A RSA-BIOMETRIC BASED USER AUTHENTICATION SCHEME FOR SMART-HOMES USING SMARTPHONE

by

Amir Mohammadi Bagha

B.Sc. in Computer Science, Amirkabir University of Technology, Iran, 2017

A Thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Computer Networks

Toronto, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

AN RSA-BIOMETRIC BASED USER AUTHENTICATION SCHEME FOR SMART-HOMES USING SMARTPHONES

Master of Applied Science

in Computer Networks

Ryerson University

Abstract

Internet of Things (IoT) is considered as one of the emerging leading technologies that allow the mainstreaming of smart homes and smart cities in the recent years, by creating a communication system for physical objects over the Internet. In a smart home (also called push-button home automation system), devices are not necessarily homogeneous in terms of topology, security protocols, computational power and communication. This nature of the devices causes some incompatibility with conventional authentication methods and the security requirements of IoT standards. This thesis proposes an RSA-Biometric based three-factor User Authentication Scheme for Smart-Homes using Smartphone (called RSA-B-ASH-S scheme). An informal security analysis of the proposed RSA-B-ASH-S scheme is provided, along with its performance evaluation in terms of computational time, storage requirements and communication overload. Furthermore, a formal analysis of the proposed RSA-B-ASH-S scheme using the Burrows-Abadi-Needham (BAN) Logic is described, showing that the proposed scheme achieves the forward secrecy property by utilizing a fresh encryption key for each session and it also satisfies the anonymity of the user by using a one-time token. A proof of concept of the proposed RSA-B-ASH-S scheme is also provided.

Acknowledgments

I would first like to thank my thesis advisor Dr. Isaac Woungang for his passionate guidance, continuous support and input. The door to Prof. Woungang's office was always open for guidance. He provided me with enough room to explore while steering me back to the right direction whenever I needed it. I am also thankful to the Department of Electrical and Computer Engineering at Ryerson University for their continuous support toward the completion of my degree, special thanks to Maninder Singh Raniyal, who helped me through the process of researching and writing this thesis, and our program administrator, Ting Hong, for her instantaneous support throughout my studies.

I must express my very profound gratitude to my parents, whose love and guidance are with me in whatever I pursue. Most importantly, I wish to thank my loving and supportive wife for her incomparable encouragement throughout my years of study and each and every challenge along the way. I am also grateful to have friends that provided me with such encouragement that allowed me to stay motivated throughout my studies.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

IoT	Internet of Things
OTP	One Time Password
D2D	Device to Device
DH	Diffie-Hellman
RSA	Rivest-Shamir-Adleman public-key cryptosystem
DoS	Denial-of-Service
DDoS	Distributed Denial of Service
MIMA	Man-in-the-middle Attack
BAN	Burrows-Abadi-Needham logic
RSA-ASH-SC	RSA-Based Two-Factor Remoted User Authentication Scheme for Smart-Home using Smart Card
iOS	iPhone Operating System (Apple)
PHP	PHP Hypertext Preprocessor
PFS	Perfect Forward Secrecy
CPU	Central Processing Unit
RAM	Random-Access Memory
AES	Advanced Encryption Standard
API	Application Programming Interface
HS	Home Serv

Chapter 1

Introduction

This Chapter presents the motivation and context of the research carried out in this thesis. It also presents the research problem, solution approach and our contributions.

1.1 Motivation and Research Problem

Internet of Things (IoT) [1] is becoming a prominent technology which makes the communication of physical objects possible over the internet. These devices (formally referred to as objects) have communication systems in place that allow them to monitor, share and collect data about their surrounding physical world. It is required that these devices initially authenticate themselves with each other using IoT-based communication protocols and/or cloud platforms. Nonetheless, the security protocols that these systems use are mostly deemed to be incompatible with IoT security requirements, due to the fact that they rely on single-factor authentication schemes [2]. One of the other authentication methods for IoT devices in the cloud is the one-time password (OTP). It has been made reported that this scheme also fails to fulfill the requirements of IoT and smart homes, due to the nature of its inflexible design.

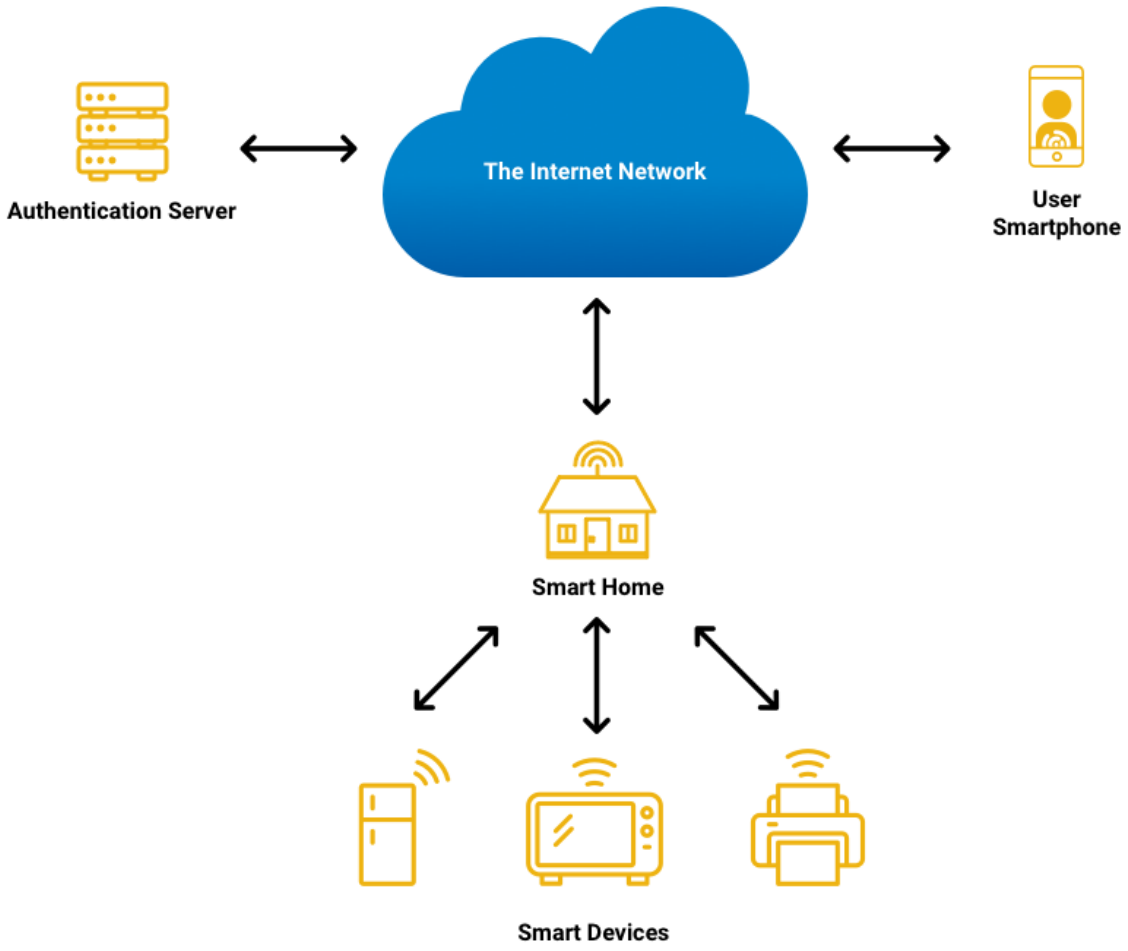

Figure 0.1: Typical Smart Home Network Architecture in IoT.

With the growth of the IoT and smart home technologies and the inevitable attention of cyber intruders and attackers to this sector, the security of the networks implemented in these systems become increasingly crucial. As the same time, smartphones have been equipped with various technologies that create new opportunities in terms of security and user authentication. The RSA-ASH-SC [3] scheme is a two-factor authentication protocol concentrating on user authentication for smart homes by utilizing smart cards (here denoted RSA-ASH-SC). This scheme, which our proposed scheme is inspired from, takes advantage of RSA security protocols and adds a second authentication layer using smart cards. One of the deficiencies of the RSA-

ASH-SC scheme is that it does not satisfy the perfect forward secrecy, which is becoming a significant security feature considering the present standards. Perfect forward secrecy means that even when long-term private key of the sender or receiver is leaked, the encrypted messages that were previously transmitted will not be exposed [4]. The RSA-ASH-SC scheme is based on asymmetric encryption for the most part, which is less viable compared to symmetric encryption, in terms of computational time. Additionally, the use of smart card authentication methods is likely to become less convenient considering the extensive adaptation of smartphones and their potential in substituting the smart cards. Our proposed RSA-Biometric Based User Authentication Scheme for Smart-Homes Using Smartphones (denoted RSA-B-ASH-S) is an enhancement to the RSA-ASH-SC scheme in the sense that it is shown to satisfy the perfect forward secrecy. Our scheme utilizes the same proposed asymmetric encryption of the RSA-ASH-SC scheme as its underlying encryption method to achieve mutual authentication and to initiate authentication. The proposed scheme not only intends to support all security features satisfied in the RSA-ASH-SC, but also seeks to meet the requirements of perfect forward secrecy. By taking advantage of current smartphone capabilities, this scheme also intends to add a third layer to the authentication process, using a biometric factor such as fingerprint or face recognition verification. The proposed scheme is to be compared to preceding related user authentication schemes against the conventional security metrics and total computational time for both authentication and login phase, separately and combined. It is also intended to assess the storage requirements and communication overloads of the proposed scheme, and validate the scheme using BAN logic [5].

1.2 Solution Approach

In order to achieve the desired security enhancements over the RSA-ASH-SC scheme, our RSA-B-ASH-S scheme is required to satisfy the perfect forward secrecy. The incorporation of both RSA

and symmetric encryption into the Diffie-Hellman (DH) key exchange protocol allows us to create a fresh encryption key for each session, create mutual authentication, and as a result, satisfy perfect forward secrecy. Another enhancement intended in our proposed scheme is the use of smartphones in substitution of smart cards, which not only creates the opportunity to obtain more hardware resources, but also enables us to add a third security layer to the scheme, using a biometric factor. The biometric factor used in this scheme is face recognition, which has a broader compatibility range compared to other factors such as fingerprint recognition. The verification of this layer is also required to be independently implemented, since the access to face recognition raw data is not yet granted in present operating system protocols. It should be noted that the intention of our approach is not to reduce the total computational time and power, but to elevate the overall security of the scheme.

1.3 Thesis Contributions

The contributions of this thesis are as follows:

- Design of an RSA-based three-factor user authentication scheme (called RSA-B-ASH-S) for smart homes using smartphones (i.e. what you have factor), a biometric factor (i.e. who you are factor) and a password (i.e. what you know factor).
- Informal security analysis of the proposed RSA-B-ASH-S scheme.
- Formal analysis of the proposed RSA-B-ASH-S scheme using the Burrows-Abadi-Needham (BAN) Logic paradigm.
- Implementation of a proof-of-concept of the proposed RSA-B-ASH-S scheme in the form of a hardware solution.

1.4 Thesis Outline

The thesis is organized as follows:

- Chapter 1 introduces the motivation and contributions of this thesis.
- Chapter 2 provides some background and related work to the subject topic of this thesis.
- Chapter 3 describes the design of the proposed RSA-B-ASH-S scheme.
- Chapter 4 presents the security analysis and performance evaluation of the proposed RSA-B-ASH-S scheme.
- Chapter 5 concludes the thesis and highlights some future work that can be carried further.

Chapter 2

Background and Related Work

2.1 Background

2.1.1 Authentication Schemes

Authentication can be defined as the process by which two communicating parties can be correctly identified over an unsecured channel to prevent illegal access to system resources [2]. In terms of user authentication, there are several ways of authenticating the identity of a user: (1) Something the user knows (such as password, personal identification number), (2) Something the user possesses (such as smart card, physical keys, tokens), (3) Something the user is (such as recognition by fingerprint, face, retina, iris), and (4) Something the user does (such as recognition by voice pattern, handwriting style, typing rhythm, and (5) A combination of the above.

Typically, there are two types of authentication schemes, namely: Single server-based and multi-server-based. In Single server-based authentication schemes, only a single server is used, which provides services to all users; and in case a user wishes to gain access to multiple network services, he/she should register his/her identity and password in different servers and maintain several user IDs and passwords, which is difficult to remember. On the other hand, in a multi-server-based authentication scheme, the users can obtain the services from multiple servers without the need of registering with each server. This solves the problem of multiple registrations encountered in the single server scenario. In a nutshell, a multi-server-based authentication scheme consists of several service servers, a Registration Centre, and multiple users in a multi-server

communication system. The Registration Centre is responsible for administering the service servers along with the registered users, in such a way that upon registration, each user is able to get the requested services from the different service servers.

The assessment of an authentication scheme can rely on several metrics, which include (1) the security features that the scheme is intended to provide and the expected types of attacks it has been designed to resist against, (2) its generated computation cost, i.e. the number of operations or the estimated time (in seconds) needed by the entities involved in order to compute the authentication messages during the execution of the scheme, and (3) its communication cost, which represents the channel overhead (in terms of number of bits) due to the transmission, by the entities involved, of the authentication messages in order to prove their identities. The main goals of an authentication scheme are to resist to a maximum possible number of available attacks while maintaining low computation and communication costs. In this thesis, our focus is on user authentication for smart homes, aiming at achieving such goals.

2.1.2 Authentication Schemes for Smart Home Environments

An example illustrating a smart home environment is shown in Fig, 2.1, where smart objects such as smart doors, smart fridges, smart TV, smart cameras, smartphones, and etc. are equipped with sensors capable of monitoring and handling the data between them through a Home Gateway (HG), known as an interface between the Internet and the home network. This HG is connected to the user interface by means of a software or a well-mounted solution. It should be noted that in some implementations, a home server (HS) is also utilized for the purpose of devices' authentication. In terms of communication technologies used in smart homes, common radio wave technologies include Ethernet, X10, 6LoWPAN, RS-485, ZigBee, Bluetooth LE, Z-Wave and Li-Wi [6].

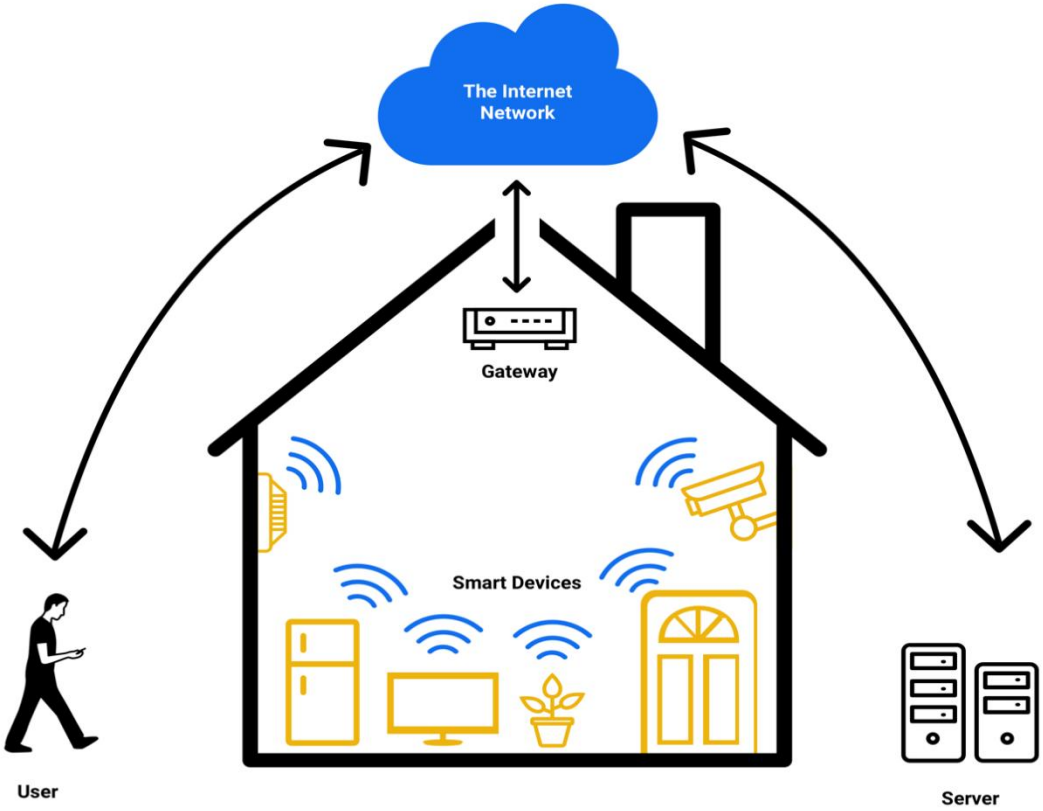

Figure 2.1: Example of a smart home.

Nowadays, with the rise of the Internet of Things (IoT) [1] as a new paradigm and driver for many applications such as smart cities, smart homes, to name a few, most of the literature on authentication schemes focuses on inter-device authentication and remote user authentication. The later can be defined as a mechanism by which a remote user is authenticated when attempting to log on a network.

As far as remote authentication in smart home environments is concerned, which is our focus of this thesis, several techniques have been explored in the literature [2], which include: password-based methods [7], digital memory-based methods [8], single-sign-on techniques based on the Kerberos protocol [9], smart cards-based approaches [10], biometrics-based approaches

[11], techniques based on Near Field Communications (NFC) [12], RSA-based techniques [13], to name a few. The RSA-B-ASH-S authentication scheme proposed in this thesis falls to this latter category of protocols since it is an RSA-based three-factor remote user authentication scheme for smart homes using smartphones (*i.e. what you have*), a biometric factor (*i.e. who you are*) and a password (*i.e. what you know*), to process the user's authentication. This RSA-B-ASH-S scheme is designed as an enhancement to the RSA-ASH-SC scheme [13] in terms of security since it is proved that it satisfies the Perfect Forward Secrecy (PFS) property, which the RSA-ASH-SC scheme was unable to achieve.

2.1.3 Considered Security Attacks and Requirements of the Proposed Authentication Protocol

Designing an authentication scheme for a smart home network such as the one depicted in Fig. 2.1, it is required that some minimal security requirements be met, which include: confidentiality, integrity, availability, and mutual authentication.

- *Confidentiality:* in smart homes, the privacy of the data exchanged between the users must be maintained in such a way that only authorized users to have access to sensitive data. In this case, cryptography techniques are recommended as solutions to secure data transactions.
- *Availability:* in smart homes, authorized users should have access to their services at all times. Therefore, the authentication scheme must be designed to counter specified types of attacks against availability such as Denial-of-Service (DoS) attacks.
- *Integrity:* in a smart home, a mechanism should be designed to ensure that the data exchanged between entities are not manipulated by an adversary.

- *Mutual authentication* (also called *two-way authentication*): this is achieved when two entities involved in the authentication process validates each other's identity. When doing so, an adversary may attempt to spoof the server by sending false information or initiating an attack. Therefore, in a smart home, the authentication scheme must be designed in such a way as to achieve a secure mutual authentication while protecting against the set of predefined types of attacks based on its design features.

In this thesis, the security attacks that are considered are: masquerading attack, replay attack, Denial of Service (DoS) & Distributed Denial of Service (DDoS) attacks, Man-in-the-middle (MITM) attack, Password-guessing attack, forward secrecy attack, Device loss attack, and Forgery attack. To evaluate the performance of the proposed RSA-B-ASH-S authentication scheme, the following security metrics are considered: computational time (in seconds), storage requirements (in bits) and communication overload (in bits). Furthermore, a proof of concept of the proposed RSA-B-ASH-S scheme is provided, along with its formal analysis using the Burrows-Abadi-Needham (BAN) Logic paradigm [5], showing that it satisfies the forward secrecy property by utilizing a fresh encryption key for each session. A proof of concept of the proposed RSA-B-ASH-S scheme is also given.

2.2 Related Work

2.2.1 Authentication Schemes for Smart Home Environment

In some of the papers described in this section, the following notations are used:

Table 2.1: Notations used in the related papers.

Notation	Description
CS	Control server
x	Master secret key chosen by CS
$\|\|$	Message concatenation operation
SID_i	Identity of SID i
GW	Home gateway node
RA	Registration authority
SDj	Smart device node
OTT_i	i^{th} user's One-Time-Token (OTT if no ambiguity)
P_i	Password of user i
$\oplus$	Exclusive OR operation
r_i	Random number generated by the smart card
T	Current timestamp
$mod()$	Modulus operation
f	Collision free one-way hash function
g	An integer and primitive element for prime factors of the private key of RSA

d	Private key components of RSA
(e, n)	Public key components of RSA
S_i	User's *i* secret information
h_i	Secret information created by the smart card issuer at the registration time
CID_i	Smart card identifier
RBF	Radial Basis Function
ID_i	i^{th} user ID
PW_i	i^{th} user's password (or PW if no ambiguity)
U_i	i^{th} user
CRT	Chinese Remainder Theorem

In [14], Yang and Shieh proposed a timestamp based and a nonce-based password authentication schemes which are based on smart cards. In each of these schemes, (1) it is not required that the password and registration tables be kept at each host, (2) the (ID_i, CID_i, h_i, S_i, e, g, n) information is recorded in the smart card, and to login the system, a user has to calculate the entities $X_i \equiv g^{r_i.pw_i} \pmod n$ and $Y_i \equiv S_i . h_i^{r_i.f(CID_i.T)} \pmod n$, then send the vector ($ID_i$, CID_i, X_i,Y_i, e, g, n, T) to the host. Upon receipt of this information, the host verifies that the equation $Y_i^e = ID_i.X_i^{f(CID_i.T)}$ holds after checking the received ID_i, CID_i and timestamp T. If that equation is satisfied, it is said that the user is authenticated; otherwise the authentication of the user has failed. A cryptanalysis of this scheme by Chan and Cheng [15] has revealed that it is vulnerable to forgery

attacks. Another cryptanalysis of the same scheme by Fan et al. [16] demonstrated that it is also vulnerable to impersonation attack. Based on these findings, an enhanced Yang and Shieh scheme was proposed [16], in which forgery can only be performed with a valid CID_i.

In [17], Liu et al. proposed a remote user authentication protocol for smart homes as a result of a cryptanalysis of the scheme proposed by Shen et al. [18] Their proposed scheme relies on the establishment of a relationship at the registration phase to check the legitimacy of the user at login time. Essentially, upon receipt of a login request from a user at the registration phase, the server checks the validity of ID_i and calculates the aforementioned relationship $CID_i = f(ID_i \oplus d)$ so that when the server receives the login request, it checks the validity of ID_i and uses it to calculate $CID'_i = f(ID_i \oplus d)$ and if $CID'_i = CID_i$, then, the user is said to be authenticated.

In [3], Raniyal proposed an RSA-based two-factor user authentication scheme using smart card (called RSA-ASH-SC). In this scheme, the Rebalanced-Multi-Power RSA scheme [10] is used as underlying RSA algorithm. It is proved that the RSA-ASH-SC scheme is 50% faster than the Om and Kumari scheme [19]. In the RSA-ASH-SC scheme the user generates the following values: $x = (f((f(PW) \oplus ID))^e \oplus f(T)) mod N$, $HXOTT = f(x \oplus OTT)$ and $y = (OTT \| T \| S_i \| f(OTT \| T \| S_i \| HXOTT) \| HXOTT)$, then it sends the (OTT, C) value to the server where $C = y^e mod(n)$. Then the server validates OTT and the timestamp and then computes $x = ((HPWID)e \oplus f(T)) modN$ and $Z = f(x \oplus OTT)$. If Z and $HXOTTi$, the extracted value from the decrypted message, are equal, then the user is said to be authenticated successfully; otherwise the authentication has failed.

In [19], Om and Kumara proposed a cryptanalysis of the remote user authentication scheme introduced in [10] (so-called Om and Reddy scheme). Based on this analysis, an enhanced secure version is designed as follows. The registration step involves the calculation of additional sensitive information $S_i = f(PW_i||ID_i)$ that the user should store in the smart card before attempting the login to the system. Afterwards, to login to the system, the user calculates the entities $x = f(f(PW_i||ID_i \oplus T) \, mod \, n, \; y = ID^x{}_i (mod \, n)$ and $C = y^e (mod \, n)$, then sends the vector (ID_i, C, T) to the server. When the server receives this vector, it verifies ID_i and the timestamp T, then it calculates $x = f(S_i \oplus T) \, mod \, n$, $y = ID^x{}_i \, mod \, n$ and $K = C^d mod \, n$. If $K = y$, the server has successfully authenticated the user; otherwise the authentication has failed. It should be emphasized that if a smart card loss attack is issued and is successful, the information S_i can be leaked, which may lead to compromised security.

In [20], Wazid et al. proposed a secure remote user authentication scheme for smart home environments, which utilizes bitwise XOR operations, one-way hash functions and symmetric encryptions and decryptions to achieve three types of mutual authentications: between the user and the home gateway node, the home gateway node and the smart device and the user and the smart device. In this scheme, each user is equipped with a smart phone capable of reading its credential information (identity, password and biometric). The user's request for authentication is handled by the home gateway node, which forwards it to the smart card. The response of this request is sent back to the home gateway node, which forwards it to the user. In this process, a registration authority is available, which securely registers the home gateway node and each smart device offline prior to activating the authentication system. Next, a user who wishes to access the smart devices has to first register at the registration authority by providing its credential information. After registration, the proposed scheme performs the following steps: user's authentication and

agreement, user's biometric and password update and verification using a fuzzy extractor, then allowance of access to the resources of the system to the user if the authentication was successful.

In [21], Reyhani and Mahdavi proposed a two-steps user authentication scheme for smart home network, which is based on the Radial Basis Function (RBF) neural network. In the registration step, the user's username and password information are hashed (using a one-way hash function) and utilized as patterns to train the RBF neural network. The authentication step consists in getting the system to perform the following tasks: (1) apply the same hash function on an entered username and password, (2) extract an output of the trained RBF neural network, and (3) compare this output against the hashed password. If there is a match, the user is said to be authenticated; otherwise, the user is qualified as illegal.

In [22], Bae and Kwak proposed a user authentication protocol in IoT environment using smart card, which is composed of three phases: (1) Registration phase – where the user and IoT server both request for registration to a third trusted party (the authentication server), which in response sends a smart card to the user as well as the secret information that it has stored and which will be needed for the login and authentication steps (encrypted password $(EncPass_i)$, $h(EncPass_i)$. In the Login and authentication phase, the server verifies the identity of the user upon request for login, then issue a session key if the user and server are both confirmed as legitimate entities. Only thereafter, the mutual authentication between the server and the user can be performed, which involves the value generated by the user and the $h(x)$ value contained in the smart card. It should be noted that a password change phase is also implemented which accounts for the situation where the user wishes to change his/her password to a new one.

In [23], Dammak et al. proposed a token-based lightweight user authentication for IoT devices (called TBLUA), which is based on a token-based method to improve the robustness of the authentication. Their scheme consists of offline smart device and home gateway registration, user registration, and token distribution between the home gateway and smart devices. Upon completion of these steps, the user can log in to the system, triggering the authentication phase, with the goal that the user be authenticated as legitimate or illegal.

In [24], Amin et al. proposed a two-factor RSA-based user authentication protocol for multi-server environments, which consists of the following steps: (1) initialization step (achieved by a trusted RC) - meant for the registration of the user U_i and the application server AS_j, (2) user login step using the smart card in which its ID and password are stored, (3) verification step - in which a session key agreement initiated by the RC is established between the U_i and AS_j, and (4) password change step – to accommodate possible changes in the registered user's password.

In [25], Zhao and Jiang proposed a cryptanalysis of the Truong et al. [26] authentication protocol based on which an enhanced scheme is derived, which can protect against server impersonation and offline password guessing attacks in a multi-server environment. Their scheme is made of the following steps: initialization and registration (which is the responsibility of the RC), authentication (between the user and the service provider), and password update phase. Furthermore, the Diffie-Hellman key exchange protocol [27] is used for key agreement and forward secrecy purposes, and the timestamps and freshness of random numbers are used to prevent against replay attacks.

In [28], Dhillon and Kalra proposed a three-factor user authentication scheme for IoT environments based on password, biometrics and smart device. Their scheme is made of: (1)

registration step – where the user and home gateway must register, (2) login step – if successful, this phase gives the user the to an IoT node and the desired resources, (2) mutual authentication – where both the user and the IoT node generates an encrypted session key based on some necessary parameters generated by the gateway node, which enables a secure communication between the user and the IoT node; and (3) password change phase – which enable password updates when needed by the user.

Unlike the aforementioned schemes, in the proposed RSA-B-ASH-S authentication scheme, symmetric encryption is utilized, and the RSA method is only used in the initial authentication phase for mutual authentication purposes and for preventing the Man-in-the-middle (MITM) attack. To satisfy perfect forward secrecy, our proposed scheme uses a fresh shared key for each session; therefore, even if the private key of the server is compromised, the attacker cannot access to any of the previous sessions' plaintext data.

2.2.2 RSA Public Key Cryptosystem

Our RSA-Biometric three-factor User Authentication scheme is an enhancement of the RSA-ASH-SC scheme [3], in which RSA [29] was primarily used as underlying algorithm. This algorithm contains three primary steps: key generation, encryption and decryption.

Key Generation Step: First, *n* the input to this algorithm, which is a security parameter also known as the length of the RSA, is given. Then, two prime numbers represented by p and q are chosen, each with half of *n*'s length long in bits. After that, $N = p.q$ is computed and the Euler's totient function $\varphi(N) = (p-1).(q-1)$ is calculated. Then, e the encryption component is chosen such that $gcd(e, \varphi(N)) = 1 < and < 1 < e < \varphi(N)$. Lastly, a private key $d = e - 1\ mod(\varphi(N))$ is generated such that the public key is (e, N) and private key is (d, N).

Encryption: Either of the private or public keys can be used to encrypt the message. However, for decryption, a second key is required. The message M should be an integer between 1 to N – 1, otherwise, it is supposed to be formatted in a way to satisfy this condition. Therefore, the Public Key Cryptography Standards, Number 1, known as PKCS#1 standard [30] is commonly used in this step, and M is encrypted as $C = M^e mod(N)$, where C is the ciphertext.

Decryption: The private components d and N are used for the decryption of C. To retrieve the original message, $C^d mod(N)$ is calculated. This calculation can be computationally exhaustive since both d and N are sizable numbers. The Chinese Remainder Theorem has been used as a solution to increase the speed of the decipher algorithm, which can increase the speed of the process up to four times [30].

2.2.3 RSA-ASH-SC User Authentication Scheme

Our proposed RSA-B-ASH-S protocol is inspired from the design of the RSA-ASH-SC scheme introduced in [3]. The latter is a two-factor authentication protocol for user authentication in smart homes using smart cards, which utilizes the Rebalanced-Multi-Power RSA [10] algorithm for asymmetric encryption. It should be noticed that although such choice of RSA-based encryption provides a public and a private key, it is much more costly than using a symmetric encryption such as AES [31] from a computational performance perspective. In the RSA-ASH-SC scheme, after the initialization and registration phases are completed, the login and authentication phases are carried out as follows.

Login phase: The smart card which functions as one of the two security factors generates $x = (f((f(PW_i) \oplus ID_i))^e \oplus f(T))\ mod\ N$, and $y = (OTT_i \parallel T \parallel S_i \parallel f(OTT_i \parallel T \parallel S_i \parallel f(x \oplus OTT_i)) \parallel f(x \oplus OTT_i))$. Then, encrypts y as $C = y^e mod\ (N)$ and sends (OTT_i, C) to the server.

Authentication phase: First, the server validates the received OTT_i, then decrypts C and extracts y using Chinese Remainder Theorem (CRT). Afterwards, it checks for the freshness and integrity of the message. Then, it computes $x\ as\ ((f(f(PW_i) \oplus ID_i))^e \oplus f(T))\ mod\ N$ and $Z\ as\ f(x \oplus OTT_i)$ and compares the Z value with the received $f(x \oplus OTT_i)$ to authenticate the request. Next, the server generates a new random token OTT_{new} for future authentication purposes and updates the corresponding database entry. It also creates the current timestamp T_{new}, and sends $(T_{new,}\{f(T_{new} \parallel OTT_{new}), OTT_{new}\}_{S_i})$ to the user as the response where, $\{\}_{S_i}$ is the symmetric encryption/decryption function with S_i as the key. The user then decrypts the received message with S_i and checks whether the message is new or not. Finally, the user replaces the existing one-time token with OTT_{new} in the card, and uses S_i for future communications.

Our primary motivation in developing the proposed scheme is to achieve the Perfect Forward Secrecy. To do so, we aim to make use of the DH algorithm [27] which creates the possibility of having a new key for each session. The underlying RSA algorithm in our proposed scheme is identical to that used in the RSA-ASH-SC scheme, however, we tried to shift the weight of the cryptography process from RSA to symmetric encryption. The proposed scheme also uses a smartphone as an alternative to the smart card since it is not only deemed to be more practical, but also it provides the possibility of adding a third layer to the authentication process using a biometric factor such as, such as fingerprint or face recognition verification. The detailed methodology of our proposed scheme is discussed thoroughly in Chapter 3.

Chapter 3

Proposed Three-Factor User Authentication Protocol for Smart Homes

3.1 High-Level Workflow of the Proposed RSA-B-ASH-S Authentication Scheme

To demonstrate the data exchange in the proposed scheme we have considered the notations given in Table 3.1.

Table 3.1: Notations used in the proposed scheme.

Notation	**Description**
$\phi(N)$	Euler's totient
n	Input security parameter for key generation algorithm
k	Distinct prime numbers in RSA key generation
s	Size of prime numbers in RSA key generation (Rebalanced)
p, q	Prime numbers used in RSA key generation
e	Encryption exponent
d	Decryption exponent
$mod()$	Modulus operation
gcd()	Greatest common divisor

$\oplus$	XOR operation
$h(),f()$	One-way Hash function
ΔT	Threshold time used to prevent replay attack
$\{\}_x$	Symmetric key encryption/decryption. Here x is symmetric key
U_i	i^{th} user
ID_i	i^{th} user's ID
PW_i	i^{th} user's password
B_i	i^{th} User's biometric impression
OTT_i	i^{th} user's One-Time-Token
Θ	A predetermined threshold for biometric verification
$\sigma()$	A symmetric parametric function for biometric factor comparison
p_{dh}	Large prime (usually larger than 1024 bits) for Diffie Hellman key exchange algorithm
n_{dh}	Size of p_{dh} in bits
a_i	Session independent random exponent chosen by i^{th} user
b_i	Session independent random exponent chosen by the server to communicate with i^{th} user
g	Generator of $Z_{p_{dh}}$
sk_i	Session key for communication of i^{th} user and the server
tsk_i	Session independent temporary key chosen randomly by the i^{th} user to encrypt the communication with the server

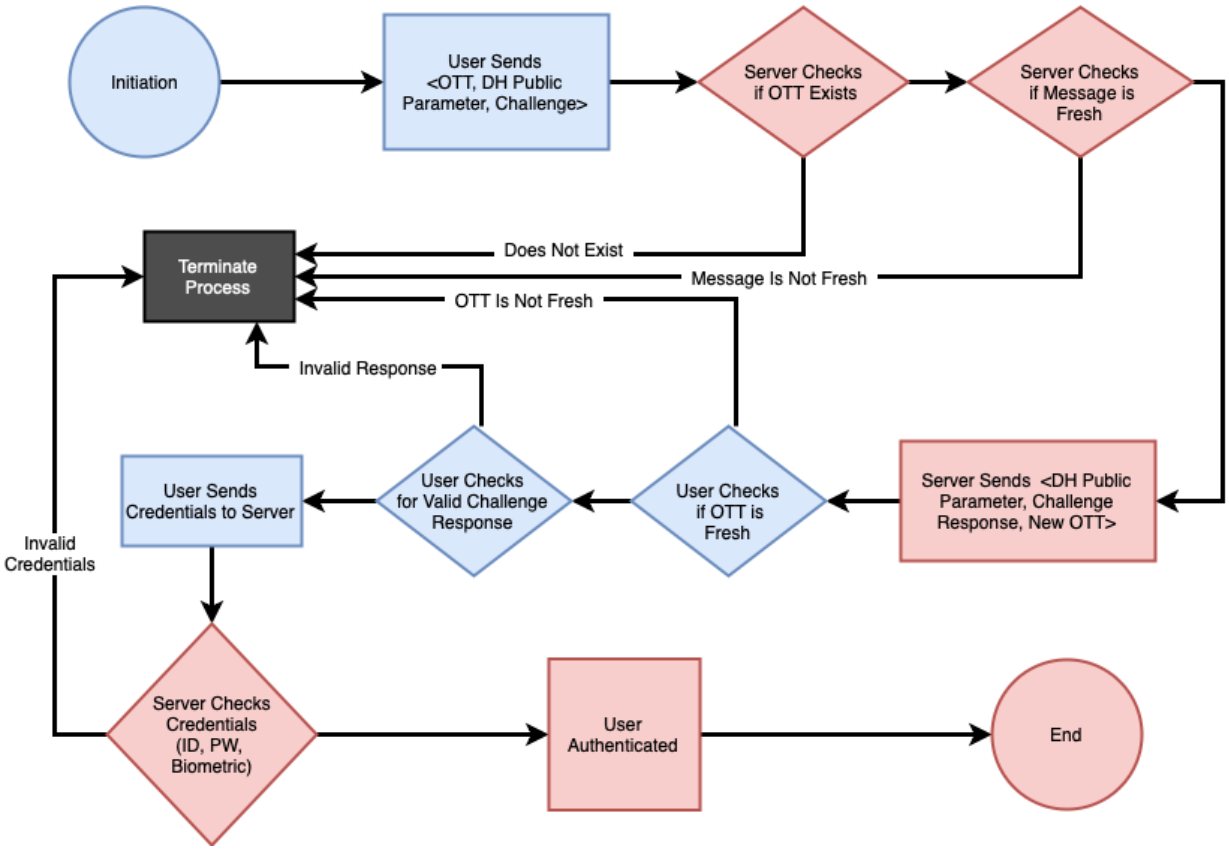

Figure 3.1: High-level workflow of the RSA-B-ASH-S authentication scheme.

The proposed RSA-B-ASH-S scheme is a three-factor remote authentication protocol, the first of which is the password (i.e. what you know factor), the second is smartphone (i.e. what you have factor) and the third is user's biometric impression (i.e. who you are factor). The user is required to register (one-time process) beforehand with the system. Each user U_i needs to choose his/her password and imprints his/her biometric impression at the sensor. The user then interactively submits the Hash of his/her password and the biometric impression to the server. The server then generates a unique ID_i and other information, which are saved in the smartphone, to be used for remote user authentication purpose. The high-level working of the proposed RSA-B-ASH-S

scheme is illustrated in Fig. 3.1. The authentication process in this scheme is initiated by the user, and ended by the server. The user-side operations are highlighted in blue and the server-side operations are highlighted in red. Each conditional operation could terminate the authentication process if given an invalid input, regardless that it is executed in the user or server-side. This process is thoroughly explained Subsection 3.2.

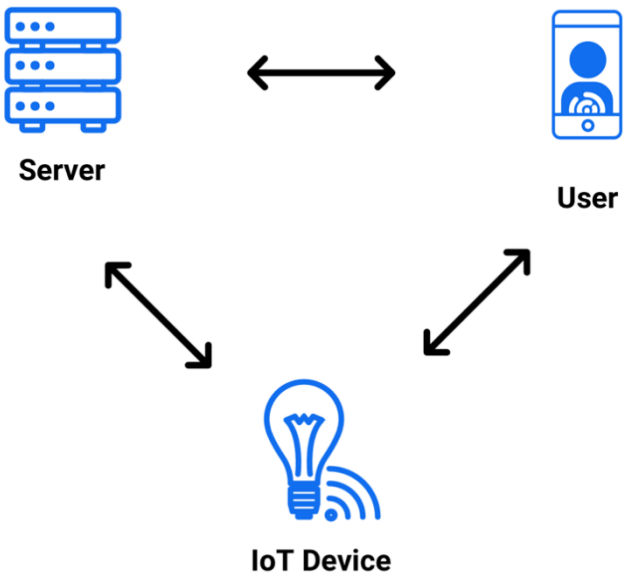

Figure 3.2: Example of authentication dynamic of an IoT structure.

As illustrated in Figure 3.2, based on the nature of the IoT structures [1], mutual authentication is required to be in place for the IoT device, the user and the server. In the system model used in our work, it is assumed that the user is equipped with an IoT device such as a smartphone, thereby the focus is on achieving a mutual authentication between the user and the server.

3.2 Description of the Proposed RSA-B-ASH-S Authentication Scheme

The proposed RSA-B-ASH-S scheme consists of four steps: Initialization phase, Registration phase, Login and Authentication phase.

Initialization phase: The RSA keys are generated as per the method described in RSA-ASH-SC [3]. The key generation algorithm takes two security parameters as inputs: n and k. First, it generates $\frac{n}{k}$ bits long two prime numbers p and q, such that $\gcd\big((p-1),(q-1)\big) = 2$. Then, it calculates $N = p^{(k-1)}.q$. Next, it generates two random numbers $r1$ and $r2$ such that $\gcd\big(r1,(p-1)\big) = 1$, $\gcd\big(r2,(q-1)\big) = 1$ and $r1 = r2\ mod(2)$. Then, it finds the integer d such that $d = r1\ mod(p-1)$ and $d = r2\ mod(q-1)$. Finally, it calculates e such that $e = d^{-1} mod\big(\phi(N)\big)$. Here, the public key is (e, N) and the private key is $(p, q, r1, r2)$, which is kept secret with the server.

The server also generates n_{dh}bits long prime p_{dh} such that $\frac{p_{dh}-1}{2}$ is also a prime. This feature makes p_{dh} a so-called "safe prime". A safe prime is called as such because it does not suffer from some attacks which may make the discrete logarithm easier. Also, it has the advantage of allowing $g = 2$ as the generator, which promotes the computational efficiency. The server additionally creates another prime number g such that g is a primitive root of p_{dh}.

Registration Phase: The user (U_i) submits a request in a secure manner to the server by sharing his/her biometric impression B_i and the hash of his/her password $(h(PW_i), B_i)$ where PW_i is the chosen password. Upon receiving this request, the server creates a random and unique ID_i for U_i. The server also creates a random one-time-token OTT_i to keep for the future authentication requests. Next, the server calculates $CR_i = h(h(PW_i), ID_i)$ then stores this value and OTT_i in the

database along with the B_i and ID_i, which is protected by the server's private key. Next, the server submits the following information $(ID_i, OTT_i, p_{dh}, g, e, N, h(), \Delta T)$ to the smartphone over a secure channel. The stepwise demonstration of the registration phase in the proposed scheme is as follows:

Registration Step 1:

$user$:

$\boldsymbol{U_i \rightarrow server}$: $\langle \boldsymbol{h(PW_i), B_i} \rangle$

Registration Step 2:

$server$:

$ID_i = \ create \rightarrow random\ and\ unique$

$OTT_i = \ create \rightarrow random\ and\ unique$

$CR_i = h(h(PW_i), ID_i)$

$database \rightarrow insert\ \langle OTT_i, ID_i, B_i, CR_i \rangle$

$\boldsymbol{server \rightarrow U_i}$: $\langle \boldsymbol{ID_i, OTT_i, p_{dh}, g, e, N, h, \Delta T} \rangle$

Login and Authentication Phase: First, U_i opens the authenticator software which makes use of the proposed authentication. U_i imprints his/her biometric B_i^* at the sensor, then inputs his/her password. The software performs the following steps. First, it generates a random number a_i and keeps it secure, then computes the Diffie-Hellman public key $PK_{user}^{dh} = g^{a_i} mod\ (p_{dh})$. Next, tsk_i, a session independent temporary key is created randomly by the user application. For the server-side application to be able to encrypt the response of the initial request asymmetrically, the tsk_i variable is needed. It should be pointed out that this value does not function as the session key, but is merely used to prevent the communication of the server's Diffie-Hellman public parameter without encryption. Finally, it generates $P_1 = \ (PK_{user}^{dh}, T, \ tsk_i)$, where T is the current

timestamp. Afterward, the software encrypts P_1 with the server's RSA public key $C_1 = (P_1)^e \ mod(N)$, then sends the following message (OTT_i, C_1) to the server. Upon receipt, the server compares OTT_i against the entries in the database. If there is a match, the server extracts (ID_i, B_i, CR_i) from the database corresponding to the OTT_i, decrypts C_1 and retrieves P_1. To decrypt C_1, it computes $M_1 = C_1^{r1} mod(p)$ and $M_2 = C_1^{r2} mod(q)$. Using CRT, it calculates $P_1 \in Z_N$ such that $P_1 = M_1 mod\ (p)$ and $P_1 = M_2 mod\ (q)$. Then, it checks whether the timestamp is recent or not, i.e. $(T_s - T) < \Delta T$, where T_s is the current timestamp of the server and ΔT is the acceptable difference. Then the server obtains U_i's DH public key $g^a mod\ (p_{dh})$ along with tsk_i. After that, it creates a random number b_i and keeps it secure, then computes the server's DH public key $PK_{server}^{dh} = g^{b_i} mod\ (p_{dh})$ and the session key $sk_i = (PK_{user}^{dh})^{b_i} mod(p_{dh})$. Then, the server creates a new random token OTT_i^{new} but it does not update U_i's token before authentication. Finally, the server computes $P_2 = h(ID_i, sk_i, P_{1,})$ and $C_2 = \{OTT_i^{new}, T, PK_{server}^{dh}, P_2\}_{tsk_i}$ where T is the current timestamp, then sends C_2 back to U_i. Hence, only the server can decrypt C_1, thus P_2 is used as a challenge and the user can authenticate the integrity of the message and its sender. Upon receipt of C_2, the client-side application decrypts it using tsk_i, therefore gains access to OTT_i^{new}, T, P_2 and the server's DH public key PK_{server}^{dh}, and finally obtains the sk_i value by computing $sk_i = (PK_{server}^{dh})^{a_i} mod(p_{dh})$. The client-side application verifies the freshness of the received message by comparing the current and received timestamps. Next, U_i confirms the integrity of the message by calculating $P_2^* = h(ID_i, sk_i, P_1)$ and checking $P_2^* = P_2$. If message was genuine, U_i creates $P_3 = h\big(OTT_i^{new}, OTT_i, h(h(PW_i), ID_i)\big)$, encrypts it along with B_i^* and the current timestamp T, producing $C_3 = \{P_3, B_i^*, T\}_{sk_i}$. Eventually, U_i sends C_3 back to the server as the last step of authentication. The server then decrypts C_3 with sk_i, confirms if the message is fresh, i.e.

$(T_s - T) < \Delta T$ and computes $P_3^* = h\left(OTT_i^{new}, OTT_i, h(h(PW_i), ID_i)\right)$ from the database entries then checks if it is identical to P_3. If this is valid, a biometric verification phase is trigger, which compares the imprinted biometric impression B_i^* with the stored B_i value. If B_i^* is validated $(\sigma(B_i^*, B_i) \leq \Theta)$, then the two values are matched successfully and the authentication phase successfully finishes, otherwise the software generates the decline message and terminates the process. Then the server authenticates the user and updates his/her OTT_i^{new} in the database. Then the user replaces his/her one-time token with OTT_i^{new}. In a nutshell, the steps of the Login and Authentication phase can be summarized as follows:

Login and Authentication Step 1:

$user$:
$B_i^* = read \rightarrow U_i\ imprinted\ biometric$
$PW_i = read \rightarrow U_i\ entered\ password$
$T = read \rightarrow current\ timestamp$
$a_i = create \rightarrow random\ number$
$tsk_i = create \rightarrow random\ number$
$PK_{user}^{dh} = g^{a_i}\ mod(p_{dh})$
$P_1 = (PK_{user}^{dh}, T, tsk_i)$
$C_1 = (P_1)^e\ mod(N)$
$\boldsymbol{U_i \rightarrow server}: \langle \boldsymbol{OTT_i, C_1} \rangle$

Login and Authentication Step 2:

$server$:
$if(database \rightarrow query(OTT_i) == \emptyset)$
    $terminate$
$M_1 = C_1^{r1}\ mod(p)$
$M_2 = C_1^{r2}\ mod(q)$
$CRT \rightarrow P_1 \in Z_n$
$T_s = read \rightarrow current\ timestamp$
$if(T_s - T > \Delta T)$
    $terminate$
$b_i = create \rightarrow random\ number$
$PK_{server}^{dh} = g^{b_i} mod\ (p_{dh})$

$sk_i = (PK_{user}^{dh})^{b_i} mod(p_{dh})$

$OTT_i^{new} = create \rightarrow random\ and\ unique$

$P_2 = h(ID_i, sk_i, P_1)$

$C_2 = \{OTT_i^{new}, T, PK_{server}^{dh}, P_2\}_{tsk_i}$

$\boldsymbol{server \rightarrow U_i: \langle C_2 \rangle}$

Login and Authentication Step 3:

$user$:

$D_2 = \{C_2\}_{tsk_i}^{-1}$

$sk_i = (PK_{server}^{dh})^{a_i} mod(p_{dh})$

$T = read \rightarrow current\ timestamp$

$if(T - T_s > \Delta T)$

$terminate$

$P_2^* = h(ID_i, sk_i, P_1)$

$if(P_2^* != P_2)$

$terminate$

$P_3 = h\left(OTT_i^{new}, OTT_i, h(h(PW_i), ID_i)\right)$

$C_3 = \{P_3, B_i^*, T\}_{sk_i}$

$\boldsymbol{U_i \rightarrow server: \langle C_3 \rangle}$

Login and Authentication Step 4:

$server$:

$D_3 = \{C_3\}_{sk_i}^{-1}$

$T_s = read \rightarrow current\ timestamp$

$if(T_s - T > \Delta T)$

$terminate$

$P_3^* = h\left(OTT_i^{new}, OTT_i, h(h(PW_i), ID_i)\right)$

$if(P_3^* != P_3)$

$exit$

$if(\sigma(B_i^*, B_i) > \Theta)$

$terminate$

$database \rightarrow update(OTT_i^{new})$

$C_4 = \{AUTHENTICATED\}_{sk_i}$

$\boldsymbol{server \rightarrow U_i: \langle C_4 \rangle;\ U_i \rightarrow update(OTT_i^{new})}$

Password/Biometric Change Phase: To update the password or the biometric impression, the user needs to be authenticated in advance. The user enters the new password and imprints

his/her biometric impression at the sensor B_i^{new} and calculates $y = h(PW_i^{new})$ then sends a password/biometric update command to the server as $CMD = (pass_{update}, \{T, y, B_i^{new}\}_{sk_i})$ where $pass_{update}$ is a known command to the server, and T is the current timestamp. After receiving the command, the server decrypts the message using sk_i and validates the timestamp T. If validated, the server computes $CR_i^{new} = h(y, ID_i)$, then updates the database corresponding to the user ID_i. The stepwise demonstration of password/biometric change phase in the proposed scheme is as follows:

Password/Biometric Change Step 1:

$user$:

$B_i^{new} = read \rightarrow U_i\ imprinted\ biometric$

$PW_i^{new} = read \rightarrow U_i\ entered\ password$

$T = read \rightarrow current\ timestamp$

$y = h(PW_i^{new})$

$CMD = (pass_{update}, \{T, y, B_i^{new}\}_{sk_i})$

$\boldsymbol{U_i \rightarrow server: \langle CMD \rangle}$

Password/Biometric Change Step 2:

$server$:

$D = \{CMD\}_{sk_i}^{-1}$

$T_s = read \rightarrow current\ timestamp$

$if\ (T_s - T > \Delta T)$

$\quad terminate$

$CR_i^{new} = h(y, ID_i)$

$database \rightarrow update(CR_i^{new}, B_i^{new})$

$\boldsymbol{server \rightarrow U_i: \langle SUCCESS \rangle}$

Chapter 4

Performance Evaluation of the Proposed RSA-B-ASH-S Scheme

4.1 Proof of Concept of the RSA-B-ASH-S Scheme

To implement the proof of concept of the proposed scheme we have used Apple iPhone 7 Plus smartphone on the client-side with iOS 12.4, running as operating system which has Quad-core (2× Hurricane + 2× Zephyr) CPU with 1.64 GHz frequency. The smartphone is equipped with a front-facing camera which enabled us to implement face recognition as the biometric authentication factor of our proposed scheme. The user-side application has been implemented by Swift 4 [32], a native language for iOS application development. On the server-side we have used a shared host with CloudLinux 6.x [33] as running operating system, which has Dual Intel(R) Xeon(R) CPU E5-2660 v4 with 2.00 GHz frequency. Considering that the server is running over a shared host, all the resources such as CPU and RAM are not completely available. The server-side implementation is over PHP [34] version 7.2, MySQL 10.1.41-MariaDB-cll-lve [35] and for the face recognition part, we have used FaceX API [36].

In both client and server-sides, the RSA encryption has been done using a key length of 2048 bits with the help of OpenSSL Lib v.1.1.1 [37] and to implement the symmetric encryption part, AES-256-GCM [31] is used. For security purposes, each session independently relies on security of AES-256-GCM. It should be emphasized that even with Frontier [38], the most

powerful and fastest supercomputer in the world, which is going to be operational in 2021, it will take millions of years to crack the 256-bit AES encryption. Also, for the Diffie Hellman key exchange algorithm, 2048 bits long prime number p_{dh} has been considered such that $\frac{p_{dh}-1}{2}$ is a prime number, which makes p_{dh} a safe prime. Hence p_{dh} is a safe prime, the generator g is presumed to be 2, which will improve the computational efficiency.

The proof of concept implementation consists of two phases: Registration phase and Login and authentication phase as described in the sequel.

Registration Phase: The proof of concept is implemented on Apple iOS platform. The biometric factor in the implemented proof of concept is face recognition using a two-dimensional picture due to the fact that the current smartphone operating systems do not provide raw access to the device biometric authentication technologies such as fingerprints or depth-powered face capturing. The iOS supported devices have high-resolution front cameras which result in higher face recognition accuracy. The face recognition process is used both in registration and authentication phases. The user captures a picture using the front camera of his/her devices, the client-side application then detects and crops the face in the picture using the iOS Core APIs [32], for efficiency purposes. The user then chooses a secure password as the "what you know" factor. The cropped picture is then encoded into a Base64 format [39], for simplicity, and then sent and saved to the server along with the SHA3-512 [40] hash of the chosen password. Upon receipt, the server generates two 512-bits long unique random values, one of which is ID_i, and the other one is OTT_i. The server then generates the SHA3-512 hash of ID_i combined with the received hashed-value of the user password. The generated value, OTT_i and the Base64 formatted user face image are then stored into the server's database. The server returns the user ID_i, as well as the one-time

token to the client application. All the incoming and outgoing data in the registration process is assumed to be sent over a secure channel. It should be noted that the raw value of the password is never saved in the client nor the server side. The user ID_i, i.e. "what you have" factor is also only stored in the device's operating system secure storage unit. The steps of this phase are shown as below in Figure 4.1.

Figure 4.1: Proof of concept registration process.

Login and Authentication Phase: If the user has registered and the ID_i is stored on the device, the user must be authenticated using his/her biometric impression as well as the selected password. The steps are as illustrated in Fig. 4.2.

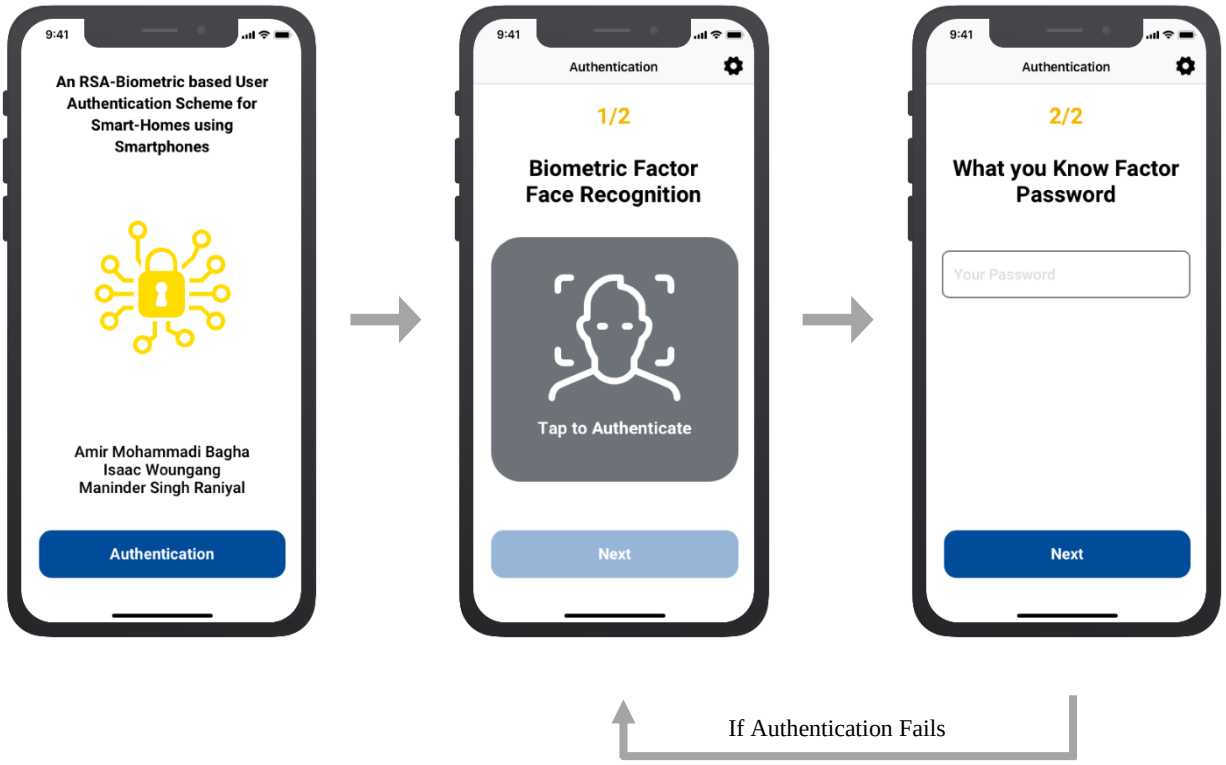

Figure 4.2: Proof of concept login and authentication process.

After the key exchange is completed, the client side and server-side have both agree on the same fresh session key, the authentication can proceed. The user application takes the face image, the inputted password and ID_i to create an encrypted authentication request using the session key and then sends the resulting request to the side. If either the biometric impression or the password does not match with the stored record from the server's database, the server declines the authentication request and the client-side application will transfer the user to the previous screen.

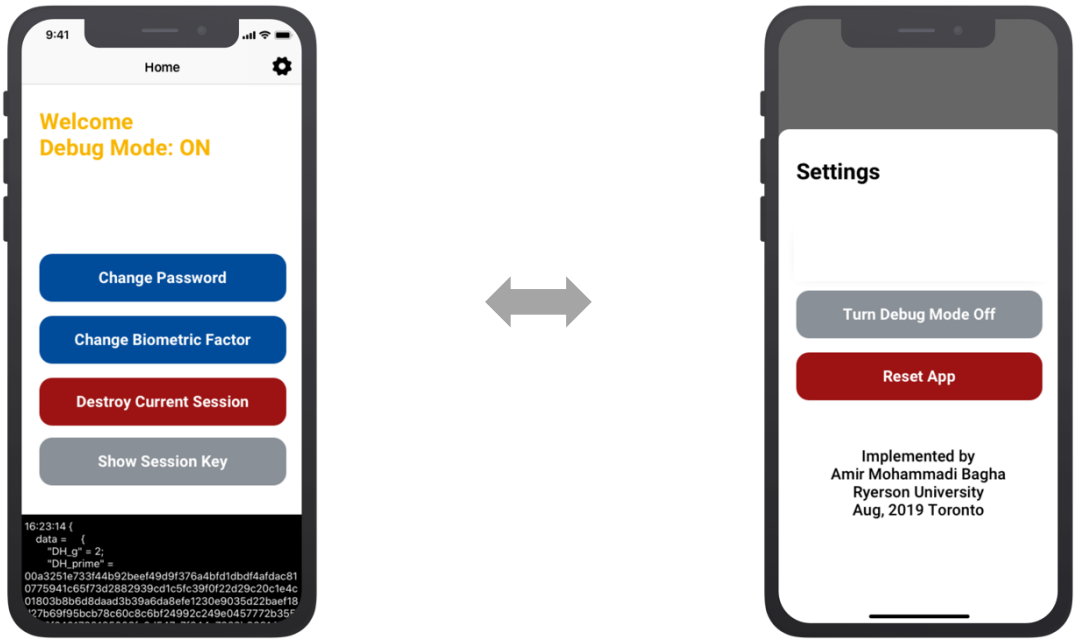

Figure 4.3: Proof of concept home page and settings views.

As shown in Fig. 4.3., once the user authenticates him/her identity using the authentication process, himself/herself can have access to the Home page and the Settings page in which he/she is able to change his/her password and/or biometric impression. The user also able to destroy the current session, turn the debug mode on or off. It should be noted that the debug mode enables the user to view all the outgoing and incoming packages, the agreed session key, and the secret random parameters, for demonstration purpose.

4.2 Informal Security Analysis of the RSA-ASH-SC Scheme

Confidentiality: In the proposed RSA-B-ASH-S scheme, all the messages are encrypted either by RSA or a symmetric key except for OTT_i, which is a one-time token updated upon each authentication completion. This value is only used for initiating the user-server communication so that the server could identify the user. Sending OTT_i without encryption enables the server to distinguish DOS vs. DDOS attacks before decrypting any messages, which make the procedure faster and ensures the availability of the server. Even if the attacker somehow gains access to tsk_i

which is not stored in the database in anyway, it is not possible to extract the session key from $g^{b_i} \bmod (p_{dh})$, since in order to access the session key, either a_i or b_i is needed. Hence p_{dh} is a prime number and g is its generator, and based on the Discrete Logarithm Problem [41], retrieving b_i from $g^{b_i} \bmod (p_{dh})$ is a NP-hard (non-deterministic polynomial-time) problem.

Masquerade attack: In the proposed RSA-B-ASH-S scheme, the user is safe from any attempts of a masquerade attack since there are no parameters sent in a plaintext format other than the OTT_i, which gets regenerated with each session. Even when the attacker gains access to the token, there is no way for he/she to masquerade the user's biometric impression and the password. Hence, in order for the attacker to gain access to the session key with which the user-server communication is encrypted with, he/she requires access to the server's private key as well as one of the session independent random exponents chosen by the server or the user.

Replay attack: In the proposed RSA-B-ASH-S scheme, not only each session has a new key, but also all the messages include a timestamp T which are valid for a short amount of time and are never sent in a plaintext format.

Denial of Service (DoS) & Distributed Denial of Service (DDoS) attacks: Upon receipt of an authentication request, the server only needs to validate OTT_i to distinguish a valid request from an attack. Even if the attacker has access to a valid OTT_i, the server can easily identify the particular user, then set short-term firewall rules to ignore the requests from that user and temporarily ban the user's access.

Perfect forward secrecy: As described earlier, the proposed RSA-B-ASH-S scheme uses a fresh key for each session. (b_i, a_i) which are the secret parameters of the key exchange algorithm, will never be saved, therefore even if the private key of the server is compromised, the session

keys will not be exposed since the attacker cannot obtain a session key unless he/she has access to either b_i or a_i of that session.

Man-in-the-middle (MITM) attack: In the proposed RSA-B-ASH-S scheme, the initial authentication message contains a challenge that would be validated by the user-side application to authenticate the server. This initial message is encrypted with the server's RSA public key, it can only be decrypted by the server. This feature of the scheme makes impossible the access to the challenge unless the attacker has access to the server's RSA private key.

Password guessing attack: In the proposed RSA-B-ASH-S scheme, to perform password guessing attack, the attacker needs to decrypt the last authentication message, which is infeasible considering the encryption of the message. Even if the attacker gets access to the private key as well as the database and successfully decrypts it, because of the fact that $h(h(PW_i), ID_i)$ was saved in the database, he/she cannot get access to the $h(PW_i)$ in a reasonable amount of time. Indeed, using ID_i as a Salt in password hashing enables us to prevent the rainbow attack. Even if the attacker in some way gets access to the password, he/she requires the user's biometric impression and his/her smartphone to get authenticated from the server.

Device loss attack: In case of user's device loss, the attacker must break through the smartphone's operating system to get the user's ID_i but by knowing only ID_i, he/she is not able to get authenticated, because the authentication procedure requires the user's password along with his/her biometric impression , which are never saved on the device.

4.3. Comparison of Authentication Schemes based on Security Attacks and Security Metrics

The proposed RSA-B-ASH-S scheme is compared against selected RSA-based variants in terms of security attacks (as per Table 4.1).

Table 4.1: Comparison of selected RSA variants in terms of security attacks and metrics.

	Yang et al. [14]	Fan et al. [16]	Yang et al. [42]	Om et al. [10]	Om et al. [19]	Shen et al. [18]	Liu et al. [17]	Chien et al. [43]	Raniyal et al. [3]	Proposed RSA-based scheme
Confidentiality	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Availability	✓	✓						✓	✓	✓
Integrity						✓		✓	✓	✓
Mutual authentication						✓	✓	✓	✓	✓
MITM				✓	✓	✓	✓		✓	✓
Device loss attack	✓	✓	✓			✓	✓	✓	✓	✓
Password-guessing attack	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Replay attack	✓	✓	✓	✓	✓	✓	✓		✓	✓
Forgery or Impersonation			✓	✓	✓		✓		✓	✓
DoS attack	✓	✓						✓	✓	✓
Forward Secrecy	✓	✓				✓	✓			✓

4.4. Comparison of Computational Performance of Authentication Schemes

The proposed RSA-B-ASH-S scheme takes advantage of an RSA-based and Diffie Hellman protocols. Considering that the computational performance and the speed of this scheme strongly rely on the proposed algorithm as well as the client-side device and the server computational power, the total authentication time is subject to the mentioned variables and scenarios. Our proposed scheme is an improvement over the RSA-ASH-SC scheme [3], which in terms of convergence speed, has considered the Rebalanced-Multi-Power RSA scheme [10] as underlying RSA algorithm, using a key length of 2048 bits.

4.4.1 Computational Performance

In terms of computational performance, we have considered the notations given in Table 4.2 sorted by complexity from high to low.

Table 4.2: Notations used for the computational time of operations.

Notation	Description
T_{exp}	The time taken by modular exponent operation
T_d	The time taken by the modular decryption exponent *(d)* operation
T_e	The time taken by the modular encryption exponent *(e)* operation
T_s	The time taken to encrypt/decrypt using the symmetric key
T_h	The time taken by hash function operation
T_{bio}	The time taken by the biometric comparison $\sigma()$ function
T_{mul}	The time taken by the modular multiplication operation
T_{xor}	The time taken by the XOR operation

In the authentication phase, the steps 1 and 3 are completed over the client-side and steps 2 and 4 are handled over to the server-side. The time taken by each step, based on the proposed scheme is listed in Table 4.3 as shown below.

Table 4.3: Total computation time needed for each step

Step	**Server-side**	**Client-side**	**Total**
1	-	$T_{exp} + T_e$	$T_{exp} + T_e$
2	$2T_{exp} + T_d + T_h + T_s$	-	$2T_{exp} + T_d + T_h + T_s$
3	-	$T_{exp} + 4T_h + 2T_s$	$T_{exp} + 4T_h + 2T_s$
4	$T_{bio} + 3T_h + 2T_s$	-	$T_{bio} + 3T_h + T_s$
Total	$2T_{exp} + T_d$ $+T_{bio} + 4T_h + 3T_s$	$2T_{exp} + 4T_h + 2T_s + T_e$	$4T_{exp} + T_d + 8T_h$ $+T_e + T_{bio} + 5T_s$

Based on the above Table 4.3, the proposed scheme is compared against the selected RSA variants in terms of computation performance. The results given in Table 4.4

Table 4.4: Comparison of selected RSA variants in terms of computation performance.

	Login Phase	**Authentication Phase**	**Total**
Yang et al. [14]	$2T_{exp} + 3T_{mul}$ $+T_h$	$T_e + T_{exp} +$ $T_{mul} + T_h$	$4T_{mul} + 3T_{exp}$ $+T_e + 2T_h$
Fan et al. [16]	$2T_{exp} + 3T_{mul}$ $+T_h$	$T_e + T_{exp} +$ $T_{mul} + T_h$	$4T_{mul} + 3T_{exp}$ $+T_e + 2T_h$

Yang et al. [42]	$2T_{exp} + 3T_{mul}$	$T_e + 2T_{exp} + T_{mul}$	$4T_{mul} + 4T_{exp} + T_e$
Om et al. [10]	$T_e + T_{exp} + T_h + T_{xor}$	T_d	$T_{xor} + T_{exp} + T_d + T_h + T_e$
Om et al. [19]	$T_e + T_{exp} + 2T_h + T_{xor}$	$T_d + T_{exp} + T_h + T_{xor}$	$2T_{xor} + 2T_{exp} + T_d + 3T_h + T_e$
Shen et al. [18]	$T_e + 2T_{exp} + 3T_{mul} + 2T_h$	$T_d + 2T_h + T_{xor}$	$T_{xor} + 2T_{exp} + T_d + 4T_h + T_e + 3T_{mul}$
Liu et al. [17]	$T_d + T_e + T_{exp} + T_{mul} + 2T_h + 2T_{xor}$	$T_e + 2T_h + T_{xor} + 3T_{mul} + 2T_{exp}$	$3T_{xor} + 3T_{exp} + T_d + 4T_h + 2T_e + 4T_{mul}$
Chien et al. [43]	$2T_h + 2T_{xor}$	$3T_h + 3T_{xor}$	$5T_{xor} + 5T_h$
Raniyal et al. [3]	$2T_e + T_s + 6T_h + 2T_{xor}$	$T_d + T_e + 2T_h + 2T_{xor} + T_s$	$4T_{xor} + T_d + 8T_h + 3T_e + 2T_s$
Proposed RSA-B-ASH-S Scheme	$2T_{exp} + 4T_h + 2T_s + T_e$	$2T_{exp} + T_d + T_{bio} + 4T_h + 3T_s$	$4T_{exp} + T_d + 8T_h + T_e + T_{bio} + 5T_s$

Table 4.5: Notations used to represent each operation.

Notation	**Description**
EXP	The total number of modular exponent operations
D	The total number of the modular decryption exponent *(d)* operations
E	The total number of the modular encryption exponent *(e)* operations
S	The total number of encrypt/decrypt operations using the symmetric key
H	The total number of hash function operations
BIO	The total number of the biometric comparison $\sigma()$ functions
MUL	The total number of the modular multiplication operations
XOR	The total number of the XOR operations

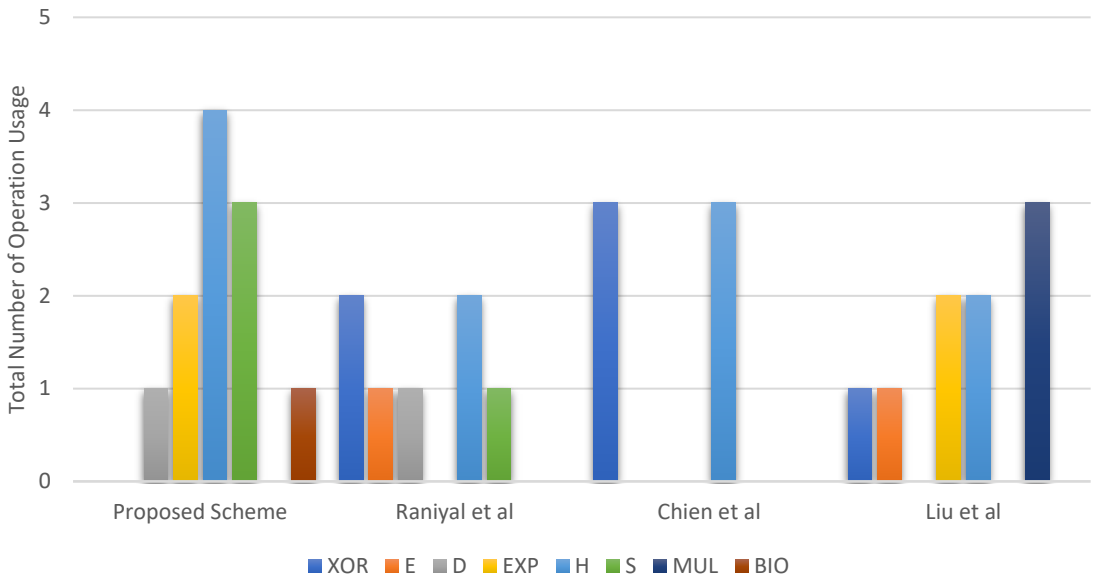

Figure 4.4: Comparison of the performance of the authentication phase in terms of computational time.

As shown in Fig. 4.5 and Table 4.3 and Table 4.4, the overall computational time of the authentication phase in the proposed scheme is increased compared to the previous schemes. This increment in time is due to the fact that our proposed scheme is satisfies the Perfect Forward Secrecy, which uses the Diffie–Hellman key exchange as underlying protocol, resulting in additional steps required to complete the authentication. The scheme also takes advantage of a biometric factor in the authentication process, which results in a 3-factor authentication method, unlike other studied schemes.

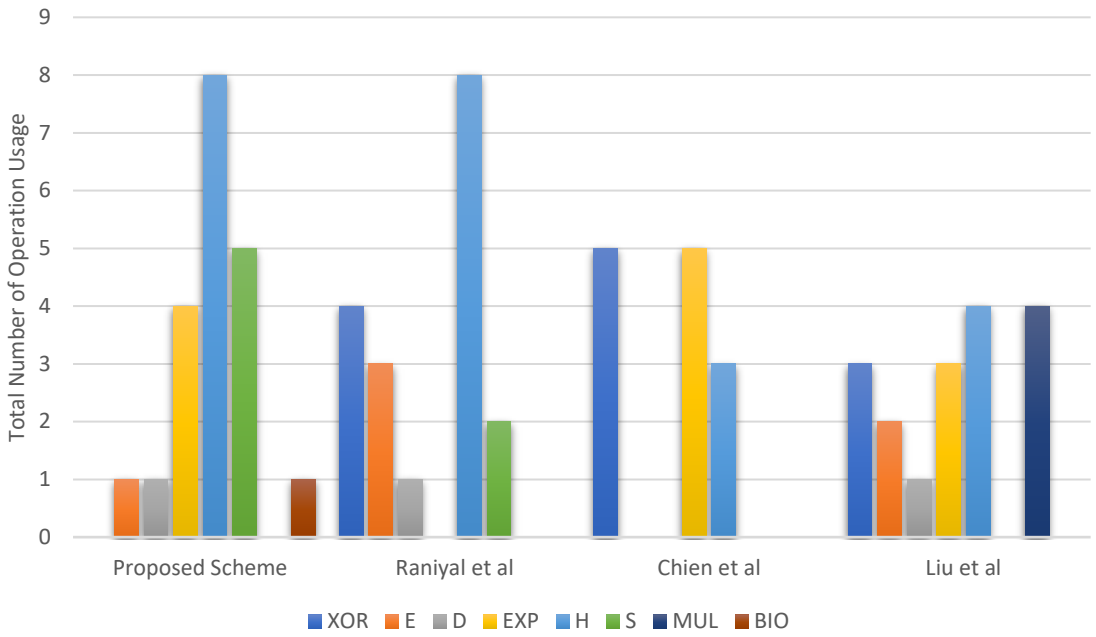

Figure 4.5: Comparison of total computational performance.

As demonstrated in Fig. 4.5, the total computational time of the proposed scheme is incremented compared to to that of all other studied schemes. To prevent Rainbow attacks, our proposed scheme uses nested hash functions to protect the data in case of any database exposure and makes the extraction of the raw data very difficult. It should be noted that the total computational time of the proposed scheme is high because use of the DH key exchange algorithm.

After performing numerous simulations of the proposed scheme over both client and server-sides, in terms of computational performance, we have captured the results in Table 4.6.

Table 4.6: Result of Simulations of the Proposed Scheme over Client-side and Server-side.

	Data length (bits)	Client-side	Server-side
T_d	2048	-	8 milliseconds
T_e	2048	0.6 milliseconds	-
T_{exp}	512	8 milliseconds	5 milliseconds
T_s	512	4 milliseconds	10 microseconds
T_h	512	0.4 milliseconds	0.05 milliseconds
T_{xor}	-	-	-
T_{mul}	-	-	-

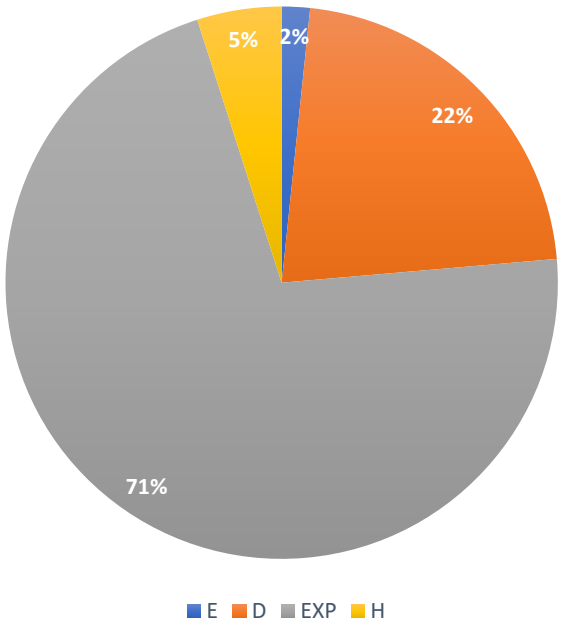

Figure 4.6: The share of each operation in percentage.

As shown in Fig. 4.6, the majority of the computational time of the login and authentication processes combined is dedicated to the modular exponential operation based on the results given in Table 4.6. The RSA decryption function has been used only once in the proposed scheme, but as demonstrated above, it takes up 22% of the whole computation in terms of time.

4.4.2 Storage Requirements

In our proposed scheme, each user node is required to store $(ID_i, OTT_i, p_{dh}, g, e, N, h, \Delta T)$. We used SHA3-512 as example of hash function, and the output of SHA3-512 is 512 bits. By applying these settings, we obtain $|ID_i|$ = $|OTT_i|$ 512 bits, while $|p_{dh}|$ was assumed to be 2048 bits for security concerns. g value is 2 so we can use short integer variable (8 bits) for storing it. ΔT can be saved into 32 bits of integer value. On the other hand, we need to store $(p, q, r1, r2)$ as server RSA private key along with (ID_i, OTT_i, B_i) for each user. The size of B_i is completely dependent on the type of biometric factor impression which in our implementation was assumed a compressed image of user's face for face recognition. Assuming our biometric factor is an image of user's face, B_i can be stored in $|B_i|$ bits. Hence, the total storage required by each user node U_i is $(3112 + |e| + |N|)$ bits, and the total storage required by the server is $\big(n \times (1024 + |B_i|)\big) + |(p, q, r1, r2)|$, where n is the number of registered users. The storage requirements are summarized in Table 4.7.

Table 4.7: Storage requirements of the user device and server-side in the proposed scheme.

	Data length (bits)
User device	$(3112 + \|e\| + \|N\|)$
Server	$\big(n \times (1024 + \|B_i\|)\big) + \|(p, q, r1, r2)\|$

4.4.3 Communication Overloads

In the authentication transmissions, the user sends $U_i \rightarrow server$: $\langle OTT_i, C_1 \rangle$ in the first step where OTT_i is 512 bits and C_1 includes the Diffie-Hellman public parameter of the user, the timestamp T and tsk_i, so C_1 is 2048 + 32 + 256 bits length. The user also sends $U_i \rightarrow server$: $\langle C_3 \rangle$ in the third step where C_3 includes $\{P_3, B_i^*, T\}$. P_3 is the output of the hash function (SHA3-512) so it is 512 bits long and B_i^* is the biometric impression of the attempting user. On the other hand, the server needs to send $server \rightarrow U_i$: $\langle C_2 \rangle$ where C_2 is the symmetrically encrypted message containing the 512-bit new one-time token, the 32-bit timestamp, the 2048-bit Diffie-Hellman public parameter of the server, and P_2 which is output of the hash function (SHA3-512) which is 512 bits long. The overall server to user communication overload is (2048 + 512 + 512 + 32). The breakdown of the communication overloads is summarized in Table 4.8.

Table 4.8: Communication overloads between user and server applications.

	Data length (bits)
User to server	3392 + $\lvert B_i \rvert$
Server to user	3104

4.5 Formal Analysis of the Proposed RSA-B-ASH-S Scheme Based on BAN Logic

4.5.1 BAN Logic Overview

The Burrows–Abadi–Needham logic, known as BAN Logic [5], was introduced in 1989 as a model to define authentication protocols and evaluate their validity. Since then, this model has been adopted and accepted by many researchers to prove that an authentication scheme can be between a user and server, in this case, user smartphone U and the server S.

The Following is a list of BAN logic notions used in this thesis to validate the Proposed RSA-Based Scheme:

$U \mid \equiv X$: U believes the statement X.

$\#(X)$: X is fresh.

$U \Rightarrow X$: U has jurisdiction over the statement X.

$U \lhd X$: U sees the statement X.

$U \mid\sim X$: U once said the statement X.

(X, Y): X or Y is one part of the expression (X, Y).

$\{X\}_Y$: X encrypted with Y.

$U \overset{sk}{\leftrightarrow} S$: sk is a secret parameter shared (or to be shared) between U and S.

$\overset{X}{\rightarrow} S$: X is public key of S. The private key associated with X is denoted with X^{-1}.

The following BAN logic rules are used to prove that the proposed authentication scheme key agreement is fulfilled successfully.

R1: Random Number Freshness

When an entity creates a random value, it believes the value is fresh.

$$\frac{U\ creates\ random\ X}{U \mid \equiv \#(X)}$$

R2: The Rule For $\overset{k}{\leftrightarrow}$-Introduction

With X indicating the essential elements for a key. Formally, it is required that U believes that S also participates in the protocol. Informally, the rule means that to believe a new session key, U must believe that the key is fresh, and U must also believe that S believes in X, so S can generate the key as well.

$$\frac{U \mid \equiv \#(K), U \mid \equiv S \mid \equiv X}{U \mid \equiv U \overset{K}{\leftrightarrow} S}$$

R3: Message Meaning

I. If U perceives X as an encrypted value with K, and believes K is a shared secret key with S, then U believes S once said X:

$$\frac{U \mid \equiv U \overset{K}{\leftrightarrow} S,\ U \lhd \{X\}_K}{U \mid \equiv S \mid\sim X}$$

II. If U perceives X as an encrypted value with K^{-1} and believes K is S public key, then U believes S once said X:

$$\frac{U \mid \equiv \overset{K}{\rightarrow} S,\ U \lhd \{X\}_{K^{-1}}}{U \mid \equiv S \mid\sim X}$$

R4: Message Freshness

If U believes X is fresh and U believes S once said X, then U believes S believes X:

$$\frac{U \mid \equiv \#(X),\ U \mid \equiv S \mid\sim X}{U \mid \equiv S \mid \equiv X}$$

R5: Hash Function

If U believes S once said $H(X)$ and U sees X, then U believes S once said X:

$$\frac{U \mid\equiv S \mid\sim H(X),\ U \lhd X}{U \mid \equiv S \mid\sim X}$$

R6: Jurisdiction

If U believes S has full control over X and U believes S believes X, then U believes X:

$$\frac{U \mid \equiv S \Longrightarrow X,\ U \mid \equiv S \mid \equiv X}{U \mid \equiv X}$$

R7: Freshness Propagation

If one parameter of an expression is fresh, then the entire expression is fresh:

$$\frac{U \mid \equiv \#(X)}{U \mid \equiv \#(X,Y)}$$

R8: Belief

If U believes X and Y, then U believes X:

$$\frac{U \mid \equiv (X,Y)}{U \mid \equiv X}$$

R9: Observation

If U perceives X and Y, then U perceives X:

$$\frac{U \lhd (X,Y)}{U \lhd X}$$

4.5.2 Goals of the Analysis of Key Exchange Part of the Proposed RSA-B-ASH-S Scheme

The core objectives of our authentication scheme analysis are listed as follows:

• G1: U believes S believes sk is a secure shared parameter between U and S:

$$U \mid \equiv S \mid \equiv U \overset{sk}{\leftrightarrow} S$$

• G2: U believes sk is a secure shared parameter between U and S.

$$U \mid \equiv U \overset{sk}{\leftrightarrow} S$$

• G3: S believes U believes sk is a secure shared parameter between U and S.

$$S \mid \equiv U \mid \equiv U \overset{sk}{\leftrightarrow} S$$

• G4: S believes sk is a secure shared parameter between U and S.

$$S \mid \equiv U \overset{sk}{\leftrightarrow} S$$

The following are the assumptions made about the initial state of the proposed scheme to analyze:

• A1: $S \mid \equiv \overset{K_S}{\rightarrow}$ S: The server believes K_S as its public key.

• A2: $U \mid \equiv \overset{K_S}{\rightarrow}$ S: The user believes K_S as the server's public key.

• A3: $S \mid \equiv U \overset{ID}{\leftrightarrow} S$: The server believes ID is a secret parameter between the server and user.

• A4: $U \mid \equiv U \overset{ID}{\leftrightarrow} S$: The user believes ID is a secret parameter between the server and user.

• A5: $U \mid \equiv U \overset{h(PW)}{\longleftrightarrow} S$: The user believes $h(PW)$ is a secret parameter between the server and user.

• A6: $S \mid \equiv U \overset{h(PW)}{\longleftrightarrow} S$: The server believes $h(PW)$ is a secret parameter between the server and user.

• A7: $S \mid \equiv U \implies B$: The server believes that only user has jurisdiction over his/her biometric.

• A8: $S \mid \equiv U \implies PW$: The server believes that only user has jurisdiction over his/her password.

4.5.3 Formal Analysis of the Proposed RSA-B-ASH-S Scheme

The analysis of our authentication scheme is shown below. D_i represents the i^{th} deduction, and M_i represents the i^{th} message in the following step-wise analysis of the proposed scheme.

D1: $\frac{U\ creates\ random\ a}{U \mid \equiv \#(a)}$: Based on R1

D2: $\frac{U\ read\ current\ timestamp\ T_1}{U \mid \equiv \#(T_1)}$

D3: $\frac{U\ creates\ random\ tsk}{U \mid \equiv \#(tsk)}$: Based on R1

M1: $U \longrightarrow S$: $\langle OTT,\ \{g^a\ mod(p_{dh}), T_1, tsk\}_{K_s}\rangle$: The user initiates the authentication procedure by sending this message to the server.

D4: $U \mid \equiv\ U \overset{tsk}{\leftrightarrow} S$: Based on M1 is encrypted with public key of the server, and based on A1 and A2, the user believes *tsk* is a secret parameter between *U* and *S*.

D5: $S \lhd (g^a\ mod(p_{dh}), T_1, tsk)$: Based on M1 is encrypted with the public key of the server, and based on R3.II, A1 and A2 the server decrypts and sees $(g^a\ mod(p_{dh}), T_1, tsk)$.

D6: $S \mid \equiv U \mid\sim (g^a\ mod(p_{dh}), T_1, tsk)$: Based on D4, D5, A1 and A2.

D7: $\frac{S \mid \equiv \#(T_1)}{S \mid \equiv \#(g^a\ mod(p_{dh}), T_1, tsk)}$: Based on R7.

D8: $\frac{S \mid \equiv \#(g^a\ mod(p_{dh}), T_1, tsk), S \mid \equiv U \mid\sim (g^a\ mod(p_{dh}), T_1, tsk)}{S \mid \equiv U \mid \equiv (g^a\ mod(p_{dh}), T_1, tsk)}$: Based on R4, D6 and D7.

D9: $\frac{S \mid \equiv U \mid \equiv (g^a\ mod(p_{dh}), T_1, tsk)}{S \mid \equiv U \mid \equiv (g^a\ mod(p_{dh}))}$: Based on R8 and D8.

D10: $\frac{S\ creates\ random\ b}{S\ |\equiv \#(b)}$: Based on R1.

D11: $\frac{S\ |\equiv \#(b)}{S\ |\equiv \#\left(g^b\ mod(p_{dh})\right)}$: Based on R7.

D12: $sk := \left(g^a\ mod(p_{dh})\right)^b mod(p_{dh})$: The server generates the session key.

D13: $\frac{S\ |\equiv \#(b)}{S\ |\equiv \#\left(\left(g^a\ mod(p_{dh})\right)^b mod(p_{dh})\right)}$: Based on R7 and D10.

D14: $\frac{S\ read\ current\ timestamp\ T_2}{S\ |\equiv \#(T_2)}$

D15: $\frac{S|\equiv \#(sk),\ \ S\ |\equiv U\ |\equiv \left(g^a\ mod(p_{dh})\right)}{S\ |\equiv U \overset{sk}{\leftrightarrow} S}$: Based on D9, R2 and D13, (G4) is met here.

D16: $\frac{S\ creates\ random\ NOTT}{S\ |\equiv \#(NOTT)}$: Based on R1.

M2: $S \longrightarrow U$: $\langle \left\{g^b\ mod(p_{dh}), h\left(ID, sk, (g^a\ mod(p_{dh}), T_1, tsk\ )\right), NOTT, T_2\right\}_{tsk}\rangle$: Server sends the respons of the initial request encrypted with tsk to the server.

D17: $U \lhd \left(g^b\ mod(p_{dh}), h\left(ID, sk, (g^a\ mod(p_{dh}), T_1, tsk\ )\right), NOTT, T_2\right)$: Based on M2 is encrypted with tsk, and based on R3.II, A1 and A2 the user decrypts and sees $\left(g^b\ mod(p_{dh}), h\left(ID, sk, (g^a\ mod(p_{dh}), T_1, tsk\ )\right), NOTT, T_2\right)$.

D18: $\frac{U\ |\equiv U \overset{tsk}{\leftrightarrow} S, U \lhd \left\{g^b\ mod(p_{dh}), h\left(ID, sk, (g^a\ mod(p_{dh}), T_1, tsk\ )\right), NOTT, T_2\right\}_{tsk}}{U\ |\equiv S\ |\sim \left(g^b\ mod(p_{dh}), h\left(ID, sk, (g^a\ mod(p_{dh}), T_1, tsk\ )\right), NOTT, T_2\right)}$: Based on R3.I.

D19: $\frac{U\ |\equiv S\ |\sim \left(g^b\ mod(p_{dh}), h\left(ID, sk, (g^a\ mod(p_{dh}), T_1, tsk\ )\right), NOTT, T_2\right)}{U\ |\equiv S\ |\sim \left(g^b\ mod(p_{dh})\right)}$: Based on R8.

D20: $U \lhd g^b\ mod(p_{dh})$: Based on D17 and R8.

D21: $sk := \left(g^b\ mod(p_{dh})\right)^a mod(p_{dh})$: The user generates the session key.

D22: $\frac{U\ |\equiv \#(T_2)}{U\ |\equiv \#\left(g^b\ mod(p_{dh}), h\left(ID, sk, (g^a\ mod(p_{dh}), T_1, tsk\ )\right), NOTT, T_2\right)}$: Based on R7.

D23:

$$\frac{U|\equiv\#\left(g^b\ mod(p_{dh}),h\left(ID,sk,\left(g^a\ mod(p_{dh}),T_1,tsk\right)\right),NOTT,T_2\right),U\ |\equiv S\ |\sim\ \left(g^b\ mod(p_{dh}),h\left(ID,sk,\left(g^a\ mod(p_{dh}),T_1,tsk\right)\right),NOTT,T_2\right)}{U\ |\equiv S|\equiv\left(g^b\ mod(p_{dh}),h\left(ID,sk,\left(g^a\ mod(p_{dh}),T_1,tsk\right)\right),NOTT,T_2\right)}$$

: Based on R4, D22 and D18.

D24: $\frac{U\ |\equiv\ S\ |\equiv\left(g^b\ mod(p_{dh}),h\left(ID,sk,\left(g^a\ mod(p_{dh}),T_1,tsk\right)\right),NOTT,T_2\right)}{U\ |\equiv S|\equiv h\left(ID,sk,\left(g^a\ mod(p_{dh}),T_1,tsk\right)\right)}$: Based on R8 and D23.

D25: $\frac{U\ |\equiv S|\equiv h\left(ID,sk,\left(g^a\ mod(p_{dh}),T_1,tsk\right)\right)}{U\ |\equiv S|\equiv U\overset{sk}{\leftrightarrow}S}$: Based on D24 and R8, (G1) is met here.

D26: $\frac{S\ |\equiv\#(a)}{S\ |\equiv\#\left(\left(g^b\ mod(p_{dh})\right)^a\ mod(p_{dh})\right)}$: Based on R7 and D1.

D27: $\frac{U\ |\equiv\ S\ |\equiv\left(g^b\ mod(p_{dh}),h\left(ID,sk,\left(g^a\ mod(p_{dh}),T_1,tsk\right)\right),NOTT,T_2\right)}{U\ |\equiv S|\equiv\ g^b\ mod(p_{dh})}$: Based on R8 and D23.

D28: $\frac{U\ |\equiv\#(sk),\ \ U|\equiv S\ |\equiv g^b\ mod(p_{dh})}{U\ |\equiv U\overset{sk}{\leftrightarrow}S}$: Based on R2, D27 and D26, (G2) is met here.

D29: $\frac{S\ read\ current\ timestamp\ T_3}{S\ |\equiv\#(T_3)}$

M3: $U\longrightarrow S$: $\left\{B,T_3,h\left(NOTT,OTT,h\left(ID,h(PW)\right)\right)\right\}_{sk}$

D30: $\frac{S\ |\equiv U\overset{sk}{\leftrightarrow}S,S\lhd\{M3\}_{sk}}{S\ |\equiv U\ |\sim\ M3}$: Based on R3.I and D15.

D31: $\frac{S\ |\equiv\#(T_3)}{S\ |\equiv\#\left(B,T_3,h\left(NOTT,OTT,h\left(ID,h(PW)\right)\right)\right)}$: Based on R7.

D32: $\frac{S\ |\equiv\#(M_3),S\ |\equiv U\ |\sim\ M3}{S\ |\equiv U\ |\equiv\ U\overset{sk}{\leftrightarrow}S}$: Based on R4, D31 and D30 (G3) is met here.

Chapter 5

Conclusion

In this thesis, after a brief overview of the related works in the area of user authentication schemes for IoT, assessing their advantages and drawbacks, we have proposed a new RSA-biometric based authentication protocol for smart-homes using smartphone (called RSA-B-ASH-S scheme). This scheme, in addition to satisfying all the security features of the authentication scheme introduced in [3], is also shown to satisfy the perfect forward secrecy, considered as a great enhancement to the RSA-ASH-SC scheme [3] when considering the increased significance of privacy and security in IoT [1].

An informal security analysis of the proposed RSA-B-ASH-S scheme is provided, along with its performance evaluation in terms of computational time, storage requirements and communication overload. Furthermore, a formal analysis of the proposed RSA-B-ASH-S scheme using the Burrows-Abadi-Needham (BAN) Logic is provided, showing that the proposed scheme achieves the forward secrecy property by utilizing a fresh encryption key for each session and it also satisfies the anonymity of the user by using a one-time token. A proof of concept of the proposed RSA-B-ASH-S scheme is also provided.

As future work, one can redesign the authentication step of the proposed RSA-B-ASH-S in such a way as to strengthen the computational efficiency. Due to the nature of IoT devices which are incorporated into the users' everyday lives, another interesting opportunity ahead for this

scheme is to take advantage of the behavioural patterns of the users as a new authentication layer meant to detect anomalies. Considering the direction of the technologies in this sector and the progress in machine learning areas, the future of behavioural pattern recognition is promising, especially in the IoT world.

Appendix

Code for the Proposed RSA-B-ASH-S Implementation

Backend (PHP files)

auth.php

```php
<?php
   require_once "connection.php";
   $step = intval($_GET['step']);
   session_start();
   if ($step<1 || $step>2)
      $step = 1;
   switch ($step) {
      case 2:
         $params  = getParameters(RequestTypes::POST, ["m"]);
         $params["m"] = base64_decode($params["m"]);
         $aes     = new AES($_SESSION['key']);
         $message = json_decode(base64_decode($aes->decrypt($params["m"])));
         if(!checkTimestamp($message->t))
            throw new Exception("Message is expired, time: " . time() . " -- your
time: " . $message->t, ErrorCode::EXPIRED);
         $user    = User::fromDB($_SESSION['token']);
         $biometricChecker = new Biometric();
         if(!$biometricChecker->compare($user->biometric, $message->b))
            throw new Exception("Biometric impression does not match",
ErrorCode::ACCESS_DENIED);
         $hashOfData = hash(hashMethod,$user->password.$user-
>token.$_SESSION['next_token'],false);
         if($hashOfData != $message->hash)
            throw new Exception("Data does not match", ErrorCode::ACCESS_DENIED);
         $user->token = $_SESSION['next_token'];
         $user->update();
         unset($_SESSION['next_token']);
         $_SESSION['token'] = $user->token;
         $_SESSION['authenticated'] = true;
         response(true,"Successfully authenticated");
         break;
      default:
      case 1:
         $params = getParameters(RequestTypes::POST, ["c1","token"]);
         $user = User::fromDB($params["token"]);
         $rsa = new RSA();
         $decryptedMessage = json_decode($rsa->decrypt($params["c1"]));
         if(!checkTimestamp($decryptedMessage->t))
            throw new Exception("Message is expired, time: " . time(),
ErrorCode::EXPIRED);
```

```
        $userDHPublicKey    = $user->getDHPublicKey(base64_decode($decryptedMessage-
>y));
        $dh                 = new DH();
        $_SESSION["server_public_dh"] = $dh->generatePublicKey();
        $_SESSION['key']        = substr($dh->getSharedKey($userDHPublicKey), 0,
64);
        $_SESSION['token']      = $user->token;
        $_SESSION['next_token'] = $user->generateToken();
        $response["c2"] = base64_encode(stringXOR($_SESSION['server_public_dh'],
$user->id));
        $response["c3"] = hash("sha3-512",$user-
>id.$_SESSION['key'].$params['c1'],false);
        $plain["token"] = $_SESSION["next_token"];
        $plain["time"]  = time();
        $aes         = new AES($_SESSION['key']);
        $response["cipher"] = base64_encode($aes-
>encrypt(base64_encode(json_encode($plain))));
        response(true,null, $response);
        break;
    }
    Database::close();
```

change.php

```
<?php
    require_once "connection.php";
    session_start();
    if (!isset($_SESSION["authenticated"]) || !$_SESSION['authenticated'])
        throw new Exception("You need to be authenticated in advance",
ErrorCode::ACCESS_DENIED);
    $params = getParameters(RequestTypes::POST, ["m"]);
    $params["m"] = base64_decode($params["m"]);
    $aes     = new AES($_SESSION['key']);
    $message = json_decode(base64_decode($aes->decrypt($params["m"])));
    if(!checkTimestamp($message->t))
        throw new Exception("Message is expired", ErrorCode::EXPIRED);
    $user = User::fromDB($_SESSION['token']);
    $user->password  = hash(hashMethod, $message->y . $user->id, false);
    $user->biometric = $message->b;
    $user->update();
    response(true,"Successfully updated");
    Database::close();
```

register.php

```
<?php
    require_once "connection.php";
    $params = getParameters(RequestTypes::POST,["password","biometric"]);
    $user   = User::create($params["password"],$params["biometric"]);
    $user->insert();
    $data["user"] = $user;
    $data["key"]  = (new RSA())->publicKey;
    $data["DH_prime"] = DH::PRIME;
    $data["DH_g"] = DH::GENERATOR;
```

```
    $data["hash_function"] = hashMethod;
    $data["symmetric_encryption"] = AES::METHOD;
    response(true,null, $data);
    Database::close();
```

user.php

```
<?php
class User implements JsonSerializable {
    public $id, $biometric, $password, $token;
    function __construct($json) {
        $jsonOBJ = json_decode($json, false);
        $this->id = $jsonOBJ->id;
        $this->biometric = $jsonOBJ->biometric;
        $this->password = $jsonOBJ->password;
        $this->token = $jsonOBJ->token ?? null;
    }
    function insert() {
        $con = Database::connect();
        $stmt = $con->prepare("INSERT INTO `users` (`id`, `password`, `token`,
`biometric`) VALUES (?,?,?,?);");
        $stmt->bind_param("ssss", $this->id, $this->password, $this->token, $this-
>biometric);
        $stmt->execute();
        $stmt->close();
    }
    function update() {
        $con = Database::connect();
        $con->query("UPDATE `users` SET    `biometric` = '$this->biometric',
                                    `token` = '$this->token',
                                    `password` = '$this->password'
                                    WHERE `id` = '$this->id';");
    }
    public function jsonSerialize() {
        $vars["id"]    = $this->id;
        $vars["token"] = $this->token;
        return $vars;
    }
    public function generateToken() {
        return hash(hashMethod,generateRandomString().$this->id,false);
    }
    public function getDHPublicKey($encrypted) {
        return stringXOR($encrypted, $this->id);
    }
    public static function fromDB($token) {
        $con = Database::connect();
        $stmt = $con->prepare("SELECT * FROM `users` WHERE `token` = ?;");
        $stmt->bind_param("s", $token);
        $stmt->execute();
        $row = $stmt->get_result()->fetch_assoc();
        $stmt->close();
        if (!$row) throw new Exception("User not found", ErrorCode::ACCESS_DENIED);
        return new self(json_encode($row));
    }
```

```
   public static function create($pass, $biometric) {
      do {
         $user["id"] = generateRandomString(); // create long unique random id
      } while(Database::getField("users", ["id" => $user["id"]], "id") != null);
      $user["password"]  = hash(hashMethod,$pass . $user["id"],false);
      $user["biometric"] = $biometric;
      $object = new self(json_encode($user));
      $object->token = $object->generateToken();
      return $object;
   }
}
```

Frontend (Swift files)

```
import UIKit
import KeychainSwift
import SVProgressHUD
var sharedKey = String()
class PasswordVC: UIViewController {
    var checkPassword = Array(repeating: false, count: 4)
    var base64Image = String()
    var isRegistered = Bool()
    var isChangePassword = Bool()
    let keychain = KeychainSwift()
    var nextBarBtn = UIBarButtonItem()
    var step1Completed = false
    override func viewDidLoad() {
   super.viewDidLoad()
        nextBtn.alpha = 0.4
        nextBtn.isUserInteractionEnabled = false
    fileprivate func register() {
        guard let password = passwordTextField.text else { return }
        let hash = password.sha3(.sha512)
        let parameters = ["biometric": base64Image, "password": hash]
        SVProgressHUD.show()
        postHTTPS(url: "http://ma.reev.ca/register.php?id=4123ji12", callBackFunc: {
(isDone, dict, errorCode) in
            SVProgressHUD.dismiss()
            self.console.addLog(text: dict.description)
            if isDone {
            guard let data = dict["data"] as? NSDictionary else { return }
                guard let user = data["user"] as? NSDictionary else { return }
                guard let id = user["id"] as? String else { return }
                guard let token = user["token"] as? String else { return }
                guard let publicKey = data["key"] as? String else { return }
                guard let prime = data["DH_prime"] as? String else { return }
                guard let generator = data["DH_g"] as? Int else { return }
                guard let symmetric = data["symmetric_encryption"] as? String else {
return }
                self.keychain.set(id, forKey: "$ID$")
                self.keychain.set(token, forKey: "$TOKEN$")
                self.keychain.set(publicKey, forKey: "$PUBLICKEY$")
                self.keychain.set(prime, forKey: "$PRIME$")
```

```
                self.keychain.set(String(generator), forKey: "$GENERATOR$")
                self.keychain.set(symmetric, forKey: "$SYMMETRIC$")
                self.keychain.set(true, forKey: "USERREGISTERED")
                let alert = ReevAlert(title: nil, hasImage: false, subtitle:
ReevLabelAttributes(title: "Successfully Registered"), message: nil)
                let okBtn = ReevAlertAction(title: "Ok", handler: {
               alert.dismiss(animated: true, completion: nil)
                    NotificationCenter.default.post(name:
NSNotification.Name("Registered"), object: nil)
                    self.navigationController?.popViewController(animated: true)
                })
                alert.addAction(action: okBtn)
                self.present(alert, animated: true, completion: nil)
            }
        }, parameter: parameters)
    }
    fileprivate func authenticationStep1() {
        guard let id = keychain.get("$ID$") else { return }
        guard let publicKey = keychain.get("$PUBLICKEY$") else { return }
        guard let token = keychain.get("$TOKEN$") else { return }
        let dh = DH()
        let y = dh.getPublicKey().xor(key: id)
        let time = Int(Date().timeIntervalSince1970)
        var c1Plain = """
{"t": "\(time)", "y": "\(Data(y.utf8).base64EncodedString())"}"""
        guard let c1Encrypted = RSA.encrypt(string:
Data(c1Plain.utf8).base64EncodedString(), publicKey: publicKey) else {return}
        print("enc: \(c1Encrypted)")
        let parameter = ["c1": c1Encrypted, "token": token]
        SVProgressHUD.show()
        postHTTPS(url: "http://ma.reev.ca/auth.php?step=1", callBackFunc: { (isDone,
dict, errorCode) in
            SVProgressHUD.dismiss()
            self.console.addLog(text: dict.description)
            if isDone {
            guard let data = dict["data"] as? NSDictionary else { return }
                guard var c2 = data["c2"] as? String else { return }
                guard let c3 = data["c3"] as? String else { return }
                guard let cipher = data["cipher"] as? String else { return }
                c2 = String(data: Data(base64Encoded: c2)!, encoding: .utf8)!
            let serverDHPublicKey = c2.xor(key: id)
                sharedKey = dh.getSharedKey(key: serverDHPublicKey)
                let verification = (id + sharedKey + c1Encrypted).sha3(.sha512)
                if verification != c3 {
                 return
            }
                print(verification)
                let aes = AESStruct(key: sharedKey)
                let plainText = aes.decrypt(message: cipher)
                var json = NSDictionary()
                do {
               json = try JSONSerialization.jsonObject(with: plainText.data(using:
.utf8)!, options: .allowFragments) as! NSDictionary
                } catch {
               self.authenticationStep1()
```

```
                }
                guard let newToken = json["token"] as? String else { return }
                guard let newTime = json["time"] as? Int else { return }
                self.keychain.set(newToken, forKey: "$NEWTOKEN$")
                if !timestamp(int: newTime) {
               return
                }
                self.step1Completed = true
                if debugMode {
               self.navigationItem.rightBarButtonItems?.append(self.nextBarBtn)
                    self.nextBtn.setTitle("Step 2", for: .normal)
                } else {
               self.authenticationStep2()
                }
            }
        }, parameter: parameter)
    }
    fileprivate func authenticationStep2() {
   let newToken = keychain.get("$NEWTOKEN$")!
   let token = keychain.get("$TOKEN$")!
   let id = keychain.get("$ID$")!
   let aes = AESStruct(key: sharedKey)
        guard let password = passwordTextField.text else { return }
        let passwordSha = password.sha3(.sha512)
        let hash = ((passwordSha + id).sha3(.sha512) + token +
newToken).sha3(.sha512)
        let mParam = """
        {"t": "\(Int(Date().timeIntervalSince1970))", "b": "\(base64Image)", "hash":
"\(hash)"}
        """
        let mParamEncrypted = aes.encrypt(message: mParam.data(using:
.utf8)!.base64EncodedString())
        SVProgressHUD.show()
        postHTTPS(url: "http://ma.reev.ca/auth.php?step=2", callBackFunc: { (isDone,
dict, errorDict) in
            SVProgressHUD.dismiss()
            self.console.addLog(text: dict.description)
            if isDone {
            self.keychain.set(newToken, forKey: "$TOKEN$")
                self.keychain.set("", forKey: "$NEWTOKEN$")
                let alert = ReevAlert(title: nil, hasImage: false, subtitle:
ReevLabelAttributes(title: "Successfully Authenticated"), message: nil)
                let okBtn = ReevAlertAction(title: "Ok", handler: {
               self.goToHome()
                })
                alert.addAction(action: okBtn)
                self.present(alert, animated: true, completion: nil)
            } else {
         }
        }, parameter: ["m": mParamEncrypted])
    }
}
```

Bibliography

[1] M. H. Miraz, M. Ali, P. S. Excell and R. Picking, "A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT), Proc. of IEEE Inter- net Technologies and Applications (ITA)," Glyndwr University, Wrexham, North East Wales, UK, 2015.

[2] E. Stobert and R. Biddle, "Workshop on Home Usable Privacy and Security (HUPS)," Workshop on Home Usable Privacy and Security (HUPS), 24 July 2013. [Online]. Available: http://cups.cs.cmu.edu/soups/2013/HUPS/HUPS13-ElizabethStobert.pdf. [Accessed 8 2019].

[3] M. S. Raniyal, I. Woungang and . K. Dhurandher, "An RSA-Based User Authentication Scheme for Smart-Homes Using Smart Card.," Ryerson University, Toronto, Ontario, 2018.

[4] A. Menezes, P. v. Oorschot and S. Vanstone, "Handbook of Applied Cryptography," CRC Press: Boca Raton, 1996.

[5] M. Burrows, M. Abadi and R. Needham, A logic of authentication, ACM Trans. on Computer Systems, vol. 8, 1990, pp. 18-36.

[6] Z. Tian, K. Wright and X. Zhou, "The DarkLight Rises: Visible Light Communication in the Dark," [Online]. Available: http://www.cs.dartmouth.edu/ xia/papers/mobicom16-darklight.pdf. [Accessed September 2019].

[7] T. Borgohain, A. Borgohain, U. Kumar and S. Sanyal, "Authentication Systems in Internet of Things," [Online]. Available: https://arxiv.org/abs/1502.00870. [Accessed September 2019].

[8] N. Shone, C. Dobbins, W. Hurst and Q. Shi, "Digital Memories Based Mobile User Authentication for IoT," Liverpool, Oct. 26-28, 2015.

[9] P. P. Gaikwad, J. P. Gabhane and S. S. Golait, "3-Level Secure Kerberos Authentication for Smart Home Systems Using IoT," IEEE Intl. Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, Sept. 4-5, 2015.

[10] H. Om and M. Reddy, "RSA based remote password authentication using smart card," Journal of Discrete Mathematical Science & Cryptography, 2012.

[11] X. Wang and W. Zhang, "An efficient and secure biometric remote user authentication scheme using smart cards," Wuhan, China, Dec 19-20, 2008.

[12] "Remote user authentication using NFC," [Online]. Available: https://www.google.ch/patents/US20110212707. [Accessed September 2019].

[13] M. S. Raniyal, I. Woungang and S. K. Dhurandher, "An Inter-Device Authentication Scheme for Smart Homes using One-Time-Password over Infrared Channel".

[14] W. H. Yang and S. P. Shieh, Password authentication schemes with smart cards, Computers and Security, vol. 18, 1999, pp. 727- 733.

[15] C. K. Chan and L. M. Cheng, Cryptanalysis of a timestamp-based password authentication scheme, Computers and Security, 2002, pp. 74-76.

[16] L. Fan, J. H. Li and H. W. Zhu, "An enhancement of timestamp based password authentication scheme," *Computers & Security,* vol. 21, pp. 665-667, 2002.

[17] Y. Liu, A. M. Zhou and M. X. Gao, "A new mutual authentication scheme based on nonce and smart cards," *Computer Communications,* pp. 2205-2209, 2008.

[18] J. J. Shen, C. W. Lin and M. S. Hwang, "Security enhancement for the timestamp-based password authentication scheme using smart cards," *Computers and Security,* vol. 7, no. 22, pp. 591-595, 2003.

[19] H. Om and S. Kumari, "omment and modification of RSA based remote password authentication using smart card," *Journal of Discrete Mathematical Science and Cryptography,* pp. 625-635, 2017.

[20] M. Wazid, A. K. Das, V. Odelu, N. Kumar and W. Susilo, "Secure Remote User Authenticated Key Establishment Protocol for Smart Home Environment," Transactions on Dependable and Secure Computing, Oct. 2017.

[21] S. Z. Reyhani and M. Mahdavi, "User Authentication Using Neural Network in Smart Home Networks," *International Journal of Smart Home,* vol. 1, no. 2, July, 2007.

[22] W. Bae and J. Kwak, "Smart Card-Based Secure Authentication Protocol in Multi-Server IoT Environment," Multimed Tools Appl., 2017.

[23] M. Dammak, O. R. M. Boudia, M. A. Messous, S. M. Senouci and C. Gransart, "Token-Based Lightweight Authentication to Secure IoT Networks," Proc. of the 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, Jan. 11-14, 2019.

[24] R. Amin, S. K. H. Islam, M. K. Khan, A. Karati, D. Giri and S. Kumari, "A two-factor RSA-based robust authentication system for multiserver environments," *Security and Communication Networks,* p. 15, 2017.

[25] Y. Zhao, S. Li and L. Jiang, "Secure and Efficient User Authentication Scheme Based on Password and Smart Card for Multiserver Environment," *Security and Communication Networks,* vol. 2018, p. 13.

[26] T.-T. Truong, M.-T. Tran, A.-D. Duong and I. Echizen, "Provable Identity Based User Authentication Scheme on ECC in Multi-server Environment," *Wireless Personal Communications,* vol. 95, no. 3, p. 2785–2801, 2017.

[27] D. Whitfield and M. E. Hellman, "New Directions in Cryptography," *Transactions on Information Theory,* Vols. IT-22, no. 6, 1976.

[28] P. K. Dhillon and S. Kalra, "Secure multi-factor remote user authentication scheme for Internet of Things environments," in *Intl. Journal of Communication Systems,* 26 Apr. 2017.

[29] R. L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, 1978.

[30] RSA Labs, "Public Key Cryptography Standards (PKCS), Number 1," RSA Labs, 9 2019. [Online]. Available: https://tools.ietf.org/html/rfc3447.

[31] M. J. Dworkin, E. B. Barker, J. R. Nechvatal, J. Foti, L. E. Bassham, E. Roback and J. F. D. Jr., "Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication, November 26, 2001.

[32] "Swift," Apple Inc., [Online]. Available: https://swift.org. [Accessed September 2019].

[33] CloudLinux, Inc., "CloudLinux OS," CloudLinux, Inc., 2010. [Online]. Available: https://www.cloudlinux.com. [Accessed 8 2019].

[34] The PHP Group, "PHP," The PHP Group, [Online]. Available: https://www.php.net.

[35] MariaDB Foundation, MariaDB Foundation, [Online]. Available: https://mariadb.org. [Accessed September 2019].

[36] FaceX, "Face Recognition APIs," FaceX, 2018. [Online]. Available: https://facex.io. [Accessed 8 2019].

[37] OpenSSL Software Foundation, OpenSSL Software Foundation, [Online]. Available: https://www.openssl.org. [Accessed 8 2019].

[38] Oak Ridge National Laboratory, 13 2 2018. [Online]. Available: https://www.olcf.ornl.gov/2018/02/13/frontier-olcfs-exascale-future/. [Accessed September 2019].

[39] W. Muła and D. Lemire, "Faster Base64 Encoding and Decoding Using AVX2 Instructions," *ACM Transactions on the Web (TWEB)*, vol. 12, no. 3, July 2018.

[40] Dworkin and M. J., "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," Federal Information Processing Standards Publication 202, August 2015.

[41] G. H. Gadiyar and V. Ramanathan Padma, "The Discrete Logarithm Problem over Prime Fields: The Safe Prime Case. The Smart Attack, Non-Canonical Lifts and Logarithmic Derivatives," Czechoslovak Mathematical Journal, Received March 20, 2017. Published online February 2, 2018..

[42] C. C. Yang, R. C. Wang and T. Y. Chang, "An improvement of the Yang-Shieh password authentication schemes," *Applied Mathematics and Computation*, vol. 162, pp. 1391-1396, 2005.

[43] H. Y. Chien, J. K. Jan and Y. M. Tseng, "An efficient and practical solution to remote authentication: smart card," *Computers & Security*, vol. 21, no. 4, p. 372–375, 2002.