P/KEY: PUF BASED SECOND FACTOR AUTHENTICATION

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Ertan UYSAL

April 2022
İZMİR

ACKNOWLEDGMENTS

I would like to earnestly acknowledge the sincere efforts and valuable time given by my respected supervisors Asst. Prof. Mete AKGÜN and Asst. Prof. Serap ŞAHİN. Without their guidance and involvement in every step of the process, I could never accomplish this. I am thankful for your guidance, support, encouragement.

I would like to thank my thesis committee members Prof. Fatih ALAGÖZ, Prof. Mehmet Ufuk ÇAĞLAYAN and Asst. Prof. Nesli ERDOĞMUŞ for their valuable insight and suggestions.

I would also like to give special thanks to my family, Hülya UYSAL, Melek Selin UYSAL, Melek KADIOĞLU, Ebru DEMİR and Raşit KADIOĞLU. Like everything I have achieved in life, I owe this thesis to their unconditional love and support.

I would like to thank my friends Ece, Aleyna, Gazi their motivation during this period, and also to my classmates Elman, Kıvanç, Mücahit and Utku. Together we had overcome many difficulties.

Finally, I would like to thank my manager Necmi TÜRER and Bosch Group to support to do master's degree in working life.

ABSTRACT

P/KEY: PUF BASED SECOND FACTOR AUTHENTICATION

Second-factor authentication mechanisms increase the security of authentication processes by implementing an additional auxiliary layer to a single factor. As a second factor, using one-time passwords (OTP) is mainly preferred due to their hardware independence and easy generation. OTP generation protocols should be evaluated in two main categories: time and security. In time-based OTP mechanisms (TOTP), client and server store a shared secret key. However, if attackers compromise the server, attackers can generate new OTPs using the key and impersonate the client. To solve this problem, protocols based on the hash chain mechanism have been proposed; however, these methods have weaknesses mainly due to the authentication speed and the limited number of OTPs they generate. This thesis proposes a server-side tamper-proof and fast response physical unclonable function (PUF) based second-factor authentication protocol on overcoming these problems. PUF is a digital fingerprint that ensures that every device produced is unique due to uncontrollable factors in the production stages of devices. It generates responses that correspond to challenges. Since PUF is based on the micro-level differences in devices, micro-level structure changes in the event of an attack, and the PUF takes to generate different responses. Although PUF is a fast response function, it is impossible to reach the challenge from the response it generates. In the proposed protocol, the PUF inside the server generates key values and used to store clients' secret seed values securely. In case of side-channel attack on server-side, the key values of the clients cannot be obtained by the attackers, as the PUF structure will be corrupted. Even if the attacker obtains the server's credentials and gains access to the system, they cannot get the secret seed values of the clients and cannot generate the OTPs. In this way, the attacker cannot authenticate by impersonating the client.

ÖZET

P/ANAHTAR: PUF TABANLI İKİNCİ FAKTÖR KİMLİK DOĞRULAMA

Kimlik doğrulama işleminin güvenliğini arttırmak amacıyla tek faktöre yardımcı ek bir katman eklenerek iki faktörlü kimlik doğrulama mekanizmaları kullanılmaktadır. İkinci faktör olarak tek kullanımlık şifre (OTP) kullanımı donanım bağımsız, kolay üretilebilmesinden kaynaklı tercih edilir. OTP üreten protokolleri temel olarak iki kategoride değerlendirmek gerekir: zaman maliyeti ve güvenlik. Zamana dayalı OTP mekanizmalarında (TOTP) istemci ve sunucu ortak bir anahtar değer saklar. Ancak sunucu saldırganlar tarafından ele geçirilirse, saldırgan anahtar bilgisini kullanarak yeni OTP üretebilir ve istemci gibi davranıp sisteme erişim sağlayabilir. Bu sorunu çözmek amacıyla hash zinciri mekanizmasına dayalı çeşitli yöntemler önerilmiştir ancak bu yöntemler temel olarak kimlik doğrulama hızı ve limitli sayıda OTP ürettiklerinden dolayı zafiyetler içerir. Bu tez çalışmasında bu sorunların üstesinden gelmek amacıyla, sunucu tarafı saldırıya karşı güvenli ve hızlı yanıt veren fiziksel klonlanmayan fonksiyonlara (PUF) dayalı iki faktörlü kimlik doğrulama mekanizması öne sürüyoruz. PUF, cihazların üretim aşamalarındaki kontrol edilemeyen faktörler sebebiyle üretilen her cihazın benzersiz olmasını sağlayan dijital parmak izidir. Meydan okuma değerlerine karşı, cevaplar üretir. PUF yapısı cihazlardaki mikro seviyedeki farklılıklardan kaynaklandığı için saldırı durumunda mikro seviyede yapı değişir ve PUF farklı cevaplar üretmeye başlar. PUF hızlı cevap üreten bir fonksiyon olmasına karşı, ürettiği yanıttan girdi değerine ulaşmak imkansızdır. Önerilen protokolde, sunucunun içindeki PUF, anahtar değerleri üretir ve ayrıca istemcilerin gizli tohum değerlerini güvenli bir şekilde saklamak için kullanılır. Sunucu tarafında yan kanal saldırısı olması durumunda, PUF yapısı bozulacağı için istemcilerin anahtar değerleri saldırganlar tarafından elde edilemez. Saldırgan, sunucunun kimlik bilgilerini alıp sisteme erişim sağlasa bile, istemcilerin gizli tohum değerlerini alamaz ve bu değerleri kullanarak OTP oluşturamaz. Bu sayede, saldırgan istemcinin kimliğine bürünerek kimlik doğrulaması yapamaz.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

LIST OF ABBREVIATION

SFA	Single Factor Authentication
2FA	Two-factor Authentication
MFA	Multi Factor Authentication
OTP	One-Time Password
PUF	Physical Unclonable Function
P/Key	PUF-based Second Factor Authenticatio
TOTP	Time-based One-time Password
HOTP	HMAC-based One-time Password
HMAC	Hash Message Authentication Code
S/KEY	S/Key Second Factor Authentication Mechanism
T/KEY	T/Key: Second-Factor Authentication From Secure Hash Chains
DUO	Google DUO

CHAPTER 1

INTRODUCTION

Authentication is the process of proving itself when a person or program wants to access a system [36]. Authentication system checks whether the person is authorized by looking at the authorized entity list. It is required on systems that are not public contain personal or group-specific information. Authentication is meaningless without identifying the user beforehand to the system. In initialization step, the authentication server registers a client and authorizes it to access the system. Authentication mechanisms can be examined in two different factor groups: Single Factor Authentication (SFA) and multi-factor authentication (MFA). SFA systems are the simplest form of authentication mechanisms [18]. In SFA, the client authenticates to the system by entering only one secret. Authentication is generally provided with a username and password. If a user has been granted access to the system beforehand, authentication is provided. SFAs are preferred because of their easy deployment [34, 29]. One-time password or facial recognition system can also be used as a single factor for authentication [9]. MFAS, which is generally used in systems with advanced security measures, aims to increase security by using various forms of authentication factors in addition to a single factor when the user logs into the system [57, 10, 7, 19]. Other factors that are independent of the first factor can be provided by biometric features, OTP, security questions, or dedicated hardware. Secret keys must be stored securely on the server in both factor groups. Otherwise, the attacker can obtain the keys and log into the system.

1.1. Motivation

Second-factor authentication which is a sub-branch of MFAs were proposed to overcome the security weaknesses of SFAs [58, 53, 63]. Second factor authentication mechanism can be provided in many ways. Password-face recognition, password-OTP

pairs can be one of these methods. Applications such as Google Authenticator [1] and DUO [2] are also applications that generate time-based OTPs and provide a second factor authentication mechanism that works integrated with many different programs. The success criterion of second factor authentication mechanisms can be evaluated with two concepts: time cost and security. It is significant that the authentication request is completed quickly, and that the confidential information of the client is kept securely. In TOTP based second factor mechanisms, secret key values are kept on the server [44, 60, 52, 33]. When authentication is required, the values sent by the client are verified with the information stored on the server. However, If the attacker can access the server, they get credentials and can act as a client and authenticate the system. This risk has been encountered not only theoretically but also in practice. RSA and Linode companies got hacked. One time password secret keys from server stolen [68]. Also, this mechanism is not resistant to side-channel attacks. Since the direct storage of the keys on the server will pose a security risk [68], hash-chain based mechanisms have been proposed [23, 33]. However, the number of OTP produced in these mechanisms is limited, after the OTPs are depleted, re-initialization is required thus OTPs are generated again. The number of OTPs produced depends on the length of the hash chain. If the hash chain is kept long, the number of OTPs will increase. Because each hash output value is an OTP. However, in this case, the verification time is delayed.

Based on these problems, in this thesis, the answer to the following question was reviewed:

- *Can we design a protocol that is resistant to side-channel attacks on the server side, respond quickly to authentication requests, and the valuable information required for authentication cannot be retrieved from the server by the attackers even if the server is compromised?*

1.2. Thesis Definition and Contributions

This thesis proposes a server-side tamper-resistant second factor authentication mechanism based on physical unclonable function (PUF). When the authentication request is received, the keys are generated within PUF in the server, it means that the client registers with the server. The PUF generates the response using the secret seed

challenge value that the client sends to the server, and this response is the secret value for the client. When client authentication is required, it generates one-time passwords using its secrets and time information and transmits them to the server. The server verifies the client's one-time passwords by generating the client's secrets again with the help of the PUF.

In the proposed protocol, as in TOTP mechanism, time information is used in OTPs, but unlike TOTP, clients' secrets are not stored on the server. The important point in this regard is that the client's secrets are generated on the server side with the help of PUF in the initialization part, and after they are transmitted to the client, they are deleted. In the authentication phase, the server regenerates the client's secrets using the PUF. Since the authentication part is based on the PUF mechanism on the server and PUFs are resistant to tampering and produce different results than in the case of an attack, the proposed protocol is resistant to side-channel attacks. Furthermore, because the secrets of clients are stored in the server's database in the hidden format, the proposed protocol is also resistant to server-side compromise.

Accordingly, the contributions of the study were expressed as follows:

- *Server-side tamper-proof second factor authentication mechanism:* In the proposed protocol, the verification process is done by the server with help of PUF. For this reason, in case of a side channel attack on the server, the PUF characteristic will change, therefore PUF will behave differently and generate different secrets than before. If attackers have these secrets, they cannot impersonate client and authenticate to the system.
- *Quick authentication response time:* Secret generation in the protocol is done with the help of PUF. It is not iterative and time-consuming as in the hash chain mechanism. Since PUFs produce fast responses against input challenge values, secret generation and authentication time is short.

- *Secrets are not stored on server-side*: Secrets are not stored on the server. The server generates secrets to verify client's authentication request and deletes them after use. Only seed values are stored in the server with hidden form thanks to XOR operation. It does not make sense for the attacker to obtain these hidden values in case of log in the system. Because attacker cannot extract the secrets of the clients.

CHAPTER 2

RELATED WORK

In this section, examined the second factor one-time password mechanisms and PUF based authentication protocols.

In one-time password (OTP) mechanisms, the password is generated based on seed and moving factors [45]. Moving factor in HMAC one-time password (HOTP) is counter value [43]. Counter value enables different OTPs to be produced within the HMAC function with a fixed seed value. When authentication occurs, the counter value is incremented by one on the server and client sides in the HOTP mechanism [43]. In this way, it is ensured that both parties generate the same hash values. However, in cases where the client does not send the generated password to the server, there may be a synchronization problem between the client and server counter values. Although the server checks the passwords sent by the client with more than one hash value in the window range, in case of synchronization problems, re-initialization is required between the client and server. Brute force attack poses a threat in HOTP mechanisms where the window size is in a wide range [45]. Time is the moving factor in the time-based one-time password (TOTP) mechanism [44]. Passwords are generated based on the time elapsed from the timestamp agreed by the client and server. Since the client and server generate a password using a shared key, OTP can be generated if unauthorized people obtain the key. S/Key protocol finds a solution to the problem of key storage on the server [23]. It hides the keys by storing the hash of the OTP sent by the client to the server [23]. However, the protocol is not time-based, and OTP is not renewed unless there is an authentication request. T/Key protocol proposed by Kogan et al. is a combination of TOTP and S/Key [33] [44] [23]. It aims to create an extra layer of security by adding time information to OTP as an additional feature to S/Key. Nevertheless, the long hash chain mechanism in this protocol causes computational costs [33]. Since the proposed P/Key model produces time-based passwords as in TOTP and T/Key mechanisms, the produced OTPs do not remain valid for a long time. Also, in our proposed P/Key protocol, keys are not directly stored on the server-side. Therefore, the protocol proposed in the thesis is

based on TOTP, S/Key, and T/Key mechanisms. A detailed description of the three models is given in Section 2.1, 2.2, 2.3.

In addition to these, Bıcakci and Baykal proposed an OTP mechanism based on asymmetric cryptography [11]. However, key generation and verification computational costs are higher than other protocols [6]. Many studies have been conducted on the use of PUF for the authentication of IoT applications [67][62] [14]. One of the significant reason for this is that there is no key storage requirement. Yoon et al. proposed an authentication mechanism based on the use of PUF on the client-side and the server sending the challenge value from the CRP database to the client when an authentication request is received [67]. Unlike P/Key, the PUF mechanism is used on the client-side. One of the most important reasons for this is that IOT client devices may be vulnerable to tampering and may carry the risk of client compromising. Wallrabenstein presented a low-cost tamper resistance authentication protocol using PUF in 2006 [62]. However, this protocol does not generate OTP and is not a second-factor mechanism.

2.1. TOTP: Time Based One Time Password Protocol [44]

Time-based one-time password scheme [44] is a protocol that allows one-time use and generates a time-based password that is valid for certain periods such as thirty seconds or one minute. This scheme is executed over the shared secret key stored on both server and client. Keys are generated by performing the HMAC operation between the shared secret key and time information both server and client. As time information, the period information from UNIX time until now is used.

UNIX time is the number of seconds that have passed since January 1, 1970. The number of seconds that have passed since this date is divided into periods. Password that created for each period is only valid for that period. TOTP is also used in software-based one-time passwords 2FA applications such as Google Authenticator [1].

2.1.1. Implementation

2.1.1.1. Initialization

1. Convert date and time to Unix Epoch Time.
2. Calculate $N = floor(T/t_s)$

   N=Total number of time steps in UNIX time

   t_s= time step.
3. Convert N to hexadecimal format.
4. Convert N into 8-byte array format and assign value to message m
5. Convert to secret key to 20 bytes array and assign the value to k
6. Calculate the HMAC using k and m

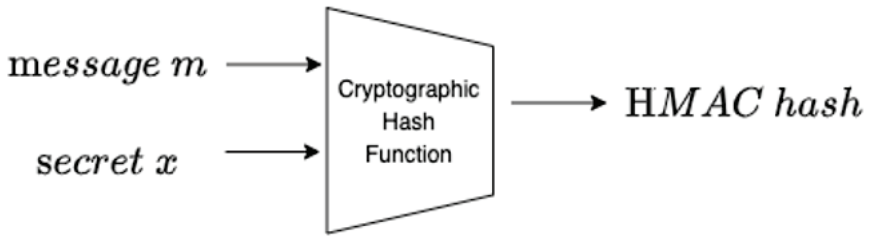

Figure 2.1. HMAC Hash Function

7. Get last four bits integer value of HMAC, this shows offset.
8. Get the first 4 bytes starting from the offset value of the HMAC hash.
9. Perform binary operation for each byte.
10. Convert binary value to integer $\acute{\imath}$
11. Select token size n

12. Token $t = i \% 10n$

13. if $t < n$, add prefix.

2.1.1.2. Authentication

Server and client create password using the above algorithm in 2.1.1.1 using the shared secret key at the same time period. The client sends the generated OTP to the server. If the OTP the server receives from the client is the same as the OTP it generates, server verifies the authentication request.

2.1.2. Protocol Review

- In the TOTP structure, shared secret key is stored on server. If the attackers compromise the server, keys can be obtained, and valid tokens can be generated. The attacker who obtains the key can impersonate the client and authenticate the system.
- As advantages of TOTP, there is no need for hardware as it generates software-based passwords. This provides benefits both in terms of cost and ease of use. If desired, it can be also integrated into special hardware.

2.2. S/Key Second Factor Authentication Mechanism [23]

For one-way hash function, it is easy to compute hashed value from left to right, however it is hard to return from the hash value to the original value. S/Key generates password based on hash chain structure and each password can only be used once [23]. Each hashing generates a password, and the passwords become invalid after it is used. Hash function takes the secret key and the salt value as input.

2.2.1. Implementation

2.2.1.1. Initialization

1. The hash function produces a hash value by taking the secret value and the salt value as input in client-side.
2. The result of hash value is re-entered into the hash function, and a new hash value is created.
3. The hashing process continues for the length of the hash chain.
4. Each hash output is also one-time password value.

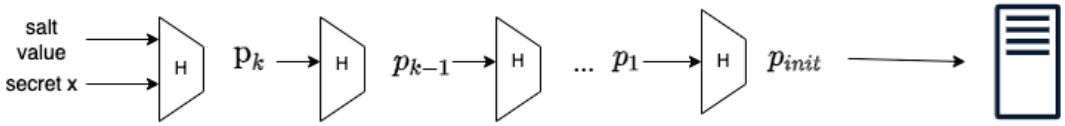

Figure 2.2. S/Key Password Generation Schema

2.2.1.2. Authentication

1. Clients compute $p_k, p_{(k-1)},,\ldots,p_1,\ p_{init}$ Then:
2. Initially, client sends p_{init} to server for storage.
3. When authentication is needed, the client uses p_1 as the first OTP, and sends it to the server for verification.
4. Server gets the p_1 then server hashes the p_1.
5. If the $H\,(p_1) = p_{init}$, then authentication is provided.
6. Server update $p_{init},$ value as p_1 for next authentication.

2.2.2. Protocol Review

- S/Key scheme is designed for a small number of login operations. The number of generated passwords is directly proportional to the length of the hash chain. However, it is not clear how the length of the hash chain will affect security [33]. Re-initialization is needed when the passwords created with the hash function are depleted. • Since the same hash function is used in each iteration, if the hash function is known, passwords can be generated by unauthorized people.

- In this protocol, the one-time-password mechanism is not time-based. For this reason, in case of no authentication for a long time, the same password can be stored on the server for a long time.

- If there is a distributed server structure, passwords must be updated in a coordinated and secure manner on each server. In this way, all servers agree on the same response [33].

- Passwords are not stored directly on the server. Since the hashed value is stored on the server, it does not make sense for the attacker to obtain this hashed value.

2.3. T/Key Second Factor Authentication with Hash Chains [33]

T/Key is the second factor authentication mechanism based on S/Key and TOTP mechanisms [33]. As in S/Key, T/Key is based on the creation of OTP by hash chain mechanism [23]. However, there is no time-based token creation in the S Key mechanism. T/Key uses time information in the hash chain mechanism, similar to TOTP [33] [23].

2.3.1. Implementation

In the T/Key mechanism, there is no key value stored on the server [33]. The server contains the previous authentication value, salt value and the last authentication time information.

2.3.1.1. Initialization

- Client chooses a secret x salt value and expiration time that can be four years from now. After expiration time expires, reinitialization required.
- Client traverses hash chain. Each element of hash chain is OTP. To hash, client uses time value for each operation like TOTP [44].
- Final element of chain that called p_{init} sends to server with salt and initial time values for first authentication.

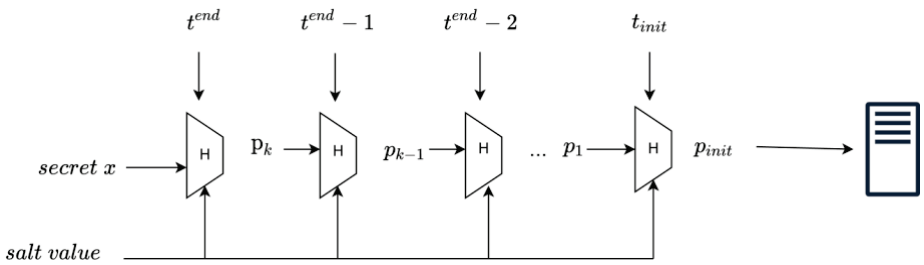

Figure 2.3. T/Key Password Generation Schema

2.3.1.2. Authentication

- During authentication, the client starts hashing from t_{end} until current time.
- Output hashed one-time password (OTP) is relayed from client to server.
- The server hashes the OTP received from the client until it reaches the t_{prev} time.
- If the $p^*_{prev} = p_{prev}$, then authentication succeed. Server stores p^*_{prev} and t^*_{prev}.

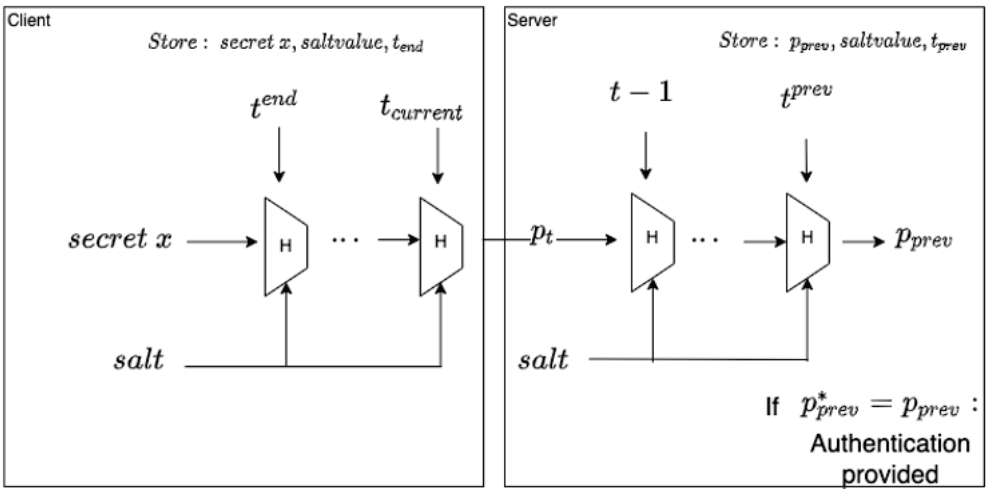

Figure 2.4. T/Key Authentication Schema

2.3.2. Protocol Review

Kogan et al. introduced a new protocol that called T/Key by adding the time information to the hash chain in S/Key [33] [23]. However, adding time information greatly increased the cost of password generation and verification [66]. Because when authentication is required, it will be necessary to perform hash operations from expiration time, t_{end} to the current time, one for each interval. If the validity gap is taken as 30 seconds for each OTP, it is necessary to generate 2^{20} hash chains for a period of one year. Although Kogan et al. proposed checkpoints to improve performance of hash chain, In the worst-case scenario, password generation and verification times are still costly [66].

If the protocol expiration time is selected shorter, the performance will increase, but the new reinitialization will occur earlier.

CHAPTER 3

BACKGROUND

3.1. Second Factor Authentication Mechanism

Second factor authentication mechanism is a branch of multi factor authentication. It provides extra security by adding an additional layer to the single factor such as password or token while authentication is taking place. The reason for this is to prevent the factor in single-factor mechanisms from being captured by the attackers and authenticated to the system. In the last 10 years, millions of usernames and passwords have been stolen from organizations or individuals. The Cybersecurity and Infrastructure Security Agency (CISA) has officially announced that single-factor authentication mechanisms are risky in terms of security [54].

Second factor authentication mechanism is based on the three factor categories: knowledge based, possession based, and inherence based. Second factor authentication mechanisms are formed by the combination of these factor types. Knowledge-based factors are usually self-determined by the user, such as a password or pin. Confidentiality of the password is significant. Because it can be used in more than one system. According to a study by Google, 13 % of people state that they use the same password all account [47]. Possession factors is those that are only owned by the user, such as an assigned USB stick or a bank card. Inference factors are factors based on the biometrics that user is. Examples include face shape, fingerprint, iris scans. Possession factors have been popularized especially by use of fingerprint and face recognition systems in authentication processes on mobile devices. However, many of these second factor authentication mechanism factors are static and must be stored securely by the verification server and client. Having static keys also pose a threat in terms of replay attack. In this respect, OTPs are more secure due to the single usage for in each session [46].

3.1.1. One Time Password (OTP)

One-time passwords are the bit strings that are automatically generated and used for a single authentication session. It usually consists of 6–8 characters. Passwords become invalid and regeneration needed if they are not used within the specified session interval. It has provided a solution to the loss and theft problems of static passwords. There are two approaches in OTP mechanisms. They can be time based, and event based. In time-based approaches, the client and server agree on a common timestamp. After this timestamp, OTP is reproduced at frequent intervals. In event-based approach, OTP' regeneration occurs by triggering the OTP generator. Both approaches have distinct relative advantages over each other. Short-term validity of time-based OTPs have benefits in terms of security, however they can consume more processing power if they are produced with complex algorithms. In the event-based approach, OTP must be stored securely because it can maintain its validity for a long time [38]. But it is more user friendly because password can be used any time.

3.2. Physical Unclonable Function

Physical Unclonable Functions (PUF) is a device-specific digital fingerprint that consists of differences in the manufacturing processes of semiconductors. These differences are uncontrollable and unpredictable. PUF is a hardware security concept that can provide low-cost hardware security by leveraging the unique inherent randomness of a device. PUF mechanism can be occurred with the help of different physical materials such as optical materials, RAMs on microchips. Values such as uncontrollable temperature, electromagnetic wave, voltage in the production processes on these physical materials cause the devices to form their own digital fingerprints and cause each device to be different from each other at the micro level. Since these differences in the microstructure can be used as a device-specific key or id, the PUF mechanism can be applied in systems that require high security.

PUF instance need to have the following properties [3] [61]:

Robustness: Responses of the PUF to the same challenge values in different time are the same or can be corrected with helper functions.

Unclonability: A PUF structure cannot be copied or imitated. It is not entirely possible to emulate the physical conditions of a PUF for a different PUF instance.

Unpredictability: Even if a sufficient number of challenge-response pairs have been obtained, it is not possible to predict beforehand the response to be generated in the PUF against a given challenge value.

Tamper-evident: Any unauthorized access attempt to the PUF causes its behavior to change and accordingly to generate different challenge-response values.

PUF has become a viable work topic for IoT security and privacy due to resource constraints and access difficulties of IoT devices and has a wide variety of uses. Implementing PUF mechanism-based authentication schemes that ensure reasonable security of RFID tags under resource constraints is an effective way to avoid RFID deployment concerns. Maurya and Bagchi propose unilateral factor authentication mechanism for use in RFID systems [41]. As a different field, In order to provide secure communication between Smart meters at consumer and the servers at the Utility Operators due to resource constraints in IoT meters, Boyapally et al. proposed PUF based authentication mechanism that performs cryptographic operations on the server [13]. There are also studies on the use of PUF in the authentication and key sharing processes of sensors in the wireless sensor network (WSN) [40].

3.2.1. Concept

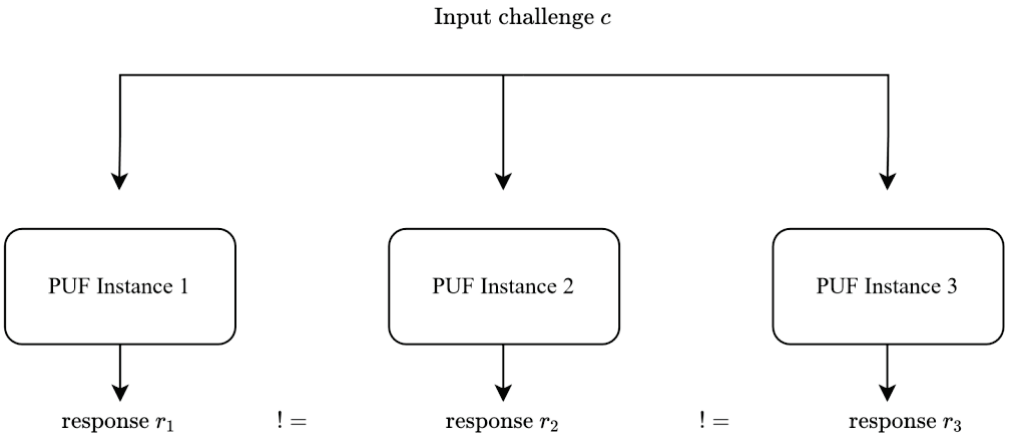

Figure 3.1. PUF Concept: Challenge-Response Pairs

Physical unclonable function construct is basically the concept of creating the output value of a unique function in the device at the micro level against an input value. The input value entering the function is called challenge, while the value created by the function is called response as shown in Figure 3.1. Challenge response pairs are unique and random. What is meant by unique is that the response corresponding to each challenge value is different from each other. PUFs produce values that are easy to calculate, and difficult to recover similar to hash functions.

3.2.2. Availability

Aubel et al. carried on a study looking for PUF components in PC components [4]. They couldn't find a way to access CPU cache and registers before initialized. However, they obtained a usable PUF structure on Nvidia GTX 295 GPU. The situation was different when they upgraded Nvidia GPU drivers. This was explained as due to the large amount of GPU SRAM being cleared on new generation GPUs. In this study, they showed that PUFs are present in the Nvidia GTX 295 graphics card and conclude that

they can be found in other graphics devices as well [4]. FPGA manufacturers in the industry have started to offer products based on PUF. Intel has created PUF-based option for AES key storage in Stratix® 10 FPGA [15]. Intel is working with Intrinsic ID company on PUF-based storage. Intrinsic-Id is a company that provides secure key generation with the help of SRAM PUF and has products on hardware security [28].

3.2.3. Types of PUF

PUFs can classify into four main categories according to their working mechanisms: Electronics, optical, radio frequency and magnetic. The PUF types that the industry focuses on are the Ring oscillator PUF, Arbiter PUF, SRAM PUF, Power Distro PUF which are obtained from microchips based on electronic mechanism [42].

3.2.3.1. Ring Oscillator PUF

Ring oscillator PUF is a mechanism based on signal transmission delay. It is formed by connecting an odd number of inverters together sequentially as shown in Figure 3.2 [12]. The reason for using an odd number of NOT gates is to reverse the incoming signal. The signal from the last inverter is given to both the input and the output. In this way, counter 1-0 signals occur with certain frequencies at the output gate. The output gate frequency is related to the delay of the transmission of the signal from the inverters. Delays in inverters cannot be controlled or predicted in systems where more than one ring oscillator is used and combined into a multiplexer as follows. For this reason, response is unpredictable and device-based due to the production stages of the inverter inside on device.

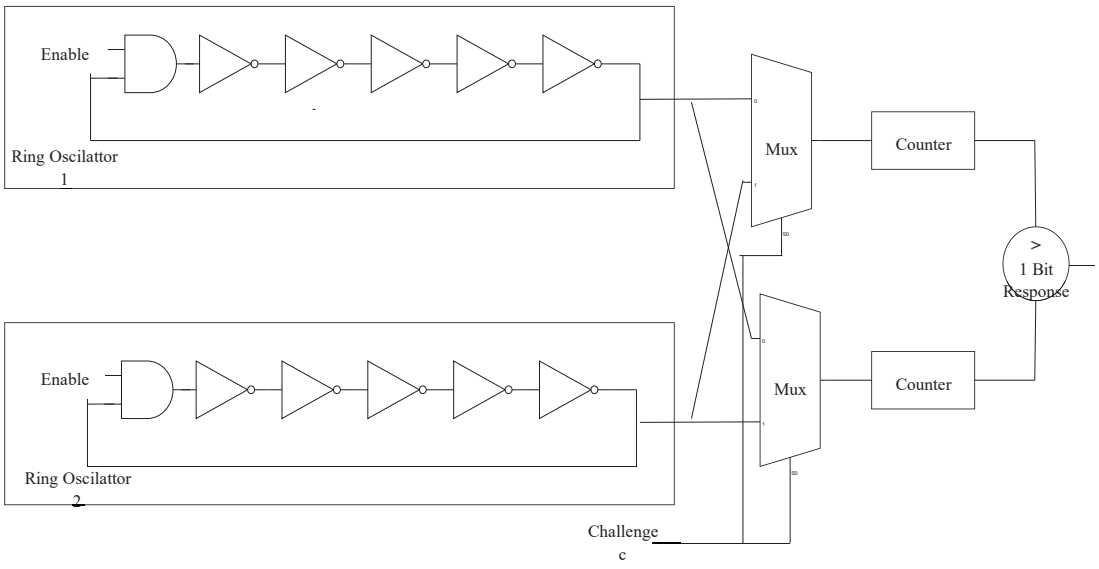

Figure 3.2. Ring Oscillator PUF

3.2.3.2. Arbiter PUF

The arbitrary PUF structure was first proposed by Lee et al. [37]. Arbiter PUF structure consists of two symmetrical flows. These flows consist of multiplexes and the outputs at the end of the streams are connected to the d flip-flop as shown below in Figure 3.3. The selector bit in each multiplex in two symmetrical streams is the challenge bit. However, due to propagation delays in the multiplexer, one of the two streams reaches the d flip-flop first. The arrival of the flows to the flip flop at different times affects the output (Q) response of the flip flop. Flip flop arrival time is unpredictable and is caused by differences in the production phase of multiplexers. Arbiter PUF implementation is shown in the Figure 3.3.

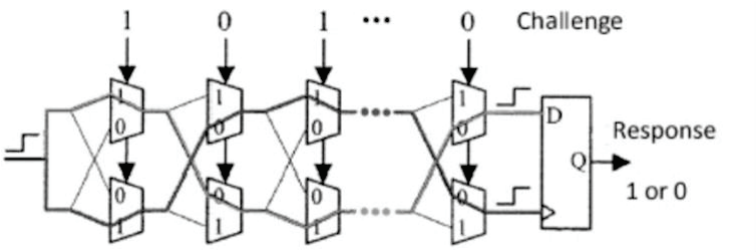

Figure 3.3. Arbiter PUF [37]

3.2.3.3. Power Distribution PUF

Power distribution PUF provides a unique device-specific characteristic due to the differences in the power distribution lines of the integrated circuit and the unpredictable internal resistances of the components. In this structure, the current coming out of the power source completes the circuit through the route. Power drops at certain points on the circuit are measured one by one with the help of power ports (PP01, PP02 etc.) [25]. Challenge in this PUF structure is the number and location of the power transmission lines, while the response is the resistance value created by these components [42].

3.2.3.4. Optical PUF

Optical PUFs, originally called "Physical One-Way Functions (POWFs)", is among the first of the different PUF type mechanisms [51]. It basically consists of five different components. These are laser beam, scattering token, speckle pattern, CCD camera and gabor hash. The core element of the optical PUF mechanism is the scattering token. Scattering tokens are formed by the random scattering of glass spheres in a certain area at a microscopic size on a transparent material. The laser beam sent onto this token creates a unique reflection based on the beam's character. This reflected speckle pattern is captured with the help of the camera. The resulting image is hashed with the help of the gabor function. The result obtained is a string of bits representing the unique value [39]. Overall scheme is depicted in Figure 3.4.

The most advantage of the optical PUF mechanism is that it consists of low-cost components. The structure that carries the PUF does not contain microelectronics.

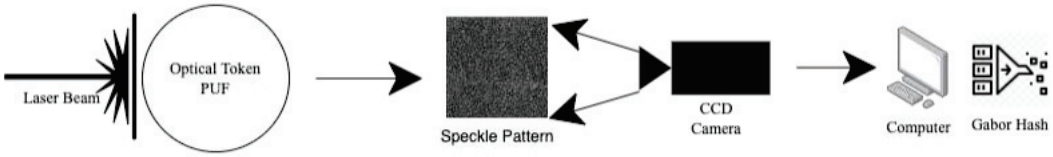

Figure 3.4. Optical PUF Structure

3.2.3.5. SRAM PUF

SRAM, as featured in Figure 3.5, is an IC element that usually consists of four or six transistors and stores one bit of data in a cell 3.5. In the SRAM PUF structure, device specific identification is provided based on the micro-level differences in the production processes of the transistors. When SRAM is power on, the 1-0 pairs stored by cells are also random and device specific. In other words, each SRAM cell sets its preferred state either 1 or 0 in the when power up. These 1-0 values in each cell can be reproduced slight differences in the same order for each chip. These minor differences can be corrected using the fuzzy extractor. It is suitable for industry use, as many devices contain SRAM. Intrinsic ID company is working on hardware security in the industry using SRAM PUF [21].

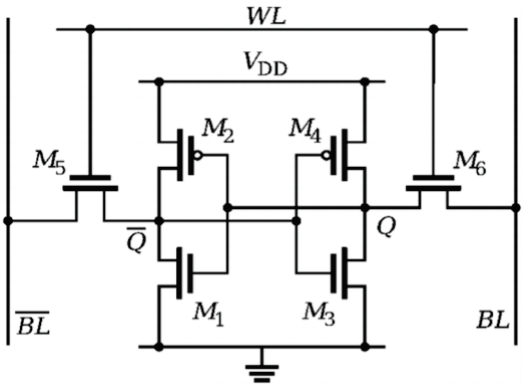

Figure 3.5. SRAM PUF Architecture

3.2.4. Strong and Weak PUF

One of the distinguishing and selective features of PUF mechanisms is strength. PUF mechanisms are classified in two levels as strong and weak PUF in terms of strength. The strengthens of PUFs is related to challenge-response pairs created from a single device. Increase in the number of CRPs also correlates with the size of the device. Since weak PUFs support few CRPs, CRP values can be obtained in the event of an attack. An attacker who has gets the CRP pairs can send the correct response values to challenges

and pretend as server. However, weak PUF produce more stable and robustness responses than strong PUFs. An example of a Weak PUF is the SRAM PUF. Since the power on state is device-specific and standard, the number of CRP is one in the SRAM PUF structure. Because there is only one challenge value: power-on status on the device [26]. In Strong PUF, there is more than one complex component in generating a response. Unlike weak PUF, there are many challenges that can be applied to the PUF. This makes it difficult for an attacker to access all of the CRPs, even if they gain access to the physical environment. In most of the Strong PUFs, a structure can be created where everyone can get a response against the challenge values [42].

3.2.5. Error Correction

Against the same challenge values in the same device, PUF is expected to produce the same response, but this may not be possible due to changing physical conditions. Especially the varying noise level affecting the PUF can be a big challenge [26]. Environmental factors such as temperature, pressure, magnetic field are other factors that change the behavior of the PUF. These factors lead to the degradation of the "digital fingerprint" characterization representing the PUF. Incorrect bits occurring in the PUF need to be corrected. Otherwise, responses consisting of PUF become meaningless, the responses cease to be device-specific information. Multiple fuzzy extractor techniques can be used to correct corrupted bits.

Forward error correction is typical error correction algorithm. It requires to creation of an error correction code (ECC). This mechanism is based on the hamming distance. The receiver stores the codewords corresponding to the bit sequence in its memory. The sender converts, adding control bits, the bit string to the corresponding codewords that the receiver stores in its memory. Sender sends the data in this format to the receiver. The receiver splits the bit sequence from the sender into parts. The receiver also converts the bit sequence to its original form using the dictionary. If there is a piece without a dictionary equivalent, the hamming distance is calculated one by one with the values in the ECC dictionary. value corresponding to the smallest result is used. But ECCs can be require costly processing time or processing power. These can be a challenge for resource restricted devices. To solve this problem, the data remanence method was

proposed by Aung et al. [5]. They achieved 99.98% stability in their study on 512 bits SRAM values.

3.2.6. Modelling Attacks

The most prominent attack type in PUF mechanisms is the modeling attack. The basic principle in the modeling attack type is create a model that predicts the responses corresponding to challenge values that can be sent to the PUF with different prediction techniques (machine learning, data mining etc.). Ruhrmair et al., using two different prediction algorithms in five different PUF types obtained an accuracy between 95% and 99.9% [55]. Hospodar et al. studied the susceptibility of 65nm CMOS Arbiter PUF to machine learning attacks. They concluded that Simple Arbiter PUFs cannot be used reliably for PUF-based authentication, and the applicability of 2-XOR Arbiter PUFs is also limited [27]. Many studies are also being carried out to build PUF that is resistant to modeling attacks. Oun and Niamat designed delay-based FPGA that has PUF characteristic. KNN achieved the highest accuracy rate of 6.8%, corresponding to the machine learning attack in the proposed PUF design. The proposed PUF design considerably prevents threats against modeling attacks [49]. Similarly, Wangetal. propose a new PUF mechanism that is resistant to modeling attacks by modifying the arbiter PUF mechanism. They create a complex CRP structure for the prediction model by changing some response values. In the model they propose, the response r value from the PUF is sent to the server in its original form or modified in a middleware and saved as r^* [64].

3.3. Side Channel Attacks

Mathematical and software-based methods often come to mind when cracking cryptographic keys and algorithms. Side-channel attack is based on making inferences by physically monitoring the system. Monitoring of power consumption, electromagnetic wave graphs, sound frequencies and timing information are some of the factors used in

side-channel attacks. It has been shown that many algorithm implementations, by used these side channel factors, tried to make inferences about secret information. Including DES, AES and RSA are vulnerable to side-channel analysis attacks, and various measures that can be taken have been suggested. Ghosh et al. performed side channel attack analysis on the hardware implementation of the RSA algorithm. They concluded that RSA have vulnerabilities in differential power attacks (DPA) [20]. The use of deep learning and machine learning techniques in combination with side-channel attacks increases the power of the attack. Kubota et al. created a Convolutional neural network-based side channel attack mechanism and evaluated its success on the AES encryption algorithm [35]. Different implementations of the same algorithm may leak different amounts and forms of side channel information. For this reason, side-channel analysis attacks are often not generalizable.

3.3.1 Power Analysis Attack

The most powerful of side-channel attack is power analysis attacks [50]. In power analysis attacks, a correlation is established between the power consumption of the cryptographic device and the confidential information, or the transactions performed, and it is tried to access the confidential information [48] [65]. A resistor is placed circuit, and the current drawn information is obtained by using the difference in voltage values at both ends of this resistor. Kocher tried power analysis attacks in first time on Data Encryption Standard (DES) and was successful [30]. We can examine power analysis attacks under two headings: simple power analysis and differential power analysis.

3.3.1.1. Simple Power Analysis

Simple Power Analysis (SPA) attacks are made using a single measurement of power consumption while the crypto device is running. Simple power analysis attack analyzes the relationship between the power consumed by the cryptographic system and the processing steps it performs. While microprocessor instructions perform different operations such as addition and multiplication, integrated circuits consume different

amounts of power. During the operation of the crypto algorithm, different operations take place depending on the key value used, and the key value can be captured with the power consumption information observed during this time.

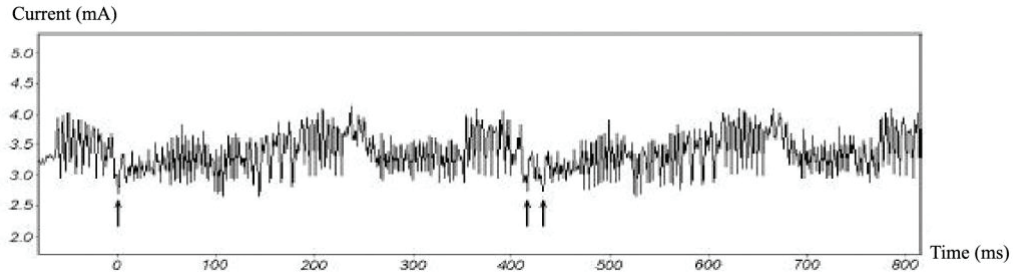

Figure 3.6. SPA Trace Showing DES Rounds 2 and 3 [30]

Some details of the algorithm can be seen more clearly when examining a high-resolution power consumption measurement of two consecutive DES rounds (rounds 2 and 3) shown in Figure 3.6. 28-bit DES key registers are rotated once (left arrow) on round 2 and twice (right arrows) on round 3 [30]. These differences are mainly due to jump operations.

3.3.1.2. Differential Power Analysis (DPA)

While SPA attacks are primarily analyzed based on visual inspection of the power fluctuations, side-channel information or confidential information can be obtained using statistical techniques and error correction methods in DPA. Small variation in power consumption due to the processed data makes observation difficult because of measurement errors and noise. DPA attacks are more difficult to implement than SPA attacks. However, they are much more powerful and irresistible than BGA attacks [31]. Using DPA, an adversary can obtain secret keys by statistically analyzing power consumption measurements from multiple cryptographic operations performed by a vulnerable smart card or other device.

3.3.2. Timing Attack

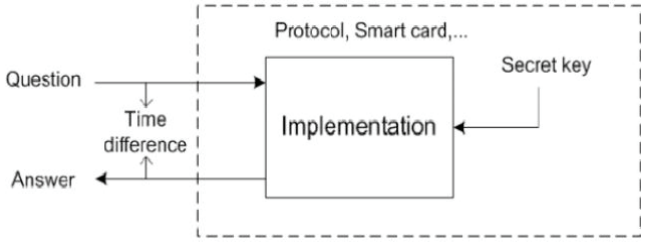

Figure 3.7. Conceptual View of The Timing Attacks [8]

In timing analysis attacks, leaked timing information by algorithms that do not have a fixed data processing time is used as side channel information. Attacker tries to compromise the system by observing the processing time of iterations in the algorithm. The reason for the formation of side channel information is that the processing time in the steps of the algorithm depends on the secret key used. The attack is more effective if there is a conditional operation in the algorithm. Since the timing characteristics of symmetric-key algorithms are not as key-dependent as those of asymmetric-key algorithms, they are more robust against timing analysis attacks. In 1996, Paul Kocher showed in the study RSA leaks key information over algorithm runtime [32].

3.3.3. Electromagnetic Attack

Signals in digital circuits consist of zeros and ones. The energy consumed when the signal goes from zero to one is about a thousand times the energy required to remain unchanged at that level. 1% of this excess energy is spent to maintain the new voltage level, about 4% is converted into heat, and the rest is emitted as electromagnetic waves.

This emitted wave may contain not only noise, but also signal leakage related to the information that is effective in its formation [17]. The Biot-Savart law used to calculate electromagnetic radiation is as follows:

$$dB = \frac{\mu_0 I d\ell \times \hat{r}}{4\pi r^2} \quad (3.1)$$

where $\hat{r}$ = unit vector, μ_0 = vacuum permeability, $d\ell$ = the length of the current carrying conductor, I = current value.

According to this equation, we can make the following inferences:

1. The intensity of electromagnetic radiation varies with the current density. Therefore, electromagnetic radiation is directly related to the size of the data [24].
2. Direction of electromagnetic radiation changes depending on the current direction.

Transient Electromagnetic Pulse Emanation Standard (TEMPEST) standards have been introduced to prevent information leakage of electromagnetic radiation [59]. In electromagnetic attack, measurement analysis operations are similar to power analysis attacks. While it is necessary to place a resistor with physical intervention between the circuit and the power source for current change measurement in power analysis, it is not needed in electromagnetic analysis attacks. The electromagnetic field can be measured by means of an antenna without interfering with the circuit [24].

3.3.4. Cold Boot Attack [22]

Cold boot attack is another side-channel attack that requires being physically close to the device. Data is stored in an encrypted, format on your device storage. When the device boots up, device stores encryption key in its RAM and uses it to encrypt and decrypt data as long as your device stays powered on. The RAM can hold data from a few seconds to a few minutes even after power off. The time it takes for data to disappear from RAM can be significantly extended by cooling the RAM. In this short period, anyone with the appropriate tool scan read the RAM and copy its contents to a secure and permanent storage on a USB stick or SD Card using a different lightweight operating system. It is possible to extract the passwords from the information obtained from the

RAM. Typically, the purpose of a cold boot attack is to illegally obtain disk encryption keys without permission.

CHAPTER 4

P/KEY: PUF BASED SECOND FACTOR AUTHENTICATION

4.1. Introduction

In this thesis, we propose a second-factor authentication protocol that uses PUFs on the server-side as s secure storage for clients' secrets. The server contains a PUF and based on the randomness of the PUF, a unique key value is generated for each client in server. The protocol also offers a time-based second factor authentication mechanism, as in the TOTP [44]. However, unlike TOTP, the clients' key are not stored on the server. The keys are generated during the authentication request on the server-side and deleted after use. Thus, if the server is compromised, the attacker cannot learn the clients' secrets. Furthermore, our protocol is resistance to side-channel attacks. This means the side-channel attacker cannot learn the secret of a client whose secrets are revealed to the server's memory during the protocol execution.

The secrets produced by the server in each authentication request for a client are needed to be the same. This is achieved with the fuzzy extractor. For this reason, it is important that the PUF is not damaged. Otherwise, PUF may produce different results for the same challenge value at different requests. This problem results in incorrect secret generation and authentication cannot be provided.

It is more suitable to use PUF structures formed in the integrated circuit so that it can be used in industry. An example of this is the SRAM PUF. Aubel et al. [4] have also shown that PUFs are present in the Nvidia GTX 295 graphics card and conclude that they may be present in other graphics devices. In the publication, it was stated that CPU manufacturers could introduce the PUF feature of the CPU by making minor changes in the hardware features.

Table 4.1. P/Key Notation Table

Symbol	Description
s_1	s_1 value represents the challenge seed value that the client sends to the server to generate the first key.
s_2	s_2 value represents the challenge seed value that the client sends to the server to generate the second key.
k_1	k_1, (i.e., $P(s_1)$) is the first key value of the client. It is obtained by putting the client seed value s_1 in the PUF.
k_2	k_2, (i.e., $P(s_2)$) is the second key value of the client. It is obtained by putting the client seed value s_2 in the PUF.
a_1	a_1 is a random value that is chosen by the client
a_2	a_2 is a random value that is chosen by the client.
c_1	c_1 is a random value that is chosen by the server.
c_2	c_2 is the random value that is chosen by the server.
d_1	d_1 is the partial seed of one of the client's secret.
d_2	d_2 is the partial seed of one of the client's secret.
r	r is a random value chosen by the client.
I	I is the time interval between two timestamps. It also determines the validity period of OTP.
M_1	M_1 is the first OTP.
M_2	M_2 is the second OTP.
P	PUF. $\{0, 1\}^{l} \rightarrow \{0, 1\}^{l}$
$\oplus$	XOR operator
$\in$	$\in$ is elements of operator
H	Hash function. $\{0, 1\}^{l} \times \{0, 1\}^{l} \rightarrow \{0, 1\}^{l}$.

4.2. Assumptions

We make the following assumptions:

- The communication channel between client and server is secured with TLS [16].
- Sadeghi [56] defined the idealized behavior of PUF. We assume that the ideal PUF is used in the protocol. This means that the PUF produces the same response against the same challenge value at different times. Also, if in case of any tampering attempt in server, the PUF is destroyed and shows different characteristics than before.

- The client and server have a synchronized clock to calculate the time period. They agree beforehand on the time interval, I, required to calculate the number of elapsed periods.

4.3. Protocol Description

The proposed second-factor authentication protocol has two phases: initialization and authentication. The server has randomly chosen seed values for each client and generates clients' secrets by evaluating these seeds with the PUF during the authentication. This means the server can verify the one-time passwords of a client with the client's secrets generated with the PUF.

4.3.1. Initialization

Initialization is a process in which a client and server agree on common secrets. These common secrets will be used to authenticate the client that wants to benefit from a service provided by the server. In the proposed initialization protocol, a client chooses four different random values a_1, a_2, s_1, s_2 and sends them to the server. Server gets these random values and performs the following steps in order:

Step 1: $u_1 = P(s_1) \oplus a_1$

Server gets s_1 challenge seed value that sent by client and put it in PUF function. Response value produced by PUF is the secret k_1 of the client. Then XOR operation is performed between $P(s_1)$ value and a_1 value. The XOR operation hides the secret k_1 within u_1.

Step 2: $c_1 \in \{0, 1\}^l$

Server chooses an l-bit random value c_1.

Step 3: $d_1 = s_1 \oplus c_1$

Server perform XOR operation between client's secret seed value, s_1, and the random value c_1. Thus, in case the server is hacked, the attacker cannot access the secret k_1 of the

clients. Because to reach the secret, k_1, the attacker must know the value of s_1 and put this value in the PUF as the challenge value. However, the server performs an XOR operation between s_1 and c_1. Thus, server hides the s_1 value in d_1.

Step 4: $delete\ a_1, s_1, P(s_1)$

In the fourth step, server deletes the values $P(s_1)$, a_1 and s_1, so that in case of an attack, the attacker cannot obtain the secret k_1 of the client.

Step 5: $u_2 = P(s_2) \oplus a_2$

As in step 1, the server generates the secret k_2 of the client with the PUF, and then XORes k_2 with a_2 value. This XOR operation hides the secret k_2 within u_2.

Step 6: $c_2 \in \{0, 1\}^{l}$

As in step 2, the server chooses an l-bit random value c_2.

Step 7: $d_2 = s_2 \oplus c_2$

The server hides the seed of the secret, k_2, by performing XOR operation between the seed value, s_2, and the random value c_2. Thus, in case the server is hacked, and unauthorized people gain the database access, the attacker cannot extract the seed of the secret k_2 from d_2.

Step 8: $delete\ a_2, P(s_2), p_2$

The server deletes the $P(s_2)$ value (k_2), a_2, and p_2, in order to prevent the attacker from accessing the secret k_2.

Step 9: The server sends u_1, u_2, c_1 and c_2 to the client.

The server sends u_1, u_2, c_1 and c_2 to the client. u_1 and u_2 hide the secrets k_1, and k_2, respectively. These values are hidden by the XOR operation as mentioned above. c_1 and c_2 values are also sent to the client so that the client's s_1 and s_2 values can be generated on the server-side in the authentication phase.

Step 10: delete c_1, c_2

The server deletes c_1 and c_2 values after sending them to the client. If these values are not deleted and stored in the server's database, the attacker having the credentials of the server can obtain the values s_1 and s_2 using the equations $s_1 = d_1 \oplus c_1$ and $s_2 = d_2 \oplus c_2$. After obtaining the s_1 and s_2 values, it can obtain the secrets k_1 and k_2 by evaluating s_1 and s_2, respectively, with the PUF.

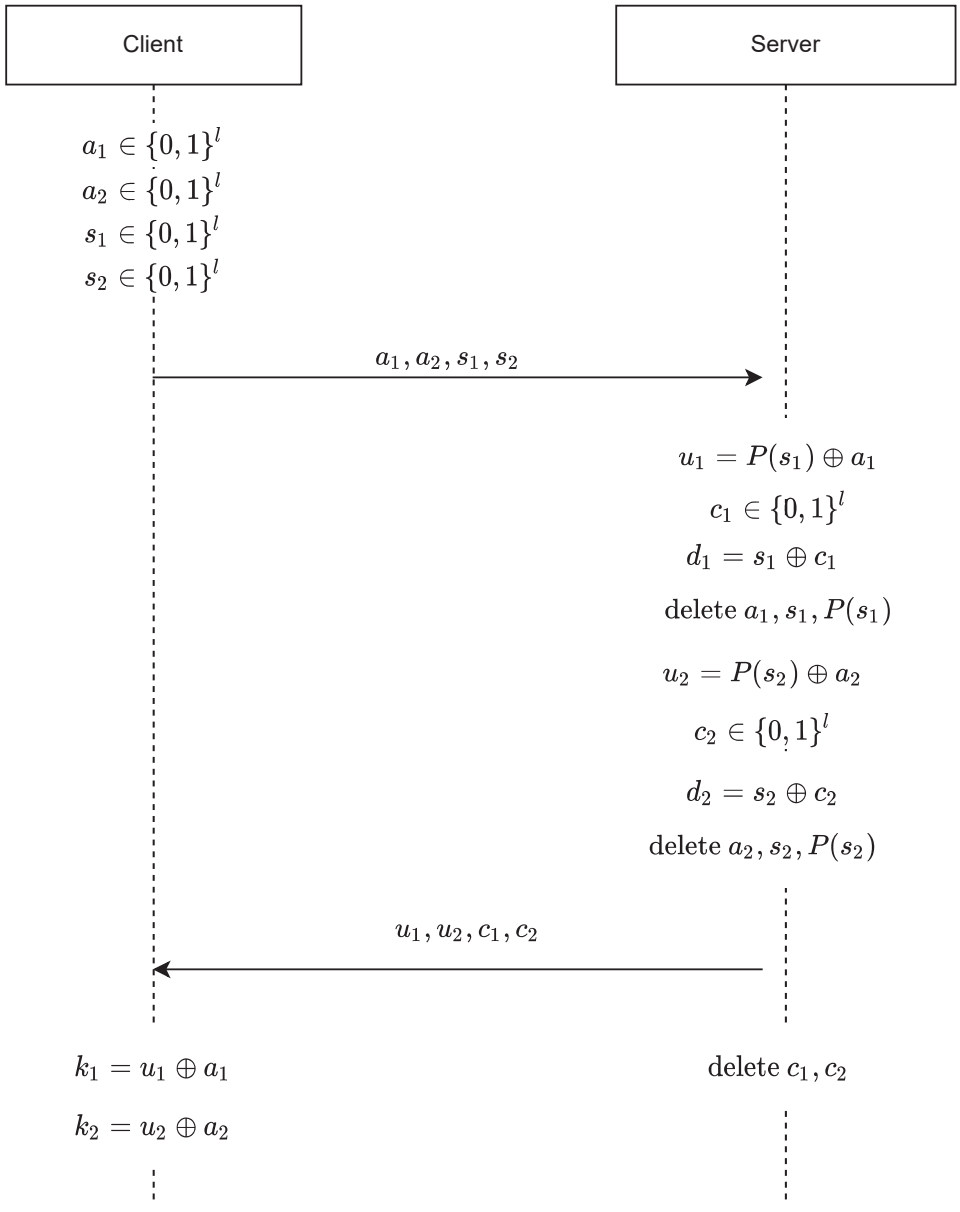

Figure 4.1. P/Key: Overall Initialization Scheme

4.3.2. Authentication

In the proposed authentication protocol, the client computes two OTPs that are valid for a certain period. Our protocol requires a synchronized clock between the client and the server like other TOTP protocols. When authentication is needed, the client calculates the elapsed time and use it in the calculation of OTPs.

In the calculation of OTPs, a random value r is used. This value is used to prevent potential modifications on OTPs by the attacker that can access them. In addition to this, r is used to make two OTPs to be dependent on each other. Thus, the server that is able to generate the secrets of the client can extract the random value r from the first OTP M_1 and use it in the verification of the second OTP M_2.

Each client secret is used to calculate only one OTP. This means one secret is used to calculate M_1 and the other is used to calculate M_1. The server uses the PUF to generate the secrets of the client sequentially to verify the authentication request of the client. In the case of physical attacks (e.g., side-channel attacks), since the characteristics of the PUF will change, at least one of the PUF execution will behave differently and incorrect key values will be generated. For this reason, the server will not be able to verify the client.

The client registers to the server in the initialization phase and obtains the s_1, s_2, c_1 and c_2 values required for authentication from the server.

The authentication protocol is depicted in Figure 4.2. The steps of the authentication protocol are as follows:

4.3.2.1 Client Side

Step 1: $r \in (0, 1)^l$

The client chooses l-bit random value r.

Step 2: $(t - t_0) / I$

The time period that OTPs will be valid is calculated. The current time is subtracted from the time t_0, and the result is divided by interval to calculate the valid time period.

Step 3: $M_1 = H(k_1, \frac{t-t_0}{I}) \oplus r$

In this step, the secret value k_1 is hashed with the time period calculated in the previous step. Then, the XOR operation is performed with the result and the r value. Thus, the first OTP is calculated. In this way, it is ensured that different values are produced in each authentication request, even if it is in the same time period.

Step 4: $M_2 = H(k_2 \oplus r, (t - t_0) / I)$

In the calculation of the second OTP M_2 firstly, XOR operation is performed with the k_2 value and r. Result is hashed with time period. Then, the generated M_1, M_2 , c_1 and c_2 values are sent to the server as an authentication request.

4.3.2.2. Server Side

The server stores d_1 and d_2 values that are generated for each registered client in the initialization phase. We assume that the partial seeds of a client's secrets are stored in the server database along with the client's descriptive and complementary information.

Step 5: $k_1 = P(d_1 \oplus c_1)$

The server calculates the seed s_1 the client by performing XOR operation between the c_1 value that the client sends and the d_1 value it stores. It gets the secret k_1 by putting s_1 value in the PUF.

Step 6: $r = M_1 \oplus H(k_1, (t - t_0)/I)$

The server hashes the time period with the secret k_1 obtained in the previous step. Then, XOR operation is performed between the M_1 value sent by the client and the hash result. Thus, the random value r is obtained. r value is authentication specific and different for each authentication request. The important point is that the server extracts the r value in

the same time period as the client generated. If the r value is extracted after a long time has passed, the r value obtained will not be the same as the one produced by the client, since the time period will change.

Step 7: $delete\ k_1, P(d_1 \oplus c_1), c_1$

The server deletes the k_1, $P(d_1 \oplus c_1)$, c_1 values. The reason for this is to ensure that in case of an attack in further steps, the attacker cannot learn the secret k_1.

Step 8: $k_2 = P(d_2 \oplus c_2)$

The server calculates the seed s_2 the client by performing XOR operation between the c_2 value that the client sends and the d_2 value it stores as in Step 5. It gets the secret k_2 by putting s_2 value in the PUF.

Step 9: $M_2 \stackrel{?}{=} H(k_2 \oplus \mathrm{r}, (\mathrm{t} - t_0)/I)$

The server has the necessary r and k_2 values to generate the M_2 message. As in the client side, it generates the OTP M_2^* and compares it with M_2. If M_2 and M_2^* are equal, the server authenticates the client.

Step 10: $delete\ k_2, P(d_2 \oplus c_2), c_2$

The server deletes the k_2, $P(d_2 \oplus c_2)$, c_2 values. The reason for this is to ensure that in case of an attack in further steps, the attacker cannot learn the secret k_2.

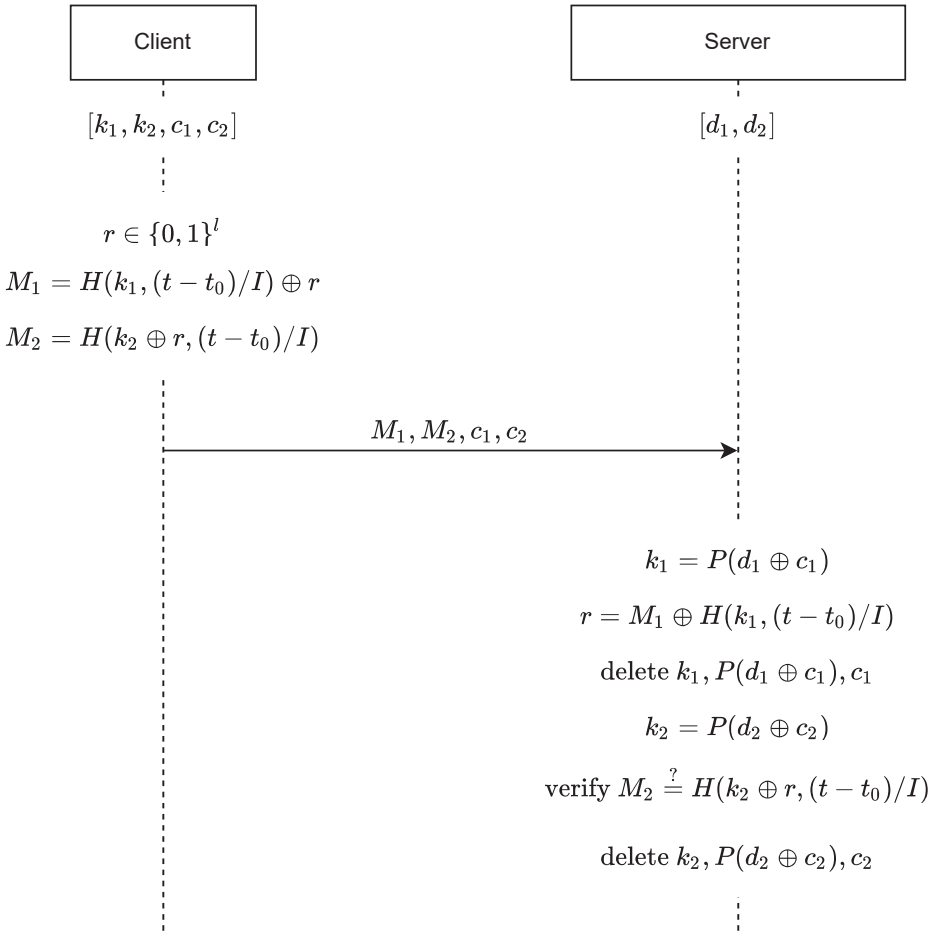

Figure 4.2. P/Key: Overall Authentication Scheme

4.4. Comparison

In the proposed protocol, the seeds of the clients' secrets are stored on the server in a hidden format. Therefore, if the server is compromised, the values obtained by the attacker cannot be used to impersonate the clients. In the protocol, if and only if an attacker attacks and accesses the server while a client performs authentication, the attacker can learn the secret keys, k_1 and k_2, of that client. However, in this case, the attacker only obtains the secrets of the client that sends authentication request. The secrets of other clients that have previously registered on the server are still safe. For this reason, the protocol is *partial resistant to server-side compromising*. In T/Key and S/Key protocols, secrets are kept hashed on the server. Thus, these protocols are also resistant

to server-side compromising. But the situation is different in TOTP. In TOTP, the attacker who takes over the server, by getting the server credentials, can generate the OTP because it obtains the secret of the clients.

In case of a side-channel attack on the server for our proposed protocol, the PUF is destroyed. This means the physical structure of the PUF will change. Therefore, the server will not be able to generate the correct secrets using the PUF. For this reason, the protocol is *resistant to side-channel attacks on server-side*. In the case of a side-channel attack in the T/Key protocol, the OTPs obtained by the attacker is not sufficient for client impersonating because OTPs are time-dependent, and the attacker cannot obtain the previous hash values to generate the future OTPs. In the S/Key protocol, the OTP of a client obtained by the side-channel attacker can be used to impersonate that client with high probability because the OTP is not time-dependent. As in T/Key, the attacker cannot obtain the previous values in the hash chain and cannot generate the future OTPs. In the TOTP mechanism, the secret of a client sending the authentication request to the server can be obtained with side-channel attacks and then can be used to impersonate the client.

Table 4.2. Comparison Table

Metric	TOTP	S/Key	T/Key	P/Key
Resistance to Server-side Side Channel Attack	Not Resistant	Resistant	Resistant	Resistant
Resistance to Server-Side Compromise Attack	Not Resistant	Resistant	Resistant	Partially Resistant
Usability Limitations (Hash Chain Usage)	No	Yes	Yes	No
Resistance to Replay Attacks (Time-based Passwords)	Yes	No	Yes	Yes

In our protocol, the hash chain structure is not used. It is costly in terms of time as many hash operations take place in systems based on the hash chain. In the protocol, since the authentication and key generation processes are done in the PUF, these processes are completed quickly. Thanks to PUF, the protocol that *responds quickly to authentication requests* on the server side is proposed.

In time-based OTP protocols, secrets are periodically renewed on the server side. For this reason, the same secret is not stored on the server for a long time. TOTP, T/Key and P/Key are time-based OTP mechanisms. However, in the S/Key protocol, the stored

secret required for verification on the server is not updated until a new authentication occurs. The secret of a client is updated when it is used in the S/Key mechanism. Because of that reason, OTP can valid for a long time. Storing the same secret on the server for a long time poses a security threat. Comparison of the proposed protocol with the other three protocols are given in Table 4.2.

CHAPTER 5

SECURITY ANALYSIS

5.1. Threat Model

We assume that the user device meets the required security requirements, does not carry any malware so that the user session cannot be hijacked by an attacker. We assume that the communication channel between the user device and the server is protected by TLS [16], thus man-in-the-middle (MITM) attacks are not possible. All TOTP schemes are vulnerable to online phishing attacks, where users' short-term one-time passwords are compromised. However, the time limit of one-time passwords makes it difficult to carry out an attack using them.

We assume that adversaries are able to access the server multiple times and obtain the necessary information to authenticate the clients. For our protocol, the information received from the compromised server is the seeds of the clients' secrets and the passwords that can be obtained from the memory during the execution of the protocol for a specific client.

We assume that attackers with physical access to the server can perform side-channel attacks [22], especially cold-boot attacks, to extract client passwords and secrets.

5.2. Formal Definition of One-time Password Protocol

A one-time password protocol is defined by the following procedures:

- PPGen(1^s): is an algorithm that outputs the password length l the given security parameter s.

- KeyGen(l): is a probabilistic polynomial-time algorithm that outputs the secrets k_1 and k_2 of the prover and the internal state of the verifier consisting of partial seeds d_1 and d_2 of the prover's secrets.

- Prover (t, k_1, k_2): is a polynomial time algorithm that takes the time t and the prover's secrets k_1 and k_2 and outputs the one-time passwords M_1 and M_2.

- Verifier (t, d_1, d_2, M_1, M_2): is a polynomial time algorithm that takes as input the time t and the verifier's internal state d_1 and d_2, and one-time passwords M_1 and M_2. It outputs accept or reject based on whether one-time passwords are verified successfully.

In order to prove the correctness, our protocol must output "accept" for every Verifier (t, d_1, d_2, M_1, M_2) call, where t is monotonically increasing and, M_1 and M_2 are produced with Prover (t, k_1, k_2).

5.3. Adversary Model

The adversary is mainly defined by specifying the actions she is allowed to take (the oracles she can query), the purpose of her attack (the definition of the game), and the way she interacts with the server and clients.

An adversary is an algorithm that can run the following oracles.

- Launch: enables the client to start a new protocol instance π at time t.

- SendServer (m, π, t): sends a message m to the server in a protocol instance π for the time t. Then, it receives the message m' as an answer.

- Result(π): returns 1, if the server verifies a client, and 0 otherwise at the end of the protocol session π.

- CorruptServer(): corrupts the server and gets the internal states of it.

5.4. Analysis

In this section, we analyze the security of the proposed protocol.

Definition 5.4.1 (Hash Function)**.** *Let* $l \in \mathrm{N}$ *be a security parameter,* $\gamma, \kappa \in \mathrm{N}$ *be polynomially bounded in* l*. A hash function* H *is defined as* $\{0, 1\}^{\gamma} \rightarrow \{0, 1\}^{\kappa}$ *with the following basic requirements:*

1. *For a given output* y_i*, it is computationally infeasible to find an input* x_i *satisfying* $h(x_i) = y_i$*.*
2. *It is computationally infeasible to find a pair* (x_i, x_j) *satisfying* $x_i \neq x_j$ *and* $h(x_i) = h(x_j)$*.*
3. *Any probabilistic polynomial time adversary who queried* H *for a polynomial number of times can distinguish the output of* H *with at most negligible probability.*

Definition 5.4.2 (Physically Unclonable Function (PUF) [56])**.** *Let* $l \in \mathrm{N}$ *be a security parameter,* $\gamma, \kappa \in \mathrm{N}$ *be polynomially bounded in* l*. An ideal PUF* P *is defined as* $\{0, 1\}^{\gamma} \rightarrow \{0, 1\}^{\kappa}$ *that has the following parameters:*

1. *For all* $c \in \{0, 1\}^{\gamma}$ *and all pairs* $(r_i, r_j) \in [P(c)]^2$*, it holds that probability* $Pr[r_i = r_j] = 1$*.*
2. *Any physical attempt to tamper the device on which* P *is implemented results in the destruction of* P*. Thus,* P *cannot be evaluated any more correctly because its behavior is changed.*
3. *Any probabilistic polynomial time adversary who queried* P *for a polynomial number of times can compute the output of* P *with at most negligible probability.*

Lemma 5.4.3. *Let A be an adversary. The advantage of* $\mathcal{A}$ *of obtaining the secrets* k_1 *and* k_2 *by corrupting a server during the execution of the initialization protocol is negligible.*

Proof. We assume that there is an adversary $\mathcal{A}$ that can learn the secrets k_1 and k_2 of a client by corrupting the server during the execution of the initialization protocol. If $\mathcal{A}$ corrupts the server while it is not interacting with any client, $\mathcal{A}$ does not learn anything

because the volatile memory is empty. In the case where $\mathcal{A}$ corrupts the server while interacting with any client, corruption time is important to determine what the attacker can learn because deletion of some values are performed two time during the protocol execution. Assume that $\mathcal{A}$ corrupts the server before the first deletion and obtains a_1, a_2, s_1, s_2, u_1, c_1 and d_1. $\mathcal{A}$ extracts the secret k_1 by computing $k_1 = u_1 \oplus a_1$ and wants to infer the secret k_2. In order to infer k_2, A has to simulate the PUF $P(.)$ but this contradicts with the security of PUF (Definition 5.4.2). Assume that $\mathcal{A}$ corrupts the server before the second deletion and obtains a_2, s_2, u_2, c_1, c_2, and d_2. $\mathcal{A}$ computes the secret $k_2 = u_2 \oplus a_2$ and wants to infer the secret k_1. In order to infer k_1, $\mathcal{A}$ has to expose it from u_1 but u_1 and a_1 are random it is not possible extract k_1 from u_1 without knowing c_1. Alternatively, $\mathcal{A}$ can calculate $P\ (d_1 \oplus c_1)$ to get the secret k_1, but this means $\mathcal{A}$ can simulate $P(.)$. This contradicts with the security of PUF (Definition 5.4.2). As a result, $\mathcal{A}$ can learn k_1 and k_2 by corrupting the server during the execution of the initialization protocol with negligible probability.

Lemma 5.4.4. Let $\mathcal{A}$ be an adversary. The advantage of $\mathcal{A}$ of obtaining the secrets k_1 and k_2 by corrupting a server during the execution of the authentication protocol is negligible.

Proof. We assume that there is an adversary $\mathcal{A}$ that can learn the secrets k_1 and k_2 of a tag by corrupting the server during the execution of the authentication protocol. If $\mathcal{A}$ corrupts the server while it is not interacting with any client, $\mathcal{A}$ does not learn anything because the volatile memory is empty. In the case where $\mathcal{A}$ corrupts the server while interacting with any client, corruption time is important to determine what the attacker can learn because deletion of some values are performed two time during the protocol execution. Assume that $\mathcal{A}$ corrupts the server before the first deletion and obtains M_1, M_2, c_1, c_2, k_1, r, d_1 and d_2. $\mathcal{A}$ knows the secret k_1 and wants to infer the secret k_2. In order to infer k_2, A has to expose it from M_2, but this contradicts with the security of hash functions (Definition 5.4.1). A knows that $k_2 = P(d_2 \oplus c_2)$ so the other way to infer k_2 is to simulate $P(.)$. This contradicts with the security of PUF (Definition 5.4.2). Assume that $\mathcal{A}$ corrupts the server before the second deletion and obtains M_1, M_2, c_1, c_2, k_2, r, d_1 and d_2. $\mathcal{A}$ knows the secret k_2 and wants to infer the secret k_1. In order to infer k_1, $\mathcal{A}$ has to expose it from M_1 but this contradicts with the security of hash functions (Definition 5.4.1). $\mathcal{A}$ knows that $k_1 = P\ (d_1 \oplus c_1)$ so the other way to infer k_1 is to simulate $P(.)$. This contradicts with

the security of PUF (Definition 5.4.2). As a result, $\mathcal{A}$ can learn k_1 and k_2 by corrupting the server during the execution of the authentication protocol with negligible probability.

Theorem 5.4.5. *The proposed authentication protocol provides second factor authentication.*

Proof. We assume that there is an adversary $\mathcal{A}$ that can generate valid M_1, M_2, c_1 and c_2 for a given client. $\mathcal{A}$ wins the security experiment if M_1 and M_2 pass the verification in the server. The communication between the client and server is secured so $\mathcal{A}$ cannot listen the channel between them.

Assume that during the execution of the initialization and authentication protocols between the client and server, $\mathcal{A}$ call CorruptServer () oracle to get the secrets of the client. This will contradict with Lemma 5.4.3 stating that $\mathcal{A}$ cannot learn the secrets of a given client by corrupting the server during the execution of the initialization protocol and with Lemma 5.4.4 stating that $\mathcal{A}$ cannot learn the secrets of a given client by corrupting the server during the execution of the authentication protocol.

A performing the server corruption can learn one-time passwords M_1 and M_2 created by the client. The time limitation of one-time passwords makes it difficult to perform an impersonation attack using them. This is the common problem of all TOTP protocols.

Additionally, an adversary who obtains server login credentials learns partial seeds of client secrets from the server database and can execute the PUF without breaking it. It is impossible for an attacker to learn secrets from partial seeds. The adversary who has accessed the server can learn the secrets of the clients that sent authentication requests to the server during the time when the adversary has the control of the server. Since we can assume that it is very difficult to learn the login credentials of the server and the detection time of an adversary who has accessed the server is very short, the probability of such an attack is negligible.

CHAPTER 6

CONCLUSION

A more secure authentication mechanism is provided with second-factor authentication systems by adding an independent layer to the single factor. One-time passwords (OTPs) are preferred as the second factor due to their easy generation, platform independence, and one-time use.In the second-factor authentication protocols based on the hash chain mechanism, OTP production is limited as it depends on the length of the hash chain. As the hash length increases, the number of OTPs increases, but the authentication speed slows down. Since the number of OTPs is limited, they require re-initialization after a certain period.

The thesis proposes a PUF-based and time-dependent second-factor authentication mechanism. PUFs are the digital fingerprint of devices and cannot be copied. PUF differs for all semiconductors as it is a natural result of the manufacturing process of semiconductors. The device-specific and fast response properties of PUF are the main reasons for its widespread use in key generation in cryptology. OTPs in the proposed protocol are generated with a combination of client-specific secrets and valid for a certain period. Unlike the TOTP mechanism, the secrets of a client are not stored on the server. They are generated during the authentication phase and deleted after verification. As the secret is generated in the PUF inside the server, we proposed a tamper-proof second-factor authentication protocol that can generate OTP fastly. In case of a side-channel attack, because the PUF is tampered with, its physical structure will change and produce different responses than before. In this way, the attacker will not be able to take over the system and will not be able to impersonate the clients. In addition, the secrets of the clients are not stored directly, but store in a hidden format in the server-side. Thus, the clients' secrets are stored securely and cannot be retrieved in case of server compromise. However, while the client is requesting authentication to the server, if the attacker compromising the server can obtain the secrets of that client. As future work, we would like to work on a solution for this problem.

As a threat to the protocol, if many clients are registered to the server and the server has to respond to many instant authentication requests, the server's response time may be delayed. How many requests the PUF can respond to instantly is a subject of research in future studies.

REFERENCES

[1] Authentication - Google Safety Center. https://safety.google/authentication. Accessed on 2022-02-11.

[2] GuidetoTwo-FactorAuthentication·DuoSecurity. https://guide.duo.com. Accessed on 2022-02-11.

[3] Mete Akgün and M Ufuk Çaglayan. Providing destructive privacy and scalability in rfid systems using pufs. *Ad Hoc Networks*, 32:32–42, 2015.

[4] Pol Van Aubel, Daniel J Bernstein, and Ruben Niederhagen. Investigating sram pufs in large cpus and gpus. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 228–247. Springer, 2015.

[5] Pyi Phyo Aung, Koichiro Mashiko, Nordinah Binti Ismail, and Ooi Chia Yee. Evaluation of sram puf characteristics and generation of stable bits for iot security. *Advances in Intelligent Systems and Computing*, page 441–450, 2019.

[6] Sergey Babkin and Anna Epishkina. Authentication protocols based on one-time passwords. *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2019.

[7] Rohitash Kumar Banyal, Pragya Jain, and Vijendra Kumar Jain. Multi-factor authentication framework for cloud computing. In *2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation*, pages 105–110. IEEE, 2013.

[8] Pourya Bayat-Makou, Ali Jahanian, and Media Reshadi. Security improvement of fpga design against timing side channel attack using dynamic delay management. In *2018 IEEE Canadian Conference on Electrical Computer Engineering (CCECE)*, pages 1–4, 2018.

[9] Leila Benarous, Benamar Kadri, and Ahmed Bouridane. A survey on cyber security evolution and threats: biometric authentication solutions. In *Biometric security and privacy*, pages 371–411. Springer, 2017.

[10] Abhilasha Bhargav-Spantzel, Anna C Squicciarini, Shimon Modi, Matthew Young, Elisa Bertino, and Stephen J Elliott. Privacy preserving multi-factor authentication with biometrics. *Journal of Computer Security*, 15(5):529–560, 2007.

[11] K. Bicakci and N. Baykal. Infinite length hash chains and their applications. *Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2002.

[12] Lilian Bossuet, Xuan Thuy Ngo, Zouha Cherif, and Viktor Fischer. A puf based on a transient effect ring oscillator and insensitive to locking phenomenon. *IEEE Transactions on Emerging Topics in Computing*, 2(1):30–36, 2014.

[13] Harishma Boyapally, Paulson Mathew, Sikhar Patranabis, Urbi Chatterjee, Umang Agarwal, Manu Maheshwari, Soumyajit Dey, and Debdeep Mukhopadhyay. Safe is the new smart: Puf-based authentication for load modification-resistant smart meters. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[14] An Braeken. Puf based authentication protocol for iot. *Symmetry*, 10(8):352, 2018.

[15] Intel Corporation. Intel®nbsp;stratix®nbsp;10 fpgas overview - high performance stratix® fpga, 2018.

[16] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176.

[17] Ulusal Elektronik and Kriptoloji Araştırma Enstitüsü Dergisi Cilt. Sayı: 3 "tempest, tempest'in keşfi ve sinyal analizi, değerlendirme kriterleri ve ölçüm sistemleri, cihaz tasarımı", 2010.

[18] Steven Feltner. Single-factor authentication (sfa) vs. multi-factor authentication (mfa), Dec 2016.

[19] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE transactions on information forensics and security*, 8(1):136–148, 2012.

[20] Santosh Ghosh, Monjur Alam, Dipanwita Roy Chowdhury, and Indranil Sen Gupta. Effect of side channel attacks on rsa embedded devices. In *TENCON 2007-2007 IEEE Region 10 Conference*, pages 1–4. IEEE, 2007.

[21] Intrinsic ID GlobalSignVideos. *Strong Device Identities Through SRAM PUF-based Certificates | Webinar*. YouTube, Sep 2017.

[22] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W.

Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, May 2009.

[23] Neil Haller. Rfc1760: The s/key one-time password system, 1995.

[24] Ersin Hatun, Elif Büyükkaya, and Sıddıka Berna Örs Yalçın. Electromagnetic radiation analysis of implementation of rsa algorithm on a raspberry pi. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2018.

[25] Ryan Helinski, Dhruva Acharyya, and Jim Plusquellic. A physical unclonable function defined using power distribution system equivalent resistance variations. *Proceedings of the 46th Annual Design Automation Conference on ZZZ - DAC '09*, 2009.

[26] Charles Herder, Meng-Day Yu, Farinaz Koushanfar, and Srinivas Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.

[27] Gabriel Hospodar, Roel Maes, and Ingrid Verbauwhede. Machine learning attacks on 65nm arbiter pufs: Accurate modeling poses strict bounds on usability. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 37–42, 2012.

[28] Intrinsic ID. Company - intrinsic id: Home of puf technology, Feb 2022.

[29] Jae-Jung Kim and Seng-Phil Hong. A method of risk assessment for multi-factor authentication. *Journal of Information Processing Systems*, 7(1):187–198, 2011.

[30] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[31] Paul Kocher, Joshua Jaffe, Benjamin Jun, et al. Introduction to differential power analysis and related attacks, 1998.

[32] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[33] Dmitry Kogan, Nathan Manohar, and Dan Boneh. T/key: second-factor authentication from secure hash chains. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[34] Radhesh Krishnan Konoth, Victor van der Veen, and Herbert Bos. How anywhere computing just killed your phone-based two-factor authentication. In *International conference on financial cryptography and data security*, pages 405–421. Springer, 2016.

[35] Takaya Kubota, Kota Yoshida, Mitsuru Shiozaki, and Takeshi Fujino. Deep learning side-channel attack against hardware implementations of aes. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pages 261–268, 2019.

[36] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[37] Jae W Lee, Daihyun Lim, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No. 04CH37525)*, pages 176–179. IEEE, 2004.

[38] Andrew Y. Lindell. Time versus event based one-time passwords. Jul 2008.

[39] Roel Maes and Ingrid Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. *Information Security and Cryptography*, page 3–37, 2010.

[40] Mahabub Hasan Mahalat, Dipankar Karmakar, Anindan Mondal, and Bibhash Sen. Puf based secure and lightweight authentication and key-sharing scheme for wireless sensor network. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(1):1–23, 2021.

[41] Pramod Kumar Maurya and Satya Bagchi. A secure puf-based unilateral authentication scheme for rfid system. *Wireless Personal Communications*, 103(2):1699–1712, 2018.

[42] Thomas McGrath, Ibrahim E. Bagci, Zhiming M. Wang, Utz Roedig, and Robert J. Young. A puf taxonomy. *Applied Physics Reviews*, 6(1):011303, 2019.

[43] David M'Raihi, Mihir Bellare, Frank Hoornaert, David Naccache, and Ohad Ranen. Hotp: An hmac-based one-time password algorithm. *The Internet Society, Network Working Group. RFC4226*, 2005.

[44] David M'Raihi, Salah Machani, Mingliang Pei, and Johan Rydell. Totp: Time-based one-time password algorithm. *Internet Request for Comments*, page 685E, 2011.

[45] n.a n.a. Otp, totp, hotp: What's the difference?: Onelogin, 0AD.

[46] n.a n.a. One-time password (otp) – more security online, Oct 2020.

[47] AIMEE O'DRISCOLL. 25+ password statistics that may change your password habits, Jan 2022.

[48] Levent Ordu. *AES algoritmasının FPGA üzerinde gerçeklenmesi ve yan kanal analizi saldırılarına karşı güçlendirilmesi*. PhD thesis, 2006.

[49] Ahmed Oun and Mohammed Niamat. Design of a delay-based fpga puf resistant to machine learning attacks. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 865–868. IEEE, 2021.

[50] Muhammet Öztemür. Aes algoritmasının bir gerçeklemesine güç analizi saldırıları. 2012.

[51] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.

[52] Woo-Suk Park, Dong-Yeop Hwang, and Ki-Hyung Kim. A totp-based two factor authentication scheme for hyperledger fabric blockchain. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 817–819. IEEE, 2018.

[53] Thanasis Petsas, Giorgos Tsirantonakis, Elias Athanasopoulos, and Sotiris Ioannidis. Two-factor authentication: is the world ready? quantifying 2fa adoption. In *Proceedings of the eighth european workshop on system security*, pages 1–7, 2015.

[54] Posted byPayam Pourkhomami. Single-factor authentication risk: Why it is bad practice, Oct 2021.

[55] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 237–249, 2010.

[56] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. Puf-enhanced rfid security and privacy. In *Workshop on secure component and system identification (SECSI)*, volume 110, 2010.

[57] Edward M Scheidt and Ersin Domangue. Multiple factor-based user identification and authentication, January 18 2005. US Patent 6,845,453.

[58] Bruce Schneier. Two-factor authentication: too little, too late. *Communications of the ACM*, 48(4):136, 2005.

[59] Aykut Sevim, Hamdi Altıner, O Serkan Ünek, and Mehmet Şam. Kurumsal yapılarda bilişim güvenliği, tempest problemi.

[60] Mariano Luis T Uymatiao and William Emmanuel S Yu. Time-based otp authentication via secure tunnel (toast): A mobile totp scheme using tls seed exchange and encrypted offline keystore. In *2014 4th IEEE International Conference on Information Science and Technology*, pages 225–229. IEEE, 2014.

[61] IngridVerbauwhedeandRoelMaes. Physicallyunclonablefunctions: manufacturing variability as an unclonable device identifier. In *Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI*, pages 455–460, 2011.

[62] John Ross Wallrabenstein. Practical and secure iot device authentication using physical unclonable functions. *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2016.

[63] Ding Wang, Debiao He, Ping Wang, and Chao-Hsien Chu. Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment. *IEEE Transactions on Dependable and Secure Computing*, 12(4):428–442, 2014.

[64] Sying-Jyan Wang, Yu-Sheng Chen, and Katherine Shu-Min Li. Modeling attack resistant pufs based on adversarial attack against machine learning. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(2):306–318, 2021.

[65] SB Ors Yalçın. *Hardware design of elliptic curve cryptosystems and side-channel attacks*. PhD thesis, PhD Thesis, Katholieke Universiteit Leuven, Belçika, 2005.

[66] Xinming Yin, Junhui He, Yi Guo, Dezhi Han, Kuan-Ching Li, and Arcangelo Castiglione. An efficient two-factor authentication scheme based on the merkle tree. *Sensors*, 20(20):5735, 2020.

[67] Seungyong Yoon, Byoungkoo Kim, Yousung Kang, and Dooho Choi. Puf-based authentication scheme for iot devices. *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020.

[68] Kim Zetter. Rsa agrees to replace securitytokens after admitting compromise. *Threat Level, Privacy, Crime and Security Online*, 2011.