of a physical or digital device, and the cloned device would be utilized to gain unauthorized authentication to a secure system. The edge device in our proposed protocol is equipped with PUF that creates a unique signature or fingerprint of that device which is impossible to clone since it depends on random process variation of the manufactured edge device. Consequently, the proposed scheme is inherently resilient to cloning attacks, as the unique PUF-generated responses make it practically impossible to replicate the edge device.

5.1.9 Physical attack

A physical or tamper attack on a device involves the attacker tampering with the device's hardware components, aiming to obtain the secret keys that are normally stored in non-volatile memory, the IoT devices are vulnerable to physical attacks due to their physical nature which are remotely located. However, the edge devices in our proposed protocol are equipped with PUFs that generate the secret keys during each session depending on the challenge-response pairs that are used to create the selection matrix, this eliminates the necessity to store the secret keys in non-volatile memory and hence creates an integrated system which is immune against physical attacks.

5.2 Formal Security Analysis

5.2.1 Preliminaries

The proposed security protocol went under formal validation using Automated Validation of Internet Security Protocols and Applications (AVISPA). AVISPA is a tool that offers a collection of applications that facilitate the building and analysis of formal models of security protocols, the security protocol models are written in High-Level Protocol Specification Language (HLPSL) [28].

The written code in HLPSL is translated into an Intermediate Format (IF), using a translator, the new format is a lower-level language format that is read directly by back-ends and presented on the AVISPA tool, and then the protocol is analyzed against the security goals to determine whether they are satisfied or violated [28].

AVISPA tool includes four back-end analyzers: OFMC (On-the-fly Model-Checker) that utilize lazy data types to dynamically construct an effective on-the-fly model for security protocols with infinite state spaces [29]. CL-AtSe (Constraint-Logic-based Attack Searcher) The input of (CL-AtSe) is a protocol defined as a collection of rewriting rules (IF format) into In a set of constraints, These constraints facilitate the detection of potential attacks on the security protocol [30]. SATMC (SAT-based Model Checker) generates a propositional formula based on a transitional state obtained from the IF specification. The propositional formula represents any breach of the security properties that can be turned into an attack [31]. TA4SP (Tree Automata-based Protocol Analyser) identifies the vulnerability of the protocol and predicts the correctness measures of the protocol by precisely estimating the attacker's capabilities [32].

5.2.2 Simulation Process

To evaluate and analyze our proposed security scheme in AVISPA, we modeled the proposed protocol using HLPSL, we defined the roles that represent system entities, sessions, the goal, and the environment of our protocol.

We have defined the roles of the main entities involved in our protocol, role(RA) which represents the role of the registration authority, role(S) which is played by the server, role (D) represents the role played by the device equipped with PUF, and role (PUF) which hypothetical role to simulate the functionality of PUF within the device.

In addition, we defined the role session which is a composite role that arranges the interactions between the four individual roles: RA, S, PUF, and D. and the role environment which defines the external environment in which the protocol operates. It includes various parameters, intruder knowledge, and the composition of the protocol session within this environment.

Finally, we had to set security goals that define desired security properties that the protocol should satisfy, such as confidentiality of some parameters such as IDs, IDd, Ns1, and M. And the authentication between server and device.

Fig 5.1 shows the HLPSL code for the definition of the S role played by the server, in this role S recognizes relevant entities (S,D, RA,PUF), the symmetric keys Ksr (between S and RA), hash function, and send/receive channels (SND, RCV), the (dy) notion indicates that the communication channel follows Dolev-Yao model. The protocol simulation starts when S receives the start signal at its initial state 0 the server generates a nonce N_{s1}, sever ID ID_s, and device ID ID_d, the server sends N_{s1}, ID_s, and ID_d after being encrypted to the RA using symmetric key K_{sr}. At state 2 the S receives N_{s1}, ID_s, ID_d,C, M from RA encrypted using symmetric key K_{sr}. At state 2 the S creates an authentication key by hashing M and sends a challenge to the device for authentication. At state 4 the server receives concatenated messages DS that include the dynamic identity of the device DID_d and a nonce, and its hashed value Hds.

Fig 5.2 illustrates the HLPSL code for the definition of RA role played by the registration authority, in this role RA recognizes relevant entities (S,D,RA,PUF), the symmetric keys K_{sr} (between S and RA), K_{rp} (between RA and PUF), hash function, and send/receive channels (SND, RCV), the (dy) notion indicates that the communication channel follows Dolev-Yao model. The role starts from state 1 by receiving the encrypted message N_{s1}, ID_s, ID_d from S using a symmetric key K_{sr}. At the same state, the server sends encrypted message N_{s1}, ID_s, ID_d,C, M to the server using K_{sr}, and another message ID_s, ID_d,C, M to the a the hypothetical entity PUF using the key K_{rp}.

Fig 5.3 shows the HLPSL code for the definition of PUF role played by the PUF, we want to highlight that PUF role in this simulation is a hypothetical role to represent the functionality of the PUF within the device. In this role PUF recognizes relevant entities (S,D,RA,PUF), the symmetric keys K_{rp} (between RA and PUF), K_{dp} (between D and PUF), hash function, and send/receive channels (SND, RCV), the (dy) notion indicates that the communication channel follows Dolev-Yao model.

```
role role_S (S,D,RA,PUF: agent, Ksr : symmetric_key, H: hash_func, SND,RCV : channel (dy))
played_by S
def=
        local
                State:nat,
                IDs,IDd,DIDd,DS,Ns1,Ns3,C,M,X1:text,
                Hds: hash(text),
                Knew : symmetric_key
        init
                State := 0
        transition
                1. State =0 /\ RCV (start) =|> State' := 2 /\ Ns1' := new() /\ IDs' := new() /\ IDd' := new()
                /\ SND({IDs'.IDd'.Ns1'}_Ksr)/\ secret (IDs,sec_IDs,{RA,S})/\ secret (IDd,sec_IDd,{RA,S})
                /\ secret (Ns1,sec_Ns1,{RA,S})

                2. State =2 /\ RCV ({IDs'.IDd'.C'.M'.Ns1'}_Ksr) =|> State' := 4 /\ Knew' := H(M) /\ SND (C')

                3. State =4 /\ RCV (DS', Hds') =|> State' := 6 /\ X1' := xor (DS, Knew) /\
                IDd' := xor (DIDd.Ns3) /\
                witness (S,D, server_device_auth, Hds) /\ witness (S,D, server_device_auth_fresh, IDd)
                %X1 := (DIDs.Ns3)

end role
```

Figure 5.1: Role of S in HLPSL code

The role starts from state 5 by receiving the encrypted message ID_s, ID_d, C, M, from RA using a symmetric key K_{rp}. At state 7, the PUF receives encrypted message C from the device using K_{dp}. At state 13, PUF sends an encrypted message ID_s, ID_d, M to the device using symmetric key K_{dp} (between PUF and D).

```
role role_RA (S,D,RA,PUF: agent, Ksr, Krp : symmetric_key, H: hash_func, SND,RCV : channel (dy))
played_by RA
def=
        local
                State:nat,
                IDs,IDd,Ns1,C,M:text
        init
                State := 1
        transition
                1. State =1 /\ RCV ({IDs'.IDd'.Ns1'}_Ksr) =|> State' := 3
                /\ C':= new() /\ M' :=new()
                /\ SND ({IDs'.IDd'.C'.M'.Ns1'}_Ksr) /\ SND ({IDs'.IDd'.C'.M'}_Krp)
                /\ secret (M,sec_M,{RA,S,PUF}) /\ secret (IDs,sec_IDs,{RA,S,PUF})
                /\ secret (IDd,sec_IDd,{RA,S,PUF})/\ secret (Ns1,sec_Ns1,{RA,S})
end role
```

Figure 5.2: Role of RA in HLPSL code

```
role role_PUF (S,D,RA,PUF :agent, Krp, Kdp : symmetric_key, H: hash_func, SND,RCV: channel(dy))
played_by PUF
def=
        local
                State :nat,
                IDs, IDd, Ns1, C,M : text
        init
                State := 5
        transition
                1. State = 5 /\ RCV ({IDs'.IDd'.C'.M'}_Krp) =|>  State' :=7
                2. State = 7 /\ RCV ({C'}_Kdp) =|> State' := 13 /\ IDs' := new () /\ IDd' := new() /\ M' := new()
                /\ SND ({IDs'.IDd'.M'}_Kdp)/\ secret (M,sec_M ,{PUF,D}) /\ secret (IDs, sec_IDs, {PUF,D})
                /\ secret (IDs, sec_IDs,{PUF,D}) /\ secret (IDd, sec_IDd, {PUF,D})
end role
```

Figure 5.3: Role of PUF in HLPSL code

Fig 5.4 indicates the HLPSL code for the definition of device role played by the device, in this role D recognizes relevant entities (S,D,RA,PUF), the symmetric keys K_{dp} (between device and PUF), hash function, and send/receive channels (SND, RCV), the (dy) notion indicates that the communication channel follows Dolev-Yao model. The role starts from state 20 by receiving a challenge C from the server that needs to authenticate the device. At state 22, the device sends an encrypted message C to the PUF using K_{dp} (between D and PUF). At state 24 the device receives the necessary information ID_s, ID_d, M using symmetric key K_{dp} (between device and PUF. Notably, while PUFs and devices are typically considered as one entity in real-world applications, for our protocol demonstration, we treat them as separate entities to simplify the illustration of the proposed protocol

Fig 5.5 and Fig 5.6 show the HLPSL code for session and environment roles

```
role role_D (S,D, RA, PUF: agent , Kdp: symmetric_key, H: hash_func, SND, RCV: channel (dy))
played_by D
def=
        local
                State: nat,
                IDs, IDd, DIDd,DS ,Ns1,Ns3, C,M,X1 : text,
                Hds: hash(text),
                Knew : symmetric_key
        init
                State := 20
        transition
                1. State =20 /\ RCV (C') =|> State' := 22 /\ SND ({C'}_Kdp)
                2. State =22 /\ RCV ({IDs'.IDd'.M'}_Kdp) =|> State':= 24 /\ Knew' := H(M)
                3. State =24 =|> State':= 30 /\ Ns3' := new() /\ DS':= xor(DIDd.Ns3, Knew)
                /\ DIDd' := xor (IDd, Ns3) /\ Hds' := H(DIDd.Ns3)/\ SND (DS'.Hds')
                /\ request (D,S, server_device_auth, Hds) /\ request (D,S, server_device_auth_fresh, IDd)

end role
```

Figure 5.4: Role of D in HLPSL code

respectively, In the session role, all roles (RA, S, D, PUF) are integrated. On the other side, in the environment role, one or more sessions are initiated. Constants (ra, s, d, puf) are defined to represent the agents (RA, S, D, PUF) respectively. In addition, (ksd, ksr, krp, kdp, knew) denote the symmetric keys shared between S and D, S and RA, RA and PUF,D and PUF, and new authentication key K_{new}, respectively. 'h' symbolizes the hash function H.

The intruder knowledge section, which represents the parameters that the intruder is assumed to possess knowledge of is determined. It is assumed that the intruder is aware of all agents (RA, S, D, PUF), and the challenge which is sent in clear C. Simulation goals are defined using the 'goal' section. Specifically, the focus lies on evaluating the secrecy of the parameters: ID_s, ID_d, N_{s1}, M.

```
role session (RA, S,D, PUF: agent, IDs, IDd: text, Ksr,Ksd,Krp,Ksp,Kdp : symmetric_key, H: hash_func)
def=
        local SND4,RCV4, SND3,RCV3, SND2,RCV2, SND1,RCV1 : channel (dy)
        composition
                role_RA (RA, S, D,PUF, Ksr, Krp, H, SND1, RCV1) /\ role_S (RA, S, D,PUF, Ksr, H, SND2, RCV2)
                /\ role_PUF (RA, S, D,PUF, Krp,Kdp, H, SND3, RCV3)/\
                role_D (RA, S, D,PUF, Kdp, H, SND4, RCV4)

end role
```

Figure 5.5: Session role in HLPSL code

```
role environment ()
def=
const ra,s,d,puf : agent, ksd,ksr,krp,kdp,knew : symmetric_key, h: hash_func, iDs, iDd, c : text,
sec_Ns1, sec_IDs, sec_IDd, sec_M, server_device_auth, server_device_auth_fresh : protocol_id

intruder_knowledge = {ra,s,d,puf,c}
composition
        session (ra,s,d,puf,iDs,iDd, ksd, krp, kdp, ksr, knew, h)
end role

goal
        secrecy_of sec_IDs, sec_IDd, sec_Ns1, sec_M
        authentication_on server_device_auth, server_device_auth_fresh
end goal

environment ()
```

Figure 5.6: Environment role in HLPSL code

5.2.3 Simulation Results

In this section, we present the simulation outcomes of our proposed protocol. Fig 5.7 illustrates the simulation results summary report based on AVISPA backend model checker, On-the-fly Model-Checke (OFMC). Fig 5.8 indicates the Security Protocol Animator (SPAN) which facilitates the interactive creation of a message sequence chart (MSC) based on the HLPSL specification protocol described earlier. Additionally, SPAN automatically generates potential attacks employing the Dolev-Yao intruder model. it demonstrates the exchange of messages among all agents during the registration and authentication phases.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/PUF_proto_latest.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.07s
  visitedNodes: 16 nodes
  depth: 7 plies
```

Figure 5.7: Proposed Protocol Simulation Summary Report

Based on the described simulation summary report, the AVISPA tool confirmed the security of the proposed scheme, thus ensuring its immunity against the defined potential security threats.

5.3 Performance Analysis

In this section, we evaluate the performance of the proposed protocol in aspects of storage and computation cost.

5.3.1 Storage Cost

We determine the storage cost of the proposed scheme (in bits) for the main communicating entities (server (S) and device (D).

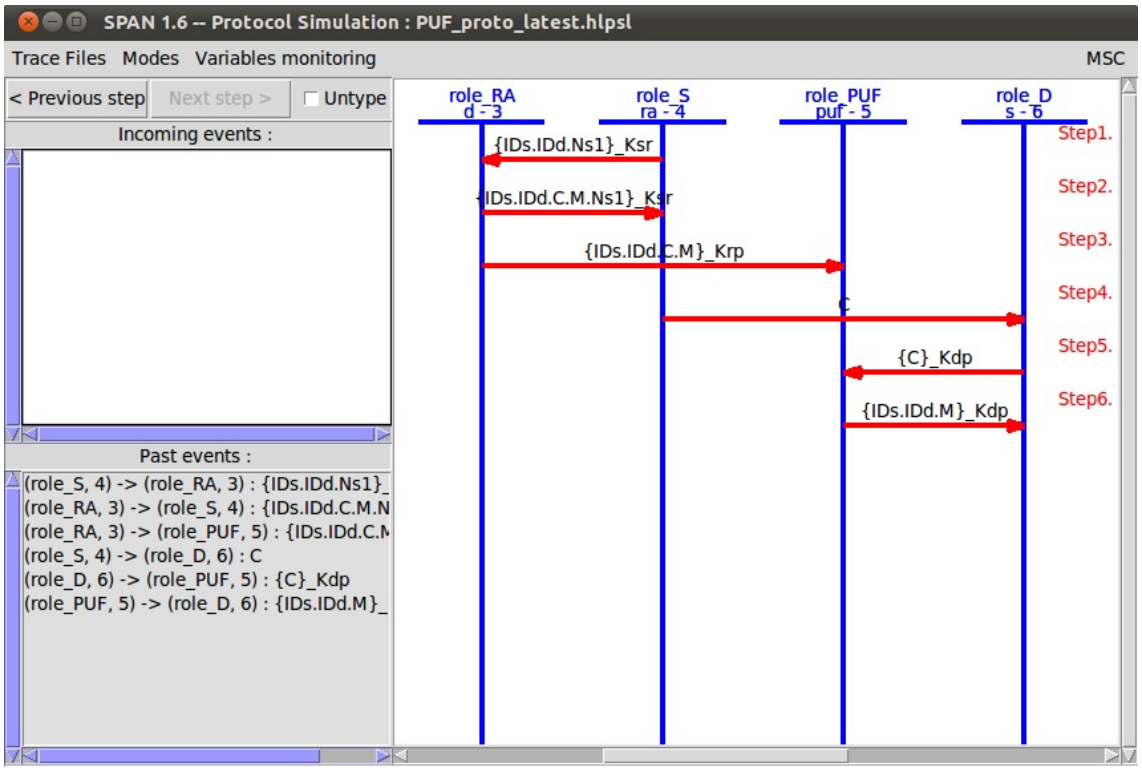

Figure 5.8: Proposed protocol simulation using AVISPA

The server (S) stores ID_s, ID_d, c, h_1, h_2, and h_3. by considering SHA3-384 as a hash function, the output of each hash function is 384 bits, while $ID_s = ID_d = c =$ 128 bits, thus the total storage cost required by the server is 3x384 + 3x128 = 1536 bits.

The device (D) stores ID_d, c, h_1^*, h_2^*, and h_3^*. by considering SHA3-384 as a hash function, the output of each hash function is 384 bits, while $ID_d = c = 128$ bits, thus the total storage cost required by the device is 3x384 + 2x128 = 1408 bits.

Hence, the total storage cost required by the proposed scheme is 2944 bits.

5.3.2 Computational Cost

In this subsection, we present the computational expenditure of the proposed protocol. We have referenced experimental data documented in [33, 34] to ensure an accurate computation cost comparison of our proposed scheme. The parameters considered to calculate the computational cost are the computational time for modular exponentiation operation denoted as T_{exp}, hash function denoted as T_{hash}, and symmetric key cryptography denoted as T_{sym}, modular exponentiation denoted as T_{me}.

Table 5.1 illustrates the time-required operations above. To estimate the computational time required in our proposed scheme, we have performed 8 hash operations,8 XOR operations, and 2 symmetric encryption operations. Due to the minor time delays for XOR operations, the required time for XOR operations is considered negligible, which yields a total computation time (8x T_{hash}) + (2 x T_{sym}). Hence the total time required is (8x0.32) + (2x5.6) = 13.76 ms.

Table 5.1: Crypto-operations and the required computational times

Crypto-operations	**Computational time**
Modular exponentiation operation T_{exp}	19.2 ms
Hash function T_{hash}	0.32 ms
Symmetric key cryptography T_{sym}	5.6 ms
Modular exponentiation T_{me}	17.1 ms

Table 5.2 shows a computation cost comparison between our proposed protocol and other state-of-the-art related security protocols. As previously noted, the computational costs, quantified as estimated time, are derived from calculated based on referenced experimental data documented in [33, 34].

Table 5.2: Computation cost comparison between the proposed protocol and other state-of-the-art related schemes in ms

Reference	**Estimated time(ms)**
Wazid et al. [35]	46.54
Shuai et al.[36]	162.72
Santoso et al.[37]	58.24
Kim et al.[38]	26.40
Alzahrani et al. [39]	52.00
Chen et al.[40]	841.8
Our proposed protocol	13.76

Table 5.2 compares security and functional features between our proposed and state-of-the-art security protocols. In conclusion, we examined that our proposed protocol provides significant performance enhancement in security and functionality against other related protocols.

Table 5.3: Security and functionality features comparison

Functionality Feature	**[35]**	**[36]**	**[37]**	**[38]**	**[39]**	**[40]**	**Our Protocol**
Mutual authentication	Yes	Yes	No	No	No	Yes	Yes
Session key agreement	Yes	Yes	No	No	Yes	Yes	Yes
User anonymity	Yes	Yes	No	No	No	Yes	Yes
Untraceability	Yes	Yes	No	No	No	Yes	Yes
Forward security	No	Yes	No	No	No	Yes	Yes
Privileged insider attack	Yes	Yes	Yes	No	Yes	Yes	Yes
Impersonation attack	Yes	Yes	No	No	No	Yes	Yes
Replay attack	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Man-in-the-middle attack	Yes	Yes	No	No	No	Yes	Yes
Formal verification (AVISPA)	Yes	No	No	No	No	No	Yes

Chapter 6

Conclusion and Future work

6.1 Conclusion

The number of IoT devices is rapidly increasing globally [41], this anticipated fast growth of IoT devices has led to serious threats for network security [42]. PUF-based hardware security is a promising lightweight security technique [43]. In our research work, we have addressed the challenges related to thermal noise associated with the PUF responses, which may lead to authentication failure. We have introduced an innovative approach that overcomes the challenges of error correction techniques designed to eliminate the noise of PUF responses. In our thesis, we have introduced a new approach that exploited the PUF noisy response to extract unique features for each PUF, a synthetic model of SRAM PUF was constructed using the Python programming language to perform statistical analysis of SRAM PUF responses, and the unique features of the SRAM PUF responses are extracted to construct its selection matrix, the SRAM PUF challenge and selection matrix pairs were created to form a unique reliable fingerprint that can be used for authentication and secret key generation using what hash chain method. Based on this approach, we have introduced a security protocol that provides secure authentication and secret key change between the main IoT entities (server, client), we have discussed in this work the three phases of our proposed protocol and the main entities involved in each phase. We have conducted an informal analysis to demonstrate that the proposed protocol is secure against well-known attacks. Then a formal analysis was conducted using the AVISPA tool that confirmed its immunity against the defined potential security threats. We have done a performance analysis of the proposed scheme in terms of

storage and computation cost, Performance analysis revealed that our scheme offers significant improvements in security and functionality compared to existing protocols.

6.2 Future work

Our potential future approaches include the following:

- Implementing the proposed security on hardware SARM PUF and performing statistical analysis for the PUF responses.
- Investigating the feasibility of using machine learning techniques and artificial intelligence to extract the unique features of PUF responses.
- Exploring the use of blockchain techniques for selection vectors generation which provides additional security for robustness for authentication.
- conducting formal analysis using a widely accepted model called BAN logic to verify the security and integrity of our proposed protocol.
- Examining potential IoT applications for implementing the proposed protocol, such as in healthcare and defence sectors.

References

[1] M. Fakroon, F. Gebali, and M. Mamun, "Multifactor authentication scheme using physically unclonable functions," *Internet of Things*, vol. 13, p. 100343, 2021.

[2] F. Gebali and M. Mamun, "Review of physically unclonable functions (pufs): structures, models, and algorithms," *Frontiers in Sensors*, vol. 2, p. 751748, 2022.

[3] Statista, "Number of Internet of Things (iot) connected devices worldwide in 2025 and 2030 (in billions)." `https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/`. Accessed: March 25, 2024.

[4] M. Al Ameen, J. Liu, and K. Kwak, "Security and privacy issues in wireless sensor networks for healthcare applications," *Journal of medical systems*, vol. 36, pp. 93–101, 2012.

[5] X. Liang and Y. Kim, "A survey on security attacks and solutions in the iot network," in *2021 IEEE 11th annual computing and communication workshop and conference (CCWC)*, pp. 0853–0859, IEEE, 2021.

[6] J. Delvaux and I. Verbauwhede, "Key-recovery attacks on various ro puf constructions via helper data manipulation," in *2014 Design, Automation and Test in Europe Conference and Exhibition (24-28 March 2014)*, pp. 1–6, 2014.

[7] R. Klöti, V. Kotronis, and P. Smith, "Openflow: A security analysis," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–6, IEEE, 2013.

[8] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual design automation conference*, pp. 9–14, 2007.

[9] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 148–160, 2002.

[10] E. Öztürk, G. Hammouri, and B. Sunar, "Towards robust low cost authentication for pervasive devices," in *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 170–178, IEEE, 2008.

[11] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[12] M. Fakroon, M. Alshahrani, F. Gebali, and I. Traore, "Secure remote anonymous user authentication scheme for smart home environment," *Internet of Things*, vol. 9, p. 100158, 2020.

[13] Y. Zheng, W. Liu, C. Gu, and C.-H. Chang, "Puf-based mutual authentication and key exchange protocol for peer-to-peer iot applications," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[14] X. Shao, Y. Guo, and Y. Guo, "A puf-based anonymous authentication protocol for wireless medical sensor networks," *Wireless Networks*, vol. 28, no. 8, pp. 3753–3770, 2022.

[15] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pp. 523–540, Springer, 2004.

[16] Z. Paral and S. Devadas, "Reliable and efficient puf-based key generation using pattern matching," in *2011 IEEE international symposium on hardware-oriented security and trust*, pp. 128–133, IEEE, 2011.

[17] I. U. Din, M. Guizani, S. Hassan, B.-S. Kim, M. K. Khan, M. Atiquzzaman, and S. H. Ahmed, "The internet of things: A review of enabled technologies and future challenges," *Ieee Access*, vol. 7, pp. 7606–7640, 2018.

[18] J. A. Manrique, J. S. Rueda-Rueda, and J. M. Portocarrero, "Contrasting internet of things and wireless sensor network from a conceptual overview," in *2016 IEEE international conference on Internet of Things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, pp. 252–257, IEEE, 2016.

[19] N. Saputro, A. I. Yurekli, K. Akkaya, and S. Uluagac, "Privacy preservation for iot used in smart buildings," *Security and Privacy in Internet of Things (IoTs): Models, Algorithms, and Implementations*, pp. 129–160, 2016.

[20] M. Fakroon, "Secure authentication schemes for internet of things (iot)," 2021.

[21] U. Rührmair, S. Devadas, and F. Koushanfar, "Security based on physical unclonability and disorder," in *Introduction to Hardware Security and Trust*, pp. 65–102, Springer, 2011.

[22] R. Maes, *Physically Unclonable Functions: Constructions, Properties and Applications*. Springer Science & Business Media, 2013.

[23] F. Gebali and M. Mamun, "SRAM physically unclonable functions for smart home iot telehealth environments," *Cybersecurity in Smart Homes: Architectures, Solutions and Technologies*, pp. 125–153, 2022.

[24] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9*, pp. 63–80, Springer, 2007.

[25] Y. Su, J. Holleman, and B. P. Otis, "A digital 1.6 pj/bit chip identification circuit using process variations," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 69–77, 2008.

[26] Python Software Foundation, *Python 3.9 Documentation*. Python Software Foundation, 2020.

[27] W. C. Navidi, *Statistics for engineers and scientists*, vol. 3. McGraw-Hill New York, 2010.

[28] "AVISPA, Automated validation of internet security protocols and applications." `http://www.avispa-project.org`.

[29] D. Basin, S. Mödersheim, and L. Vigano, "Ofmc: A symbolic model checker for security protocols," *International Journal of Information Security*, vol. 4, pp. 181–208, 2005.

[30] M. Turuani, "The cl-atse protocol analyser," in *International conference on rewriting techniques and applications*, pp. 277–286, Springer, 2006.

[31] A. Armando and L. Compagna, "Satmc: a sat-based model checker for security protocols," in *Logics in Artificial Intelligence: 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004. Proceedings 9*, pp. 730–733, Springer, 2004.

[32] Y. Boichut, P.-C. Héam, O. Kouchnarenko, and F. Oehl, "Improvements on the genet and klay technique to automatically verify security protocols," in *Proc. AVIS*, vol. 4, p. 84, 2004.

[33] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "Iot security: ongoing challenges and research opportunities," in *2014 IEEE 7th international conference on service-oriented computing and applications*, pp. 230–234, IEEE, 2014.

[34] D. He, N. Kumar, J.-H. Lee, and R. S. Sherratt, "Enhanced three-factor security protocol for consumer usb mass storage devices," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 1, pp. 30–37, 2014.

[35] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo, "Secure remote user authenticated key establishment protocol for smart home environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 391–406, 2017.

[36] M. Shuai, N. Yu, H. Wang, and L. Xiong, "Anonymous authentication scheme for smart home environment with provable security," *Computers & Security*, vol. 86, pp. 132–146, 2019.

[37] F. K. Santoso and N. C. Vun, "Securing iot for smart home system," in *2015 international symposium on consumer electronics (ISCE)*, pp. 1–2, IEEE, 2015.

[38] H. J. Kim and H. S. Kim, "Authhotp-hotp based authentication scheme over home network environment," in *International Conference on Computational Science and Its Applications*, pp. 622–637, Springer, 2011.

[39] B. A. Alzahrani, "Secure and efficient cloud-based iot authenticated key agreement scheme for e-health wireless sensor networks," *Arabian Journal for Science and Engineering*, vol. 46, no. 4, pp. 3017–3032, 2021.

[40] C.-L. Chen, T.-T. Yang, and T.-F. Shih, "A secure medical data exchange protocol based on cloud environment," *Journal of medical systems*, vol. 38, pp. 1–12, 2014.

[41] IoT Analytics, "Number of connected Internet of Things (IoT) devices." `https://iot-analytics.com/number-connected-iot-devices/`. Accessed: March 25, 2024.

[42] A. Giaretta, S. Balasubramaniam, and M. Conti, "Security vulnerabilities and countermeasures for target localization in bio-nanothings communication networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 665–676, 2015.

[43] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure pufs," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, pp. 670–673, IEEE, 2008.