Designing Secure Authentication Schemes for Distributed Systems

Thesis for the Degree of Doctor of Philosophy (PhD)

by name: Norbert Oláh

Supervisor: Dr. Andrea Pintér-Huszti

University of Debrecen

Doctoral Council of Natural Sciences and Information Technology

Doctoral School of Informatics

Debrecen, 2022

Hereby I declare that I prepared this thesis within the Doctoral Council of Natural Sciences and Information Technology, Doctoral School of Informatics, University of Debrecen in order to obtain a PhD Degree in Informatics at Debrecen University.
The results published in the thesis are not reported in any other PhD theses.
Debrecen, 2022.....

.....
signature of the candidate

Hereby I confirm that Norbert Oláh candidate conducted his studies with my supervision within the Theoretical computer science, data security and cryptography Doctoral Programme of the Doctoral School of Informatics between 2017 and 2021. The independent studies and research work of the candidate significantly contributed to the results published in the thesis.
I also declare that the results published in the thesis are not reported in any other theses.
I support the acceptance of the thesis.
Debrecen, 2022.....

.....
signature of the supervisor

Designing Secure Authentication Schemes for Distributed Systems

Dissertation submitted in partial fulfilment of the requirements
for the doctoral (PhD) degree in Informatics

Written by Norbert Oláh certified Business Informatics MSc.

Prepared in the framework of
the Doctoral School of Informatics of University of Debrecen
Theoretical computer science, data security and cryptography programme

Dissertation advisor: Dr. Andrea Pintér-Huszti

The official opponents of the dissertation:

Dr. .....
Dr. .....

The evaluation committee:

chairperson:	Dr. .....
members:	Dr. .....
	Dr. .....
	Dr. .....
	Dr. .....

The date of the dissertation defence: 2022 .....

Contents

1 Introduction **2**
- 1.1 Historical background . . . 2
- 1.2 Presentation overview and our results . . . 7
- 1.3 Credits . . . 9

2 Cryptographic Protocol Building Blocks **11**
- 2.1 Merkle Tree . . . 11
- 2.2 Elliptic Curves Cryptography . . . 13
  - 2.2.1 Shamir's Secret Sharing . . . 16
  - 2.2.2 Description of Cryptographic Primitives . . . 17
- 2.3 Communication Channels . . . 18

3 Background of the Security Analysis Techniques **20**
- 3.1 AVISPA . . . 21
- 3.2 ProVerif . . . 23
  - 3.2.1 Applied π Calculus . . . 23
  - 3.2.2 ProVerif . . . 25
    - Security properties . . . 27
- 3.3 The Concept of Provable Security . . . 28

4 Distributed Cloud Authentication Protocols **30**
- 4.1 Cloud Authentication Protocol Using a Merkle Tree . . . 31
  - 4.1.1 Our Contribution and Literature . . . 31
  - 4.1.2 The Proposed Scheme . . . 34
    - Registration . . . 34
    - Authentication . . . 36
    - Synchronization . . . 37
  - 4.1.3 Security Analysis . . . 38
    - Security Requirements . . . 39
    - Adversarial Model . . . 39
    - Mutual Authentication and Key Secrecy . . . 40
    - Formal Model . . . 40

Security Properties 40
Key Freshness and Confirmations 43
4.2 Scalable Distributed Authentication for Cloud Services . . . 44
4.2.1 Our Contribution and Literature 44
4.2.2 The Proposed Scheme 46
Setup 47
Scalability 48
Authentication Phase 49
4.2.3 Security Analysis 52
Cloud Authentication Security Model 52
The threshold hybrid corruption model 54
Practical Issues 64
Java Simulation Result 65

5 Provably Secure Identity-Based Remote Password Registration 67
5.1 Our Contribution and Literature 67
5.2 Our Proposed Registration Protocol 71
5.2.1 Setup 71
5.2.2 Registration Phase 72
5.3 Security Analysis 74
5.3.1 Registration Security Model 74
5.3.2 Security Proof 81
5.4 Efficiency 91
Computation Cost 91
Comparison with Other Schemes 92

6 Scalable, Password-Based and Threshold Authentication for Smart Homes 93
6.1 Our Contribution and Literature 93
6.2 The Proposed Scheme 98
6.2.1 Setup Phase 99
6.2.2 Authentication Phase 100
First Subphase 101
Second Subphase 102
Third Subphase 104
6.3 Security Analysis 105

6.3.1 Security Validation 106
6.4 Computation and Communication Cost Analysis 109
6.4.1 Devices 110
6.4.2 Communication and Computation Costs of the Protocol 110
Computation Cost 110
Communication Cost 112

7 Appendix **114**

Summary **119**

Összefoglaló **138**

Bibliography **158**

A. List of papers of the author and citations to them **171**

B. List of talks of the author **174**

C. Acknowledgements **176**

D. Köszönetnyilvánítások **177**

Chapter 1

Introduction

The present dissertation demonstrates three new entity authentication schemes and a user registration protocol, which is necessary before the first identity verification. Distributed identity verification is carried out by multiple participants to secure cloud computing services and smart home environments. Via formal analysis we demonstrate that the protocols fulfil the necessary security requirements. Our solutions are more efficient than the current practical and theoretical schemes.

One of the essential issues during online communication is the secure authentication between the participants. The proper authentication serves to avoid the different attacks (e.g. impersonation attack). However, in the case of inproper authentication, user access control, confidentiality and integrity of user data are not provided. The authentication schemes require several security properties, which depend on the attributes of environments. One of the most widely used authentication methods is based on short secrets like passwords. The registration process (the initial phase of the protocol) must be executed before the authentication, but it receives insufficient attention in the scientific literature.

1.1 Historical background

Information systems are only as secure as their weakest point. One of the essential security issues during online communication is the secure authentication between the participants. Entity authentication is based on the possession of secret information, which is known and verified only by the entities participating in the process.

In scientific papers, one-factor and two-factor authentication solutions are used primarily. In the case of one-factor authentication, the user is usually protected only by a password, which is a string of characters used

to verify the identity of a user. Since it is easy to use and deploy, password usage is a widespread form of user authentication. Passwords are applied in many cryptographic schemes and systems, e.g., password authentication schemes or Password-Based Key Derivation Function (PBKDF) ([97, 118, 29, 74]).

The password is also often used as authentication information in key exchange protocols. In the case of a password-authenticated key exchange (PAKE), a user and a server establish a session key for a secure channel after user registration ([16, 27, 28, 85, 68]). In [28] a Diffie-Hellman-based password-authenticated key exchange protocol is proposed. Their protocol provides formal security proof in the random oracle model against passive and active adversaries. Two-factor authentication provides a higher level of security than one-factor authentication, as it provides an extra security factor for the login process. Two-factor authentication forces the adversary to expand the implementation of his attacks even if the username and the password are known. Several two-factor authentication schemes have been proposed that depend on a long cryptographic secret and a short password ([96, 79]). In 2011, Choudhury *et al.* demonstrated a two-factor authentication protocol ([40]) for cloud computing where one of the factors is the smart card, and the other one is the password. They applied an out-of-band channel. Later, Chen and Jiang detected an impersonation attack against the scheme of Choudhury *et al.* and proposed a new authentication framework which did not use the out-of-band channel ([33]). Compared to [33], we proposed a more efficient authentication solution which only uses fast cryptographic operations (hash, xor). Our authentication protocol for a cloud environment ([58]) applies two-factor authention based on a Merkle tree.

Furthermore, forward secrecy is crucial and appears in several schemes ([55, 66, 98]). Pointcheval and Zimmer ([98]) demonstrated a multi-factor authentication scheme applying a password, a long cryptographic secret, and biometric data as well.

In a multi-server environment usually, threshold password-authenticated key exchange protocols are designed. First, the threshold variant of PAKE was recommended in [86]. They apply multiple servers in a threshold scheme to prevent server corruption and handle password vulnerability. These solutions assume that threshold-number servers may become cor-

rupt, and not all servers are colluding. Devriş Işler and Alptekin Küpçü introduced a scheme ([62]) where the protocol ensures that multiple storage providers can be employed, and the adversary needs to corrupt the login server and threshold-many storage providers to be able to mount an offline dictionary attack. Later, they proposed a new framework ([63]) for distributed single password protocols (DiSPP). In their protocol, users store the secret among storage providers (e.g. personal devices, online storage providers) and access it by using their password. Mario Di Raimondo and Rosario Gennaro recommended two threshold password-authenticated key exchanges ([99]) where the protocols require $n > 3t$ servers to work. They enforce a transparency property: from the point of view of the client, the protocol should look exactly like a centralized KOY protocol ([67]). Password-Protected Secret Sharing (PPSS) scheme with parameters (t, n) was formalized by Bagherzandi et al. ([9]). Jarecki et al. ([65]) presented the first round-optimal PPSS scheme, requiring just one message from user to server and from server to user, and proved its security in the challenging password-only setting where users do not have access to an authenticated public key. The scheme uses an Oblivious Pseudorandom Function (PRF) and builds the first single-round password-only Threshold-PAKE protocol in the Common Reference String model (CRS) and Random Oracle Model (ROM). In [59], we designed a scalable and distributed authentication protocol for cloud services, which applies parallel multiple servers for the user authentication. We introduced the new threshold hibrid corruption model for analyzing corrupt participants on the system and our solution provides the property of robustness. Compared to our earlier proposed protocol ([58]), we use secret splitting technique and we also achieved the scalable property.

The vulnerabilities of password schemes are well-known. Users may choose "weak" passwords or not change the default passwords, which are accessible for attackers to guess. Another criticism of passwords is that if any site where a specific password is reused becomes compromised and the system administrators do not follow the best industry practices, the participants' other accounts may also become compromised by the attacker who can guess the password with an offline attack. Several attacks aim to figure out passwords, applying dictionary, rainbow tables or brute-force attacks. The "password-cracking tools" can be very efficient (such as

Hashcat [54] and John the Ripper [95]), and it can be assumed that the hash values of the leaked password are not safer than plain passwords when compromised by an attacker ([44, 84, 26]).

Before the authentication, the remote registration of passwords is one of the most important security aspects and the initial step of all remote password-based protocols, but does not get enough attention. In cryptographic password-based protocols, password registration is often skipped assuming that the passwords are set securely and known to the parties before the protocol is executed and implemented. During the implementation of registration, the chosen passwords are transmitted to the server through a secure channel (e.g. Transport Layer Security (TLS) channel) and users create or activate their account with their password through the verification email. Hence the client's password is revealed to the server. However, the TLS implementations are rather complex with the use of certificates, as the users need to manage and update them. Registrations may be incomplete, and one of the shortcomings is when the TLS channel is not used, it may lead to a breach and leakage of confidential data (which conflicts with the General Data Protection Regulation (GDPR)). During login, passwords usually appear in cleartext at the server, and security can be harmed if the TLS channel is established with a compromised server's public key (a major concern given today's common Public Key Infrastructure failures). To improve the security of password registration, Kiefer and Manulis introduced a new family of protocols called Blind Password Registration (BPR) for Verifier-Based Password-Authenticated Key Exchange (VPAKE) ([71]) and two-server PAKE ([72]). They proposed Zero-Knowledge Password Policy Checks (ZKPPC), which enables blind registration. BPR protocol can be executed over the TLS channel established between the client and the server. They define a security model for stand-alone blind password registration protocols, which meets the requirements of compliance with regulations and resistance to dictionary attacks. We proposed a new password registration protocol, which is more cost-effective than the above-mentioned TLS-based and the other blind solutions ([71], [72]). It is not necessary to manage certificates or execute Zero-Knowledge (ZK) proof. During our registration, we apply the bilinear mapping, MAC and hash function, hence we achieve a favorable computational time.

Identity-Based Cryptography (IBC) is an alternative to Certificate-Based Cryptography, which was first proposed by Shamir in 1984 ([108]). The basic idea of identity-based cryptography involves an identity-based asymmetric key pair where an arbitrary string can be used as a user public key. For this, a trusted authority or Private Key Generator (PKG) is needed to generate private keys from public keys and a master secret key. The PKG also publishes public information required for all encryption, decryption, signature, and verification algorithms in the system. Boneh and Franklin ([24]) formalized the notion of Identity-Based Encryption (IBE), which uses bilinear pairings over elliptic curve groups. In the IBE setting, a user's public key can be any arbitrary string, which is typically an e-mail address. There is no need for Bob to go to the Certificate Authority to verify Alice's public key. In this way, an IBE can greatly simplify certificate management. However, IBC has a well-known disadvantage. If the PKG is corrupted and its secret key is revealed, all messages and secret keys are compromised. A cross-platform identity-based system using WebAssembly is proposed in [114]. They recommend an efficient library called CryptID, which is an open-source IBC implementation for desktops, mobiles, and Internet of Things (IoT) platforms. In the case of IoT devices, they are often very vulnerable due to weak protection (weak or default passwords) and poor maintenance. Numerous studies have addressed the security vulnerabilities of IoT devices ([73, 70, 2]). Bugs have been found in a wide range of devices, including routers ([116]), smart cams ([106]), baby monitors ([18]) and smart plugs ([82]). We proposed a new threshold and password-based, distributed, mutual authenticated key agreement with key confirmation protocol designed for a smart home environment. The proposed protocol is an identity-based, scalable and robust scheme that forces the adversary to corrupt $l-1$ smart home devices to perform an offline dictionary attack, where l is the password threshold. Short-lived identity-based key pair and bilinear pairing are applied in our protocol.

Proving the security requirements of cryptographic protocols is a challenging issue. Several protocols had been thought to be secure until a simple attack was found. We apply different security analysis methods for our protocols. We use the AVISPA tool to analyse the smart home system, which applies High-Level Protocol Specification Language (HLPSL).

HLPSL is based on a fragment of the Temporal Logic of Actions and a translation into the rewrite-based formalism Intermediate Format (IF). We also use the ProVerif tool to analyse a cloud authentication protocol based on applied Pi calculus. Moreover, we provide provable security analysis based on the polynomial reduction technique for the recommended registration and a cloud authentication protocols.

1.2 Presentation overview and our results

The present work consists of three authentication schemes and a user registration protocol.

The first chapter contains the scientific background of the user authentication schemes and solutions.

In the second chapter, we detail the cryptographic primitives applied in our protocols and the necessary preliminaries.

Chapter 3 covers automated security analysis tools and give the details of the concept of provable security.

In Chapter 4, two distributed authentication are proposed for cloud services. In the Chapter 4.1, we demonstrate a two-factor authentication scheme for cloud computing services using a Merkle tree ([57]). We have extended the scheme in [58] and also provided a security analysis in applied-pi calculus. A leaf of the Merkle tree is the hash of a password share, and the root element is verified in order to confirm the correctness of the whole one-time password. In the registration phase, the secret keys are exchanged generating a large amount of one-time passwords between the user and the cloud servers. In the authentication phase, a distributed authentication is applied where the randomly chosen server verifies the correctness of the one-time password with the usage of the Merkle tree. Comparing the efficiency of our authentication phase to the work of [33, 40], we found that our scheme is more efficient, since only hash calculations are performed. A message authentication key (MAC) exchange is also provided to guarantee data origin integrity for the latter interactive communication.

In Chapter 4.2, we recommend a mutual authentication protocol with key agreement ([59]), where the identity verification is carried out by multiple servers applying secret splitting technology on server side. The pro-

tocol results in a session key, which provides the confidentiality of the subsequent messages between the participants. In our solution, we also achieve robustness and scalability. To show that the proposed protocol is provably secure, we apply the threshold hybrid corruption model. We assume that among the randomly chosen k servers, there is always at least one uncorrupted server and the authentication server reveals only the long-lived keys. We prove that the protocol is secure in the random oracle model if MAC is universally unforgeable under an adaptive chosen-message attack, the symmetric encryption scheme is indistinguishable under chosen plaintext attack, moreover, ECCDH assumption holds in the elliptic curve group.

In Chapter 5, a password registration scheme is demonstrated based on the identity-based cryptography, *i.e.* both the user and the service provider are authenticated by their short-lived identity-based secret key. For secure storage, a bilinear map with a salt is applied, therefore in case of an offline attack the adversary is forced to calculate a computationally expensive bilinear map for each password candidate and salt, which slows down the attack. We introduce a new adversarial model with a new secure password registration scheme. We show that the proposed protocol is based on the assumptions that the Bilinear Diffie-Hellman problem is computationally infeasible, the bilinear map is a one-way function and MAC is existentially unforgeable under an adaptive chosen-message attack. The proposed protocol suits our smart home user authentication scheme where the values of the bilinear map are stored on the IoT devices. Our cloud scheme ([58]) can also be easily modified with the proper long-lived key setting to be compatible with our registration scheme.

In Chapter 6, we present a threshold and password-based, distributed, mutual authenticated key agreement with key confirmation protocol for a smart home environment ([60]). In our proposed cloud authentication schemes ([57], [58], [59]), we assume that the cloud servers are always available. However, the devices can be of various types in smart home systems, which means some devices are battery-powered or resource-constrained and might not be available. We need to consider this property of smart homes, and we propose a new smart home user authentication scheme with a secret sharing technique, where we require k devices to be available out of n ones, which can be chosen dynamically. The proposed protocol

is a scalable and robust scheme, which forces the adversary to corrupt $l-1$ smart home devices, where l is the threshold, in order to perform an offline dictionary attack. The protocol is designed to achieve password-only setting, and end-to-end security if the chosen IoT devices are also authenticated besides the user. We also provide a security analysis of the protocol in AVISPA. We apply the On-the-fly Model-Checker and the Constraint-Logic-Based Attack Searcher to perform the protocol verification for bounded numbers of sessions. We show that the proposed protocol provides secrecy of the session key and mutual authentication of the user and the device manager. Since efficiency is a crucial aspect, we implemented our protocol to measure the computation and communication costs. We demonstrate that our solution is appropriate and eligible for smart homes.

1.3 Credits

Results of *Ditributed Cloud Authentication Protocols* are based on

> A. Huszti, N. Oláh, *A simple authentication scheme for clouds*, In Proceedings of 2016 IEEE Conference on Communications and Network Security (CNS), IEEE, (2016), 565–569.,
>
> A. Huszti, N. Oláh, *Security analysis of a cloud authentication protocol using applied pi calculus*, International Journal of Internet Protocol Technology, vol. 12, no. 1, (2019), 16 – 25. SJR: Q4 and
>
> A. Huszti, N. Oláh, *Provably Secure Scalable Distributed Authentication for Clouds*, Lecture Notes In Computer Science 12579., Chapter 10, (2020), 188–210. CORE: B

Results of *Scalable, password-based and threshold authentication for Smart Homes* are based on

> A. Huszti, Sz. Kovács, N. Oláh, *Scalable, password-based and threshold authentication for Smart Homes*, Int. J. Inf. Secur. 21, https://doi.org/10.1007/s10207-022-00578-7, (2022), 707— 723. SJR: Q2, IF: 2.067

Results of *Provably Secure Identity-Based Remote Password Registration* are based on

> Cs. Bertók, A. Huszti, Sz. Kovács, N. Oláh, *Provably Secure Identity-Based Remote Password Registration*, to appear in Publicationes Mathematicae Debrecen. SJR: Q2, IF: 0.636

Chapter 2

Cryptographic Protocol Building Blocks

This chapter details the cryptographic primitives applied in our protocols and the necessary preliminaries. We specify, inter alia, the Merkle tree, the types of communication channels, review the theory of elliptic curves and the related bilinear pairings. Furthermore we define the cryptographic primitives and provide the message authentication code and encryption scheme definitions. As a first step, let's check the Merkle tree.

2.1 Merkle Tree

In Chapter 4.1, we recommend a two-factor authentication scheme for cloud computing used by a Merkle tree. The main idea is to provide shared responsibility of handling the one-time password, *i.e.* the one-time password is stored and shared among the cloud servers. We apply a Merkle tree or hash tree ([90]) for verifying the correctness of the one-time password. A Merkle tree is a binary tree where each non-leaf node is labeled with the hash value of the concatenation of the labels of its two children.

Let $H()$ and $||$ denote a hash calculation and a concatenation operation, respectively. We extend the general Merkle tree with an extra level (K_{i_j}). This level contains the shares of the one-time password, and the servers update these values later in the synchronization phase. There are 2^n values denoted by K_{i_j}, where $i \in \{1, \ldots, 2^{n-1}\}$ and $j \in \{1, 2\}$, the Merkle tree leaf node labels T_{i_j} are the hash values of K_{i_j}, *i.e.* $T_{i_j} = H(K_{i_j})$. All the other node values Y_k are calculated as a hash of two concatenated children labels (*e.g.* $Y_1 = H(T_{1_1}||T_{1_2})$). Eventually, a tree with 2^n leaves and all together with $2^{n+1} - 1$ nodes is built up.

Especially, we label the root node by Y_r and its two children values by Y_{d_1} and Y_{d_2}. You can see an example of our Merkle tree in Figure 2.1.

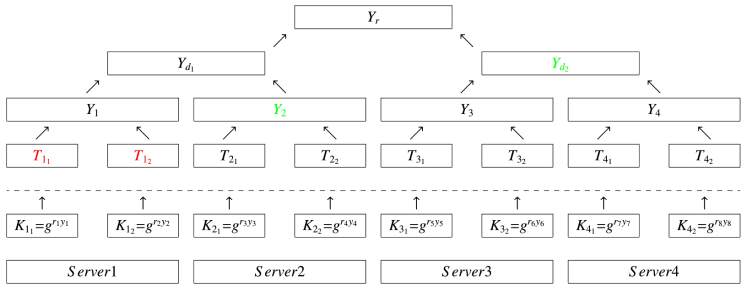


Figure 2.1. A Merkle tree with 8 leaves

The most important advantage of the Merkle tree is the low modification cost of a leaf value. In our protocol, after each authentication, the values T_{i_j}, for a given i and $j = 1, 2$ need to be updated, consequently all the node values in the path to the root are modified (including the root node value). The new node labels can be computed with the help of the actual siblings, thus we do not have to rebuild the whole tree. Let us see the example of Figure 2.1. We suppose that T_{1_1} and T_{1_2} are reset and we would like to compute the new Y_r. We can calculate the new $Y_1 = H(T_{1_1}||T_{1_2})$, then we get Y_{d_1} and Y_r via $Y_{d_1} = H(Y_1||Y_2)$ and $Y_r = H(Y_{d_1}||Y_{d_2})$, where Y_2, Y_{d_2} are the siblings. We emphasize that in case of 2^n leaves, only n hash calculations are needed for computing the new root node value.

The Merkle tree construction is applied to obtain efficient one-time password verification. For each user there is a Merkle tree, where the values T_{i_j} yield the one-time password shares, therefore we get the whole one-time password as $T = T_{1_1}||T_{1_2}||T_{2_1}||\ldots||T_{2_2^{n-1}}$. The Merkle tree is stored on user and server sides as well. On server side the one-time password is stored distributed among the cloud servers.

2.2 Elliptic Curves Cryptography

We review some of the theory of elliptic curves over finite fields, further details can be found in [89]. Let $\mathbb{F}_q$ denote the finite field of q elements, where q is a prime or a prime power. We apply E to define elliptic curve over $\mathbb{F}_q$, where $q = p^m$ and p is the characteristic of $\mathbb{F}_q$. If $p > 3$, then $E(\mathbb{F}_q)$ is the set of all solutions in $\mathbb{F}_q \times \mathbb{F}_q$ to an affine equation

$$y^2 = x^3 + ax + b,$$

with $a, b \in \mathbb{F}_q$ and the $4a^3 + 27b^2 \neq 0$, together with an extra point O, called the point at infinity. One can equip $E(\mathbb{F}_q)$ with a binary operation by using the "chord and tangent" process. On this way one gets an Abelian group, where the point at infinty is the neutral element. The main advantage of applying elliptic curves in cryptography is that the same level of security is ensured with shorter key lengths comparing to other public-key systems.

Elliptic Curve Cryptography (ECC) is based on the infeasibility of Elliptic Curve Discrete Logarithm Problem, where the definition is the following:

2.2.1 Definition. Let $G \in E(\mathbb{F}_q)$ be a point of order n, and let $< G >$ be the subgroup of $E(\mathbb{F}_q)$ generated by G. The Elliptic Curve Discrete Logarithm Problem is to determine the value of $a \in \mathbb{Z}_n$ in the equation $A = aG$, for a given point $A \in < G >$.

One of the most often used key agreement protocol is the Diffie-Hellman protocol ([41]). The purpose of the Diffie-Hellman protocol is to enable two parties to securely exchange a session key which can then be used for encrypting and authenticating messages. The security of the Elliptic Curve Diffie-Hellman protocol is based on the infeasibility of calculating elliptic curve discrete logarithms. The algorithm works as follows:

- Alice and Bob select random secret values a, $b \in \mathbb{Z}_n^*$, respectively.
- Alice and Bob calculate aG and bG, respectively, where G is a generator in $E(\mathbb{F}_q)$ and send their values to each other.
- Both of them compute the shared secret $K = a(bG) = a(bG) = abG$.

Our password hashing scheme in Chapter 5 and the long-lived keys in Chapter 6 based on the bilinear pairings on elliptic curves. Let us review the definition of the admissible bilinear map ([24]).

2.2.2 Definition. Let $\mathbb{G}$ be an additive group and $\mathbb{G}_{\mathbb{T}}$ be a multiplicative group of order q for some large prime q. A map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ is an admissible bilinear map if satisfies the following properties:

1. Bilinear: We say that a map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ is bilinear if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}$ and all $a, b \in \mathbb{Z}$.

2. Non-degenerate: The map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_{\mathbb{T}}$. Since $\mathbb{G}$, $\mathbb{G}_{\mathbb{T}}$ are groups of prime order, if P is a generator of $\mathbb{G}$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_{\mathbb{T}}$.

3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}$.

The Weil and Tate pairings are examples of such constructions. Typically, $\mathbb{G}$ is an elliptic-curve group and $\mathbb{G}_{\mathbb{T}}$ is the multiplicative group of a finite field. The Bilinear Diffie-Hellman Problem is strongly related to the bilinear map.

We also use the following hard problem as well and give formalization of the definitions:

2.2.3 Definition. *Bilinear Diffie-Hellman Problem.* Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ be a bilinear map on $(\mathbb{G}, \mathbb{G}_{\mathbb{T}})$. Given (P, aP, bP, cP) for some $a, b, c \in Z_q^*$, compute $\hat{e}(P, P)^{abc}$.

2.2.4 Definition. *One-way Pairing [50].* Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ be a bilinear map on $(\mathbb{G}, \mathbb{G}_{\mathbb{T}})$. We say that $\hat{e}$ is a one-way pairing if for any polynomial time (in a security parameter κ) algorithm $\mathcal{A}$ that takes as input $G \in \mathbb{G}$ and $g \in \mathbb{G}_{\mathbb{T}}$ and produces as output an element of $\mathbb{G}$ the probability $Pr[\hat{e}(G, \mathcal{A}(G, g)) = g]$ is negligible. The probability is taken over the possible values of G and g.

2.2.5 Definition. (Computational Diffie-Hellman Problem) Let q be a prime $a, b \in \mathbb{Z}_q^*$ and $\mathbb{G}$ be a cyclic multiplicative group of order q. For a given (g, g^a, g^b) (with $g \in \mathbb{G}$) compute g^{ab}.

2.2.6 Lemma. *Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map defined as in Definition 2.2.2. If the Computational Diffie-Hellman (CDH) Problem is infeasible in $\mathbb{G}_\mathbb{T}$ then $\hat{e}$ is a one-way pairing.*

Proof: Let G be a generator of $\mathbb{G}$. Then it can be seen that $\hat{e}(G, G) = g \in \mathbb{G}_T$ is also a generator of $\mathbb{G}_T$. Suppose now that $\hat{e}$ is not one-way, thus for every given $g^a, g^b \in \mathbb{G}_T$ we can find (in polynomial time) $aG, bG \in \mathbb{G}$ such that $\hat{e}(G, aG) = g^a$ and $\hat{e}(G, bG) = g^b$. However in this case $\hat{e}(aG, bG) = \hat{e}(G, G)^{ab} = g^{ab}$ which contradicts the infeasibility of the CDH problem. □

In the case of password hashing scheme we need an efficient way to map passwords first into $\mathbb{Z}_p$ (with p given in the Section 6.4), then these elements of $\mathbb{Z}_p$ into a point on the curve. Mapping passwords into $\mathbb{Z}_p$ can be done easily by concatenating the ASCII value of each character, then taking the result mod p. For mapping messages (passwords) from $\mathbb{Z}_p$ to an elliptic curve over $\mathbb{Z}_p$ it is desirable to choose an "almost always" injective encoding. A similarly good property would be that the mapping is efficiently computable and reversible, so we can easily obtain both the encoded message from an (almost) arbitrary element of $\mathbb{Z}_p$ and also our initial message (password) from (almost) any given curve point. Finally for cryptographic algorithms it is also desirable that the mapping is surjective, since otherwise our possible message (password) space is unnecessarily limited. Unfortunately general encodings which fulfills these requirements are very scarce, however in [48] the authors provide such a mapping.

Since the results in [48] are formulated in a more general form, stating them here is out of the scope of the present dissertation, thus we only provide their main result in a simplified form and for the exact technical details we refer to [48] Sections 1 and 2. For this paragraph, let q be an odd prime congruent to 3 modulo 4, g a positive integer and $E : y^2 = x^{2g+1} + a_1 x^{2g-1} + \cdots + a_g x$ an elliptic curve over $\mathbb{Z}_q$. Denote by $\left(\frac{\cdot}{q}\right)$ and $\sqrt{\cdot}$ the Legendre symbol and the square root over $\mathbb{Z}_q$. Finally let

$\varepsilon(x) = \left(\frac{f(x)}{q}\right)$. The proposed encoding is

$$tr : \mathbb{Z}_q \longrightarrow E$$
$$x \mapsto \left(\varepsilon(x) \cdot x,\, \varepsilon(x)\sqrt{\varepsilon(x) \cdot f(x)}\right)$$

Since $\varepsilon(x) \in \{-1, 0, 1\}$ and for every $x \in \mathbb{Z}_q$ $f(-x) = -f(x)$ then it is clear that $(\varepsilon(x)\sqrt{\varepsilon(x) \cdot f(x)})^2 = f(\varepsilon(x) \cdot x)$ holds. Let T denote the set of roots in $\mathbb{Z}_q$ of $f(x)$ and W the set consisting of the points of E in the form of $(x, 0)$, and the point at infinity. In [48] the authors proved that the encoding tr induces a bijection $\mathbb{Z}_q \setminus T \longrightarrow E \setminus W$. In our case the elliptic curve is $E : y^2 = x^3 + x$ over $\mathbb{Z}_q$, where the prime q is given in Section 5.4 and Section 6.4. It can be verified that both requirements stated in [48] are fulfilled thus the theorem holds for E. Thus based on the results stated in [48] tr is "almost always" a bijection.

It can also be seen that this mapping is efficiently reversible, because for any point $P = (x, y) \in E$ the numerical value of the original message (password) is either x or $-x$.

2.2.1 Shamir's Secret Sharing

In Chapter 6, we also apply secret sharing where we using a (k, n) threshold scheme. A secret S can be divided into n shares in a way that $k \leq n$ will be the threshold number of the shares and we must be able to compute S. Thus $k - 1$ or fewer shares leave S completely undetermined. We apply secret sharing for the IoT devices to construct the password.

Shamir's Secret Sharing threshold scheme is based on polynomial interpolation. We need at least k points to define a polynomial of $k - 1$ degree. We set a polynomial of $k - 1$ degree to divide a secret S into shares: $f(x) = a_0 + a_1x + ... + a_{k-1}x^{k-1}$, where $a_i \in \mathbb{F}_q$ are chosen randomly for $i = 1, \ldots, k - 1$. Secret $S = f(0) = a_0$ and the shares will be: $s_1 = f(1), \ldots, s_n = f(n)$.

To find the secret S, Lagrange polynomials are applied. There is an optimized approach, where

$$S = f(0) = \sum_{j=0}^{k-1} f(x_j) \prod_{\substack{m=0 \\ m \neq j}}^{k-1} \frac{x_m}{x_m - x_j}.$$

2.2.2 Description of Cryptographic Primitives

In this part we define the cryptographic primitives, which are used in our solutions:

2.2.7 Definition. A message authentication code (or MAC) is a tuple of polynomial-time algorithms (Key_M,MAC,Ver) such that:

1. The key-generation algorithm Key_M takes as input the security parameter 1^κ and outputs a key K with $|K| \geq \kappa$. Key_M is probabilistic.

2. The tag-generation algorithm MAC takes as input a key K and a message $m \in \{0,1\}^*$, and outputs a tag t. We write this as $t := MAC_K(m)$. We assume that MAC is deterministic.

3. The verification algorithm Ver takes as input a key K, a message m, and a tag t. It outputs a bit b, with $b = 1$ meaning valid and $b = 0$ meaning invalid. We assume without loss of generality that Ver is deterministic, and so write this as $b := Ver_K(m,t)$.

Consider the experiment for a message authentication code (Key_M, MAC, Ver), an adversary $\mathcal{A}$, and security parameter κ, as follows. The message authentication experiment $Exp^{eforge}_{MAC}(\mathcal{A})$ is the following:

1. A random key K is generated by running $Key_M(1^\kappa)$.

2. The adversary $\mathcal{A}$ is given input 1^κ and oracle access to $MAC_K(.)$. The adversary eventually outputs a pair (m,t).

3. The output of the experiment is defined to be 1 if and only if $Ver_K(m,t) = 1$ and m was never asked from the oracle $MAC_K(.)$ before.

2.2.8 Definition. A message authentication code (Key_M, MAC, Ver) is *existentially unforgeable under an adaptive chosen-message attack*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, $Pr[Exp^{eforge}_{MAC}(\mathcal{A}) = 1]$ is negligible.

In Chapter 4.1, servers should be able to use a secret channel to exchange data in an encrypted manner.

2.2.9 Definition. A symmetric encryption scheme is a tuple of probabilistic polynomial-time algorithms (Key_E, Enc, Dec).

1. The key-generation algorithm Key_E takes as input the security parameter 1^κ and outputs a random key $K \in \{0,1\}^\kappa$.

2. Enc is an encryption algorithm that takes inputs key K and plaintext $m \in \{0,1\}^*$, and outputs a ciphertext $c = Enc_K(m)$.

3. Dec is a deterministic decryption algorithm that takes inputs key K and ciphertext c, and outputs the plaintext $m = Dec_K(m)$.

To define secure encryption, let $\mathcal{A}$ be an adversary and consider the experiment $Exp_{Enc}^{ind-cpa}(A)$ as follows.

1. A random key K is generated by running $Key_E(1^\kappa)$.

2. The adversary $\mathcal{A}$ is given input 1^κ and oracle access to $Enc_K(.)$, and $\mathcal{A}$ outputs $m_0, m_1 \in \{0,1\}^*$ with $|m_0| = |m_1|$.

3. A random bit b is chosen and a ciphertext $Enc_K(m_b)$ is computed and given to $\mathcal{A}$.

4. $\mathcal{A}$ continues to have oracle access to $Enc_K(.)$, and outputs a bit b'.

5. We define $\mathcal{A}$'s advantage to be $Adv_{Enc}^{ind-cpa}(A) = |Pr[b' = b] - 1/2|$.

2.2.10 Definition. We say that a symmetric encryption scheme is *indistinguishable under chosen plaintext attack* if $Adv_{Enc}^{ind-cpa}(A)$ is negligible in κ for any Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$.

2.3 Communication Channels

In the case of cryptographic protocols, the parties involved exchange information through various types of communication channels.

Public channel. Public channels transmit all information without applying cryptographic algorithms. Attackers are able to tap the message, and the identity of the sender can be traced back.

Authenticated channel. An authenticated channel is resistant to unauthorized tampering but it does not provide condentiality against the eavesdropper adversary.

Secure authenticated channel. A secure authenticated channel is resistant to unauthorized tampering and provides confidentiality as well. In order to realize a secure channel between two parties, first the participants run a key-exchange protocol to obtain a session key. The sender encrypts the message and concatenates it with a tag computed by applying a message authentication function to the ciphertext. Encryption and authentication are performed via keys derived from the session key. Verification and decryption are executed analogously.

Chapter 3

Background of the Security Analysis Techniques

This section covers the various security techniques we use to analyze our proposed protocols. We present two automated tools and give the details of the concept of provable security.

A protocol verification aims to prove that the protocol is secure. However, the complexity of security protocols makes it challenging to analyse them. Informal arguments about the security of protocols are not reliable. There are many formal methods for analysing the security protocols, and this topic remains very active in the research community. Formal methods for analysing security protocols fall into two main categories.

Model-checking methods consider a finite number of possible protocol behaviours, allowing you to check whether the protocol meets the specified security conditions. These methods are sound but not complete, i.e. they are generally more suitable for finding attacks against protocols than for proving their security.

Theorem-proving methods consider all possible protocol behaviours and check whether they meet a set of security conditions. These methods are generally more suitable for proving the security protocols.

Lowe ([83]) has developed an effective method which has been used to find a previously unknown attack on the Needham-Schroeder public key protocol ([92]). This method is applied for verifying security protocols using Failures Divergences Refinement Checker (FDR), a model checker for the process algebra (Communicating Sequential Processes - CSP) ([103]). A comprehensive introduction to the method can be found in [104], including the background of CSP. In the protocol, each participant is modelled as a CSP process representing the protocol steps performed by the participant, and communication is modelled by the notion of channels.

The US Naval Research Laboratory (NRL) developed a special-purpose software tool known as the Analyzer ([88]), which is a Prolog program for cryptographic protocol analysis. The Analyzer is hybrid, possessing features of both a model checker and a theorem prover. Searching begins from an insecure state, examining if that state can be reached from the initial state and if so, an explicit attack has been found. Lemmas may be proven to demonstrate that infinite classes of states are unreachable, which can prove that all paths to the insecure state start in unreachable states.

The Burrows-Abadi-Needham (BAN) logic ([30]) is one of the most popular protocol logics, which is an early formal method for analyzing authentication and key establishment protocols. Proofs constructed in the BAN logic tend to be short and easily obtained. For analysing a protocol, one must transform the actual or concrete protocol into an idealised protocol. We distinguish three types of primitive objects in the syntax of the BAN logic: principals, keys and nonces, and we express the protocol messages as a logic formula. While several researchers have proposed extensions or improvements over the BAN logic, others criticised it because it may not accurately reflect the protocols ([93, 91]).

3.1 AVISPA

In this section the necessary informations are provided which are required to understand the AVISPA Automated Validation of Internet Security Protocols and Applications) tool. In AVISPA, we can define protocols and their security properties. It applies the High-Level Protocol Specification Language (HLPSL) which is a modular, role-based language. AVISPA contains two types of roles where basic roles represent each participant role and composition roles act for scenarios of basic roles. The roles are independent of the other roles. In the initial step, roles get information by parameters and communicate with each other's used by the channels. With the use of HLPSL correct protocol behavior described in the specification is achieved.

For implementing distinct state-of-the-art automated analysis techniques, we also integrate various back-ends. AVISPA contains four different formal verification approaches (i.e., On-the-fly Model-Checker,

Constraint-Logic-based Attack Searcher, SAT based Model-Checker and Tree Automata-based Protocol Analyser) that can formally validate security properties of a protocol.

The On-the-fly Model-Checker (OFMC) ([10, 11, 12, 43]) executes protocol forgery and bounded session verification as well, where OFMC considers both typed and untyped protocol models. OFMC's effectiveness is due to the integration of a number of symbolic, constraint-based techniques, which are correct and complete, in the sense that no attacks are lost nor new ones are introduced by them. These techniques are the lazy intruder technique or the constraint differentiation technique. OFMC also implements a number of efficient search heuristics. It also provides support for modeling an intruder who is capable of performing guessing attacks on weak passwords, and for the specification of algebraic properties of cryptographic operators. OFMC can be employed for protocol falsification and also for verification for a bounded numbers of sessions - without bounding the messages an intruder can generate.

The Constraint-Logic-based Attack Searcher (CL-AtSe) is different from the OFMC since it uses constraint solving to perform the protocol falsification and verification for bounded numbers of sessions ([34, 35, 36, 37, 38]). The protocol messages can be typed or untyped, and the pairing can be considered to be associative or not. It is significant that several properties of the XOR operator can be handled as well. CL-AtSe is built in a modular way and is thus open to extensions for handling algebraic properties of cryptographic operators. CL-AtSe performs several kinds of optimizations to reduce, and often eliminate redundancies or useless branches in the protocol's symbolic execution.

In AVISPA the SAT-based Model-Checker (SATMC) ([4, 5, 6, 7]) could be also applied which considers the typed protocol model and performs both protocol falsification and bounded session verification by reducing the input problem to a sequence of invocations and state-of-the-art SAT solvers. Finally, the TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) back-end ([22, 23]) performs unbounded protocol verification by approximating the intruder knowledge with the help of regular tree languages and rewriting. Genet and Klay introduced an extension of an approximation method based on tree automata ([51, 52]) for verifying security protocols.

Internally, the attack conditions are specified in terms of temporal logic, but useful and concise macros are provided for the two most frequently used security goals, authentication and secrecy. When as we give the goal predicates declaration which is explicitly on the right-hand sides of HLPSL transitions and are translated into corresponding IF facts, these are applied to specify secrecy and different forms of authentication. Among the goal facts *secrecy* declares which values should be kept secret. This goal declaration in the goal section describes that anytime the intruder learns a secret value, even if it is not explicitly given, then it should be considered an attack. The evaluation of secrecy remains in effect until the end of the protocol run. Since it has effect only after the events have been issued, secrecy events should be defined at the earliest possible time, when the secret term has been generated in the respective role(s).

The *witness and request predicates* are goal facts related to authentication. They are used to check whether a participant is right in believing that its intended peer is present in the current session, has reached a certain state, and agrees on a certain value, which typically is fresh. They always appear in pairs with identical third parameter. Further information about the AVISPA can be found in [3, 8].

3.2 ProVerif

In this section we present the ProVerif software package (actual version is 2.04). ProVerif is another tool in which we can formalize our cryptographic protocols and automatically analyse their security. The following cryptographic primitives are provided symmetric and asymmetric encryptions; digital signatures; hash functions; bit-commitments; and non-interactive zero-knowledge proofs. It is notable that the analysis of formalized protocol is evaluated in point of an unbounded number of sessions and an unbounded message space in ProVerif.

3.2.1 Applied π Calculus

Term and process grammar

In this section we briefly review the applied π calculus that is based

on the π calculus. Detailed description of this topic can be found in [1]. A signature $\sum$ is a set of function symbols, each with an arity. A function symbol f with arity 0 is a constant symbol. The set of terms is built from names, variables, and function symbols. Let us denote channel names by a, b, c, and names of any sort by m, n. Also, let x, y, z range over variables.

$L, M, N, T, U, V ::=$	**terms**
$a, b, c, \ldots, k, \ldots, m, n, \ldots, s$	name
x, y, z	variable
$f(M_1, ..., M_l)$	function application

The grammar for processes is the following:

$P, Q, R ::=$	**processes**
$\mathbf{0}$	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction (new)
$if\ M = N\ then\ P\ else\ Q$	conditional
$u(x).P$	message input
$\overline{u}\langle N\rangle.P$	message output

Replication of process $!P$ means infinite number of copies of P running in parallel. Name restriction process $\nu n.P$ creates a new, private name and behaves as P. Finally, process $u(x).P$ is ready to input from channel u, then to run P with the message replaced for the formal parameter x, and process $\overline{u}\langle N\rangle.P$ is ready to output N on channel u, then to run P. We extend processes with active substitution. We denote the substitution that replaces the variable x with the term M by $\{M/x\}$.

Operational semantics

Given a signature $\sum$ we equip it with an equational theory, *i.e.* an equivalence relation on terms that is closed under substitution of terms for variables. We write $E \vdash M = N$ for equality and $E \nvdash M = N$ for inequality in the theory associated with $\sum$. Operational semantics of the applied-pi calculus is defined in terms of structural equivalence ($\equiv$) and internal reduction ($\rightarrow$). Structural equivalence captures rearrangements of parallel compositions and restrictions, and the equational rewriting of the terms in a process. Internal reduction defines the semantics of process synchronizations and conditionals. Observational equivalence ($\approx$) cap-

tures the equivalence of processes with respect to their dynamic behavior.

3.2.2 ProVerif

ProVerif handles input files encoded in a variant of the applied π calculus which supports types. ProVerif takes the model of the protocol and the security properties that we want to prove as input. ProVerif supports cryptographic primitives, modeled by rewrite rules or by equations. By using constructors and destructors with rewrite rules cryptographic primtives can be represented (see next section). Constructors and destructors cannot model all cryptographic operations, therefore to deal with these limitations (e.g. modelling modular exponentiation) ProVerif uses equational theory. For example, the Diffie-Hellman key agreement can be modelled by using equations:

$$exp(exp(g, x), y) = exp(exp(g, y), x)$$

where $g : G$ is a constant and $exp(G, Z) : G$ is modular exponentiation. Symmetric encryption schemes can be modelled by equations as well. In this model decryption always succeeds. However, rewrite rules should be preferred when they are sufficient to give the security property, ProVerif handles them more efficiently.

It is important to mention that ProVerif does not support all equational theories. For some equations infinite number of rewrite rules are generated and ProVerif does not terminate. A typical example for this is the associativity, which prevents the modeling of primitives such as XOR or groups. Another one is the equation of $f(g(x)) = g(f(x))$. The solution for this problem is considering the extensions of ProVerif ([78]), or there are other protocol verifiers, e.g. Maude-NPA ([88], [45]) and Tamarin([107]), that support more equational theories, at the cost of a more computationally expensive verification or by requiring user intervention.

ProVerif is able to verify security properties, including secrecy, mutual authentication and observational equivalence properties. The adversary intercepts all messages, computes, and sends all messages it has, following the Dolev-Yao model. ProVerif translates the protocol into a set of Horn clauses, and the security properties into derivability queries on these

clauses and determines whether a fact is derivable from the clauses. Horn clauses are first order logic formulas of the form $F_1 \wedge \cdots \wedge F_n \implies F$, where $F_1, \ldots, F_n, F$ are facts. An algorithm based on resolution with free selection is applied. If the fact is not derivable, then the security property holds, otherwise there might be an attack. The derivation may give a false attack, because the Horn clause representation makes some abstractions. One can find more details in [21] and [20].

Cryptographic primitives

We give formalization for cryptographic primitives as follows:

- *Hash function, message authentication code*
  We represent a one-way hash function as a unary function symbol H with no rewrite rules. The absence of an inverse for H models the onewayness of H. Similarly we denote a one-way Mac function as a binary function symbol mac, where the first argument corresponds to the secret key of Mac.

  $fun\ H(bitstring) : bitstring.$
  $fun\ mac(bitstring, bitstring) : bitstring.$

- *Symmetric encryption*
  We take binary function symbols *encrypt* and *decrypt* for encryption and decryption, respectively, with the rewrite rule: $decrypt(encrypt(M, k), k) = M$. Here M represents the plaintext and k is the secret key.

  $fun\ encrypt(bitstring, bitstring) : bitstring.$
  $reduc\ forall\ M : bitstring, k : bitstring;$
  $decrypt(encrypt(M, k), k) = M.$

- *Asymmetric encryption*
  In the case of asymmetric encryption, we have to generate a keypair, a public and a secret key. We have an unary function symbol pk for generating the public key, where the secret key is the argument. Similarly to symmetric encryption, we represent asymmetric encryption and decryption with binary function symbols *aenc* and

$adec$ with the rewrite rule of $adec(aenc(M, pk(k)), k) = M$, where M denotes the plaintext and k is the secret key.

$type\ skey.$
$type\ pkey.$
$fun\ pk(skey) : pkey.$
$fun\ aenc(bitstring, pkey) : bitstring.$
$reduc\ forall\ m : bitstring, k : skey;$
$adec(aenc(M, pk(k)), k) = M.$

- *Digital signatures*
  In order to formalize digital signatures that also employ secret and public keys, we use function symbol spk for generating public keys, and binary function symbols $sign$, $checksign$. Digital signatures rely on a pair of signing keys of types $sskey$ (private signing key) and $spkey$ (public signing key). We interpret digital signatures with message recovery, meaning we have $checksign(sign(M, k), spk(k)) = M$, where M is the message and k is the secret key.

Security properties

$type\ sskey.$
$type\ spkey.$
$fun\ spk(sskey) : spkey.$
$fun\ sign(bitstring, sskey) : bitstring.$
$reduc\ forall\ M : bitstring, k : sskey;$
$checksign(sign(M, k), spk(k)) = M.$

ProVerif is capable of analyzing properties reachability, correspondence assertions and observational equivalences. These capabilities are demonstrated to permit the analysis of secrecy and authentication properties. Furthermore, in the analysis other properties can be considered like privacy, traceability, and verifiability. When a property cannot be proved, ProVerif can reconstruct the attack and try to reconstruct an execution trace that falsifies the desired property. We mention that ProVerif is sound, but not complete. If ProVerif results that a property is satisfied, the model guarantees the property, but ProVerif may not be able to prove a property that holds.

For security evaluations ProVerif uses queries that might be a fact or a correspondence. In the case of *reachability*, the query is a fact, we test whether the fact holds. Especially, we query whether a term m is secret for the attacker: *query attacker* : m.

A *correspondence* is a form of $F \Longrightarrow H$, which means if F holds, then H also holds. We define events in the model as important stages and we test whether if event a has been executed, then event b has been previously executed. The *query ev* : $a(x, y) \Longrightarrow ev : b(y, z)$. means that for all x, y, for each occurrence of $a(x, y)$, there is a previous occurrence of $b(y, z)$ for some z. For proving one-to-one relationship we apply injective correspondences. The *query evinj* : $a(x, y) \Longrightarrow evinj : b(y, z)$. means that for each occurrence of the event $a(x, y)$, there is a *distinct* earlier occurrence of the event $b(y, z)$ for some z.

The notion of indistinguishability is called *observational equivalence* in the formal model, denoted by $P \approx Q$. Two processes are observationally equivalent if an active adversary cannot distinguish them. Further information can be found in [21].

3.3 The Concept of Provable Security

In Chapter 4.2 and Chapter 5, we present two provably secure protocols. The first protocol provides secure password registration, while the second ensures mutual key authentication and mutual key confirmation. Our security models are based on the indistinguishability-based model (see [14], [15]) proposed by M. Bellare and P. Rogaway in 1993.

In Chapter 5 we also apply the generic bilinear group (GBG) model, which was introduced by D. Boneh and et. al. in [25]. In the GBG model, two random encodings Ω_0, Ω_1 of the additive group $\mathbb{Z}_q^+$ are considered. Let $\mathbb{Z}^+$ denote the positive integers, q is a large prime and injective maps $\Omega_0, \Omega_1 : \mathbb{Z}_q^+ \to \Theta$ are employed, where Θ is a bitstring set and $|\Theta| = q$. We assume that $\mathbb{G} = \{\Omega_0(x) | x \in \mathbb{Z}_q^+\}$ and $\mathbb{G}_\mathbb{T} = \{\Omega_1(x) | x \in \mathbb{Z}_q^+\}$. In the GBG model, an oracle executes the group operation and takes two encodings of group elements as input, then outputs an encoding of a third element. The group is allowed for a pairing operation, which is an additional oracle. We give oracles $\mathbb{Q}_\mathbb{G}$, $\mathbb{Q}_{\mathbb{G}_\mathbb{T}}$ to execute the group operation on $\mathbb{G}, \mathbb{G}_\mathbb{T}$ and an oracle $\mathbb{Q}_\mathbb{P}$ to calculate a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathbb{T}$. For any operations

on groups, the adversary must issue the associated group queries to the polynomial time adversary $\mathcal{F}$ to get the results.

For any $a, b \in \mathbb{Z}_q$, queries $\mathbb{Q}_{\mathbb{G}}$, $\mathbb{Q}_{\mathbb{G}_\mathbb{T}}$ and $\mathbb{Q}_{\mathbb{P}}$ possess the following properties:

- $\mathbb{Q}_{\mathbb{G}}(\Omega_0(a), \Omega_0(b)) \rightarrow \Omega_0(a + b \ mod \ q)$
- $\mathbb{Q}_{\mathbb{G}_\mathbb{T}}(\Omega_1(a), \Omega_1(b)) \rightarrow \Omega_1(a + b \ mod \ q)$
- $\mathbb{Q}_{\mathbb{P}}(\Omega_0(a), \Omega_0(b)) \rightarrow \Omega_1(ab \ mod \ q)$

We detail the security model. Let 1^κ denote the string consisting of κ consecutive 1 bits, where $\kappa \in \mathbb{N}$ is a security parameter. *ID* is the union of the finite, disjoint, nonempty sets of $Client = \{1, 2, \ldots, T_1(\kappa)\}$ and $Server = \{1, 2, \ldots, T_2(\kappa)\}$, where $T_i(\kappa)$, $i \in 1, 2$ is a polynomial bound on the number of participants in κ for some polynomial function T_i. Each participant is modelled by an oracle $\prod_{I,J}^{l}$, which simulates a participant I executing a protocol session in the belief that it is communicating with another participant J for the lth time, where $l \in \{1, \ldots, T_3(\kappa)\}$ for some polynomial function T_3. Oracles keep transcripts, which contain all messages they have sent and received and the queries they have answered. The oracles are available to the adversaries, and we can model the various kinds of attacks via the queries to these oracles. In the concept of the applied proof technique, we assume that a problem is hard to solve and prove that the solution is secure under this assumption. The proof that a given cryptographic protocol is secure as long as an underlying problem is hard proceeds by presenting an explicit polynomial reduction. This reduction shows how to convert an adversary $\mathcal{A}$ that succeeds in "breaking" the construction with non-negligible probability into an efficient algorithm $\mathcal{A}'$ that succeeds in solving the problem we assumed to be hard. It contradicts the initial assumption if an efficient algorithm solves the hard problem with non-negligible probability. We state that our protocol is computationally secure since no efficient adversary $\mathcal{A}$ can succeed in breaking protocol with probability that is not negligible.

Chapter 4

Distributed Cloud Authentication Protocols

In this chapter, we propose two distributed authentication protocols for cloud services. Our solutions apply multiple servers for user authentication. In both solutions, we put great emphasis on the handle of corrupt participants. In the first solution, we present a two-factor authentication protocol based on Merkle tree, however, it does not provide the scalability. In our second protocol we focus on robustness and scalability, where we apply secret sharing technique. First, let's check the distributed system and the cloud computing definitions.

Distributed systems play an essential role in our daily lives, making them more efficient and convenient. Several definitions of distributed systems have been formalized in the literature, and we use the definition specified in [111]:

4.0.1 Definition. A distributed system is a collection of independent computers that appears to its users as a single coherent system.

A distributed system consists of multiple, autonomous computers that communicate through a network even while completing their task. The goal of a distributed system is to solve a single problem by breaking it down into several tasks where each task is computed by a computer of the system. Distributed systems can run in a cloud infrastructure as well. Secure user authentication is an important issue of cloud services. If it is breached, confidentiality and integrity of the data or services may be compromised. In the case of Software as a Service model, the cloud service provider takes responsibility for securing all the data from unauthorized access.

4.0.2 Definition. Cloud computing is a model for enabling ubiquitous,

convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort and service provider interaction.

4.1 Cloud Authentication Protocol Using a Merkle Tree

In this section, a two-factor authentication scheme is proposed for cloud environments. Besides a static password, a one-time password is suggested by applying a Merkle tree for identity verification. A security analysis is carried out in the case of outsider adversaries, and the protocol is formalized in applied pi calculus. In the ProVerif framework, we show that our authentication protocol meets the standard security requirements of a key exchange protocol.

The results of this section are contained in our papers [57], [58]. This paper is a joint work with Andrea Huszti.

4.1.1 Our Contribution and Literature

In scientific papers, one-factor and two-factor authentication solutions are being used in general. In 2000, Hwang and Li suggested a new remote user authentication scheme ([61]), which is based on smart cards. Instead of a password verification table they applied the ElGamal encryption scheme, but an impersonation attack was found. In 2002, Chien, Jan and Tsien proposed a password-based authentication ([39]), which does not use a verification table and the passwords were chosen freely. Ku and Chen proved that Chien *et al*'s scheme was vulnerable to several attacks ([76]). In [46] an asymmetric authentication system is proposed, where a USB drive is used as a hardware key, and it is combined with asymmetric cryptography.

Two-factor authentication provides higher security level than one-factor authentication, because it provides an extra security factor for the login process. In 2011, Choudhury *et al* showed a two-factor authentication protocol ([40]) for cloud computing where one of the factors is the smart card and the other one is the password. They applied an out of band channel. Later, Chen and Jiang detected an impersonation attack

in Choudhury *et al*'s scheme and proposed a new authentication framework which did not use the out of band channel ([33]).

In practice, OpenStack is one of the most popular cloud computing software. The OpenStack Identity service ([94]) supports multiple methods of authentication, including user name and password, Lightweight Directory Access Protocol (LDAP), and external authentication methods (e.g. Kerberos). LDAP provides simple and SASL ([102]) authentication for users. In general, either the clear-text password is sent through TLS or a password-based challenge-response mechanism is applied.

Another popular application is called Kerberos ([101]), which was developed at MIT to provide secure authentication for UNIX networks. User authentication is based on secret keys stored on the Kerberos Key Distribution Center (KDC) server. Kerberos uses temporary certificates called tickets, which contain the credentials that identify the user to the servers on the network. There are concerns about Kerberos, including Golden Ticket Attack ([77, 110]), presented on the 2015 RSA Conference slide deck. Let's suppose an adversary has a domain or local admin access on an Active Directory domain. In that case, he might be able to get the secret key for the KDC server, a golden ticket, and this way, he can manipulate Kerberos tickets to get unauthorized access. This is a vulnerability due to the centralized structure of the Kerberos protocol. It shows the weakness of the centralized KDC server against attacks when an adversary becomes a full system administrator.

Instead of a centralized structure of authentication, a distributed authentication system is recommended. The advantage of a shared system is that external attackers need to attack multiple servers simultaneously, which increasing the attack cost. Companies that store passwords for a large number of enterprises are primary targets for hackers. The recent hack of OneLogin, an IAM (Identity and Access Management) provider, proves that the credentials may not be as safe as we are led to believe. Our design goal is to increase security level against these attacks by providing shared responsibility.

First, we review our basic protocol [57] (Figures 4.1 - 4.3). We differentiate three participants in the scheme. *Users* (U) ask for services from the cloud service provider. The cloud service provider consists of several *cloud servers* (C_i) and an *authentication server* (AS). A cloud server

which is chosen randomly proceeds the steps of the user authentication. The authentication server manages the cloud servers. Each cloud server C_i possesses shares T_{i_1} and T_{i_2}, where T_{i_1} is used for user authentication, T_{i_2} is necessary for calculating the MAC key. The node values Y_k are stored separately by a dedicated cloud server called authentication server. The correctness of the root node value Y_r verifies the correctness of the whole one-time password.

One cloud server is chosen randomly to proceed the authentication with the user. Let C_i denote this randomly chosen server. The user provides the one-time password share T_{i_1} and the two-child-node values of the root node: Y_{d_1}, Y_{d_2}. Since C_i stores the values T_{i_1}, T_{i_2} and Y_r, the correctness of the share T_{i_1} and Y_{d_1}, Y_{d_2} can be verified via $Y_r = H(Y_{d_1}||Y_{d_2})$.

After each login, the shares T_{i_1}, T_{i_2} are reset to get the new one-time password. The efficient root path modification is applied with the help of the existing siblings, therefore a new root node value is also calculated. Please notice, during the authentication process only T_{i_1} is revealed, the remaining part of the one-time password is not disclosed, accordingly only one of the Y_{d_j}, $j \in \{1, 2\}$ values is recalculated. If $i \leq 2^{n-2}$ (*i.e.* T_{i_1} is in the subtree corresponding to Y_{d_1}), Y_{d_1} is changed, if $i > 2^{n-2}$ (*i.e.* T_{i_1} is in the subtree corresponding to Y_{d_2}), Y_{d_2} is modified.

By employing 2^{n-1} cloud servers we get a Merkle tree with height n. Altogether $2^n + 2^{n+1} - 1$ hash calculations are necessary to build the tree and in n steps the Merkle tree update is completed. Computing the hash values is efficient if the number of hashes is less than 2^{40}. This limit means application of millions of servers in the authentication process, which is not necessary.

In our solution in [57] several cloud servers are applied, and the one-time password is stored distributed. In contrast to that, in [33] and in [40] and in case of the practical solution above only *one* cloud server verifies users' authenticity. In our protocol an attack can be successful only if the adversary possesses all password shares known by the servers.

We propose a two-factor authentication scheme in [57], which is based on a *static password* and a *one-time password* generated by a smart card. A Merkle tree is applied, where a leaf is the hash of a password share, and the root element is verified in order to confirm the correctness of the

whole password. In the registration phase the secret keys are exchanged generating large amount of one-time passwords between the user and the cloud servers. Comparing the efficiency of our authentication phase to the work of [33, 40], it is showed that the scheme is more efficient since only hash calculations are performed. A message authentication key (MAC) exchange is also provided for guaranteeing data origin integrity for the latter interactive communication.

In [58] the extended scheme is given and also provided a security analysis in applied-pi calculus.

In the registration phase an index value is added to the static salted password sent and stored hashed, resulting different hash values for different servers. The protocol is formalized in applied-pi calculus and the Proverif framework is used. With Proverif events and injective queries, we have proved that the protocol provides secure key exchange, *i.e.* provides secure mutual authentication, key secrecy, key freshness and proof of knowledge of the new key on both server and user sides during the authentication phase. We consider outsider adversaries in the Dolev-Yao model.

4.1.2 The Proposed Scheme

Users (U), 2^{n-1} cloud servers (C_i, where $i \in \{1, \ldots, 2^{n-1}\}$) and an authentication server (AS) contribute to the scheme. We assume that each cloud server possesses an asymmetric key pair: $SK_{C_i} = (y_i, z_i)$, $PK_{C_i} = (g^{y_i}, g^{z_i})$, where g is a generator element of a cyclic group, and we randomly choose $y_i, z_i \in \mathbb{Z}_q$, where q is a prime. Either the multiplicative group of a finite field or the elliptic curve additive group can be chosen where the discrete logarithm or the elliptic curve discrete logarithm problem, respectively, is hard. The three phases: registration, authentication and synchronization are as follows.

Registration

Registration is the first phase of the protocol (Figure 4.1). During the registration all keys and the system parameters are generated. We assume that each user U possesses a smart card that is protected by a PIN code. Besides the static password, secret Diffie-Hellman keys are also exchanged

between the user and the cloud servers. The Merkle tree is built up on the user and server sides as well. It is important to mention that each cloud server receives only a secret share, hence neither of the servers knows the whole secret.

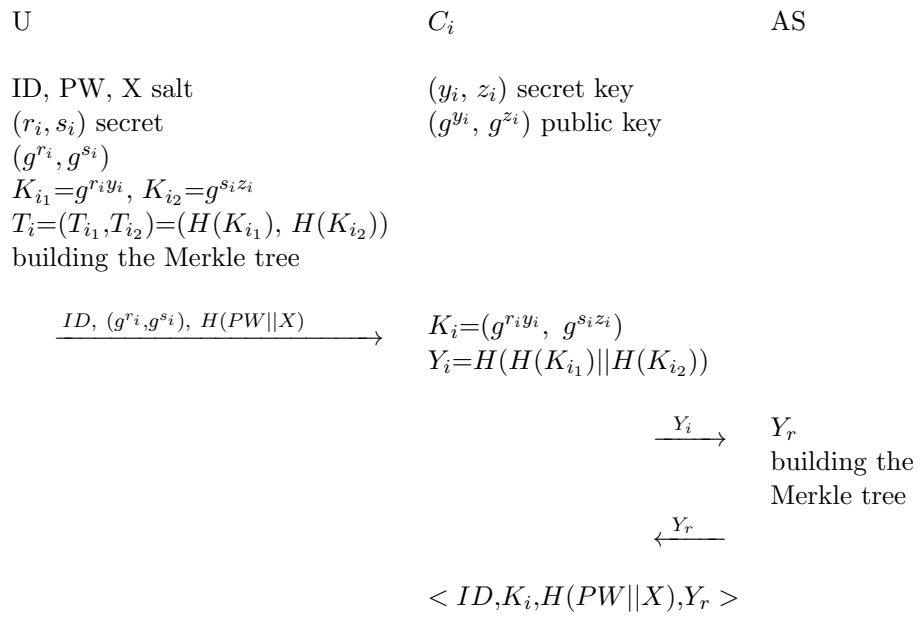


Figure 4.1. Registration

Let g denote a generator element in a cyclic group G. U chooses a username (ID), a static password (PW), two secret random values (r_i, s_i) for each cloud server and calculates (g^{r_i}, g^{s_i}). U computes the Diffie-Hellman keys $K_i = (K_{i_1} = g^{r_i y_i}, K_{i_2} = g^{s_i z_i})$ that are *stored securely* on the smart card and calculates $T_i = (T_{i_1} = H(K_{i_1}), T_{i_2} = H(K_{i_2}))$. In the next step U creates the Merkle tree from the leaves T_i and stores it on the smart card.

We assume that a secret random value X, which is different for different cards, is also *stored securely* on this card. This value X is considered to be a salt for the static password, which protects against dictionary and rainbow table attacks. U sends $(ID, (g^{r_i}, g^{s_i}), H(PW||X||i))$ through an *authenticated channel* to C_i, where $i \in \{1, \ldots, 2^{n-1}\}$.

C_i also computes the Diffie-Hellman key $K_i = (K_{i_1} = g^{r_i y_i}, K_{i_2} = g^{s_i z_i})$ and the value $Y_i = H(H(K_{i_1})||H(K_{i_2}))$. C_i sends ID, Y_i to AS through a secret, authenticated channel. The secret, authenticated channel is necessary, since Y_i should be kept secret, otherwise the verification values Y_{d_1} and Y_{d_2} can be calculated easily. AS creates the Merkle tree from the values Y_i and computes Y_r. AS sends Y_r to every cloud server through authenticated channels.

Cloud servers store the username (ID), the seeds of the one-time password share (K_i), the root node of the Merkle tree (Y_r) and the hash of the salted static password with the server index ($H(PW||X||i)$) for each user.

Authentication

In the second phase (Figure 4.2) the mutual authentication between the user and a randomly chosen cloud server, furthermore a MAC key exchange are processed. The cloud server verifies the correctness of the static password, the share of and also the whole one-time password.

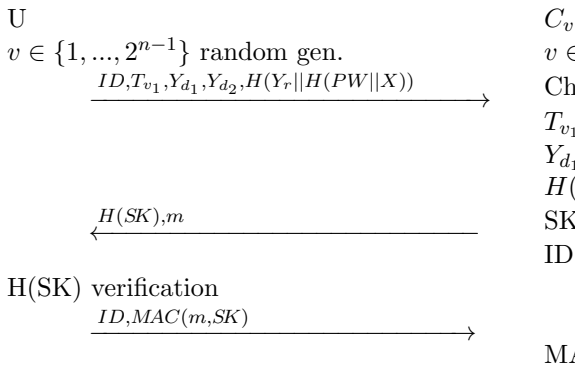


Figure 4.2. Authentication

As a first step, a random public value $v \in \{1, \ldots, 2^{n-1}\}$ based on the actual time is generated on both sides. This value is known by the user and each cloud server. The value v denotes the index of the cloud server that executes the authentication.

U sends his username ID, the corresponding one-time password share T_{v_1} and the two child nodes of the root node Y_{d_1}, Y_{d_2} to the chosen cloud

server C_v. U also computes and sends the value $H(Y_r||H(PW||X||v))$, where Y_r is the actual root node value and $H(PW||X||v)$ is the hash of the salted static password concatenated with the chosen server index. Since Y_r is different for each authentication run, a different hash value $H(Y_r||H(PW||X||v)$ is sent. Calculating $H(PW||X||v)$ from $H(Y_r||H(PW||X||v))$ is hard, therefore $H(PW||X||v)$ is kept secret. The value T_{v_1} and either Y_{d_1}, or Y_{d_2} are recalculated after each login, hence they are different for different authentication run.

C_v authenticates the user by verifying the correctness of the one-time password share T_{v_1}, the validity of the whole one-time password via Y_{d_1}, Y_{d_2}, by comparing the stored Y_r to $H(Y_{d_1}||Y_{d_2})$. C_v also verifies the static password by checking the correctness of $H(Y_r||H(PW||X||v))$.

The *MAC* secret key $SK = H(Y_r||T_{v_2})$ is computed by C_v if all input values are valid. After each authentication phase both Y_r and T_{v_2} are modified, thus *SK* is different for each login. The server generates a random message m and $(H(SK), m)$ is sent back to U. C_v stores *ID*, m and *SK* for verification purposes.

U also computes $SK = H(Y_r||T_{v_2})$ and checks the correctness of the received $H(SK)$. If the hash value is correct, the mutual authentication of U and C_v is completed and U is assured that C_v knows the secret MAC key SK.

In the following step U calculates $MAC(m, SK)$ and sends *ID*, $MAC(m, SK)$ to C_v.

C_v is also informed that U knows SK by verifying the correctness of the MAC value received with the help of the previously stored *SK* and m.

Since during the mutual authentication and MAC key exchange only hash values are calculated, good efficiency results have been achieved.

Synchronization

In the synchronization phase (Figure 4.3) the Merkle tree is updated by modifying the secret shares of C_v, on both the user and the server sides. C_v sends the new node Y_v' to the authentication server. *AS* calculates the new root node Y_r', and sends the new root value to all servers.

U calculates $K_v' = (K_{v_1} * g, K_{v_2} * g)$, where g is the generator element. Therefore, the number of keys K_v that can be generated is $|G|$, where $|G|$ denotes the size of G. By calculating the new leaves

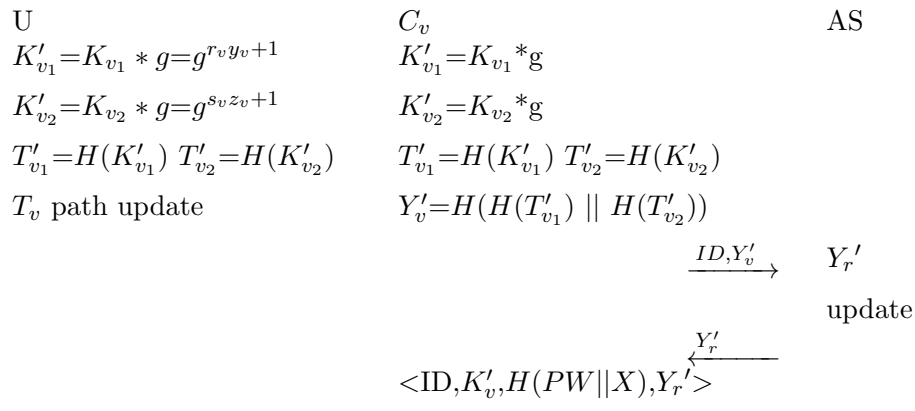


Figure 4.3. Synchronization

$T'_v = (H(K'_{v_1}), H(K'_{v_2}))$, the user U computes the new internal nodes of the root path.

C_v also calculates K'_v and T'_v in the same way and computes $Y'_v = H(T'_{v_1}||T'_{v_2})$, then sends ID, Y'_v to AS through a secret, authenticated channel. The secret, authenticated channel is essential since Y'_v should be kept secret, otherwise Y'_{d_1} or Y'_{d_2} can be calculated.

AS computes the new internal nodes of the root path and sends ID, Y'_r to all cloud servers through authenticated channels.

During the phase of synchronization due to the Merkle-tree structure the new one-time password is set efficiently, the new root node value is calculated in n steps.

The most challenging issue for the user and the cloud servers is to be correctly synchronized. The cloud servers and also the user should store the same root value. It is essential to update secret values right after a successful authentication on both sides. If a server is unavailable or the user loses his/her smartcard, a new registration is needed.

4.1.3 Security Analysis

In this section we present the security analysis of the protocol. We define the security requirements and the adversarial model and provide an analysis in applied pi calculus with the help of the Proverif tool.

Security Requirements

We analyse the protocol as a key exchange scheme, hence the typical security requirements for mutual entity authentication schemes and also the key related requirements are considered. We prove the following four properties:

1. Authentication of both parties
   (a) Authentication of users: Adversaries should not be able to impersonate a legal user and achieve illegal access to the user data.
   (b) Authentication of the server: Adversaries should not be able to impersonate a legal cloud server.
2. Secrecy of the MAC key: During the key exchange the newly generated key is a confidential datum and an adversary should not have any information about the new key.
3. Key freshness: During a protocol run a new, randomly chosen key should be exchanged so that a protocol execution could not be successfully finished with an old, already used key exchanged.
4. Both parties should verify that the other party knows and is able to use the new MAC key.

Adversarial Model

The goals of the adversary are to successfully exchange a key with the server or to get an illegal access to the cloud services. In our security analysis we consider only outsider attacks.

An *outsider attack* means that an adversary does not collude with any of the cloud servers, the authentication server or a user. It means the legal participants of the protocol do not reveal secret information to the adversary. We apply the Dolev-Yao model ([42]) for our analysis. In the Dolev- Yao model an adversary has the following properties:

1. The adversary has complete control over the entire network.

2. He acts as a legitimate participant, can intercept and compose any message and is limited by the constraints of the cryptographic methods used.

3. An adversary can initiate the protocol with any party, and can be a receiver to any party.

Mutual Authentication and Key Secrecy

Let us examine the requirements of user and server authentication and key secrecy at first. For the security analysis of mutual authentication and key secrecy concerning outsider attacks we use the ProVerif ([21]) software package.

Formal Model

We formalize the protocol, as follows. We differentiate three processes. In the main process (Figure 4.4) identification numbers for the participants, secret and public keys are generated. In the appendix (Chapter 7), there are two sub-processes representing the protocols for the user (Figure 7.1) and the server (Figure 7.2). An unbounded number of sub-processes running in parallel are considered. We model the interactions between the user and the server chosen randomly, we omit formalizing the communication among the authentication server and the cloud servers.

During the registration phase both the user and the server calculate the Merkle tree from the leaves that are randomly generated by the user and exchanged securely. We have set an authenticated, secure channel by applying an asymmetric encryption and a digital signature.

We have formalized the authentication phase in a way that the user authenticates himself to the same cloud server twice. After the first authentication, the Merkle-tree is updated on server and user sides, and the second authentication is performed. All the additional authentications happen in the same way.

Security Properties

In Section 3.2 the Proverif properties are presented and for security analysis we concentrate on the authentication phase. To follow interactions

```
(*Main process*)
process
new y1: exponent;
new y2: exponent;
new y3: exponent;
new y4: exponent;

new S_synch: exponent;
new id: bitstring;
new idS: bitstring;

let S1 =exp(g, y1) in
let S2 =exp(g, y2) in
let S3 =exp(g, y3) in
let S4 =exp(g, y4) in

out(c,(S1,S2,S3,S4)); (*public keys*)
new sskU:sskey;
new eskS:skey;
let spkU=spk(sskU) in out(c,spkU);
let epkS=pk(eskS) in out(c,epkS);
 ((!User(id,idS,S1,S2,S3,S4,sskU,spkU,epkS,S_synch)) |
(!Server(id,idS,y1,y2,y3,y4,eskS,epkS,spkU,S_synch)))
```

Figure 4.4. Main process

between the participants during the authentication phase, please check Figure 4.5.

In order to run the security queries, eight events are defined. Events $User\_auth\_start$ and $Server\_auth\_start$ show the end of the registration phase for the user and the server, respectively. Events $User\_auth\_end$ and $Server\_auth\_end$ are executed at the end of the first authentication. The other four events: $User\_auth2\_start$, $Server\_auth2\_start$, $User\_auth2\_end$, $Server\_auth2\_end$ refer to the beginning and the end of the second authentication on user and server sides.

We apply injective correspondences for the security analysis of the user and server authentication. We prove user authentication in a way that we test whether for each occurrence of the event $User\_auth\_end$ there is a distinct earlier occurrence of the event $User\_auth\_start$.
query sk:bitstring; inj-event(User_auth_end(sk))==>
inj-event (User_auth_start(sk)).

In a similar way we show the authentication of the server. We check whether for each occurrence of $Server\_auth\_end$ there is a distinct occurrence of $Server\_auth\_start$.
query sk:bitstring; inj-event(Server_auth_end(sk)) ==>
inj-event(Server_auth_start(sk)).

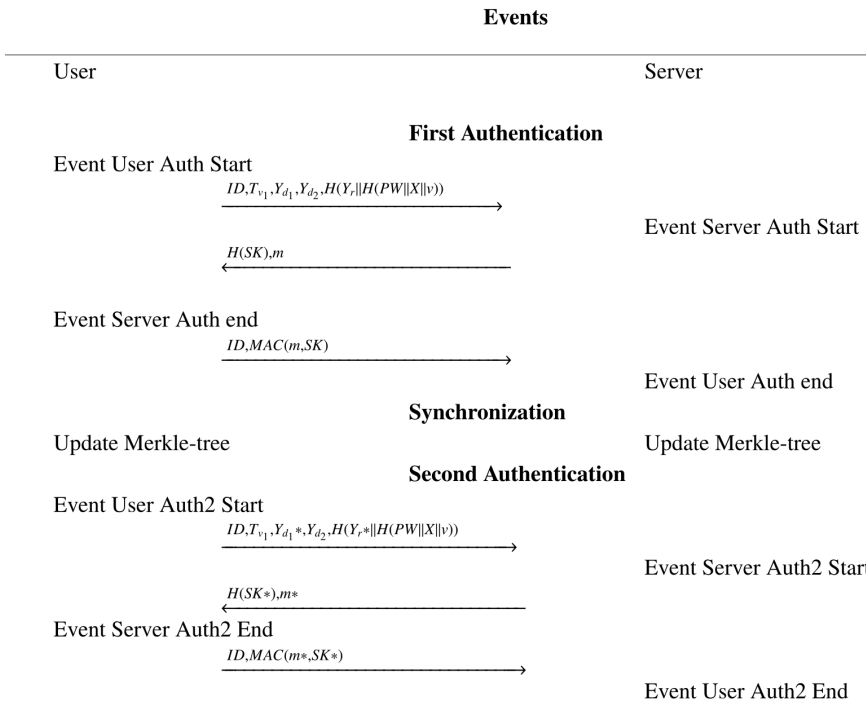


Figure 4.5. Events

By running the following queries we test whether the adversary is able to gain enough information to successfully proceed a man-in-the-middle attack by eavesdropping all messages sent during the first authentication.

query sk:bitstring; inj-event(User_auth2_end(sk))==>
inj-event(User_auth2_start(sk)).
query sk:bitstring; inj-event(Server_auth2_end(sk)) ==>
inj-event(Server_auth2_start(sk)).

Secrecy of the MAC key is also evaluated with the query *query attacker(SK).* by testing whether the key SK is reachable by the adversary.

All the queries above return with the value true, therefore user and server authentications and key secrecy hold in our model.

Key Freshness and Confirmations

Assuming a successful mutual authentication, *key freshness* holds. The MAC key SK, which is modified after each authentication execution during the synchronization phase, is the hash of the concatenation of Y_r and T_{v_2}.

The last requirement, *i.e.* both parties confirm that the other party knows the new MAC key, also holds. During the authentication phase the user verifies the correctness of $H(SK)$ after receiving message $(H(SK), m)$. If $H(SK)$ is valid, U calculates the MAC value for m applying SK, $MAC(m, SK)$. The server checks whether the MAC value is correct. If it is valid, the server confirms that SK is known by the user. Applying challenge and response technology, the protocol finishes successfully only if the parties receive correct replies.

4.2 Scalable Distributed Authentication for Cloud Services

In this section, we present a mutual authentication protocol with key agreement, where the identity verification is carried out by multiple servers applying secret splitting technology on server side. The protocol results in a session key, which provides the confidentiality of the subsequent messages between the participants. In our solution, we also achieve robustness and scalability. To show that the proposed protocol is provably secure, we apply the threshold hybrid corruption model. We assume that among the randomly chosen k servers, there is always at least one uncorrupted server and the authentication server reveals only the long-lived keys.

The results of this chapter are contained in our paper [59]. This paper is a joint work with Andrea Huszti.

4.2.1 Our Contribution and Literature

In a multi-server environment usually, threshold password-authenticated key exchange protocols are designed. First, the threshold variant of PAKE was recommended in [86]. Devriş Işler and Alptekin Küpçü introduced a scheme [62] where the protocol ensures that multiple storage providers can be employed, and the adversary needs to corrupt the login server and threshold-many storage providers to be able to mount an offline dictionary attack. Later, they proposed a new framework [63] for distributed single password protocols (DiSPP). Mario Di Raimondo and Rosario Gennaro recommended two threshold password-authenticated key exchanges [99] where the protocols require $n > 3t$ servers to work. They enforce a transparency property: from the point of view of the client, the protocol should look exactly like a centralized KOY protocol [67]. Password-Protected Secret Sharing (PPSS) scheme with parameters (t, n) was formalized by Bagherzandi et al. [9]. Jarecki et al. [65] presented the first round-optimal PPSS scheme, requiring just one message from user to server and from server to user, and proved its security in the challenging password-only setting where users do not have access to an authenticated public key. A secure distributed password verification protocol is presented in [31]. This scheme applies the verifiability of the V-OPRF (verifiable oblivious

pseudorandom function) and supports thresholds and robustness.

We give the details and the novelty of our solution, compared to the previous propositions. The main goal of *key exchange* protocols is to set up a shared secret key between two or more entities. In case of *key agreement*, both entities contribute to the joint secret key by providing information from which the key is derived. In mutual authentication parties who engage in a conversation in which each gains confidence that it is the other with whom he speaks. In protocols providing implicit key authentication, each participant is assured that no one other than the intended parties can learn the value of the session key. A key agreement protocol that provides mutual implicit key authentication is called an *authenticated key agreement* protocol (or AK protocol). A key agreement protocol provides key confirmation (of B to A) if A makes sure that B possesses the secret key.

Our main goal is to design an authenticated key agreement with key confirmation protocol (AKC) which takes advantage of distributed systems. Our protocol would fit into these systems and takes advantage of the capabilities of these systems like robustness, scalability and greater availability. We assume there are thousands of servers in a cloud system therefore we reject the single-server authentication (e.g. Kerberos) and instead we propose the multi-server authentication. It is important to note that a single point of failure occurs typically in single-server solutions. If the server is unavailable, the provider usually needs to ensure replication to tackle the failure of their servers. Our scheme consists of n servers and the user randomly selects $k \leq n$ ones for each authentication. Besides the randomly chosen k servers a server called authentication server is also chosen randomly in our solution. In the authentication phase, the k servers use their long-lived keys and the user's authentication is verified by the chosen server via the correct MAC values. Even if one or more servers fail out of the n servers, the client can still choose k servers randomly. So if one or more servers break down or become corrupt, the service provider will be able to service and authenticate the users securely. Compared to our earlier proposed protocol ([58]), we use secret splitting technique and we also achieved the scalable property.

During authentication, instead of securely constructing the secret password from its shares, a random challenge generated by the client is con-

structed and verified by the authentication server. The randomly chosen participating servers are able to compute their challenge shares with the help of the password-based long-lived key set during registration and send them to the authentication server. In this way, the confidentiality of the password is assured. We focused heavily on making the protocol effective and we achieved promising efficiency results. The related protocols in the literature in several cases employ asymmetric cryptographic primitives which are considered slow compared to symmetric solutions and hash functions. The results of our protocol can be led back to the facts that the session key is generated by Elliptic Curve Diffie-Hellman (ECDH) key exchange, moreover MAC, xor operations and symmetric encryption are applied.

Finally, we also provide the security proof of the protocol and we demonstrate that the protocol is provably secure. For our protocol, we extended the Bellare and Rogaway security model in [19] to prove that our multi-device scheme is secure and we introduced the threshold hybrid corruption model. We assume that among the randomly chosen k servers, there is always at least one uncorrupted and the authentication server reveals at most the long-lived keys. We prove that the proposed protocol is a secure AKC protocol in the random oracle model, assuming the Elliptic Curve Computational Diffie-Hellman (ECCDH) assumption holds in the elliptic curve group, if MAC is universally unforgeable under an adaptive chosen-message attack and the symmetric encryption scheme is indistinguishable under chosen plaintext attack.

4.2.2 The Proposed Scheme

In this section we propose a multi-server authenticated key agreement protocol with key confirmation. We assume that a secret symmetric, long-lived key is exchanged between each client and server during client registration. In the authentication phase, a client chooses several servers randomly to participate. The client verifies the identity of each server and one of the randomly chosen servers, the authentication server, proceeds the steps of the client authentication. Mutual authentication of the participants is based on the correctness of a calculation, where the secret, long-lived symmetric key is used. Besides the mutual authentication of the participants a secret, session key is exchanged between the client and

the authentication server.

During registration, the client sets password-based long-lived keys with all the n servers. In such a system, in addition to the aspect of robustness, the property of scalability is also important. To achieve this, we propose a simple solution in which the client accesses the long-lived keys by using a password. We assume that a client software is running on the client device (e.g. smartcard, mobile phone etc.) that requires a password from the user to initiate the authentication process. After the client gives the password the client software generates the long-lived keys and the execution of authentication begins. The correctness of the password is verified by the servers not the client software, hence a client device does not store any information about the password.

During authentication a server only with the knowledge of the symmetric, long-lived key K_i, where $i \in \{1, \ldots, k\}$ generated from the client password, is able to calculate the challenge value given by the client. The authentication server authenticates the client by verifying the correctness of all the k challenge values received from the participating servers.

In the proposed protocol servers communicate on secure channels to each other. We prefer one server chosen randomly that communicates to the client, hence the client does not need to communicate to all the k servers in parallel and build secure channels.

During the design of the protocol, the efficiency of authentication is ensured by MAC and other fast cryptographic algorithms (hash, xor operation, symmetric encryption). The protocol is provably secure and the necessary model and the formal proof are given. We apply distributed authentication, thus we extend the model with the concept of threshold hybrid corruption.

Setup

In the setup phase the system parameters are generated and the registration is executed between the client and the servers. We differentiate two participants: A *client* (I) asks for services and the *servers* $(J_1, \ldots, J_n)$. We denote by $\{0,1\}^*$ the set of all binary strings of finite length. If x, y are strings, then $x||y$ denotes the concatenation of x and y. Let $\oplus$ denote a bitwise xor calculation. During setup all system parameters and keys are generated. Let E denote an elliptic curve defined over a finite

field $\mathbb{F}_q$ and $G \in E(\mathbb{F}_q)$ a point of order $\mathbf{n}$. Elliptic curve parameters are chosen in a way that the system resists all known attacks on the elliptic curve discrete logarithm problem in $< G >$. Let σ denote the length of an elliptic curve point binary representation. We also represent n servers ($J_i,\ i \in \{1,\ldots,n\}$) as a bitstring with length σ. System parameters par are given by $par = (E, q, \mathbf{n}, G, H, H_0, MAC)$, where $H : \{0,1\}^* \to \{0,1\}^\nu$, $H_0 : \{0,1\}^* \to \{0,1\}^\iota$ are cryptographic hash functions and ν, ι are not necessarily different, ι is the size of the secret session key being exchanged. $MAC : \{0,1\}^* \to \{0,1\}^\nu$ is a MAC function. System parameters are publicly known. Password-based long-lived secret symmetric keys $(K_1,\ldots,K_n)$ between the client and each server are exchanged securely. To provide message confidentiality between the servers each server possesses $n-1$ symmetric encryption keys for secure communication and Server $J_i, i \in \{1,\ldots,n\}$ stores $\overline{K}_j, j \in \{1,\ldots,n-1\}$. These keys are short-term and exchanged securely.

Scalability

In this section, we present an algorithm providing scalability of our protocol. Let *KKDF* denote a Keyed Key Derivation Function that for a message m and a key generates a secret key K, i.e. $K = KKDF_{key}(m)$. Let the message be the password psw and the $key = H(salt||psw)$, and c be the number of iteration. Salt is a short (12 - 48 bits) random piece of data that is concatenated with the password before hashing to increase resistance against offline attacks. In case of a weak password the randomly chosen salt value significantly slows down the offline attacks (e.g. brute force attacks). Value $KKDF_{key}(psw)$, and secret shares

$$KKDF^{c}{}_{key}(psw), KKDF^{c+1}{}_{key}(psw), \ldots, KKDF^{c+n-2}{}_{key}(psw)$$

are calculated for the $n-1$ servers. Finally let

$$K_n = KKDF_{key}(psw) \oplus KKDF^{c}{}_{key}(psw) \oplus \cdots \oplus KKDF^{c+n-2}{}_{key}(psw)$$

for the nth server. If the number of servers are increased in the cloud, we take the secret part of one of the servers and divide it into as many parts as the number of new servers we want to add to the system. In general, for increasing n servers with k new ones, a $KKDF^{l}{}_{key}(psw)$ is chosen to

be scaled and $KKDF^{t}{}_{key}(psw)$, where $t = c+n, \ldots, c+n+k-1$ are calculated. The secret share for the chosen server is modified to

$$K_{new} = KKDF^{l}{}_{key}(psw) \oplus KKDF^{c+n}{}_{key}(psw) \oplus \ldots \oplus KKDF^{c+n+k-1}{}_{key}(psw).$$

The device stores for each server a list of numbers of iteration. The list has either one element, or if it is scaled by k new ones, than $k+1$ elements. If we decrease the number of servers, we take the iteration numbers of the deleted servers and add them to the stored list of a remaining server. Observe, that to calculate a secret share from another share the key of the KKDF, the salt and the password are needed. Hence even if we scale a corrupted share, the new shares cannot be calculated. The salt is stored only on the client device and the password is known only by the client.

Authentication Phase

In the authentication phase, we utilize the benefits of the distributed system to perform multiple server authentication. In this phase, mutual authentication between a client and the randomly chosen servers is processed with a key agreement. Client I randomly chooses k servers out of the n. The k servers verify whether I possesses the long-lived symmetric keys $(K_1, \ldots, K_k)$. At the end of the authentication, a secret session key ssk is exchanged. Figures 4.6, 4.7 and 4.8 show the processes of authentication.

As the first step of the authentication, I selects k servers that are involved in the authentication. A $v \in \{1, \ldots, k\}$ value is generated by a time-based pseudorandom number generator. Let J_v denote the authentication server, which performs the authentication on server side. J_v communicates with the client and server J_i, where $i = 1, \ldots, k-1$. After entering the correct password, the client software calculates the long-lived keys from the password and the salt that is stored on the device. I generates random bitstrings $t_1, \ldots, t_k$ and calculates hash values $w_1 = H(t_1), \ldots, w_k = H(t_k)$ and w, such that $w = H(w_1||w_2|| \ldots ||w_k)$. Random values $r_1, \ldots, r_k \in \{0,1\}^{\sigma}$ and $x \in \mathbb{Z}^{*}_{\mathbf{n}}$ are generated, too. Subsequently, I creates the first message M_1, which is sent to the chosen authentication server J_v. I computes $m_0 = H(w)$ and $m_v = (MAC_{K_v}(r_v \oplus xG \oplus J_v) \oplus w_v)||r_v||xG$, where MAC is calculated with the long-lived symmetric key K_v. Moreover, m_v comprises xG, which is an elliptic curve point and represented by a bitstring that is necessary for the key agreement, it is the client

I J_v

$(K_1, \ldots, K_k)$, G K_v, G

$K_i = KKDF_{key}^{c+i}(psw)$, where $key = H(salt||psw)$

$K_n = KKDF_{key}(psw) \oplus \cdots \oplus KKDF^{c+n-2}{}_{key}(psw)$

$t_1, \ldots, t_{k-1}, t_v$; $r_1, \ldots, r_{k-1}$, r_v; x random

$w_1 = H(t_1), \ldots, w_v = H(t_v)$

$w = H(w_1|| \ldots ||w_{k-1}||w_v)$

$m_0 = H(w)$

$m_i = (MAC_{K_i}(r_i \oplus J_i) \oplus w_i)||r_i$

$m_v = (MAC_{K_v}(r_v \oplus xG \oplus J_v) \oplus w_v)||r_v||xG$

$$\xrightarrow[\text{public channel}]{M_1 = I||J_1||...||J_k||m_0||...||m_k}$$

Figure 4.6. Authentication - Client process

message of the ECDH key exchange. The first message also consists of $m_i = (MAC_{K_i}(r_i \oplus J_i) \oplus w_i)||r_i$. Authentication of the participants is based on the correct calculation of the MAC values.

J_v receives message M_1 and forwards each m_i together with I to server J_i. Each server receives $I||m_i$, where $m_i = p||o$. Server J_i calculates $MAC_{K_i}(o \oplus J_i)$, where K_i is the long-lived key exchanged between the client and the server. Each server calculates $w_i' = p \oplus MAC_{K_i}(o \oplus J_i)$. Therefore, a server is able to calculate a valid w_i' only with the knowledge of K_i, the secret, long-lived key exchanged with I before. Value w_i' is sent back to J_v encrypted. J_v calculates w' from all w_i' received, and checks whether $H(w) = H(w')$ holds. If they are equal, then J_v makes sure about the authenticity of the client. Thereafter J_v generates a random value $y \in \mathbb{Z}_{\mathbf{n}}^*$ and computes the secret session key $ssk = H_0(yxG)$. J_v calculates response $M_2 = h||yG$, where $h = H(ssk||yG||xG||w)$ and yG is the server message of the EC Diffie-Hellman key exchange.

I receives $M_2 = h||yG$ and calculates the secret session key $ssk = H_0(yxG)$ and $h' = H(ssk||yG||xG||w)$. If $h = h'$, then I is confirmed that server J_v knows the secret session key, and the randomly chosen servers know the secret long-lived keys, hence their identity is verified. As a last

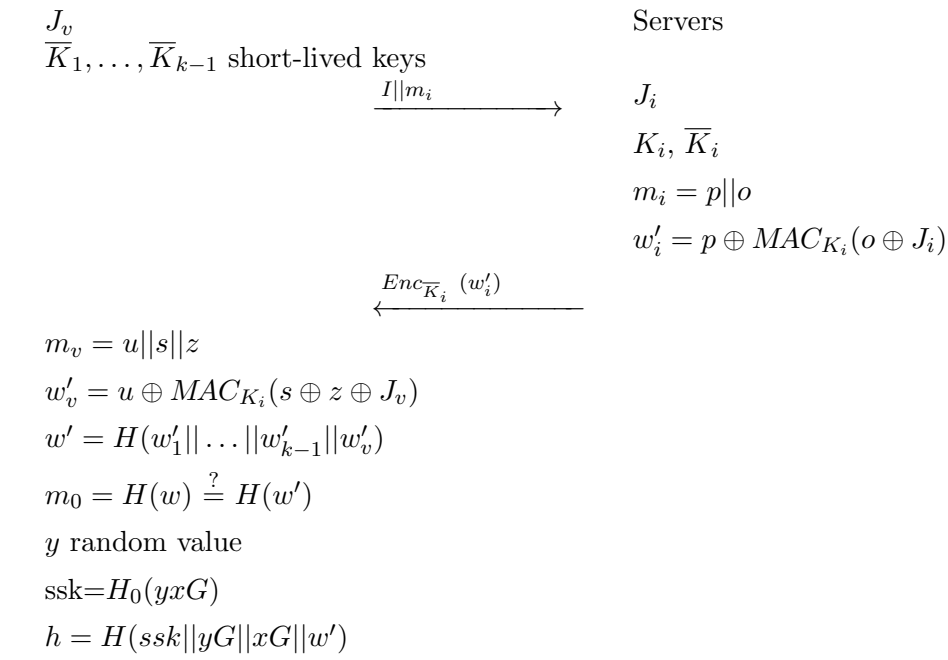


Figure 4.7. Authentication - Cloud servers communication

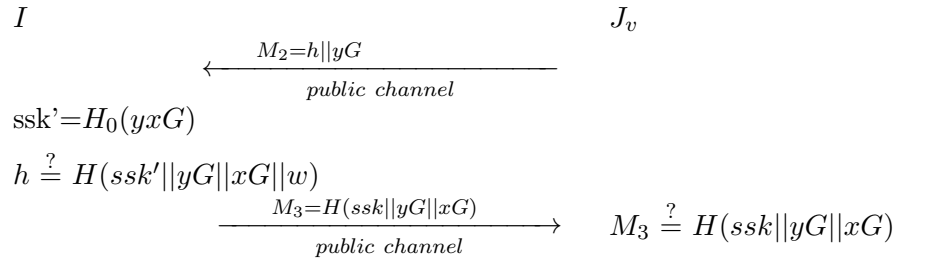


Figure 4.8. Authentication - Final process

step $M_3 = H(ssk||yG||xG)$ is computed and sent to J_v. J_v verifies the message received from the client and if it is correct, J_v confirms that I knows the secret session key.

In the authentication phase, the random value w can be calculated on server side only if the servers know the long-lived symmetric keys, hence

the client is able to verify the identity of multiple servers by checking h. On the other hand after calculating w on server side involving keys K_i, m_0 is checked. Correct m_0 proves that the client possesses K_i, hence identity of the client is verified as well. Value r_i ensures that the MAC value m_i is fresh for every authentication in order to avoid replay attack. Basically, the secret session key is created via an authenticated key agreement protocol based on random values (x, y) generated by the client and the selected server. These values are sent securely so the attackers cannot gain any information about them. Considering the time complexity, the authentication phase is very efficient, since there is only one scalar multiplication on both sides besides the hash, MAC and xor operations.

4.2.3 Security Analysis

In this section we provide a formal security proof of the proposed protocol. Basic requirements of the proposed protocol are mutual authentication of the participants, key secrecy, key freshness and key confirmation. Secure mutual authentication of participants prevents impersonation attack, the new key should be kept secret, and an old key shouldn't be exchanged successfully. At the end parties should confirm that the other party is able to use the new session key.

Known-key security and forward secrecy are also considered. If known-key security holds, disclosure of a session key does not jeopardize the security of other session keys. Forward secrecy holds if long-term secrets of one or more entities are compromised and the secrecy of previous session keys is not affected.

In 4.2.3 section, we specify the base concept of our cloud authentication security model. We also give the notions of partnering, reveal, corrupt queries and model semantic security of the session key via Test query. Here, we extend and generalize the security model as follows.

Cloud Authentication Security Model

The mutual key authentication with mutual key confirmation protocols are called an *authenticated key agreement with key confirmation* protocol (or an AKC protocol)[13], which fulfill the above. We have extended the

indistinguishability-based model proposed by M. Bellare and P. Rogaway in 1993 (see [14], [15], [19], [13]). Our goal of applying multiple servers is to take into account the situation when the verifier server is corrupted, *i.e.* the long-lived keys and other login information stored in the server database are hacked. Multiple servers together provide secure user authentication, if at least one of the verifier servers is uncorrupted.

We detail the security model in the case of our cloud protocol, where each protocol run k servers are chosen. We assume that at least one of the k servers is not corrupted. Participant J_v chosen randomly out of the k servers conducts the protocol on server side, J_v communicates with I and the other $k-1$ servers. We assume that each participant $I \in Client$ holds long-lived symmetric keys exchanged with each server $J_i \in Server$, $i \in \{1, 2, \ldots, n\}$ during previous registration.

Adversarial model

The adversary $\mathcal{A}$ is neither a client nor a server. $\mathcal{A}$ is a probabilistic polynomial time Turing Machine with a query tape where oracle queries and their answers are written. $\mathcal{A}$ is able to relay, modify, delay or delete messages. We assume that $\mathcal{A}$ is allowed to make the following queries.

`Send`$(\prod_{I,J}^{l}, M)$: This oracle query models an active attack, allows $\mathcal{A}$ to send the message M to oracle $\prod_{I,J}^{l}$, and the oracle returns a message (m) that the user instance sends in response to the message M. $\prod_{I,J}^{l}$ following the protocol steps also provides information whether the oracle is in state (δ) **Accepted**, **Rejected** or $*$. The query enables $\mathcal{A}$ to initiate a protocol run between participants I and J by query `Send`$(\prod_{I,J}^{l}, \lambda)$. The oracle replies: m, δ.

`Reveal`$(\prod_{I,J}^{l})$: This models an insecure usage of a session key. If oracle $\prod_{I,J}^{l}$ is in state **accepted**, holding a secret session key ssk, then this query returns ssk to $\mathcal{A}$. The oracle replies: ssk.

`Corrupt`$(\prod_{I,J}, K'_{I,J})$: This oracle query models the corruption of participant I. Replying to this oracle query a participant oracle $\prod_{I,J}$ replies long-lived keys $K_{I,J}$ and I's state, *i.e.* all the values stored by the participant I, moreover $\mathcal{A}$ is allowed to replace the stored

long-lived keys with any valid keys of $\mathcal{A}$'s choice $K'_{I,J}$. The oracle replies: $K_{I,J}, state_I$.

`Test`$(\prod^{l}_{I,J})$: This oracle query models the semantic security of the secret session key. It is allowed to be asked only once in a protocol run. If participant I has **accepted** holding a secret session key ssk, then a coin b is flipped. If $b = 1$, then ssk is returned to the adversary, if $b = 0$, then a random value from the distribution of the session keys is returned.

We define $\mathcal{A}$'s advantage, the probability that $\mathcal{A}$ can distinguish the session key held by the queried oracle from a random string, as follows:

$$Adv^{\mathcal{A}}(\kappa) = |Pr[\text{guess correct}] - 1/2|.$$

Participants' oracle instances are terminated when they finish a protocol run. They are in state **accepted**, if they decide to accept holding a secret session key denoted by ssk, after receipt of properly formulated messages. An oracle can be in state accepted before it is terminated. An oracle is **opened** or **corrupted**, if it has answered a query `Reveal`$(\prod^{l}_{I,J})$ or `Corrupt`$(\prod_{I,J}, K'_{I,J})$, respectively.

Moreover adversary $\mathcal{A}$ is given access to $MAC(.)$ and $Enc(.)$ oracles as well.

The threshold hybrid corruption model

A model is a *strong corruption model* ([13]), if long-lived keys $K_{I,J}$ and all the values stored (e.g. randomly chosen secret values) by the participant I during the protocol run are transferred to $\mathcal{A}$. In case of the *weak corruption model* only the long-lived keys $K_{I,J}$ are transferred or replaced, the adversary does not completely compromise the machine. Other values generated and stored during the protocol run are not revealed.

4.2.1 Definition. We call a model *threshold hybrid corruption model*, if we assume that the client is uncorrupted and there are at least $n - k + 1$ uncorrupted servers out of the n servers, if k servers are chosen randomly for AKC. Moreover, the server chosen to communicate with the client is

1. uncorrupted, or

2. corrupted weakly and among the remaining servers there is at least one uncorrupted.

During the attack an experiment of running the protocol with an adversary $\mathcal{A}$ is examined. After generating the keys and system parameters, $\mathcal{A}$ initializes all participant oracles and asks polynomially number of oracle queries including $\texttt{Send}(\prod_{I,J}^{l}, M)$, $\texttt{Reveal}(\prod_{I,J}^{l})$, $\texttt{Corrupt}(\prod_{I,J}, K'_{I,J})$ to the participant oracles. Finally $\mathcal{A}$ asks a $\texttt{Test}(\prod_{I,J}^{l})$ query.

In order to give the definition of a secure AKC protocol, we need to review the definition of conversation and matching conversation from [19]. They were also formalized in [14].

4.2.2 Definition. Consider an adversary $\mathcal{A}$ and a participant oracle $\prod_{I,J}^{s}$. We define the *conversation* $C_{I,J}^{s}$ of $\prod_{I,J}^{s}$ as a sequence of

$$C_{I,J}^{s} = (\tau_1, \alpha_1, \beta_1), (\tau_2, \alpha_2, \beta_2), \ldots, (\tau_m, \alpha_m, \beta_m),$$

where τ_i denotes the time when oracle query α_i and oracle reply β_i are given $(i = 1, \ldots, m)$.

Naturally $\tau_i > \tau_j$, iff $i > j$. $\mathcal{A}$ terminates after receiving the reply β_m, *i.e.* does not ask more oracle queries. During a conversation the initiator and responder oracles are differentiated. $\prod_{I,J}^{s}$ is an initiator oracle if $\alpha_1 = \lambda$, otherwise it is a responder. Consider the definition for matching conversation when the number of protocol flows is odd.

4.2.3 Definition. Running protocol P in the presence of $\mathcal{A}$, we assume that the number of flows is $R = 2\rho - 1$, $\prod_{I,J}^{s}$ is an initiator and $\prod_{J,I}^{t}$ is a responder oracle that engage in conversations C and C', respectively.
C' is a *matching conversation* to C, if there exist $\tau_0 < \tau_1 < \cdots < \tau_{R-1}$ and $\alpha_1, \beta_1, \ldots, \beta_{\rho-1}, \alpha_\rho$ such that C is prefixed by:

$$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), \ldots, (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho),$$

and C' is prefixed by:

$$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}).$$

C is a *matching conversation* to C', if there exist $\tau_0 < \tau_1 < \cdots < \tau_R$ and $\alpha_1, \beta_1, \ldots, \beta_{\rho-1}, \alpha_\rho$ such that C' is prefixed by:

$$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}), (\tau_{2\rho-1}, \alpha_\rho, *),$$

and C is prefixed by:

$$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), \ldots, (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho).$$

If C is a matching conversation to C' and C' is a matching conversation to C, then $\prod_{I,J}^s$ and $\prod_{J,I}^s$ are said to have had *matching conversation.*

Matching conversation formalizes real-time communication between entities I and J, it is necessary to define authentication property of an AKC protocol. We give the definition of the event $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ that is a modified version of the definition given in [19]. We leave out the requirement that $J \in \mathit{Server}$ is uncorrupted. In our multi-server setting each client communicates with a server that can be corrupted weakly, if there is at least one uncorrupted server from the k servers.

4.2.4 Definition. $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ denotes an event when in a protocol P in the presence of an adversary $\mathcal{A}$ assuming a threshold hybrid corruption model, there exist

1. a client oracle $\prod_{I,J}^s$ which is accepted, but there is no server oracle $\prod_{J,I}^t$ having a matching conversation with $\prod_{I,J}^s$, or
2. a server oracle $\prod_{I,J}^s$ which is uncorrupted and accepted, but there is no client oracle $\prod_{J,I}^t$ having a matching conversation with $\prod_{I,J}^s$, or
3. a server oracle $\prod_{I,J}^s$ which is corrupted weakly and accepted, but there is no client or no uncorrupted server oracle having a matching conversation with $\prod_{I,J}^s$.

In order to give the definition of a secure AKC, it is essential to define the notion of *freshness* and *benign adversary.*

4.2.5 Definition. A $k+1$-tuple of oracles containing one client and k server oracles is fresh, if in the threshold hybrid corruption model the client

oracle and the server oracle with which it has had a matching conversation are unopened. We call an oracle *fresh*, if it is an element of a fresh $k+1$-tuple.

4.2.6 Definition. An adversary is called *benign* if it is deterministic, and restricts its action to choosing a $k+1$ tuple of oracles containing one client and k server oracles, and then faithfully conveying each flow from one oracle to the other, with the client oracle beginning first.

4.2.7 Definition. A protocol is a *secure AKC protocol* if,

1. In the presence of the benign adversary the client and the server oracle communicating with the client always accept holding the same session key ssk, and this key is distributed uniformly at random on $\{0,1\}^{\kappa}$.

and if for every adversary $\mathcal{A}$

2. If in a threshold hybrid corruption model there is a server oracle $\prod_{I,J}^{l}$ having matching conversations with a client oracle and if $\prod_{I,J}^{l}$ is weakly corrupted, $\prod_{I,J}^{l}$ has matching conversation with an uncorrupted server oracle, then the client oracle and oracle $\prod_{I,J}^{l}$ both accept and hold the same session key ssk.

3. The probability of $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ is negligible.

4. If the tested oracle is fresh, then $Adv^{\mathcal{A}}(\kappa)$ is negligible.

4.2.8 Theorem. *The proposed protocol is a secure AKC protocol in the random oracle model, assuming MAC is* universally unforgeable under an adaptive chosen-message attack *and symmetric encryption scheme is* indistinguishable under chosen plaintext attack*, moreover* ECCDH assumption holds *in the elliptic curve group.*

Proof: The conditions 1 and 2 hold, since the steps of the protocol are followed and with the assumption that the MAC and the encryption scheme provides correct verification and decryption, respectively. Moreover the hash function is a random oracle.

Let's look at condition 3. Consider an adversary $\mathcal{A}$ and suppose that $Pr[\texttt{No-Matching}^{\mathcal{A}}(\kappa)]$ is non-negligible. There are two cases: either the server, or the client oracle is accepted.

- Case 1.

  Let $\mathcal{A}$ `succeeds` denote the event that in $\mathcal{A}$'s experiment there is a server oracle $\prod^{t}_{J_v,I}$ that is *accepted*, but there is no client oracle $\prod_{I,J_v}$ having matching conversation to $\prod^{t}_{J_v,I}$.

  We assume that

  $$Pr[\mathcal{A} \texttt{ succeeds}] = n_S(\kappa),$$

  where $n_S(\kappa)$ is non-negligible.

  We construct a polynomial time adversary $\mathcal{F}$ that is able to proceed an existential forgery against MAC under an adaptive chosen message attack. $\mathcal{F}$'s task is to generate a valid (m, t) message-tag pair, where m was never asked from the oracle $MAC_K(.)$ for a security parameter κ. $\mathcal{F}$ simulates the key generation Γ and answers $\mathcal{A}$' oracle queries.

  $\mathcal{F}$ randomly picks $I \in Client$ and $J_1, \ldots, J_k \in Server$, moreover randomly chooses $J_v \in \{J_1, \ldots, J_k\}$ and $J_u \in \{J_1, \ldots, J_k\}$. Let $\Delta = \{I, J_1, \ldots, J_k\}$ denote identities of protocol participants. $\prod_{J_v,I}$ denotes the server oracle that communicates to the client I, $\prod_{J_u,J_v}$ oracle denotes the uncorrupted server oracle that is in connection with server J_v. If $u = v$, then the server communicating with the client is uncorrupted. $\mathcal{F}$ also chooses randomly a particular session $l \in \{1, \ldots, T_3(\kappa)\}$. Given security parameter κ, adversary $\mathcal{F}$ randomly chooses values $K_1, \ldots, K_k$ as long-lived keys exchanged between client I and servers $J_1, \ldots, J_k$, and $\overline{K}_1, \ldots, \overline{K}_{k-1}$ as encryption keys exchanged between J_v and $J_1, \ldots, J_{k-1}$. $\mathcal{F}$ runs $\mathcal{A}$ and answers $\mathcal{A}$'s queries as follows.

  1. $\mathcal{F}$ answers H_0, H hash oracle queries at random (like a real random oracle would).
  2. $\mathcal{F}$ answers `Corrupt` queries according to Π, reveals long-lived keys K_i, internal states and encryption keys $\overline{K}_i$ for corrupted servers. Queries to the uncorrupted server and the client oracles

are refused. If $u \neq v$, then J_v is corrupted weakly, hence to the corrupt query $\mathcal{F}$ answers only K_i and $\overline{K}_i$.

3. $\mathcal{F}$ answers `Reveal` queries as specified in Π. This query is refused if it is asked from $\prod_{I,J_v}$ or $\prod_{J_v,I}$.

4. $\mathcal{F}$ answers `Send` queries according to Π with the knowledge of the keys, if they are not sent to $\prod_{J_v,I}$ and $\prod_{J_u,J_v}$. $\mathcal{F}$ answers queries to $\prod_{J_u,J_v}$ by choosing $\overline{K}_u$ randomly. If $\mathcal{A}$ does not involve $\prod_{J_v,I}$ as a server oracle which communicates to the client oracle $\prod_{I,J_v}$ and other server oracles $\prod_{I,J_i}$, then $\mathcal{F}$ gives up. If $\mathcal{A}$ involves $\prod_{J_v,I}$ as an initiator oracle, then $\mathcal{F}$ gives up. Otherwise $\mathcal{A}$ asks query $\texttt{Send}(\prod_{J_v,I}, M_1)$, where

$$M_1 = I||J_1|| \dots ||J_k||m_0|| \dots ||m_k,$$

where $m_j = (MAC_{K_j}(r_j \oplus J_j) \oplus w_j)||r_j$ for corrupted servers, $r_j \in \{0,1\}^\sigma$ are chosen randomly, $m_v = (MAC_{K_v}(r_v \oplus xG \oplus J_v) \oplus w_v)||r_v||xG$ for random $x \in \mathbb{Z}_q^*$ and m_u for uncorrupted J_u.

$\mathcal{A}$ asks hash oracle queries $H(.)$ to get w_i, $\mathcal{F}$ answers these queries. If there is a J_t, $t = 1..k$ in M_1 such that $J_t \notin \Delta$, then $\mathcal{F}$ gives up. $\mathcal{A}$ asks MAC oracle queries to calculate m_u, $\mathcal{F}$ answers these queries using his/her oracle $MAC_K(.)$. Eventually $\mathcal{A}$ creates a valid m_u. $\mathcal{F}$ calculates the valid $(\overline{m}, \overline{t})$ MAC forgery, where $\overline{t} = MAC_K(\overline{m})$, as follows. If $u = v$, then $m_u = p||o||f$, $\overline{t} = p \oplus w_u$ and $\overline{m} = o \oplus f \oplus J_u$, where w_u is generated via oracle $H(.)$. If $u \neq v$, then $m_u = p||o$, $\overline{t} = p \oplus w_u$, $\overline{m} = o \oplus J_u$. If $\overline{m}$ was asked to oracle $MAC_K(.)$ before, then $\mathcal{F}$ gives up. $\mathcal{F}$ answers query $\texttt{Send}(\prod_{J_v,I}, M_1)$ with $H(H_0(yxG)||yG||xG||w)||yG$, where $y \in \mathbb{Z}_q^*$ randomly chosen and w is calculated from w_i values of the corrupt servers and w_u. If some later time $\mathcal{A}$ does not asks $\texttt{Send}(\prod_{J_v,I}, H(H_0(xyG)||yG||xG))$, then $\mathcal{F}$ gives up, otherwise $\prod_{J_v,I}$ gets accepted.

$\mathcal{F}$ answers query $\texttt{Send}(\prod_{I,J_v}, M_2)$, as follows. If $M_2 \neq H(H_0(yxG)||yG||xG||w)||yG$, then $\mathcal{F}$ gives up, otherwise replies $H(H_0(yxG)||yG||xG)$.

Finally, $\mathcal{F}$ responses $(\overline{m}, \bar{t})$ to the challenger. If $\mathcal{A}$ succeeds with non-negligible probability, then $\mathcal{F}$ outputs a valid forgery $(\overline{m}, \bar{t})$, where $\overline{m}$ was never asked to oracle $MAC_K(.)$ before.

Assume that $\mathcal{A}$ is successful, event $\mathcal{A}$ `succeeds` happens with $n_S(\kappa)$ non-negligible probability. Hence following the algorithm above $\mathcal{F}$ calculates a valid $(\overline{m}, \bar{t})$ pair. We show that $\mathcal{F}$ wins its experiment with non-negligible probability. The probability that $\mathcal{F}$ chooses correct participants Δ, session l and succeeds is

$$\xi_1(\kappa) = \frac{n_S(\kappa)}{T_1(\kappa)T_2(\kappa)\binom{T_2(\kappa)-1}{k-1}T_3(\kappa)} - \lambda(\kappa), \tag{4.2.1}$$

where $\lambda(\kappa)$ denotes the probability that $\mathcal{F}$ previously calculated the flow. Since $n_S(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and $\lambda(\kappa)$ is negligible and

$$\xi_1(\kappa) \geq \frac{n_S(\kappa)}{T_1(\kappa)T_2(\kappa)T_2(\kappa)^{k-1}T_3(\kappa)} - \lambda(\kappa),$$

thus $\xi_1(\kappa)$ is non-negligible. That contradicts the security assumption of MAC, hence $n_S(\kappa)$ must be negligible.

- Case 2.

  Let $\mathcal{A}$ `succeeds` denote the event that in $\mathcal{A}$'s experiment there is a client oracle $\prod^s_{I,J_v}$ that is *accepted*, but there is no server oracle $\prod_{J_v,I}$ having matching conversation to $\prod^s_{I,J_v}$. We assume that

  $$Pr[\mathcal{A} \texttt{ succeeds}] = n_C(\kappa),$$

  where $n_C(\kappa)$ is non-negligible.

  We can construct a polynomial time adversary that is able to distinguish two plaintexts under chosen plaintext attack against the symmetric encryption scheme.

  Challenger generates a key K and flips a bit b. $\mathcal{F}$ is given an oracle access to $Enc_K(.)$. $\mathcal{F}$'s task is to output a bit b' on inputs m_0, m_1 chosen by $\mathcal{F}$ and m_b. $\mathcal{F}$ picks the protocol participants and a session $l \in \{1, \ldots, T_3(\kappa)\}$, let $\Delta = \{I, J_1, \ldots, J_k\}$ denote the identities.

Similarly to Case 1, $\prod_{I,J_v}$ denotes the client, $\prod_{J_v,I}$ the communicating server and $\prod_{J_u,J_v}$ the uncorrupted server oracle. If $u = v$, then the server communicating with the client is uncorrupted. $\mathcal{F}$ simulates the key generation Γ in the same way as in Case 1. $\mathcal{F}$ generates long-lived keys K_i and symmetric encryption keys $\overline{K_i}$ for corrupted servers for the security parameter κ. $\mathcal{F}$ answers $\mathcal{A}$'s oracle queries as follows.

$\mathcal{F}$ answers queries to oracles $H(.), H_0(.)$, `Corrupt`, `Reveal` in the same way as in Case 1. $\mathcal{F}$ answers `Send` queries according to Π with the knowledge of the keys of corrupted servers, if they are not sent to $\prod_{I,J_v}$, $\prod_{J_v,I}$ or $\prod_{J_u,J_v}$. If $\mathcal{A}$ does not involve $\prod_{J_v,I}$ as a server oracle which communicates to the client oracle $\prod_{I,J_v}$ and other server oracles $\prod_{I,J_i}$, then $\mathcal{F}$ gives up. We consider the case when $\prod_{J_v,I}$ is weakly corrupted ($u \neq v$). If $\mathcal{A}$ does not invoke $\prod_{I,J_v}$ as an initiator oracle, then $\mathcal{F}$ gives up, otherwise $\mathcal{A}$ asks oracle query $\texttt{Send}(\prod^s_{I,J_v}, \lambda)$. $\mathcal{F}$ responses $M_1 = I||J_1|| \ldots ||J_k||m_0|| \ldots ||m_k$, with $m_j = (MAC_{K_j}(r_j \oplus J_j) \oplus w_j)||r_j$ and $m_v = (MAC_{K_v}(r_v \oplus xG \oplus J_v) \oplus w_v)||r_v||xG$, where $r_i \in \{0,1\}^\sigma, x \in \mathbb{Z}^*_{\mathbf{n}}, w_i$ and the MAC key K_u for the uncorrupted server are randomly chosen by $\mathcal{F}$. Some time later $\mathcal{A}$ asks oracle queries to $Enc(.)$ and eventually asks query $\texttt{Send}(\prod_{J_u,J_v}, I||m_u)$.

$\mathcal{F}$ answers with $Enc_K(m_b)$. $\mathcal{F}$ perfectly simulates uncorrupted server $\prod_{J_u,J_v}$ to $\mathcal{A}$, since without the knowledge of the MAC key $\mathcal{A}$ cannot verify correctness of m_u. $\mathcal{A}$ some time later asks xyG to oracle $H_0(.)$ and $H_0(xyG)||yG||xG||w$ to oracle $H(.)$ and asks query $\texttt{Send}(\prod_{I,J_v}, H(H_0(xyG)||yG||xG||w)||yG)$. $\mathcal{F}$ answers queries, replies $H(H_0(xyG)||yG||xG)$, gets accepted and checks whether $w = H(w_1|| \ldots ||m_0|| \ldots ||w_{k-1})$, where $w_1, \ldots, w_{k-1}$ denote the random values generated for corrupted servers. If the equality holds, then $\mathcal{F}$ outputs bit $b' = 0$, otherwise $b' = 1$.

Assume that $\mathcal{A}$ is successful, event $\mathcal{A}$ `succeeds` happens with $n_C(\kappa)$ non-negligible probability. $\mathcal{F}$ outputs the correct b'. We show that $\mathcal{F}$ wins its experiment with non-negligible probability. For the analysis the probability that $\mathcal{F}$ chooses the correct participants Δ, session l

and succeeds is calculated:

$$\xi_2(\kappa) = \frac{n_C(\kappa)}{T_1(\kappa)T_2(\kappa)\binom{T_2(\kappa)-1}{k-1}T_3(\kappa)} - \lambda(\kappa), \qquad (4.2.2)$$

where $\lambda(\kappa)$ denotes the probability that $\mathcal{F}$ previously calculated the flow, including the case of uncorrupted $\prod_{J_v,I}$, when $\mathcal{F}$ calculates correct MAC message-tag pair for $\prod_{J_v,I}$. Similarly to Case 1. $\xi_2(\kappa)$ is non-negligible, if $n_C(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and $\lambda(\kappa)$ is negligible. That contradicts the security assumption of the symmetric encryption, hence $n_C(\kappa)$ must be negligible.

We turn to condition 4. Consider an adversary $\mathcal{A}$ and suppose that $Adv^{\mathcal{A}}(\kappa)$ is non-negligible.

- Case 3.

  Let $\mathcal{A}$ `succeeds` against $\prod^s_{I,J}$ denote the event that $\mathcal{A}$ asks `Test`$(\prod^s_{I,J})$ query and outputs the correct bit. Hence

  $$Pr[\mathcal{A} \texttt{ succeeds}] = \frac{1}{2} + n(\kappa),$$

  where $n(\kappa)$ is non-negligible.

  Let A_κ denote the event that $\mathcal{A}$ picks either a server or a client oracle $\prod^s_{I,J}$ and asks its `Test` query such that oracle $\prod^s_{I,J}$ has had a matching conversation to $\prod^t_{J,I}$.

  $$Pr[\mathcal{A} \texttt{ succeeds}] = Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa]Pr[A_\kappa] +$$

  $$Pr[\mathcal{A} \texttt{ succeeds}|\overline{A}_\kappa]Pr[\overline{A}_\kappa]$$

  According to the previous section $Pr[\overline{A}_\kappa] = \mu(\kappa)$, where $\mu(\kappa) \in \{n_C(\kappa), n_S(\kappa)\}$ and $Pr[A_\kappa] = 1 - \mu(\kappa)$, where $\mu(\kappa)$ is negligible, hence

  $$\frac{1}{2} + n(\kappa) \leq Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa]Pr[A_\kappa] + \mu(\kappa)$$

  and we get

  $$\frac{1}{2} + n_1(\kappa) = Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa],$$

for a non-negligible $n_1(\kappa)$. Let B_κ denote the event that for given aG, bG adversary $\mathcal{A}$ or any other oracle besides $\prod^s_{I,J}$ or $\prod^t_{J,I}$ asks abG to oracle $H_0(.)$. $Pr[\mathcal{A}$ `succeeds`$|A_\kappa] = Pr[\mathcal{A}$ `succeeds`$|A_\kappa \wedge B_\kappa]Pr[B_\kappa|A_\kappa] + Pr[\mathcal{A}$ `succeeds`$|A_\kappa \wedge \overline{B}_\kappa]Pr[\overline{B}_\kappa|A_\kappa]$.
Since $Pr[\mathcal{A}$ `succeeds`$|A_\kappa \wedge \overline{B}_\kappa] = \frac{1}{2}$,

$$\frac{1}{2} + n_1(\kappa) \leq Pr[\mathcal{A} \text{ } \texttt{succeeds}|A_\kappa \wedge B_\kappa]Pr[B_\kappa|A_\kappa] + \frac{1}{2}, \quad (4.2.3)$$

hence $Pr[B_\kappa|A_\kappa]$ is non-negligible.

We construct a polynomial time adversary $\mathcal{F}$ that for given aG, bG calculates $ECCDH(aG, bG) = abG$. $\mathcal{F}$ picks the protocol participants, $\Delta = \{I, J_1, \ldots, J_k\}$ denotes the identities. Similarly to Case 1, $\prod_{I,J_v}$ denotes the client, $\prod_{J_v,I}$ the communicating server and $\prod_{J_u,J_v}$ the uncorrupted server oracle. If $u = v$, then the server communicating with the client is uncorrupted. $\mathcal{F}$ sets $par = (E, q, \mathbf{n}, G, H, H_0, MAC)$ public parameters, where $G \in E(\mathbb{F}_q)$ is a generator of order $\mathbf{n}$. $\mathcal{F}$ also simulates the key generation Γ in the same way as in Case 1.

$\mathcal{F}$ answers queries to oracles $H(.), H_0(.)$, `Corrupt`, `Reveal` in the same way as in Case 1. Let $T_4(\kappa)$ denote the polynomial bound on the number of queries allowed to ask to oracle $H_0(.)$. $\mathcal{F}$ randomly picks $j \in \{1, \ldots, T_4(\kappa)\}$, assuming that jth query will be on abG. $\mathcal{F}$ answers `Send` queries according to Π except for queries to $\prod^s_{I,J_v}$ and $\prod^t_{J_v,I}$. $\mathcal{F}$ generates messages to $\prod^s_{I,J_v}$ and $\prod^t_{J_v,I}$ in a way that instead of choosing x, y randomly inserts aG, bG as inputs. The MAC computations of message M_1 are calculated by the randomly chosen keys K_i and aG or bG is inserted. M_2 sent to $\prod^s_{I,J_v}$ is constructed as a concatenation of h and bG or aG, respectively, where h is a freshly generated random value. If $\mathcal{A}$ does not ask j queries to $H_0(.)$, then $\mathcal{F}$ gives up. After the jth query $\mathcal{F}$ stops and outputs the jth query. If $\prod^s_{I,J_v}$ and $\prod^t_{J_v,I}$ do not have matching conversation, then $\mathcal{F}$ gives up.

The probability that $\mathcal{F}$ succeeds is at least

$$\xi_3(\kappa) = \frac{n_1(\kappa)}{T_1(\kappa)T_2(\kappa)\binom{T_2(\kappa)-1}{k-1}T_3(\kappa)^2T_4(\kappa)} - \mu(\kappa),$$

that is non-negligible. This contradicts to the ECCDH assumption, hence $n_1(\kappa)$ and $Adv^{\mathcal{A}}(\kappa)$ must be negligible.

□

The proposed AKC scheme possesses known-key and forward secrecy. Key parameters xG and yG for each run are chosen independently and randomly, hence session keys are also independent and random. The adversary is able to calculate the session key from xG and yG only if he or she computes ECCDH function.

If the adversary asks $\texttt{Reveal}(\prod_{I,J}^{l})$ and receives a session key, the session keys of the subsequent runs are independent from the revealed session key, hence the scheme is known-key secure with the ECCDH assumption.

In the proposed AKC values x, y are not transmitted for key parameters xG and yG, hence if long-term secret keys are compromised, the adversary faces ECCDH assumption for the previous session keys.

Practical Issues

Efficiency was an important aspect during the design of our protocol. In the protocol, the session key is generated by ECDH key exchange, and the other operations are hash and xor operations, which are extremely fast. After the security analysis, a java application was created to simulate the protocol in a real environment. The system is divided into two applications, the first one is the client, and the other one is the application for implementing the servers. Both applications were run in a microsoft azure cloud environment for authentic simulation. We worked with constant parameters in the simulation, because the difference in user names and passwords was negligible in terms of performance. Clients and the selected server communicate on the public channel. The communication between the servers occurs through a encrypted channel. In connection with their operation, it should be noted that the server programs monitor the incoming authentication requests in the background. Furthermore, when running the client program, the user selects the servers that will perform the authentication. After calculating the first message, the user sends it to the server that communicates with the service provider and communicates with the other servers. During the connection, the steps described in the protocol are implemented. If the connection fails, the

server rejects the connection request and keeps track of the channel.

Java Simulation Result

Java simulation was tested with six servers, one of which received the client authentication request and the remaining five servers were required for authentication. Time means the time interval from the first step of the protocol to the successful connection. The average connection time was then calculated based on the received time results. For the cloud environment, we tested the standard package and the p3v2 package. The Standard Package contains 100 Azure compute units [56] and the p3v2 package includes 840 Azure compute units. The concept of the Azure Compute Unit (ACU) provides a way of comparing compute (CPU - Central Processing Unit) performance across Azure SKUs (Stock-keeping-Units). This will help you to identify easily the SKU which is the most likely to satisfy your performance needs. ACU is currently standardized on a Small (Standard_A1) VM (Virtual Machine) being 100 and all other SKUs then represent approximately how much faster that SKU can run a standard benchmark.
Figure 4.9 demonstrates the results:

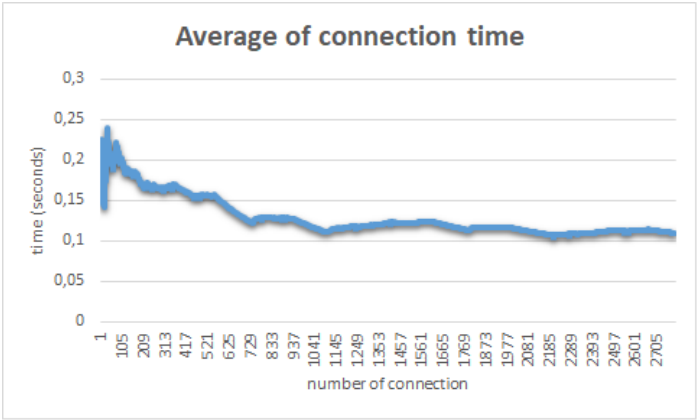


Figure 4.9. Average connection times of the successful authentications between the client and the servers.

The average connection time is 0,1092 second per connection when we use the p3v2 package. In the standard package the result of average connection time is 0,1537 second per connection. This time contains the mutual authentication including the authentication of selected servers and generating a session key. Table 4.1 shows the result of the comparison, where the performance evaluation is based on the running time of the protocol compared to two other available solutions: the Optimal Distributed Password Verification used by three cloud servers (ODPV ([31])) and a password-authenticated key exchange protocol (KOY protocol) by Katz et al. ([67]).

Scheme	**Login Total**
KOY protocol	1,063s
ODPV (3 cloud servers)	0,285 s
Our proposition	0,1092 s

Table 4.1. Average execution time of protocols

It is an important issue in our protocol that the user gives his/her password and after that the long-lived keys are available. The static password is comfortable for the user and the long-lived keys provide the appropriate security level. Since in each authentication the values are random and fresh, the key freshness holds and the protocol execution could not be successfully finished with old, already used values and keys.

Chapter 5

Provably Secure Identity-Based Remote Password Registration

In this chapter, we demonstrate the Certificate-Less Secure Blind Registration Protocol (CLS-BPR), which can be an ideal alternative for the traditional registration methods (email or TLS/SSL connection). During the registration, we apply the bilinear mapping, MAC and hash function, hence we achieve a favorable computational time. The proposed protocol suits our smart home user authentication scheme ([60]) where the values of the bilinear map are stored on the IoT (Internet of Things) devices. Our cloud scheme ([58]) can also be easily modified with the proper long-lived key setting to be compatible with our registration scheme. We also provide a detailed security analysis and prove that our registration protocol is based on the assumptions that solving the Bilinear Diffie-Hellman problem is computationally infeasible, the bilinear map is a one-way function and MAC is existentially unforgeable under an adaptive chosen-message attack, where the bilinear map is considered in the generic bilinear group model and the hash functions are random oracles.

The results of this chapter are contained in our paper [17]. This paper is a joint work with Csanád Bertók, Andrea Huszti and Szabolcs Kovács.

5.1 Our Contribution and Literature

The remote registration of passwords is one of the most important security aspects and the initial step of all remote password-based protocols,but does not get enough attention. To improve the security of password registration, Kiefer and Manulis introduced a new family of protocols

called Blind Password Registration (BPR) for Verifier-Based Password-Authenticated Key Exchange (VPAKE) [71] and two-server PAKE [72]. They proposed Zero-Knowledge Password Policy Checks (ZKPPC) which enables blind registration. Users register their chosen password with a server and prove that it suits the password policy without revealing any information about the password, which prevents password leakage from the server. BPR protocol can be executed over the TLS channel established between the client and the server. They define a security model for stand-alone blind password registration protocols which fulfil the requirements of policy compliance and dictionary attack resistance. OPAQUE [75] is an asymmetric PAKE protocol containing a password registration scheme as well. Password registration can be offline, PKI-based or out-of-band. Blindness and resistance against offline attacks are considered.

Most registration processes are based on passwords where the vulnerabilities are well-known. Users may choose "weak" passwords or not change the default passwords, which are easy for attackers to guess. Another criticism of passwords is that if any site where a certain password is reused becomes compromised and the system administrators do not follow the best industry practices, the participants' other accounts may also become compromised by the attacker who can guess the password with an offline attack. Several attacks aim to figure out passwords, applying dictionary, rainbow tables or brute-force attack. An attacker conducting an offline attack hashes each password guess. While many standard hashing algorithms were designed to make execution quicker, certain ones were deliberately designed to be slow to hinder attackers from conducting an offline attack. For example, the bcrypt hashing scheme [113] can be configured with a cost factor that exponentially increases its execution time by requiring a sequential series of computations. Besides the password, usually a salt value is also used. Salt is a short (12 - 48 bits) random piece of data that is concatenated with the password before hashing. It is then stored with the hash of the password information. An attacker who successfully steals the password file or database is forced to run an exhaustive, computationally expensive offline attack to find the users' passwords from the salted hashes.

In our solution, Identity-Based Cryptography is only used for password registration, where the master secret key is changed daily. This way,

the new entrant receives a short-lived secret key from the Private Key Generator (PKG) server, thus eliminating vulnerability of the secret key in Identity-Based Cryptography. Corruption of the secret key does not result in the change of the public key, with new system parameters new secret keys are generated.

We assume that the server has an extra secret key besides the identity-based key pair. This secret key ensures secrecy and prevents the attacker from accessing the password information from the earlier registration even when the PKG becomes corrupt. In this system, PKG may be distributed, which can further increase security and applying the scheme can suit distributed systems.

In our scheme, a device (e.g. smart card) or an application is required, which generates the salt value. It also checks the password which is chosen by the user. We assume that the password policy, which is demanded, is applied on the client side (such as JavaScript API or other application). This device is used by the user only once during the process of registration. During transmission, mutual authentication of participants and confidentiality of the password and the salt are ensured by applying the temporary identity-based key pair. The server calculates the result of the bilinear map of the password and the salt, and stores the calculated value along with the salt.

Our registration scheme can be applied for standard user login applications, the server receives the password in the usual way, then performs the bilinear mapping with the salt and compares it with the stored value. Our scheme also fits to those authentication processes, when the bilinear mapping is performed on the client side, and the value is sent to the server for verification. The proposed protocol is suitable for the registration phase of the two-factor or one-factor password-based authentication process, depending on whether the device is used for authentication or not.

Note that our proposed registration is also fully blind, as users' passwords and temporary PKG secret key are not known by the server during the registration and subsequent authentication.

In contrast to traditional registration solutions, our solution does not require a TLS channel and can also omit the associated certificate management, which can be efficiently implemented in a corporate or educa-

tional institution. According to our implementation, our protocol is more cost-effective than the above-mentioned TLS-based and the other blind solutions ([71], [72], [75]). It is not necessary to manage certificates or execute costly digital signature.

For password verification and storage, bilinear mapping is used, which meets the requirements of password storage (one-way function). In addition, the bilinear mapping applied for password storage is a "slow" function and it can be extended for multi-rounds.

The registration we recommend is flexible, which is optimal for SSO and Kerberos, but it is also suitable for systems where different passwords must be applied for each service. The bilinear map of the password and the salt can be used as a long-lived symmetric key and applied for entity authentication or session key generation.

Comparison	Certificates	Blind	Interactions	Online
BPR - two server	yes	yes	3	yes
BPR - VPAKE	yes	yes	3	yes
TLS-based	yes	no	4	yes
OPAQUE (offline)	no	yes	2	no
OPAQUE (PKI-based)	yes	yes	3	yes
Our proposition	no	yes	2	yes

We have also formalized the security analysis of the registration protocol. Unlike other schemes ([71], [72]) besides the password hashing scheme we also consider the interactions, when the password information is sent securely. Consequently, we prove that our solution is secure against online attacks as well. We introduce the definition for a secure password registration scheme, provide an adversarial model and show that our scheme is provably secure. Security of the proposed registration protocol is based on the assumptions that solving the Bilinear Diffie-Hellman problem is computationally infeasible, the bilinear map is a one-way function and MAC is existentially unforgeable under an adaptive chosen-message attack, where the bilinear map is considered in the generic bilinear group model and the hash functions are random oracles.

Comparing the offline part of our scheme to [71] and [72], our protocol is still resistant against offline attacks even when the server is corrupted and the client is weakly corrupted.

5.2 Our Proposed Registration Protocol

In this section, we introduce a password registration protocol with password and salt confirmation, *i.e.* the client is able to confirm that the server knows the map of the correct password and the salt.

The protocol consists of a Setup and a Registration phase. During the Setup system parameters and keys are generated for the participants, during the Registration phase the client sends its password information to the server and confirms that the server has received the verification value. The protocol fulfils all the necessary requirements, such as password secrecy, mutual authentication and resistance against offline attacks.

5.2.1 Setup

We differentiate two participants: A *client* (C) requesting registration and a *server* (S). During the setup, all system parameters and keys are generated for the identity-based environment. A Private Key Generator (PKG) generates the identity-based secret keys for the participants. We denote the set of all binary strings of finite length by $\{0,1\}^*$. A security parameter k and the descriptions of groups $\mathbb{G}, \mathbb{G}_{\mathbb{T}}$ of order q are given, where q is a large prime, and the bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ and the function tr from Section 2.2 are made publicly available. The descriptions include polynomial time (in k) algorithms to compute the group operations in $\mathbb{G}, \mathbb{G}_{\mathbb{T}}$ as well as $\hat{e}$.

We build up the identity-based environment as follows. Let P be a generator of $\mathbb{G}$. PKG chooses a random $\alpha \in \mathbb{Z}_q^*$ and generates parameters $P, \alpha P$. The master secret key (msk) for the system is α and the system parameters par are given by $par = (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \hat{e}, tr, P, \alpha P, H, MAC)$, where $H : \{0,1\}^* \to \{0,1\}^\iota$ is a cryptographic hash function, and ι is the size of the long-lived key being exchanged. $MAC : \{0,1\}^* \to \{0,1\}^\nu$ is a Message Authentication Code function, where ν, ι are not necessarily different. System parameters par are publicly known.

Identities (e.g. e-mail address) denoted by ID_C and ID_S are generated for the participants. Public keys are derived, *i.e.* $PK_C = Q_C = tr(ID_C)$ and $PK_S = Q_S = tr(ID_S)$. The PKG calculates the participants' secret keys $SK_C = \alpha Q_C$ and $SK_S = \alpha Q_S$. The server randomly generates $x \in \mathbb{Z}_q^*$, and then sends $(Q_S, x\alpha P)$ to the PKG.

5.2.2 Registration Phase

In the registration phase, a client (C) registers to the server (S) and sends the chosen salted password securely with the salt. An identity-based setting is applied, all the benefits of Identity-Based Cryptography are utilized, *i.e.* we leave the chain of trust (long certificate chains) and the Public Key Infrastructure. We take advantages of the characteristics of the bilinear map $\hat{e}$ including bilinearity and one-way function. In this phase, mutual authentication between the client and the server is processed. Moreover, at the end of this phase S stores the identity of C, the salt and the salted password securely received from C. A long-lived key K is also exchanged. Figure 5.1. and Figure 5.2 show setup and the process of registration between the client and the server.

Client (C)	PKG	Server (S)
	$\alpha \in \mathbb{Z}_q^*$ (msk)	$x \in \mathbb{Z}_q^*$ secret key
	Public information:	
	$P, \alpha P, x\alpha P$	
$Q_C = tr(ID_C)$ (PK_C)		$Q_S = tr(ID_S)(PK_S)$
αQ_C (SK_C)		αQ_S (SK_S)

Figure 5.1. Setup

During the setup, system parameters including $P, \alpha P$, the public keys Q_C, Q_S and $(Q_S, x\alpha P)$ are made public.

- C chooses a random value $z \in \mathbb{Z}_q^*$ which serves as a salt and a password psw. C computes the encoding from Section 2.2 to get $R = tr(psw)$. Subsequently, C creates a message $m = \hat{e}(Q_S, zx\alpha P + \alpha Q_C) \cdot \hat{e}(zP, R)$ and a verification value $V = H(\hat{e}(Q_S, zx\alpha P + \alpha Q_C)||K)$, where $||$ denotes the concatenation of the messages. $K = H(\hat{e}(zP, R))$ serves as a key that is transferred with the server. Values Q_C , m, V and zP are sent to server S. Value zP is the salt and stored in the server's password database. The salt is needed for S to verify the validity of $\hat{e}(zP, R)$. Authentications of the client and the message are based on the short-lived identity-based secret key αQ_C and the correctness of V.

- After receiving the registration request message (Q_C, m, V, zP) from

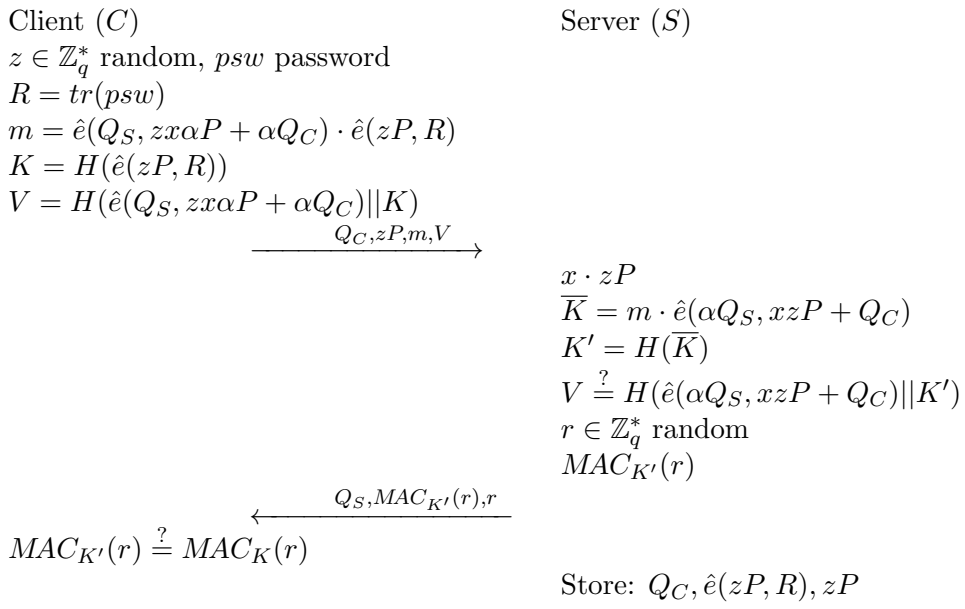


Figure 5.2. Password registration protocol

C, S computes $\overline{K} = m \cdot \hat{e}(\alpha Q_S, xzP + Q_C)$, where αQ_S is S's short-lived secret key. Then S computes the value $V' = H(\hat{e}(\alpha Q_S, xzP + Q_C)||K')$, where $K' = H(\overline{K})$ and checks whether $V = V'$ holds. If they are equal, then S is sure of the authenticity of the client and the validity of the other values $\overline{K}$ and zP. S stores $Q_C, \hat{e}(zP, R)$ and zP in the database. Thereafter S generates a random value $r \in \mathbb{Z}_q^*$ and computes a MAC value $MAC_{K'}(r)$. S sends a response $(Q_S, MAC_{K'}(r), r)$.

- C receives the S message and calculates the MAC value applying $K = H(\hat{e}(zP, R))$. If $MAC_K(r)$ is correct, then C is successfully authenticated by S, and C also confirms that server S knows $\hat{e}(zP, R)$.

In the proposed password registration protocol, the client chooses a password (psw) and the salt (z) is generated. The bilinear map - a one-way map - of the password and the salt ($\hat{e}(zP, R)$) is securely sent and stored on server side. The authenticity of message $\hat{e}(zP, R)$ and $z\alpha P$ is verified by the server as follows. The identity of the sender is verified by

calculating $\hat{e}(zP, R)$ from message m and xzP, applying secret server key αQ_S. Data integrity of the messages is verified by checking the correctness of V. The salt information (zP) is sent randomized to provide its confidentiality. Confidentiality of $\hat{e}(zP, R)$ is assured. Value $\hat{e}(Q_S, zx\alpha P + \alpha Q_C)$ randomized by x is multiplied by $\hat{e}(zP, R)$.

The client is able to confirm that the server has received and stored the correct password and salt information by checking the correctness of the $MAC_{K'}(r)$ value. In order to prevent replay attacks, value r ensures that the MAC value is fresh for every registration. Considering the time complexity, this password registration is very efficient, since there is only one bilinear map calculation on user side and one bilinear mapping on server side besides the MAC, hash operations and point multiplication by a scalar.

5.3 Security Analysis

In this section we state that our proposed scheme is a secure password registration protocol. First we review informally the security requirements.

For a secure password registration protocol basic requirements are mutual authentication of the participants, password secrecy during transmission and resistance against offline attacks. Secure mutual authentication of participants prevents adversaries to impersonate a legal user or server. During the password registration, the newly generated password is confidential, an adversary should not have any information about it. It is essential that password information should be stored on server side in a way that it should be secure against offline attacks (e.g. dictionary attack, rainbow tables). By the end of the protocol the client is able to verify that the server knows and stores the proper password information.

5.3.1 Registration Security Model

In [71] authors consider the dictionary attack resistance property, *i.e.* server learns nothing about the password in the verifier. Passive attacks are considered in which the adversary must not be able to retrieve the password from the password verifier faster than with a brute-force attack.

In their security model three oracles are listed. The `Execute(C,S)` oracle models a passive attack that executes the protocol. It returns the protocol transcript and the state of the server instance S. Oracle `Send`(C, S, m) models an active attack that sends message m from client instance C, to server instance S. It returns the server's answer m if there exists any. Oracle `Finalise(C,S,psw)` takes a client, server pair (C, S) and a password psw as input, and returns 1 iff there exists a server instance that accepted password verifier information for psw.

In [72] the adversary has access to oracles `Setup, Send, Execute and Corrupt` for interaction with the protocol participants. *Password blindness* is defined by a distinguishing experiment where the attacker, after interacting with the oracles, outputs a challenge comprising of two passwords, two clients, and a pair of servers. After a random assignment of passwords to the two clients, the adversary interacts with the oracles again and has to decide which client possesses which password.

We provide a security model that considers online attacks in addition to resistance against offline attacks. Our proposed model takes the whole registration process into account unlike [72] and [71]. We have regard to all communication messages between the client and the server as well. Hence mutual authentication of the participants and password secrecy are also studied during transmission. We define security goals for password registration protocols that consider the whole registration process assuming the minimum requirements.

Each participant holds an identity-based key pair generated by the *PKG oracle* during the setup. The final output of the registration is the password information denoted by psw_data_C that is necessary to verify the identity of C.

A registration protocol run is considered to be successful, if the participants confirm the password information. Participants' oracle instances are terminated when they finish a protocol run. An oracle can be in state accepted before it is terminated. The server is in state accepted, if it decides to store the password information psw_data_C. The client is in state accepted, if it confirms that server stores the correct psw_data_C, after receipt of properly formulated messages.

We give the definition of a registration protocol as follows. In general a protocol determines what step a participant instance should take as a

response to the adversarial message.

5.3.1 Definition. A registration protocol is a pair $P = (\Pi, \Gamma)$ of probabilistic polynomial time (in the security parameter κ) computable functions, where Π specifies how (honest) players behave and Γ generates key pairs for the participants.
Π takes as input:

κ:	the security parameter;
I:	identity of the sender;
J:	identity of the intended recipient;
pk_I, sk_I:	identity based key pair of I;
pk_J:	identity based public key of J;
$tran$:	ordered set of messages transmitted and received by I in this run of the protocol;

$\Pi(\kappa, I, J, pk_I, sk_I, pk_J, tran)$ outputs a triple (m, δ, η), where:

$m \in \{0,1\}^* \cup \{*\}$:	the next message to be sent from I to J ($*$ indicates no message is sent);
$\delta \in \{Accept, Reject, *\}$:	$C's$ current decision ($*$ indicates no decision yet reached);
$\eta \in \{psw\_data_C, *\}$:	client's password information stored by the server, ($*$ indicates no password information is stored);

Adversarial Model

We assume that the adversary is $\mathcal{A} \notin ID$, *i.e.* neither a user nor a server. $\mathcal{A}$ is a probabilistic polynomial time Turing Machine with an access to the participants' oracles, *i.e.* it has a query tape where oracle queries and their answers are written. $\mathcal{A}$ is able to relay, modify, delay or delete messages. $\mathcal{A}$ is allowed to make the following queries that model adversarial attacks.

`Send`$(\prod^i_{I,J}, M)$: This oracle query models an active attack. $\mathcal{A}$ sends the message M to oracle $\prod^i_{I,J}$ that returns a message m, which is sent by the user instance as a response to M. Besides m oracle $\prod^i_{I,J}$ also provides information whether the oracle is in state (δ) **Accepted**, **Rejected** or $*$. The query enables $\mathcal{A}$ to initiate a protocol run between participants I and J by query `Send`$(\prod^i_{I,J}, \kappa)$.

`Corrupt`$(\prod^i_{I,J})$: This oracle query models the corruption of participant I. This oracle query models an adversary hacks into the machine. Replying to this oracle query a participant oracle $\prod^i_{I,J}$ provides information about I's asymmetric secret keys and state, *i.e.* all the values calculated and stored by participant I. If I is a server, then both its secret key and psw_data_C are returned. If I is a client, the password itself and the salt are given as well.

`Reveal`$(\prod^i_{I,J})$: This models an insecure usage of a password. If oracle $\prod^i_{I,J}$ is in state accepted, holding a password psw, then this query returns psw to $\mathcal{A}$. This query models the attacks, when the adversary persuades a participant to leak the password, e.g. via a social engineering attack.

`Test`$(\prod^i_{I,J})$: This oracle query models the semantic security of the password. It is allowed to be asked only once in a protocol run. If participant I is in state accepted, then a coin b is flipped. If $b = 1$, then psw is returned to the adversary, if $b = 0$, then a random value from the distribution of the password is returned.

We define $\mathcal{A}$'s advantage, the probability that $\mathcal{A}$ can distinguish the password held by the queried oracle from a random string, as follows:

$$Adv^{\mathcal{A}}(\kappa) = |Pr[\text{guess correct}] - 1/2|.$$

`Execute`(C, S) : This oracle models a passive attack. It takes new unopened client and server instances and proceeds honest executions of the protocol. If there is a record for client C on server S then it aborts, otherwise it outputs the transcript of the protocol and S's states after the execution, *i.e.* all values including password verification information and the salt are stored.

`Finalise`(C, S, psw) : This oracle query models the verification of a password psw. Takes a client, server pair (C, S) and a password psw as input, and returns 1 iff there exists a server instance that is in state accepted and stores (C, psw_data_C), where psw_data_C is the verification data of the input psw. We assume that no `Send` was queried for (C, S). Otherwise, return 0.

An oracle is **opened** or **corrupted**, if it has answered a query $\texttt{Reveal}(\prod^i_{I,J})$ or $\texttt{Corrupt}(\prod^i_{I,J})$, respectively. We differentiate strong and weak corruption models. In the case of the weak corruption model only the asymmetric keys are transferred, the adversary does not completely compromise the machine. Other values generated and stored during the protocol run are not revealed. A model is called strong corruption model if asymmetric keys and the state (including the password) are also revealed. This is the case when the state is revealed via a malware installed on the machine. Applying these secret values the adversary is able to calculate messages that might be sent to a participant oracle with query $\texttt{Send}(\prod^i_{I,J}, M)$.

The output of the oracle `Execute` with query `Finalise` makes it possible to model dictionary-like attacks. Oracle `Finalise` models the attack, when the adversary verifies a client's password stored on server side. We emphasize that `Send` is not queried, since a passive attack is formalized. This refers to the attack when an adversary has access to the transcripts of the protocol and the password database or files. Oracle `Execute` is queried to model the generation of transcript and password data. We consider honest executions, but participants might be corrupted. While modelling offline attacks we assume that the client is weakly corrupted. We note that the success of a password extraction via an online attack is measured by oracle `Test`.

There are concurrent and non-concurrent security models. The concurrent model assumes that several copies of the protocol can be processed concurrently, *i.e.* several instances of the same participant can be active simultaneously. For the non-concurrent model at most one participant instance can be active per participant.

During the attack an experiment of running a protocol $P = (\Pi, \Gamma)$ in the presence of an adversary $\mathcal{A}$ is examined. First Γ is run to generate keys, system parameters for all participants, then $\mathcal{A}$ initializes all participant oracles and asks polynomially number of oracle queries including $\texttt{Send}(\prod^i_{I,J}, M)$, $\texttt{Reveal}(\prod^i_{I,J})$, $\texttt{Corrupt}(\prod^i_{I,J})$ to the participant oracles and queries $\texttt{Execute}(C, S)$ and $\texttt{Finalise}(C, S, psw)$. Finally $\mathcal{A}$ asks a $\texttt{Test}(\prod^i_{I,J})$ query.

In order to give the definition of a secure registration protocol, we need to review the definition of conversation and matching conversation from

[19]. They were also formalized in [14].

Matching conversation and formalizes real-time communication between entities I and J, it is necessary to define authentication property of a protocol. We refer to definitions `Matching conversation` and the event $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ in Chapter 4.2 (Section 4.2.2, 4.2.4).

We review the definition of min-entropy for a dictionary from [71]. Passwords are considered as character strings, where the distribution of characters depends on the used character set Σ, character positions and the password string itself.

5.3.2 Definition. Let D_n denote the dictionary, from which the passwords with length n are chosen. Let $\mathbb{D}_\Sigma$ denote the probability distribution in password psw of characters from a character set Σ. Min-entropy for D_n containing passwords $psw = (c_0, \ldots, c_{n-1})$ is defined as

$$\beta_{D_n} = - \max_{psw \in D_n} \sum_{i=0}^{n-1} \mathbb{D}_\Sigma(c_i) log_2 \mathbb{D}_\Sigma(c_i)$$

A password registration scheme is secure if the values stored on the server-side leak as little information as possible on the password, *i.e.* an attacker can not retrieve the password from the password verification value more efficiently than by performing a brute-force attack over the dictionary.
We need to define the benign adversary, which is the following.

5.3.3 Definition. An adversary is called *benign* if it is deterministic, and restricts its action to choosing a tuple of oracle containing one client and one server oracle, and then faithfully conveying each flow from one oracle to the other, with the client oracle beginning first.

Before proving the security of our protocol, we need to formalize what we consider a secure registration.

5.3.4 Definition. A protocol is a *secure registration protocol* if

- Online resistance:

1. In the presence of the *benign adversary* the client oracle and the server oracle communicating with the client oracle are always accepted. The server stores the password verification value confirmed by the client.

and for every adversary $\mathcal{A}$

2. If there is an uncorrupted client oracle having matching conversations with an uncorrupted server oracle, then they always accept. The server stores the password verification value confirmed by the client;
3. For uncorrupted server and client oracles the probability of $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ is negligible;
4. For the tested oracle $Adv^{\mathcal{A}}(\kappa)$ is negligible. If it is a client oracle, then it is unopened;

- Offline resistance:

5. If for all dictionaries D_n adversary $\mathcal{A}$ generates at most t tuples (C, S, psw), then

$$Pr[\texttt{Finalise}(C, S, psw) = 1] \leq \frac{t}{2^{\beta_{D_n}} \cdot t_{pre}} + \mu(\kappa),$$

where $\mu(\kappa)$ is negligible, $\frac{t}{2^{\beta_{D_n}} \cdot t_{pre}}$ denotes the probability that $\mathcal{A}$ finds psw by trying t number of (C, S, psw) tuples, β_{D_n} is the min-entropy for dictionary D_n and t_{pre} denotes the computational cost to calculate the input value of the one-way function from the password.

A function μ is *negligible* if for every positive polynomial $p(.)$ there exists an N such that for all integers $n > N$ it holds that $\mu(n) < \frac{1}{p(n)}$. In the definition above, only the necessary security assumptions are given. According to element three, we assume that participants do not disclose their secret keys in order to assure mutual authentication. To provide semantic security of the password, it is assumed that the client does not reveal the password. In the case of offline attacks, we assume that besides having access to the transcripts and the password information stored on

the server-side, the adversary gains the secret key of the server and the client. We do not assume that the server and the client are uncorrupted. We only assume that the client is unopened and does not reveal the password.

We define two security models. In the case of client-server protocols, clients are usually assumed to be malicious, *i.e.* they deviate form the steps of the protocol and they can apply any type of strategy to attack. The servers providing some service are usually considered to be honest, meaning they do not launch any attack or honest-but-curious model, *i.e.* they initiate only passive attacks, not leaving any trace of the attack. Depending on whether the server is honest or honest-but-curious, we differentiate **honest and honest-but-curious models.** In [72] and [71] an honest model is used.

5.3.2 Security Proof

Protocol security is considered in an extended random-oracle model, hash functions and the bilinear map are replaced with random mapping.

5.3.5 Theorem. *The proposed password registration protocol is resistant against online attacks in the honest-but-curious model, assuming MAC is* existentially unforgeable under an adaptive chosen-message attack*, solving the* Bilinear Diffie-Hellman problem *is computationally infeasible, moreover, the bilinear map is considered in the generic bilinear group model and the hash functions are random oracles.*

Proof: Proving conditions (1) and (2) of Definition 6. is trivial, since the steps of the protocol are followed by the client and the server and they are always accepted. The server stores the password verification value confirmed by the client. Let us consider condition (3), which holds if the assumption that the MAC is existentially unforgeable under an adaptive chosen-message attack and the Bilinear Diffie-Hellman Problem holds. Moreover the hash functions are random oracles. Let us see it in details.

Consider an adversary $\mathcal{A}$ and suppose that $Pr[\texttt{No-Matching}^{\mathcal{A}}(\kappa)]$ is non-negligible. There are two cases: either the server, or the client oracle is accepted.

- Case 1
  Let $\mathcal{A}$ `succeeds` denote the event that in $\mathcal{A}$'s experiment there is a server oracle $\prod_{S,C}$ that is *accepted*, but there is no client oracle $\prod_{C,S}$ having matching conversation to $\prod_{S,C}$.
  We assume that
  $$Pr[\mathcal{A} \texttt{ succeeds}] = n_S(\kappa),$$
  where $n_S(\kappa)$ is non-negligible. We construct a polynomial time adversary $\mathcal{F}$ that for given $a\mathbf{P}, b\mathbf{P}, c\mathbf{P}, \mathbf{P}$ calculates $BDH(a\mathbf{P},b\mathbf{P},c\mathbf{P},\mathbf{P}) = \hat{e}(\mathbf{P},\mathbf{P})^{abc}$. $\mathcal{F}$ randomly picks $C \in Client$ and $S \in Server$. Let $\Delta = \{C, S\}$ denote the identities of protocol participants. $\prod_{S,C}$ denotes the server oracle that communicates to the client C. $\mathcal{F}$ also chooses randomly a particular session $t \in \{1, \ldots, T_3(\kappa)\}$. Given security parameter κ, adversary $\mathcal{F}$ sets $par = (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \mathbb{Q}_{\mathbb{P}}, tr, \mathbf{P}, c\mathbf{P}, H, MAC)$, for calculating hash values, encodings and bilinear map $\mathcal{F}$ calls hash, tr and $\mathbb{Q}_{\mathbb{P}}$ oracles, respectively. To make the proof easier to follow let $a\mathbf{P}$ denote the value returned by oracle $\mathbb{Q}_{\mathbb{G}}$ as a result of applying group operation a times for any $a \in Z_q$ and the answer of oracle query $\mathbb{Q}_{\mathbb{G}_{\mathbb{T}}}$ for inputs c, d is denoted by $c \cdot d$. $\mathcal{F}$ also sets public keys $Q_S = a\mathbf{P}, Q_C = b\mathbf{P}$, whenever $tr(ID_S)$ or $tr(ID_C)$ are asked $\mathcal{F}$ answers $Q_S = a\mathbf{P}, Q_C = b\mathbf{P}$, respectively. $\mathcal{F}$ randomly chooses value $\bar{x} \in \mathbb{Z}_q^*$ and sets $\bar{x}c\mathbf{P}$ as a server public value. Value $\bar{x}c\mathbf{P}$ is sent to oracle PKG. $\mathcal{F}$ runs $\mathcal{A}$ and answers $\mathcal{A}$'s queries as follows.

  1. $\mathcal{F}$ answers H hash and tr encoding oracle queries at random (like a real random oracle would), except if ID_S or ID_C is asked.
  2. $\mathcal{F}$ answers `Corrupt` queries according to Π, reveals secret keys, internal states and secret values. Queries to the current server and the client oracles are refused.
  3. $\mathcal{F}$ answers `Reveal` queries as specified in Π. This query is refused if it is asked from $\prod_{S,C}$, since $\prod_{S,C}$ does not hold the password.
  4. If $\mathcal{A}$ does not involve $\prod_{C,S}$ as a client oracle which communicates to the server oracle $\prod_{S,C}$, then $\mathcal{F}$ gives up. If $\mathcal{A}$

does not invoke $\prod_{C,S}$ as an initiator oracle, then $\mathcal{F}$ gives up. Otherwise $\mathcal{A}$ generates a password psw and a random $\bar{t}$ value, and calculates $z\mathbf{P}$ and $\mathbf{R}$. Eventually $\mathcal{A}$ asks bilinear map oracle queries $\mathbb{Q}_{\mathbb{P}}(.)$ to get $\hat{e}(Q_S, z\bar{x}c\mathbf{P} + cQ_C)$ and $\hat{e}(z\mathbf{P}, \mathbf{R})$, $\mathcal{F}$ answers these queries. $\mathcal{A}$ calculates m, where $m = \hat{e}(Q_S, z\bar{x}c\mathbf{P} + cQ_C) \cdot \hat{e}(z\mathbf{P}, \mathbf{R})$. $\mathcal{A}$ asks hash oracle query $H(.)$ of $\hat{e}(z\mathbf{P}, \mathbf{R})$ from $\mathcal{F}$ to get key K. If K is a previously used value, then $\mathcal{F}$ gives up. $\mathcal{A}$ asks hash oracle query $H(.)$ to get V, where $V = H(\hat{e}(Q_S, z\bar{x}c\mathbf{P} + cQ_C)||K)$. $\mathcal{F}$ answers the hash query and if V is a previously used value, then $\mathcal{F}$ gives up. $\mathcal{A}$ asks query $\texttt{Send}(\prod_{S,C}, M)$, where

$$M = Q_C||z\mathbf{P}||m||V.$$

Since $\mathcal{F}$ knows $\hat{e}(z\mathbf{P}, \mathbf{R})$, $\mathcal{F}$ calculates K and answers $Q_S||$ $MAC_K(r)||r$, where r is random. If some later time $\mathcal{A}$ does not ask the $\texttt{Send}(\prod_{C,S}, Q_S||\ MAC_K(r)||r)$, then $\mathcal{F}$ gives up, otherwise $\prod_{S,C}$ gets accepted. $\mathcal{F}$ calculates and outputs $\hat{e}(Q_S, z\bar{x}c\mathbf{P} + cQ_C) \cdot \hat{e}(Q_S, c\mathbf{P})^{-z\bar{x}}) = \hat{e}(\mathbf{P}, \mathbf{P})^{abc}$.

5. $\mathcal{F}$ answers `Execute` and `Finalise` queries as specified in Π. Query `Finalise` is refused if `Send` was queried before.

   Assume that $\mathcal{A}$ is successful, event $\mathcal{A}$ `succeeds` happens with $n_S(\kappa)$ non-negligible probability. We show that $\mathcal{F}$ wins its experiment with non-negligible probability. For the analysis the probability that $\mathcal{F}$ chooses the correct participants Δ, session t and succeeds is:

$$\xi_1(\kappa) = \frac{n_S(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)} - \lambda(\kappa),$$

   where $\lambda(\kappa)$ denotes the probability that $\mathcal{F}$ previously calculated the flow. $\xi_1(\kappa)$ is the multiplication of probabilities which contains the $n_S(\kappa)$ probability of that $\mathcal{A}$ `succeeds`, $\frac{1}{T_1(\kappa)}, \frac{1}{T_2(\kappa)}, \frac{1}{T_3(\kappa)}$ denote the probability that the correct client, server participants and the appropriate session are chosen. The $\xi_1(\kappa)$ is non-negligible, if $n_S(\kappa)$ is non-negligible, $T_i(\kappa)$

(i=1,. . . ,3) is polynomial in κ and $\lambda(\kappa)$ is negligible. That contradicts the security assumption of the BDH problem, hence $n_S(\kappa)$ must be negligible.

- Case 2.
  Let $\mathcal{A}$ `succeeds` denote the event that in $\mathcal{A}$'s experiment there is a client oracle $\prod_{C,S}$ that is *accepted*, but there is no server oracle $\prod_{S,C}$ having matching conversation to $\prod_{C,S}$. There are two cases: either $\mathcal{F}$ is able to proceed an existential forgery against MAC under an adaptive chosen message attack, or we will show how to construct BDH problem solver $\mathcal{F}$ that uses an adversary $\mathcal{A}$.

  – Case 2.1
    We assume that

    $$Pr[\mathcal{A} \texttt{ succeeds}] = n_{C_{2_1}}(\kappa),$$

    where $n_{C_{2_1}}(\kappa)$ is non-negligible. We construct a polynomial time adversary $\mathcal{F}$ that is able to proceed an existential forgery against MAC under an adaptive chosen message attack. $\mathcal{F}$'s task is to generate a valid (m, t) message-tag pair, where m was never asked from the oracle $MAC_K(.)$. $\mathcal{F}$ picks the protocol participants and a session $t \in \{1, \ldots, T_3(\kappa)\}$, let $\Delta = \{C, S\}$ denote identities. Let $\prod_{C,S}$ denote the client and $\prod_{S,C}$ the server oracle. $\mathcal{F}$ sets $par = (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \mathbb{Q}_{\mathbb{P}}, tr, P, \alpha P, H, MAC)$, where α chosen randomly. To make the proof easier to follow let $a\mathbf{P}$ denote the value returned by oracle $\mathbb{Q}_{\mathbb{G}}$ as a result of applying group operation a times for any $a \in Z_q$ and the answer of oracle query $\mathbb{Q}_{\mathbb{G}_{\mathbb{T}}}$ for inputs c, d is denoted by $c \cdot d$. The key generation Γ is simulated as follows. $\mathcal{F}$ sets public keys as $Q_S = tr(ID_S), Q_C = tr(ID_S)$, and $\alpha tr(ID_S)$ or $\alpha tr(ID_C)$, respectively. $\mathcal{F}$ answers $\mathcal{A}$'s oracle queries as follows.

    1. $\mathcal{F}$ answers queries to oracles $H(.), tr(.), \mathbb{Q}_{\mathbb{P}}$, `Corrupt`, `Reveal` in the same way as in Case 1.
    2. $\mathcal{F}$ answers `Send` queries according to Π with the generated random values z, s. If $\mathcal{A}$ does not involve $\prod_{S,C}$ as a server oracle which communicates to the client oracle $\prod_{C,S}$,

then $\mathcal{F}$ gives up. If $\mathcal{A}$ does not invoke $\prod_{C,S}$ as an initiator oracle, then $\mathcal{F}$ gives up, otherwise $\mathcal{A}$ asks oracle query $\texttt{Send}(\prod_{C,S}, \lambda)$. $\mathcal{F}$ responses

$$M_1 = Q_C||zP||m||V$$

with $m = \hat{e}(Q_S, zx\alpha P + \alpha Q_C) \cdot \hat{e}(zP, R)$ and $V = H_2(\hat{e}(Q_S, zx\alpha P + \alpha Q_C)||K)$, where z is random and R is generated with the tr oracle. The αQ_C secret key is calculated by $\mathcal{F}$.
If some later time $\mathcal{A}$ does not ask the $MAC_K(.)$ oracle queries, then $\mathcal{F}$ gives up. Otherwise $\mathcal{F}$ answers these queries using oracle $MAC_K(.)$. Eventually $\mathcal{A}$ creates

$$M_2 = Q_S||\bar{t}||\overline{m}$$

and calls $Send(\prod_{C,S}, M_2)$. If $\overline{m}$ was asked to oracle $MAC_K(.)$ before, then $\mathcal{F}$ gives up. If $\bar{t} \neq MAC_K(r)$, then $\mathcal{F}$ gives up, otherwise $\prod_{C,S}$ gets accepted. $\mathcal{F}$ responses $(\overline{m}, \bar{t})$ to the challenger. If $\mathcal{A}$ succeeds with non-negligible probability, then $\mathcal{F}$ outputs a valid forgery $(\overline{m}, \bar{t})$, where $\overline{m}$ was never asked to oracle $MAC_K(.)$ before.
Assume that $\mathcal{A}$ is successful, event $\mathcal{A}$ `succeeds` happens with $n_{C_{2_1}}(\kappa)$ non-negligible probability. Hence following the algorithm above $\mathcal{F}$ calculates a valid $(\bar{m}, \bar{t})$ pair. We show that $\mathcal{F}$ wins its experiment with non-negligible probability. The probability that $\mathcal{F}$ chooses correct participants Δ, session t and succeeds is

$$\xi_{2_1}(\kappa) = \frac{n_{C_{2_1}}(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)} - \lambda(\kappa),$$

where $\lambda(\kappa)$ denotes the probability that $\mathcal{F}$ previously calculated the flow. Since $n_{C_{2_1}}(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and $\lambda(\kappa)$ is negligible thus $\xi_{2_1}(\kappa)$ is non-negligible. That contradicts the security assumption of MAC, hence $n_{C_{2_1}}(\kappa)$ must be negligible.

– Case 2.2

Let $\mathcal{A}$ `succeeds` denote the event that in $\mathcal{A}$'s experiment there is a client oracle $\prod_{C,S}$ that is *accepted*, but there is no server oracle $\prod_{S,C}$ having matching conversation to $\prod_{C,S}$. We assume that

$$Pr[\mathcal{A} \texttt{ succeeds}] = n_{C_{2_2}}(\kappa),$$

where $n_{C_{2_2}}(\kappa)$ is non-negligible. In this case we can construct a polynomial time adversary that for given $\mathbf{P}, a\mathbf{P}, b\mathbf{P}, c\mathbf{P}$, calculates $\hat{e}(\mathbf{P}, \mathbf{P})^{abc}$.

$\mathcal{F}$ picks the protocol participants and a session $t \in \{1, \ldots, T_3(\kappa)\}$, let $\Delta = \{C, S\}$ denote identities. $\prod_{C,S}$ denotes the client and $\prod_{S,C}$ the server oracle. $\mathcal{F}$ sets $par = (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \mathbb{Q}_{\mathbb{P}}, tr, \mathbf{P}, c\mathbf{P}, H, MAC)$. To make the proof easier to follow let $a\mathbf{P}$ denote the value returned by oracle $\mathbb{Q}_{\mathbb{G}}$ as a result of applying group operation a times for any $a \in Z_q$ and the answer of oracle query $\mathbb{Q}_{\mathbb{G}_{\mathbb{T}}}$ for inputs c, d is denoted by $c \cdot d$. The key generation Γ is simulated similarly to Case 1., hence $Q_S = a\mathbf{P}$, $Q_C = b\mathbf{P}$ and randomly chooses values $\bar{x}, \bar{z} \in \mathbb{Z}_q^*$ and sets $\bar{x}c\mathbf{P}$ as a server public value and computes $\bar{z}\mathbf{P}$. Value $\bar{x}c\mathbf{P}$ is sent to oracle PKG. $\mathcal{F}$ answers $\mathcal{A}$'s oracle queries as follows.

$\mathcal{F}$ answers queries to oracles $H(.), tr(.), \mathbb{Q}_{\mathbb{P}}$, `Corrupt`, `Reveal` in the same way as in Case 1.

$\mathcal{F}$ answers `Send` queries as follows. If $\mathcal{A}$ does not involve $\prod_{S,C}$ as a server oracle which communicates to the client oracle $\prod_{C,S}$ or $\prod_{C,S}$ is not an initiator oracle, then $\mathcal{F}$ gives up. Otherwise $\mathcal{A}$ asks oracle query `Send`$(\prod_{C,S}, \lambda)$. $\mathcal{F}$ responses

$$M_1 = Q_C || \bar{z}\mathbf{P} || m_1 || V_1$$

with $m_1 \in \mathbb{G}_{\mathbb{T}}$ chosen randomly and V_1 is a fresh random value chosen by the random oracle as a hash value.

$\mathcal{A}$ eventually asks oracle $\mathbb{Q}_{\mathbb{P}}$ to calculate values $\hat{e}(a\mathbf{P}, \bar{z}\bar{x}c\mathbf{P} + cb\mathbf{P})$ and $\hat{e}(\bar{z}\mathbf{P}, R)$ for some random value R and the hash oracle for $\hat{e}(\bar{z}\mathbf{P}, R)$. If these oracle queries were asked before, then $\mathcal{F}$ gives up, otherwise answers the queries. $\mathcal{F}$ multiplies $\hat{e}(a\mathbf{P}, \bar{z}\bar{x}c\mathbf{P} + cb\mathbf{P})$ and $\hat{e}(\bar{z}\mathbf{P}, R)$ and verifies whether

the result is m_1. If the result is not m_1, then $\mathcal{F}$ gives up. Otherwise if some time later oracle $H(.)$ is asked for $\hat{e}(a\mathbf{P}, \bar{z}\bar{x}c\mathbf{P} + cb\mathbf{P})||H(\hat{e}(\bar{z}\mathbf{P}, R))$, then V_1 is answered. $\mathcal{A}$ generates a random value r_S and calculates $MAC_K(r_S)$ and asks query $\texttt{Send}(\prod_{C,S}, M_2)$, where

$$M_2 = Q_S||MAC_K(r_S)||r_S.$$

If M_2 is not valid or not asked, then $\mathcal{F}$ gives up, otherwise $\prod_{C,S}$ gets accepted.

Since $\mathcal{A}$ asked $\hat{e}(\bar{z}\mathbf{P}, R)$ from oracle $H(.)$, $\mathcal{F}$ is able to output $m_1 \cdot \hat{e}(\bar{z}\mathbf{P}, R)^{-1} \cdot \hat{e}(a\mathbf{P}, c\mathbf{P})^{-\bar{z}\bar{x}} = \hat{e}(\mathbf{P}, \mathbf{P})^{abc}$.

Assume that $\mathcal{A}$ is successful, event $\mathcal{A}$ `succeeds` happens with $n_{C_{2_2}}(\kappa)$ non-negligible probability. $\mathcal{F}$ outputs the solution of BDHP. We show that $\mathcal{F}$ wins its experiment with non-negligible probability. The probability that $\mathcal{F}$ chooses the correct participants Δ, session t and succeeds is:

$$\xi_3(\kappa) = \frac{n_{C_{2_2}}(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)} - \lambda(\kappa)$$

where $\lambda(\kappa)$ is the probability that the flow was already calculated before. Similarly to Case 2.1 $\xi_3(\kappa)$ is non-negligible, if $n_{C_{2_2}}(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ. That contradicts the assumption of Bilinear Diffie-Hellman, hence $n_{C_{2_2}}(\kappa)$ must be negligible.

We turn to condition (4). Consider an adversary $\mathcal{A}$ and suppose that $Adv^{\mathcal{A}}(\kappa)$ is non-negligible.

- Case 3.

  Let $\mathcal{A}$ `succeeds` against $\prod^s_{C,S}$ denote the event that $\mathcal{A}$ asks $\texttt{Test}(\prod^s_{C,S})$ query and outputs the correct bit. Hence

  $$Pr[\mathcal{A} \texttt{ succeeds}] = \frac{1}{2} + n(\kappa),$$

  where $n(\kappa)$ is non-negligible.

Let A_κ denote the event that $\mathcal{A}$ picks either a server or a client oracle $\prod_{C,S}^{s}$ and asks its `Test` query such that oracle $\prod_{C,S}^{s}$ has had a matching conversation to $\prod_{S,C}^{t}$.

$$Pr[\mathcal{A} \texttt{ succeeds}] = Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa]Pr[A_\kappa]$$

$$+Pr[\mathcal{A} \texttt{ succeeds}|\overline{A}_\kappa]Pr[\overline{A}_\kappa]$$

According to the previous section $Pr[\overline{A}_\kappa] = \mu(\kappa)$, where $\mu(\kappa) \in \{n_{C_{2_1}}(\kappa), n_{C_{2_2}}(\kappa), n_S(\kappa)\}$ and $Pr[A_\kappa] = 1 - \mu(\kappa)$, where $\mu(\kappa)$ is negligible, hence

$$\frac{1}{2} + n(\kappa) \leq Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa]Pr[A_\kappa] + \mu(\kappa)$$

and we get

$$\frac{1}{2} + n_1(\kappa) = Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa],$$

for a non-negligible $n_1(\kappa)$. We have two cases.

Let B_κ denote the event that for given $Q_C, Q_S, x\alpha P, zP, \alpha P, P, m, V$ adversary $\mathcal{A}$ asks K to oracle $H(.)$, where $K = e(zP, R)$ with $R = tr(psw)$ for password psw. This is the case of 2.2 ($n_{C_{2_2}}(\kappa)$), where we showed how to construct BDH problem solver $\mathcal{F}$ that uses an adversary $\mathcal{A}$. Moreover $\mathcal{F}$ also breaks the one-wayness of the bilinear map given in Definition 2.2.4 with $\mathcal{A}$, since R is asked from oracle $\mathbb{Q}_{\mathbb{P}}(.)$.

$$Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa] = Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge B_\kappa]Pr[B_\kappa|A_\kappa]$$

$$+Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge \overline{B}_\kappa]Pr[\overline{B}_\kappa|A_\kappa]$$

Since $Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge \overline{B}_\kappa] = \frac{1}{2}$,

$$\frac{1}{2} + n_1(\kappa) \leq Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge B_\kappa]Pr[B_\kappa|A_\kappa] + \frac{1}{2},$$

hence $Pr[B_\kappa|A_\kappa]$ is non-negligible. We construct a polynomial time adversary $\mathcal{F}$ that for given Q_C, Q_S, $x\alpha P$, zP, αP, P, m, V calculates psw.

$$\xi_4(\kappa) = \frac{n_1(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)}$$

that is non-negligible if $n_1(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and denotes the same as in Case 2.2. This contradicts to the BDHP assumption, hence $n_1(\kappa)$ and $Adv^{\mathcal{A}}(\kappa)$ must be negligible.

Let see the other case when C_κ denotes the event that $\mathcal{A}$ is able to recover K itself, and thus carries out MAC existential forgery. This is the case of 2.1. Moreover $\mathcal{A}$ also calculates psw having K and zP, *i.e.* breaks one-wayness of the bilinear map.

$$Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa] = Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge C_\kappa]Pr[C_\kappa|A_\kappa]$$

$$+Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge \overline{C}_\kappa]Pr[\overline{C}_\kappa|A_\kappa]$$

Since $Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge \overline{C}_\kappa] = \frac{1}{2}$,

$$\frac{1}{2} + n_1(\kappa) \leq Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge C_\kappa]Pr[C_\kappa|A_\kappa] + \frac{1}{2},$$

hence $Pr[C_\kappa|A_\kappa]$ is non-negligible. $\mathcal{F}$ proceeds MAC existential forgery non-negligibly and also breaks $\mathcal{F}$ one-wayness of the bilinear map with non-negligible probability.

We construct a polynomial time adversary $\mathcal{F}$ that for given Q_C, Q_S, $x\alpha P$, $zP, \alpha P, P, m, V$ calculates K and psw.

$$\xi_5(\kappa) = \frac{n_1(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)}$$

that is non-negligible if $n_1(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and denotes the same as in Case 2.1. This contradicts to the MAC or the one-way pairing assumption, hence $n_1(\kappa)$ and $Adv^{\mathcal{A}}(\kappa)$ must be negligible.

□

5.3.6 Theorem. *The proposed password registration protocol is resistant against offline attacks in the random oracle model, if the bilinear map is a one-way pairing and the client is weakly corrupted.*

Proof: Let $\mathcal{A}$ `succeeds` against $\prod_{S,C}$ denote the event that $\prod_{S,C}$ is accepted and $\mathcal{A}$ is able to output a valid (S, C, psw) tuple. Hence

$$Pr[\mathcal{A} \texttt{ succeeds}] = n_d(\kappa),$$

where $n_d(\kappa)$ is non-negligible. We construct an efficient algorithm that breaks one-wayness of the bilinear map, for given $\mathbf{P}, z\mathbf{P}, \hat{e}(z\mathbf{P}, R)$ outputs R.

We construct a polynomial time adversary $\mathcal{F}$, which picks the protocol participants $\Delta = \{C, S\}$ and a session $s \in \{1, \ldots, T_3(\kappa)\}$. $\mathcal{F}$ sets $par = (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \mathbb{Q}_{\mathbb{P}}, tr, \mathbf{P}, \alpha\mathbf{P}, H, MAC)$ and simulates the key generation Γ similarly to Case 1. of the proof of Theorem 2. $\mathcal{F}$ answers $\mathcal{A}$'s oracle queries as follows.

Adversary $\mathcal{F}$ answers $H(.)$ hash oracle query at random. For `Corrupt` query $\mathcal{F}$ answers secret keys of the participant oracles and the state of the server oracle. Adversary $\mathcal{F}$ refuses oracle queries `Reveal` and `Send`.

Adversary $\mathcal{F}$ answers to the `Execute` oracle the transcripts generated by honest executions of the protocol with the help of the secret keys and reveals password information and the salt values stored by the server oracle. After polynomial number of executions, either for the given transcript values $(\alpha Q_S, x, z\mathbf{P}, m = \hat{e}(Q_S, zx\alpha\mathbf{P} + \alpha Q_C) \cdot \hat{e}(z\mathbf{P}, R))$ or for the given password information stored by the server $(\hat{e}(z\mathbf{P}, R), z\mathbf{P})$ adversary $\mathcal{A}$ eventually generates valid (C, S, psw). Adversary $\mathcal{F}$ outputs $R = tr(psw)$. The following probability is calculated

$$\xi_6(\kappa) = \frac{n_d(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)} - \frac{t}{2^{\beta_{D_n}} \cdot t_{pre}},$$

where $\frac{t}{2^{\beta_{D_n}} \cdot t_{pre}}$ denotes the probability that $\mathcal{A}$ finds psw by trying t number of (C, S, psw) tuples, where β_{D_n} is the min-entropy for dictionary D_n and t_{pre} denotes the computational cost to calculate the input value of the bilinear map from the password. Since t is polynomially bounded

in κ and $n_d(\kappa)$ is non-negligible $\xi_6(\kappa)$ is non-negligible, that contradicts to the bilinear pairing one-wayness assumption. □

5.4 Efficiency

In order to confirm the results obtained, we implemented the protocol for performance evaluation. The implementation was created in Python 3.9, and performed on an average personal computer with an AMD Ryzen 5 2600 processor, which has 6 cores and 12 threads with a clock rate of 3.4 GHz to 3.9GHz, 16 GB of 3600MHz RAM, and an M.2 NVMe SSD with 3200 MB/s writing and 3500 MB/s reading speed. The elliptic curve and its parameters can be checked:

- The used elliptic curve: $y^2 = x^3 + x$ which is a supersingular curve over $\mathbb{Z}_p$ with $p = 731329576256467855322039941411215536384068289627312830254310277821058411810144462486413246228592183502383911176278505421042514024101864935444574549103938 7$
- In $\mathbb{Z}_q^*$ and in $\mathbb{G}, \mathbb{G}_T$ we use $q = 730750818665451459101842416358141509827966402561$

Computation Cost

Table 5.1 summarizes the numbers of the main operations on both server and client side. We can realize that the most applied operation - the hash - is also the fastest operation in the registration. We note that our registration implementation is single threaded. The reason for not implementing a multithreaded version is that the bottleneck in the implementation of the underlying computation of Tate pairings and scalar multiplications, which we did not focus in our work. However, even so the runtime of our protocol is convincing and registering multiple users at the same time can be extremely fast.

Table 5.2 shows the average execution time of the protocol's main operations. The operations run 10000 times to make the run time more accurate. The bilinear pairing is the most expensive operation, but still its run time is under 0.01 seconds.

Operation	User	Server
Hash	5	3
EC scalar mult.	3	2
Bilinear pairing	3	2

Table 5.1. Number of operations

Operation	Time
HMAC	0,0000011
EC scalar mult.	0,002880
Bilinear pairing	0,007043

Table 5.2. Execution time of protocol's operation)

Comparison with Other Schemes

The performance evaluation is based on the running time of the protocol compared to three other available solutions: the Blind Password Registration Protocol Verifier Password-Authenticated Key Exchange (BPR VPAKE), the two-server version of BPR and the Transport Layer Security (TLS) handshake. Table 5.3 shows the result of comparison. All of our tests are repeated 100 times to make sure to get a precise result. The performance tests of the BPR protocols were completed on a laptop with an Intel Core Duo P8600 at 2.40GHz. We provide the computational time for only the TLS protocol run, the registration process takes more time, since an e-mail-based verification is also needed. Our registration protocol achieves better results in term of efficiency.

Scheme	Client	Server	Full
BPR- two server	1,4 s	0,68 s	2,76 s
BPR - VPAKE	0,72 s	0,67 s	1,5 s
TLS			0,168 s
Our proposition	0,072 s	0,023s	0,095 s

Table 5.3. Execution time of protocols

Chapter 6

Scalable, Password-Based and Threshold Authentication for Smart Homes

In this chapter a threshold and password-based, distributed, mutual authenticated key agreement with key confirmation protocol described for a smart home environment. The proposed protocol is a scalable and robust scheme that forces the adversary to corrupt $l - 1$ smart home devices to perform an offline dictionary attack, where l is the threshold. The protocol was designed to achieve password-only setting, and end-to-end security if both the user and the chosen IoT devices are authenticated. We also provide a security analysis of the smart home protocol in the AVISPA framework.

The results of this chapter is contained in our paper [60]. This paper is a joint work with Andrea Huszti and Szabolcs Kovács.

6.1 Our Contribution and Literature

Wireless physical communication attackers can intercept communications more easily, channels may reveal sensitive information regarding user interaction, behavior, lifestyle or physical activity. IoT devices are often very vulnerable due to weak protection (weak or default passwords) and poor maintenance. Numerous studies have addressed the security vulnerabilities of IoT devices [73, 70, 2]. Bugs have been found in a wide range of devices, including routers [116], smart cams [106], baby monitors [18] and smart plugs [82]. However, there are many propositions and ideas which

try to fix these vulnerabilities.

6.1.1 Definition. IoT is a network of devices containing the hardware, software, firmware, and actuators which allows the devices to connect, interact, and freely exchange data and information.

Smart homes are a special use case of the IoT paradigm.

6.1.2 Definition. A smart home is a home equipped with lighting, heating, and electronic devices that can be controlled remotely on a smartphone or computer.

As the definition shows, users can control their home's electronic systems (smart bulbs, smart locks, sensors, etc.) even from remote locations by using smartphone applications. In smart homes, a large variety of devices interact over the local network using different communication technologies (e.g., Wi-Fi, Bluetooth, Zigbee, Z-Wave, LoRa, cellular network) to allow data flow between other devices [87]. Zigbee or Z-Wave IoT devices are usually connected to a smart home hub constructing a mesh network [87] and accessing LAN through the hub. IoT devices are connected directly to the home Internet router and communicate with the smart home hub via the LAN or the cloud.

Smart home systems also have other unique features that place additional design requirements. Since most devices have limited computing resources and lack or have minimal security functionalities (no authentication and plain text data transmission) several recommendations are available to handle these aspects, such as Software Defined Networking (SDN), which offers programmability, agility and centralized management. In [64] a privacy-preserving communication scheme is proposed for SDN enabled smart homes (PCSS), which aims to ensure user and smart device authentication, privacy for data and user queries. In most cases, there is a central node or a smart home hub which is responsible for managing communications with the other nodes of the local network and outside. This proposition applies lightweight cryptography (xor, hash function, AES) for the authentication between the user and the smart device using a central controller, focuses on data storage and provides privacy to the searchable encrypted user queries. Moreover, this scheme does not use the distributed properties of the smart home and addresses other issues that we consider.

Smart devices send sensitive data via a wireless connection to the central node, where immediate analysis and decision are performed to send back control commands [81], [112]. In smart homes EdgeOS_H [32] and HomePad [119] are two solutions to address edge-based IoT issues.

Recently several security vulnerabilities have been exploited in smart home hub devices. The smart home hub is an appealing target for cybercriminals as they can serve as an entry point for remote attacks. In July 2019, researchers at BlackMarble found a weakness with Zipato's ZipaMicro smart hubs where exploiting Pass-the-Hash security flaws, an adversary could open a smart lock connected to the hub [80]. ESET IoT Research has also found several vulnerabilities in three different hubs [49]. Attackers were able to gain hardcoded passwords or change passwords.

Jian Shen et.al. [109] proposed a one-to-many group authentication protocol with a group key establishment between personal digital assistance and each sensor node. They demonstrated a certificateless authentication protocol without pairings based on certificateless cryptography between PDA and application provider, using ECC algorithms. A novel security protocol is introduced by Mazhar Rathore et.al. [100], which simplifies the mutual authentication and key exchange among smart home devices. The protocol leverages Identity-Based Cryptography (IBC), thus alleviating the requirement of storing and managing public key certificates. The protocol maintains the security of the honest devices' private keys, even when the admin device is compromised. They used a secret sharing technique for the new device registration and applied bilinear pairing in the authentication phase. Formal Language Identity-Based Cryptography is provided the finegrained cryptographic access control practical in [115].

Our goal is to propose a password-based multi-device authentication scheme for smart homes to reduce security vulnerabilities.

The proposed protocol is designed typically for smart home environment (Figure 6.1). There are several IoT devices and at least one central node or edge, therefore instead of centralized authentication (e.g., Kerberos) we propose a multi-device authentication. The central node or edge is called the *device manager*. If one or more devices break down or become compromised, the system will still be able to authenticate the user in a secure way. Hence we thoroughly utilize the capabilities of these systems like *robustness* and *greater availability*. In our proposed cloud authentica-

tion scheme ([59]), we assume that the cloud servers are always available. However, the devices can be of various types in smart home systems, which means some devices are battery-powered or resource-constrained and might not be available. We need to consider this property of smart homes, and we propose a new smart home user authentication scheme with a secret sharing technique, where we require k devices to be available out of n ones, which can be chosen dynamically.

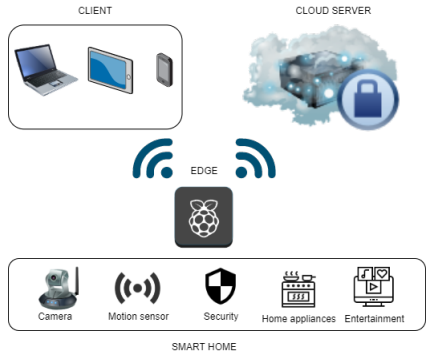


Figure 6.1. Smart Home

The scheme is an authenticated key exchange protocol with key confirmation (AKC) which takes advantage of the distributed IoT system. The client's password is shared among the smart home devices. Thus, several sensors and devices together verify the correctness of the user password. Attackers need to attack multiple devices simultaneously in order to impersonate the user successfully. Distributed storage of the passwords provides resistance against offline attacks as well. We accomplish the *password-only setting*, hence a user needs to know only a password. Since smart home devices (e.g., cameras) generate a lot of sensitive data, *confidentiality of data* needs to be ensured during the communication between the parties, and besides the identity verification of the user and the smart home a session key is also generated.

Our scheme is designed to be an AKC between the user and the device manager. However if the user would like to connect to an IoT device directly, the proposed protocol provides end-to-end security as well.

Since there are resource constrained devices, we put a great emphasis

on *efficiency* during the design. The session key is generated by Elliptic Curve Diffie-Hellman key exchange, moreover hash, MAC, xor operations are applied. The device manager and other devices use only one exponentiation.

The protocol consists of two phases: setup and authentication. During the setup phase the password is chosen by the user and split among the devices. Whenever the user logs in to the smart home system the authentication phase is run, the password given by the user is verified by the devices.

The number of devices in a smart home system is changing. The more devices there are, the higher security the system should provide, *i.e.,* the password verification should be processed by more devices. The proposed protocol is *scalable* in an efficient way. The password is not necessarily changed every time the number of devices is increased, hence the shares already set for the installed devices are not changed. For the newly registered devices new shares of the same password are set.

To provide higher security level, during the authentication phase the user chooses a random value called *authentication value*, which is securely split among the devices. For the authentication value a threshold based on the number of devices, called *authentication threshold* is set. At least threshold number of devices are necessary to construct the authentication value from its shares. The authentication threshold is greater than or equal to the password threshold. A device calculates its authentication share with the help of its symmetric key based on its password share. Consequently, the smart home system authenticates the user successfully only if the authentication value is calculated, *i.e.,* only if at least authentication threshold number of devices participate. Increasing the number of devices results in a larger authentication threshold, hence greater security level is achieved. Similarly, the number of devices can be decreased. Reregistration is required only if it goes below the threshold of the password secret sharing.

The smart home is also authenticated. The user verifies whether the devices are able to correctly calculate the password and the authentication value. Valid verification value is constructed only if the devices possess valid password shares.

Secret sharing algorithm and bilinear map are adopted to provide resis-

tance against offline attacks. To construct the password at least threshold number of shares are necessary. Let l denote the password threshold and assume that $l-1$ devices are compromitted. With the knowledge of $l-1$ password shares the adversary can launch a dictionary attack. For each possible password the authentication value should be constructed from its shares for the verification. Besides the hash, these calculations are also required for each dictionary element that slows down the attack. We apply a bilinear map for storing the hash value of the password and the salt. A hash and a bilinear map calculation together should be carried out for each possible password that also slows down the attack.

We have validated the security properties of the proposed protocol by using an automated, security protocol validation tool, named AVISPA. We show that our protocol provides mutual authentication of the participants and the generated fresh session key is kept secret. We formalize the protocol in HLPSL and also define the above security goals for the analysis. AVISPA supports three types of goal predicates:witness (for weak authentication), request (for strong authentication), and secret.

6.2 The Proposed Scheme

In this section an AKC protocol is proposed that is carried out between a user and a smart home environment including a device manager connected to the IoT devices. During the *setup phase* the user sets up his smart home system, chooses a password that is split into shares. The shares are securely stored on each device including the device manager. From the stored value a password share-based long-lived symmetric secret key K_i is calculated, where $i = 1, \ldots, n$, where n is the number of devices.

A share-based long-lived key K_i depends on a secret share of the password and the salt value. The shares are static until the password is valid. Whenever the password is changed new shares are generated. If the number of devices is increased, the user with the help of the setup module sets a new share of the same password for the new device. The stored values, hence the long-lived keys of the other IoT devices that are already installed are not modified, *i.e.,* by increasing the number of devices the password threshold is still the same.

In the authentication phase mutual authentication of the user and the

smart home system is processed. The authentication is run by the user and o randomly chosen devices, where $k \leq o \leq n$ (k is the authentication threshold, *i.e.,* at least k devices are necessary for calculating the authentication value). The authentication threshold is based on the number of devices and set by the device manager. Hence by increasing the number of devices the authentication threshold is also increased. In case the number is decreased, the authentication threshold can also be decreased. The authentication threshold is greater than or equal to the password threshold.

At the beginning of the authentication phase short-lived symmetric keys ($\overline{K}_i$) are exchanged securely between the edge and each device, moreover the user chooses a *random authentication* value that is split among all the devices. The secret shares of the authentication value are sent securely to the edge that randomly chooses o devices, where o is greater than or equal to the authentication threshold. The edge transmits the authentication shares securely to the devices, which are able to calculate the authentication share only with the password share-based long-lived keys. The o devices together calculate a value which depends on the user's password, the salt and the chosen authentication value and send it to the user. If this value is correct, devices prove the knowledge of the password share-based long-lived keys, hence the smart home system is authenticated. Moreover, the hash of this value is also transmitted to the edge. If it is correct, then the password and the salt are also valid, hence the user is authenticated as well.

In addition to mutual authentication of the participants, a secret session key is also exchanged between the user and the device manager applying Elliptic Curve Diffie-Hellman key exchange.

It is assumed that a client software is running on the client device (e.g., laptop, mobile phone etc.) that requires the password from the user to initiate the authentication process. Having the password, the share-based long-lived keys K_i are calculated with the help of a salt value stred by the client sofware and the execution of authentication begins.

6.2.1 Setup Phase

There are two participants in our protocol. One of them is the *IoT system* including the manager device and the IoT devices $(J_1, \ldots, J_n)$ and the other one is the *user* (I), which queries services and data. The set of

all binary strings of arbitrary length is denoted by $\{0,1\}^*$. If x, y are strings, then $x||y$ denotes the concatenation of x and y. Let $\oplus$ denote an exclusive or calculation. During setup, all system parameters and long-lived keys are generated. Let E denote an elliptic curve defined over a finite field $\mathbb{F}$ and $G \in E(\mathbb{F})$ be a point of order $\mathbf{n}$. Elliptic curve parameters are chosen in a way that the system resists all known attacks on the elliptic curve discrete logarithm problem in $< G >$. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of order q for some large prime q, map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be an admissible bilinear map. System parameters par are given by $par = (E, \mathbb{F}, \mathbf{n}, G, \mathbb{G}, \mathbb{G}_T, \hat{e}, H, H_0, H_1, MAC)$, where $H : \{0,1\}^* \to \{0,1\}^\nu$, $H_0 : \{0,1\}^* \to \{0,1\}^\iota$, $H_1 : \{0,1\}^* \to \{0,1\}^\sigma$ are cryptographic hash functions and ν, ι, σ are not necessarily different, ι is the size of the secret session key being exchanged. $MAC : \{0,1\}^* \to \{0,1\}^\nu$ is a MAC function. The system parameters are publicly known.

During the setup phase the user chooses a password psw, the client software generates and securely stores a random salt value z and a random polynomial for the Shamir secret sharing of psw. The secret shares s_i, where $i = 1, \ldots, n$ are generated and values $\hat{e}(P, Q_z)$, where $Q_z = H(psw||z)$ and $\hat{e}(s_i P, Q_z)$ are sent and stored by the devices. The password share-based long-lived symmetric secret keys $K_i = H(\hat{e}(s_i P, Q_z))$ are calculated for the authentication during the authentication scheme. If a user wants to set new devices to the smart home system, they need to give the password to the client software, which generates new extra shares s_i for the same polynomial, where $i > n$. This way the construction includes the property of scalability.

6.2.2 Authentication Phase

We assume that $\hat{e}(P, Q_z)$ and $\hat{e}(s_i P, Q_z)$ are stored by the devices and $K_i = H(\hat{e}(s_i P, Q_z))$ are calculated by the client software whenever the password is given. Moreover all IoT devices possess symmetric encryption keys $(\overline{K}_1, \ldots, \overline{K}_n)$ for authenticated encryption of the messages sent to the manager device. These are short-term keys and exchanged securely in the beginning of the authentication phase.

The authentication phase consists of three main subphases. The first subphase is carried out by the client software. A secret, random authentication value w is chosen and split into shares with Shamir Secret Sharing.

These shares and a hash value m_0 based on the w, the password and the salt value are transferred securely to the manager device.

During the second subphase randomly chosen smart home devices calculate their password share-based long-lived symmetric secret keys $K_i = H(\hat{e}(s_iP, Q_z))$, construct and also verify the user's knowledge of the value $\hat{e}(P, Q_z)^{w+psw}$ that is based on the password, the salt and the secret, random authentication value w.

In the third subphase a secret symmetric key is exchanged between the user and the manager device and the user verifies whether the smart home system is able to calculate $\hat{e}(P, Q_z)^{w+psw}$, *i.e.,* whether the devices possess the password shares and the salt.

First Subphase

In the first subphase (Figure 6.2), J_v denotes the manager device, which manages the authentication process on the IoT side. It communicates with the user and the other $n-1$ devices. After entering the correct password to the client software, it calculates $K_i = H(\hat{e}(s_iP, Q_z))$, where $\hat{e}(s_iP, Q_z)$ is stored securely and s_i, $i = 1, \ldots, n$ are the secret password shares.

For the Shamir Secret Sharing of a randomly chosen value w, the client software generates a random polynomial $g(x)$ with a degree large enough, where $w = g(0)$. The degree, hence also the threshold depend on the number of devices. The larger the number of devices the larger the threshold is. The threshold - the authentication threshold - is greater than or equal to the password threshold. The secret shares are calculated and denoted by w_i, where $i = 1, \ldots, n$. More details are provided in Section 2.2.1.

I creates the first message M_1 and it is sent to the manager device J_v. I computes $m_0 = H_0(\hat{e}(P, Q_z)^{psw+w})$, where $Q_z = H(psw||z)$ and also calculates $m_v = MAC_{K_v}(r_v \oplus xG \oplus J_v) \oplus w_v||r_v||xG$, where MAC is calculated with the password share-based long-lived symmetric key K_v. Moreover, m_v comprises xG, which is an elliptic curve point represented by a bitstring that is necessary for the key agreement and it is the client message of the elliptic curve Diffie-Hellman key exchange and an r_v random value. The first message also contains $m_i = MAC_{K_i}(r_i \oplus J_i) \oplus w_i||r_i$, where r_i is random and $i = 1, \ldots, n$ and $i \neq v$.

Since w is chosen randomly, for each authentication value w and the shares w_i are different. Therefore values m_i, $i = 0, \ldots, n$ are random.

Second Subphase

During the second phase (Figure 6.3) the identity of the user is verified by the smart home system. The device manager chooses o devices randomly, where $k \leq o \leq n$ and k denotes the authentication threshold. These devices - which might include the manager device as well - together authenticate the user.

The device manager also receives m_v. If the manager device is in the set of the chosen devices, then $\hat{o} = o$, *i.e., o* devices verifies value m_0, otherwise $\hat{o} = o+1$, *i.e.,* besides the chosen devices' shares the manager device's share is also applied for the calculation. Hence, the manager device calculates $w'_v = MAC_{K_v}(b \oplus c \oplus J_v) \oplus a$ and $f_v = \hat{e}(P, Q_z)^{w'_v} \hat{e}(s_v P, Q_z)$, where $m_v = a||b||c$, values $\hat{e}(P, Q_z)$, $\hat{e}(s_i P, Q_z)$ are stored by the manager device and K_v is its password share-based long lived key.

After J_v receives message M_1, it forwards each m_i along with I to

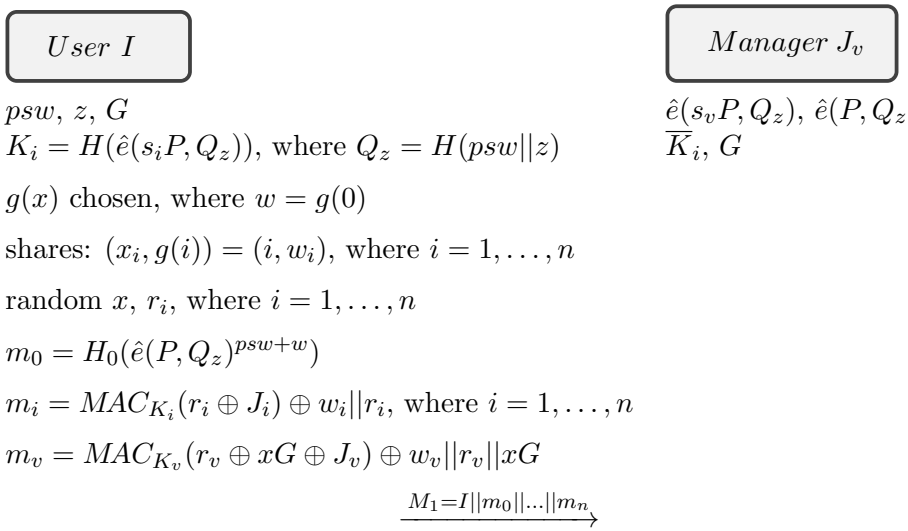


Figure 6.2. Authentication - Client process

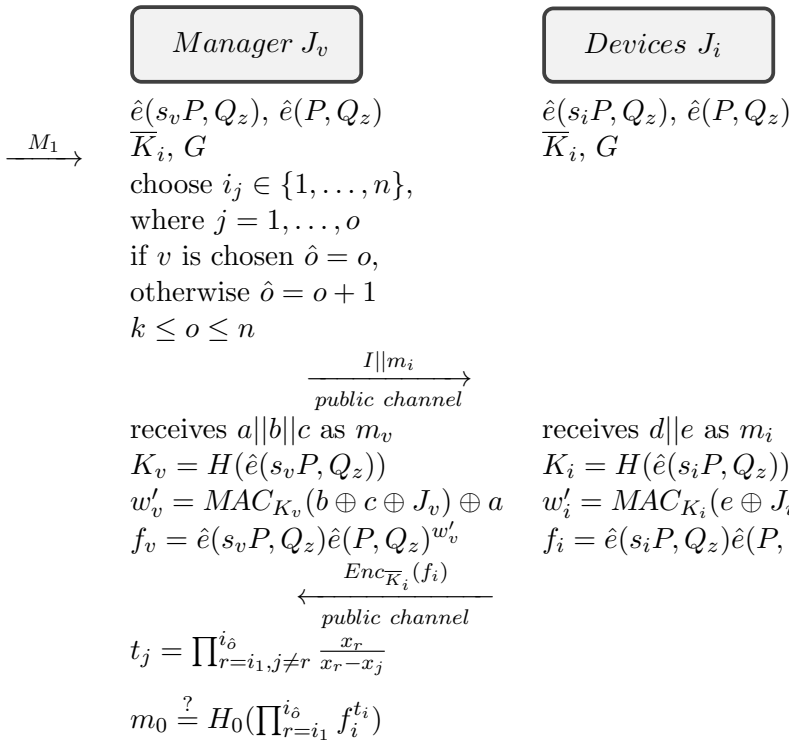


Figure 6.3. Authentication - Devices' process

devices J_{i_j}, where $i_j \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, o\}$. Each device calculates $K_i = H(\hat{e}(s_i P, Q_z))$ and $w_i' = MAC_{K_i}(e \oplus J_i) \oplus d$, where $m_i = d||e$. J_i computes and transmits $f_i = \hat{e}(P, Q_z)^{w_i'} \hat{e}(s_i P, Q_z)$ encrypted with authenticated encryption to the manager device, where values $\hat{e}(s_i P, Q_z)$ and $\hat{e}(P, Q_z)$ are stored by the device. Hence its message authentication and confidentiality are provided by the short-lived symmetric key $\overline{K}_i$. The manager device checks whether $m_0 = H_0(\prod_{r=i_1}^{i_\delta} f_i^{t_i})$ holds, where $t_j = \prod_{r=i_1, j \neq r}^{i_\delta} \frac{x_r}{x_r - x_j}$ and $\prod_{r=i_1}^{i_\delta} f_i^{t_i} = \hat{e}(P, Q_z)^{w+psw}$. If the equality holds the manager device verifies the identity of the user and the integrity of $c = xG$.

Third Subphase

Figure 6.4 demonstrates the steps of the third subphase. J_v generates a random value $y \in \mathbb{Z}_n^*$ and computes the secret session key $ssk = H_0(yxG)$. J_v calculates response $M_2 = h||yG$, where $h = H_1(ssk||yG|| \prod_{i=i_1}^{i_\delta} f_i^{t_i})$ and yG is the manager device message in the EC Diffie-Hellman key exchange.

I receives $M_2 = h||yG$ and calculates the secret session key $ssk' = H_0(yxG)$ and $h' = H_1(ssk'||yG||\hat{e}(P, Q_z)^{w+psw})$. If $h = h'$, then I confirms that manager device J_v knows the secret session key, and the IoT devices

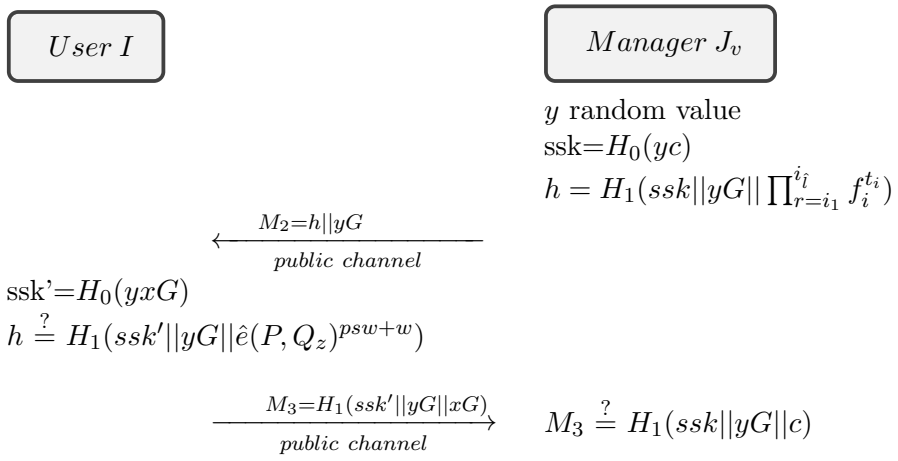


Figure 6.4. Authentication - Final process

possess the password share-based long-lived secret keys, hence they are authenticated.

Eventually, $M_3 = H_1(ssk'||yG||xG)$ is computed and forwarded to J_v. J_v verifies the message received from the client and if it is correct, J_v confirms that I knows the secret session key.

In the authentication phase, the random value $\hat{e}(P, Q_z)^{w+psw}$ can be calculated by the manager device only if the devices possess the password share-based long-lived symmetric keys, hence the user is able to verify the identity of multiple devices just by checking h. On the other hand, m_0 is checked by calculating $\hat{e}(P, Q_z)^{psw+w}$ involving keys K_i. Correct m_0 proves that the user knows psw and the client software accesses the salt value z, hence the identity of the user is verified as well. Value r_i ensures that the MAC value m_i is fresh for every authentication in order to avoid replay attack. Basically, the secret session key is created via an authenticated key agreement protocol based on random values (x, y) generated by the user and the edge. These values are sent securely so the attacker is not able to gain any information about them.

The authentication phase was also proved efficient in terms of time complexity since there are only symmetric encryptions and one exponentiation per device. Both sides use hash, MAC and xor operations, which are also fast operations. Our protocol can be considered robust, because l out of n IoT devices are required for authentication. If a device is not available, the manager device will not choose it.

6.3 Security Analysis

In this section a detailed security analysis of the proposed AKC protocol is provided. One of the indispensable security requirements is the mutual authentication of the participants. Secure mutual authentication of participants prevents adversaries from impersonating a legal user or the device manager and gaining illegal access to sensitive data.

Another security goal is key secrecy, *i.e.,* an adversary should not possess any information about the new key. During a protocol run, a new randomly chosen session key should be exchanged between the participants, a protocol execution could not be successfully completed with an

old key exchanged before. At the end, parties should be able to verify that the other party knows and is able to use the new session key. We also consider known-key security and forward secrecy properties. Known-key security preserves the security of session keys after disclosure of a session key. Disclosure of a session key should not jeopardize the security of other session keys. Forward secrecy holds if long-term secrets of one or more entities are compromised and the secrecy of previous session keys is not affected.

Formal methods have been proved to be a good choice for uncovering flaws of incorrectly designed security protocols.

6.3.1 Security Validation

In this part we analyse the security properties of our protocol. Our main goal besides mutual authentication of participants is providing that at the end of the protocol the attacker is not able to gain any information about the exchanged new session key. We formalize the protocol, the security goals and the full version of them can be checked in the appendix (Chapter 7). The following part shows the user's role, the user's steps in the protocol formalized in AVISPA. We apply the OFMC and CL-AtSe and execute the attacker simulation. The results of the security analysis show that the attacker cannot impersonate the legal participants or get the session key.

init
State:=0
transition
1.State=0 /\ RCV(start) =—>
State':=2 /\ R1':=new() /\ R2':=new() /\ X':=new() /
W1':=new() /\W2':=new()/
MAC1':= H(Kac.R1') / MAC2':=H(Kab.R2'.exp(G,X'))/
W':=exp(exp(exp(P,Psw),W1'),W2')/
M1':=H(W')/
SND(xor(MAC2',W2').R2'.exp(G,X').H(W')) /\ SND(xor(MAC1',W1').R1') /
**
witness(U,DM,user_dm_w,W')
5.State=2 /\ RCV(exp(G,Y')) =—>
State':=7 /\ SSK':=H(exp(exp(G,Y'),X))

6.State=7 /\ RCV(HE') /\ H(W.SSK.exp(G,Y))=HE' =—>
State':=9 /\ Secret':=new() /\ M3':=H(SSK.exp(G,Y).exp(G,X)) /
secret(Secret', sec:1, {U,DM}) /\ SND(M3'.{Secret'}_SSK) /
request(U,DM,user_dm_SSK,SSK)

Let us move on to the formalization of the key secrecy and authentication goals with the AVISPA tool. We formalize the protocol in HLPSL (Appendix Figure 7.3 - 7.8). This language is based on roles: basic roles for representing each participant role, and composition roles for representing scenarios of basic roles. Each role is independent from the others, getting some initial information by parameters, communicating with the other roles by channels. The intruder is modeled using the Dolev-Yao model. In AVISPA we can apply the secret, witness and request goal facts. We use these facts to demonstrate that our protocol is secure and we verify the m_0, h, SSK values in the AVISPA model . The secret is used to show that the session key (SSK) is secret and witness and request serve to prove authentication of participants (m_0, h). In the goal section four goals for authentication and one goal for secrecy are specified:

- secrecy_of sec_1
- authentication_on user_dm_w
- authentication_on dm_user_He
- authentication_on dm_user_SSK
- authentication_on user_dm_SSK

The m_0 value contains the password and w fresh value and we try to find an attack with the authentication_on user_dm_w. The authentication_on dm_user_He goal is similar to authentication_on user_dm_w and we would like to check He (which we denote with h in the protocol) value. Finally, the authentication_on dm_user_SSK and authentication_on user_dm_SSK for the verification of the key confirmation requirement, therefore the mutual authentication is achieved in our protocol as well. In our protocol we have used witness and request for three purposes:

- The user authenticates the device manager on the value of He

- Device manager authenticates the user on the value of w
- The device manager authenticates the user and vica versa on the value of SSK. We abuse strong authentication on SSK here to express that SSK should be generated freshly (and not replayed).

We formalized these statements:

- $secret(Secret, sec\_1, \{U, DM\})$
- $witness(DM, U, dm\_user\_He, HE)$
- $witness(U, DM, user\_dm\_w, W')$
- $request(DM, U, dm\_user\_SSK, SSK)$
- $request(U, DM, user\_dm\_SSK, SSK)$

We describe an example of these statements. The goal fact $secret(Secret, sec\_1, \{U, DM\})$ declares the information "Secret" as secret shared by the agents of the set containing U and DM, moreover $sec1$ protocol id identifies the secrecy goal in the goal section. We check with this fact whether the intruder can calculate the session key SSK. The goal fact $witness(DM, U, dm\_user\_He, HE)$ is for a (weak) authentication property of the device manager DM by user U on HE. The fact declares that the device manager, who is an agent, is a witness to the information HE. The goal will be identified by the constant dm_user_He in the goal section. The goal fact $request(U, DM, user\_dm\_SSK, SSK)$ is for a strong authentication property of the device manager DM by user U on SSK, it declares that agent user U requests a check of the value SSK and this goal will be identified by the constant $user\_dm\_SSK$ in the goal section. The declaration witness represents that value h in the protocol is fresh and generated by the device manager for the user and request represents the user's acceptance of the session key (SSK) that was created by the device manager for the user. The other request and witness statements are working in a similar way. We need to require the strong authentication which prevents replay attacks. Applying the back-ends model checker (OFMC, ATSE) we check whether the intruder can execute the replay attack. Using the Dolev-Yao model the back-ends model checkers verify whether

there is any man-in-the-middle attack possible by the intruder. If any of the back-end tools finds a trace in which the request event is preceded by a witness event originated by an agent other than U, an attack will be reported.

In OFMC's case, the total number of nodes visited is 2262, while the depth is 4 piles with 5.48 seconds search time. The results of CL-AtSe protocol analysis demonstrate that the number of states analyzed is 26 out of which 15 states can be reached, the translation time is 0.07 seconds and the computation time is 0.15 seconds. The output proves that the proposed protocol is safe against active and passive attacks. The known-key and forward secrecy hold in the proposed protocol as well. The parameters xG and yG of the session key are chosen independently and randomly for every protocol run, hence the new session keys are also independent and random. The session key can be calculated by the adversary from xG and yG only if he or she computes ECCDH function. The adversary faces ECCDH assumption for the previous session keys even if long-term secret keys are compromised.

6.4 Computation and Communication Cost Analysis

Since efficiency is an important aspect, evaluation of the computation and communication cost of the protocol is essential. Overall, the session key is generated by ECDH key exchange, there are hash, MAC and xor calculations, which are considered fast operations. Moreover, the device and the device manager apply one exponentiation.

In the efficiency analysis the number of devices participating in the identity verification is an important parameter. In order to make communication secure and message size acceptable, the appropriate value has to be specified. We propose to apply around ten to fifty devices for the protocol, because the communication cost is still small and also ensures security.

- The used elliptic curve: $y^2 = x^3 + x$ which is a supersingular curve over $\mathbb{Z}_p$ with $p = 731329576256467855322039941411215536384068289627312830254310277821058411810144462486413246228592183502383911176278505421042514024101864935444574549103938 7$

- In $\mathbb{Z}_q^*$ and in $\mathbb{G}, \mathbb{G}_T$ we use
  $q = 730750818665451459101842416358141509827966402561$

6.4.1 Devices

We implemented the steps of the protocol in python program language in order to test it in a real environment. We applied and focused on three devices (PC, Raspberry PI 4, ESP32). We used a PC with the following parameters: AMD Ryzen 5 2600 - 6 cores, 12 threads, 3.4 GHz - 3.9 GHz 16 GB RAM and an M.2 NVMe SSD with 3200 MB/s writing - 3500 MB/s reading speed. The Raspberry Pi is a popular small on-board computer (SBC). In a smart home, we assume that a manager device is a more powerful device that can be matched with a Raspberry PI. Raspberry has all the software you need for basic computing. Although Raspbian is the OS officially recommended by the manufacturer, countless other operating systems are available. In our case, we apply a RasPi, which includes Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz processor. It has 2 GB RAM and a class 10 memory card. The ESP32-WROOM-32 is a high-performance, generic Wi-Fi + BT + BLE MCU module that targets a wide range of applications, from low-performance sensor networks to the most demanding tasks such as audio encoding, music streaming and MP3 decoding. The essence of the module is the ESP32-D0WDQ6 chip. The embedded chip is scalable and adaptive. There are two CPU cores that can be controlled separately and the frequency of the CPU clock can be adjusted from 80 MHz to 240 MHz.

6.4.2 Communication and Computation Costs of the Protocol

In this section, we detail the communication and computation costs of the protocol. During the identity verification, we can parallelize the operations on the IoT side. Even if we apply more devices, the time does not increase. Furthermore, this parameter denoted by o can be adjusted dynamically in our protocol.

Computation Cost

Table 6.1 summarizes the computational requirements for each device:

Operation	User device	Manager Device	IoT Devices
Exponentation	1	1	1
SHA 256	n+6	4	0
AES 128	0	o-1 or o	1
HMAC	n	1	1
EC scalar mult.	n+3	2	0

Table 6.1. Number of operations

We demonstrate the execution time of the operations in seconds (Table 6.2). The authentication protocol can be executed within 1 second even for a large number of devices. In addition to mutual authentication, this execution time also includes the session key exchange and the key confirmation. This is considered acceptable for the user.

Operation	PC	ESP32	Raspberry
Exponentation	0,0002143	0,01582	0,001685
SHA 256	0,0000002	0,00002	0,0000008
AES 128	0,0000033	0,00155	0,0000016
HMAC	0,0000011	0,0082	0,0000059
EC scalar mult.	0,002880	0,2945	0,0205

Table 6.2. Execution time of protocol's operations (second)

Figure 6.5 shows the execution time for the client depending on the number of IoT devices.

Table 6.3 shows the computational performance of the authentication phases with different devices. We have selected a threshold authentication

Threshold	Raspberry+ESP32	PC+ESP32	PC
2-5 Threshold	0,233989	0,0548115	0,0294602
3-6 Threshold	0,2544973	0,0576961	0,0323448
5-10 Threshold	0,3365273	0,0692279	0,0438766

Table 6.3. Performance evaluation of our Threshold Authentication (in seconds).

system [63] which is similar to our system. Their runtime results are

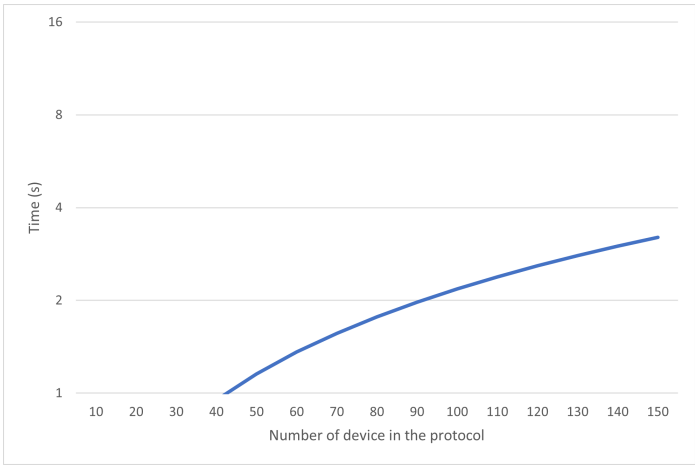


Figure 6.5. Execution time for the client with different number of sensors

compared to our ones for the different number of devices and thresholds. According to [53], about 500 devices will be connected per household in 2022, hence large number of devices and threshold should be considered. Our proposition provides a better result for $n \geq 10$ number of devices and for $o \geq 5$ threshold (Table 6.4).

Threshold	**2-5**	**3-6**	**5-10**
Işler, Küpçü - DSPP	0,00806	0,01171	0,01833
Our proposition	0,0150602	0,0150648	0,0150766

Table 6.4. Performance comparison (in seconds).

Today, achieving computing capacity and adequate security are important considerations for IoT devices. The cost of manufacturing affects the capabilities of these devices, however we need to ensure security. These aspects were also taken into account during the design of our protocol.

Communication Cost

We assume that the identifiers of participants are 32-bit. In this case 4,294,967,296 (2^{32}) unique IDs are possible. During authentication, the

hash and MAC values are 256 bits, the AES symmetric ciphertexts are 128 bits, and the elliptical curve points are 448 bits. Table 6.5 summarizes the bits sent by the client, device manager and the devices. The manager always participates in the authentication. If the manager is among the randomly chosen o devices, then $o-1$ IoT devices are chosen, otherwise o. We can see that the costs of communication and computation depend

Client	$(n+1)\cdot(32+256)+n\cdot 128+448+256$
Manager	(o-1 or o) $\cdot(32+256+128)+256+448$
IoT	(o-1 or o)$\cdot 128$

Table 6.5. communication cost of the protocol

largely on the number of devices involved in the authentication. The computation cost is negligible because the manager device and the client device can execute these operations extremely quickly. Moreover, we emphasize that the IoT devices compute in parallel and this phase does not increase significantly the execution time of the protocol even if we add more devices to the smart home. Figure 6.6 demonstrates how communication cost and data traffic are growing on the network, where the number of devices is growing fourteen times; however, the communication cost is growing ten times.

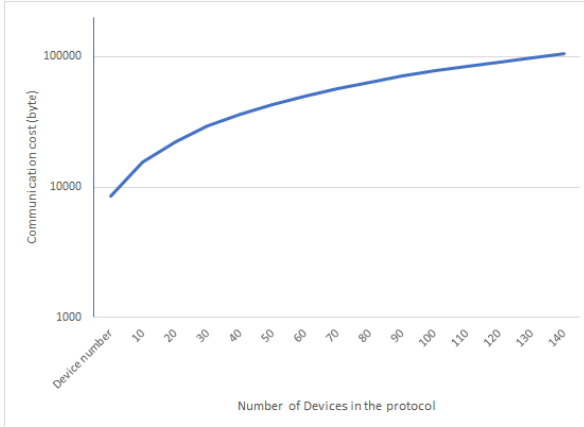


Figure 6.6. Communication cost of the protocol with different number of sensors

Chapter 7

Appendix

The appendix of Chapter 4.1. and Chapter 6.:

```
(*User process*)
let User(id:bitstring, idS:bitstring,
S1U:G,S2U:G,S3U:G,S4U:G,sskU:sskey,spkU:spkey,epkS:pkey, S_synch:exponent)=

(*Authentication 1*)
let M1=H2((Y7,H_PW)) in
let USK=H2((Y7,TT2)) in
event User_auth_start(USK);
out(c,(id,TT1,Y5,Y6,M1));
in(c,(M2:bitstring,serverm:bitstring));
let E2=H2((USK)) in
if E2=M2 then
event Server_auth_end(USK);
let M3=mac(USK,serverm) in
out(c, (id,M3));

(*Synchronization*)
event User_sync_start;
(*Merkle-tree update*)
let T1_2=exp(T1,S_synch) in
let TT1_2= H(T1_2) in
let Y1_2= H2(TT1_2) in
let Y5_2= H2((Y1_2,Y2)) in
let Y7_2= H2((Y5_2,Y6)) in

(*Authentication 2*)
let M1_2=H2((Y7_2,H_PW)) in
let USK_2=H2((Y7_2,TT4)) in
event User_auth2_start(USK_2);
out(c,(id,TT3,Y5_2,Y6,M1_2));
in(c,(M2_2:bitstring,serverm_2:bitstring));
let CheckSK=H2((USK_2)) in
if CheckSK=M2_2 then
event Server_auth2_end(USK_2);
let M3_2=mac(USK_2,serverm_2) in
out(c, (id,M3_2)).
```

Figure 7.1. User process

```
(*Server process*)
let Server(id:bitstring,idS:bitstring, y1: exponent, y2: exponent,y3: exponent,
y4: exponent, eskS:skey,epkS:pkey,spkU:spkey,S_synch:exponent)=

(*Authentication 1*)
let SK=H2((YS7,TS2)) in
in(c,(id:bitstring,TT1:bitstring,Y5:bitstring,Y6:bitstring,M1:bitstring));
let C1=H2((YS7,H_PWS)) in
if TS1=TT1 then
if YS7=H2((Y5,Y6))  then
  if C1=M1 then
        let M2=H2(SK) in
        new serverm:bitstring;
event Server_auth_start(SK);
out(c,(M2,serverm));
in(c, (id:bitstring,M3:bitstring));
let Checkmac=mac(SK,serverm) in
if M3=Checkmac then
event User_auth_end(SK);

(*Synchronization*)
event Server_sync_start;
let T1_2=exp(T1,S_synch) in
(*Merkle-tree update*)
let TS1_2=H(T1_2) in
let YS1_2= H2(TS1_2) in
let YS5_2= H2((YS1_2,YS2)) in
let YS7_2= H2((YS5_2,YS6)) in
event Sync_end;

(*Authentication 2*)
in(c,(id:bitstring,TS3_2:bitstring,Y5_2:bitstring,Y6:bitstring,M1_2:bitstring));
let CheckMessage1_2=H2((YS7_2,H_PWS)) in
if TS3=TS3_2 then
if YS7_2=H2((Y5_2,Y6))  then
 if CheckMessage1_2=M1_2 then
let SK_2=H2((YS7_2,TS4)) in
let M2_2=H2(SK_2) in
new serverm_2:bitstring;
event Server_auth2_start(SK_2);
out(c,(M2_2,serverm_2));
in(c, (id:bitstring,M3_2:bitstring));
let Checkmac_2=mac(SK_2,serverm_2) in
if M3_2=Checkmac_2 then
out(c,test);
event User_auth2_end(SK_2).
```

Figure 7.2. Server process

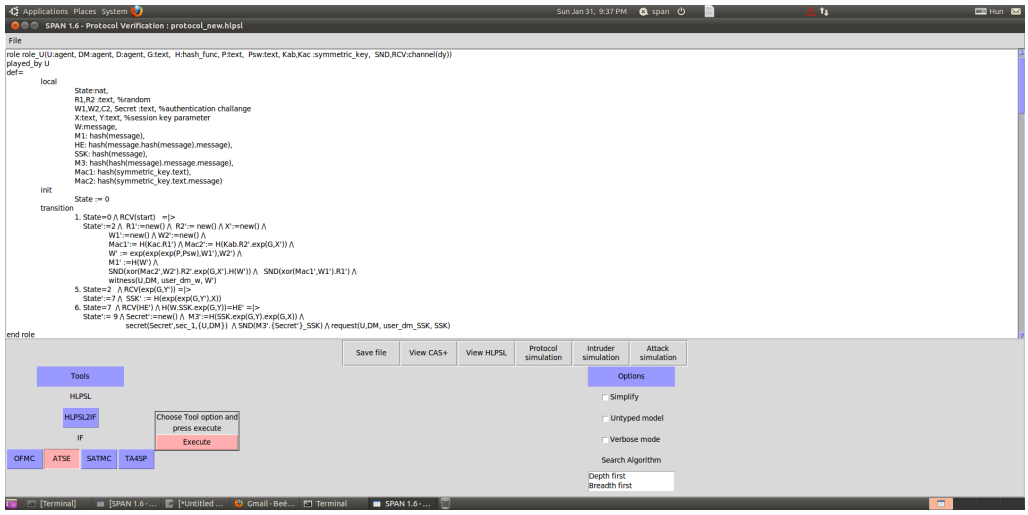


Figure 7.3. HLPSL specification of user's role

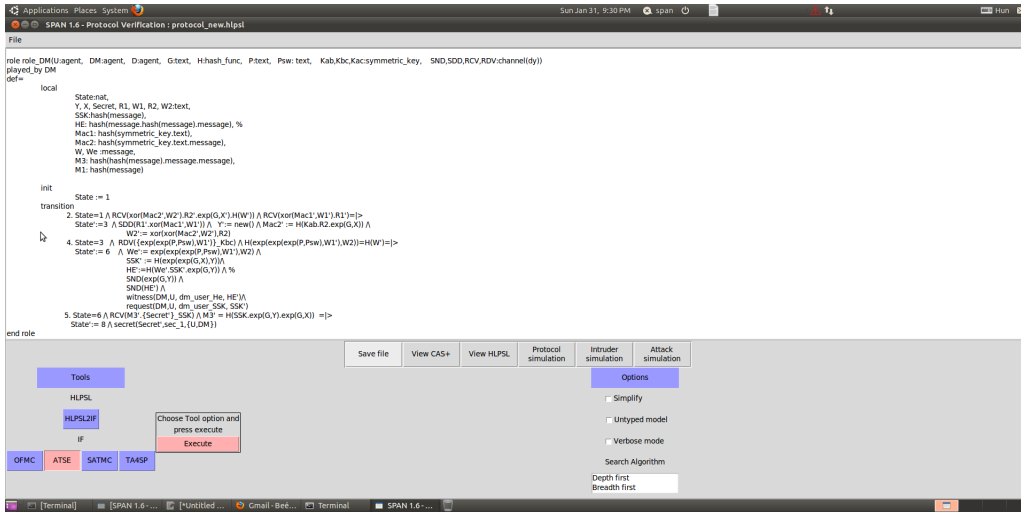


Figure 7.4. HLPSL specification of device manager's role

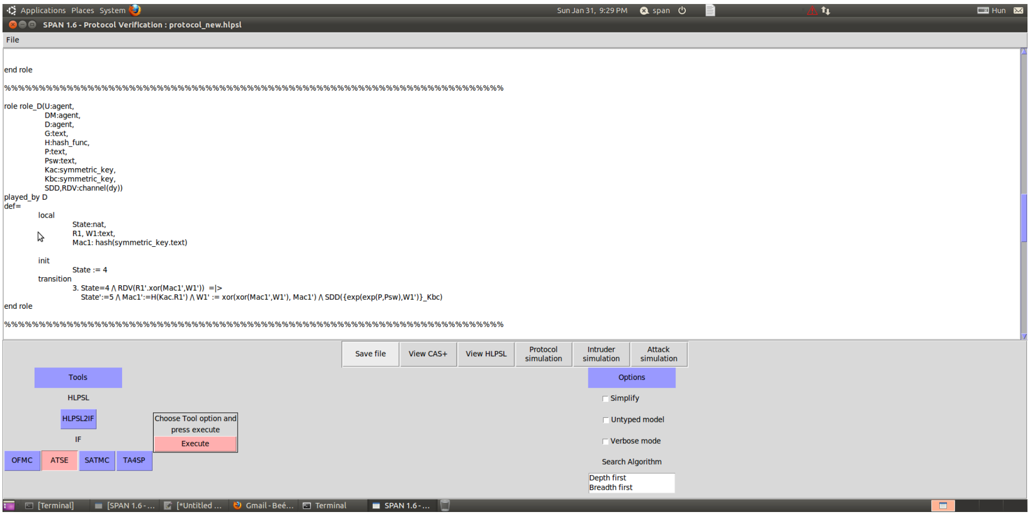


Figure 7.5. HLPSL specification of device's role

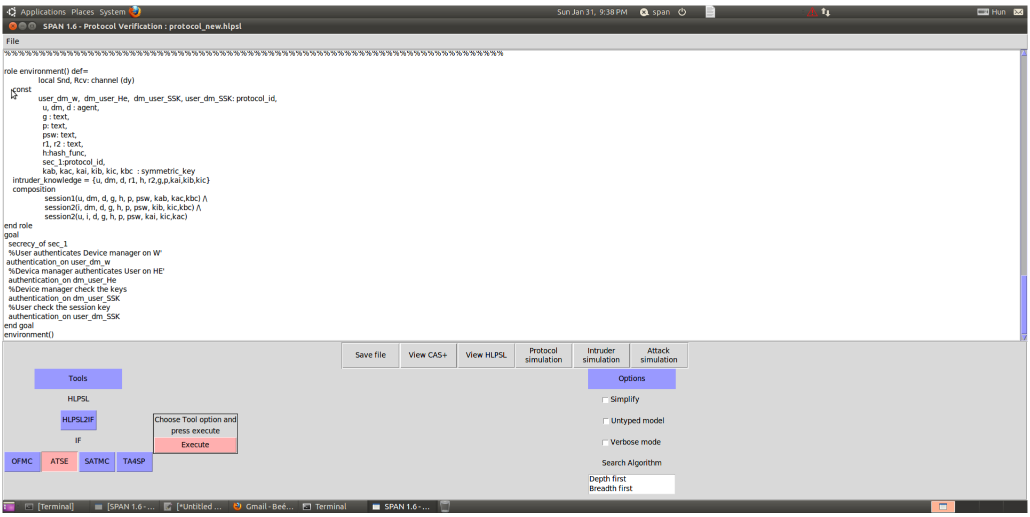


Figure 7.6. HLPSL Specification of role environment

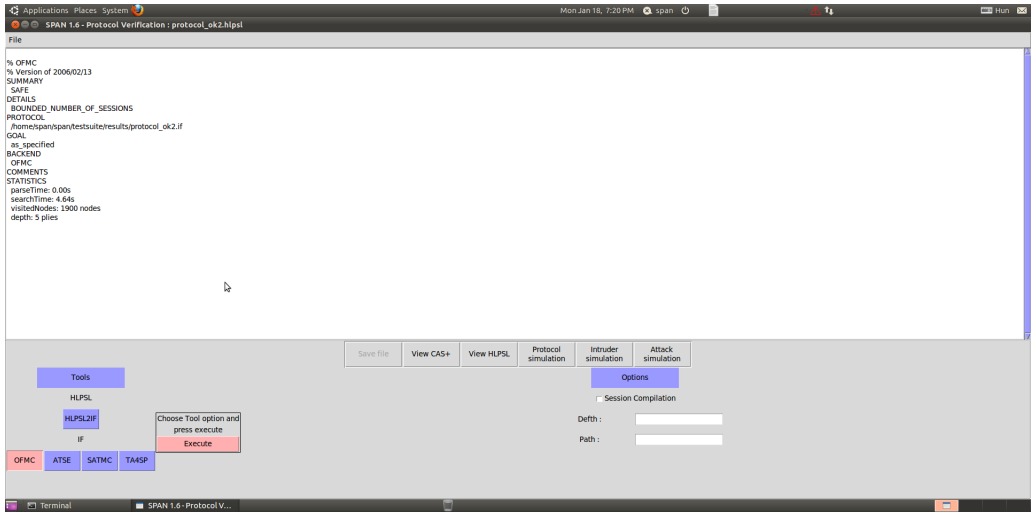


Figure 7.7. OFMC output of the proposed protocol verified in AVISPA

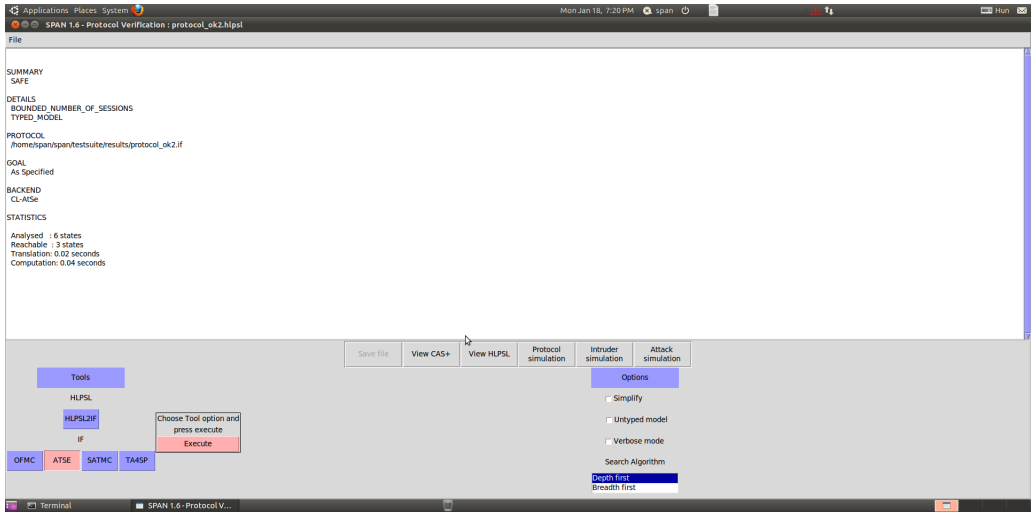


Figure 7.8. CL-AtSe output of the proposed protocol verified in AVISPA

Summary

One of the essential issues during online communication is the secure authentication between the participants. The proper authentication serves to avoid the different attacks (e.g. impersonation attack). However, in the case of improper authentication, user access control, confidentiality and integrity of user data are not provided. The authentication schemes require several security requirements, which depend on the attributes of environments. One of the most widely used authentication methods is based on short secrets like passwords. The registration process must be executed before the authentication, but it receives insufficient attention in the scientific literature.

The present dissertation demonstrates three new entity authentication schemes and a user registration protocol, which is necessary before the first identity verification. Distributed identity verification is carried out by multiple participants to secure cloud computing services and smart home environments. Via formal analysis we demonstrate that the protocols fulfil the necessary security requirements. Our solutions are more efficient than the current practical and theoretical schemes.

The first chapter contains the scientific background of the user authentication schemes and solutions.

In the second chapter, we detail the cryptographic primitives applied in our protocols and the necessary preliminaries.

Chapter 3 covers automated security analysis tools and gives the details of the concept of provable security.

In Chapter 4, two distributed authentication protocols are proposed for cloud services.

In Chapter 5, a password registration scheme is demonstrated based on the identity-based cryptography, *i.e.* both the user and the service provider are authenticated by their short-lived identity-based secret key.

In Chapter 6, we present a threshold and password-based, distributed, mutual authenticated key agreement with key confirmation protocol for a smart home environment.

Cloud Authentication Protocol Using a Merkle Tree

In Chapter 4.1, a two-factor authentication scheme for cloud computing services using a Merkle tree is demonstrated ([57]). In contrast to [33, 40] and the practical solutions, where only *one* cloud server verifies the users' authenticity, our solution applies multiple servers for user authentication. We have extended the scheme in [58] and also provided a security analysis in applied pi calculus. In our protocol, an attack can be successful only if the adversary possesses all password shares known by the servers. Comparing the efficiency of our authentication phase to the work of [33, 40], our scheme is more efficient, since only hash calculations are performed. The results of this chapter are contained in our papers ([57, 58]). These papers are joint work with Andrea Huszti.

The user is authenticated with a static and a one-time password on the service provider's side at a randomly selected server that can verify the one-time password by using a Merkle tree (Figure 8.9). A leaf of the Merkle tree is the hash of a password share, and the root element is verified in order to confirm the correctness of the whole one-time password.

The protocol has three phases: registration, authentication, and synchronization.

In the registration phase (Figure 8.10), the secret keys are exchanged generating a large amount of one-time passwords between the user and the cloud servers. Each cloud server (C_i) possesses an asymmetric key pair: $SK_{C_i} = (y_i, z_i)$, $PK_{C_i} = (g^{y_i}, g^{z_i})$, where g is a generator element of

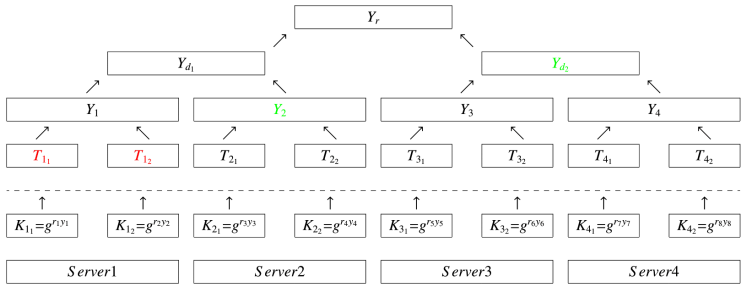


Figure 8.9. A Merkle tree with 8 leaves

a cyclic group, and $y_i, z_i \in \mathbb{Z}_q$ are random.

In the authentication phase (Figure 8.11), the mutual authentication between the user and a randomly chosen cloud server (C_v), furthermore a MAC key exchange are processed. A message authentication key (MAC) exchange is also provided to guarantee data origin integrity for the latter interactive communication.

After authentication, a synchronization step (Figure 8.12) follows and the password of the selected server is updated with the path associated with the tree.

For security analysis, the protocol is formatted in ProVerif and the results of the ProVerif test show that the specified security criteria are met. We analyse this protocol as a key exchange scheme hence the typical security requirements for mutual entity authentication schemes, and also the key related requirements are considered. We formalise the protocol in ProVerif and prove four properties:

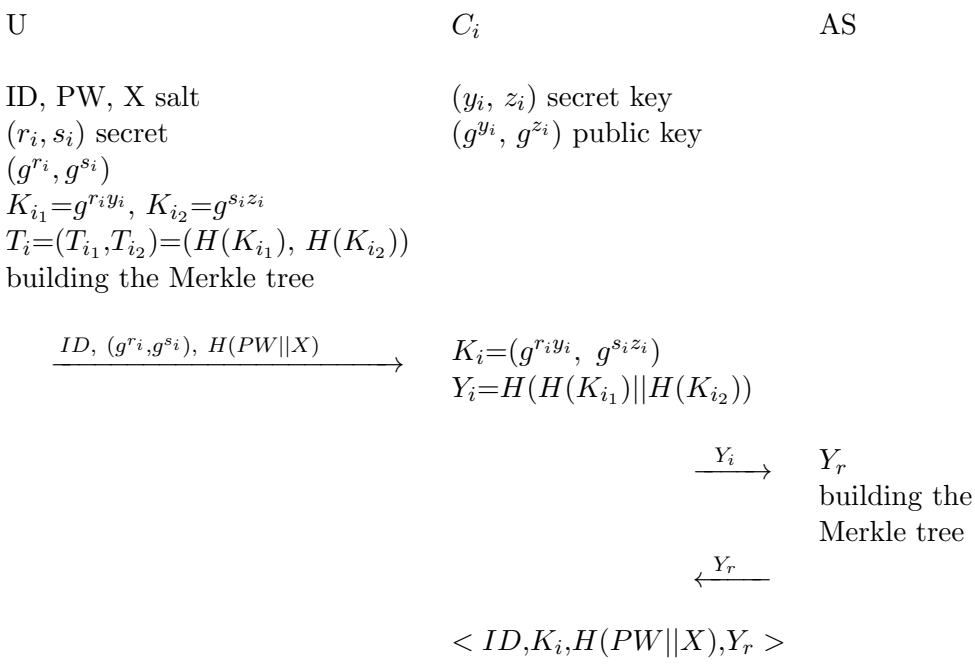


Figure 8.10. Registration

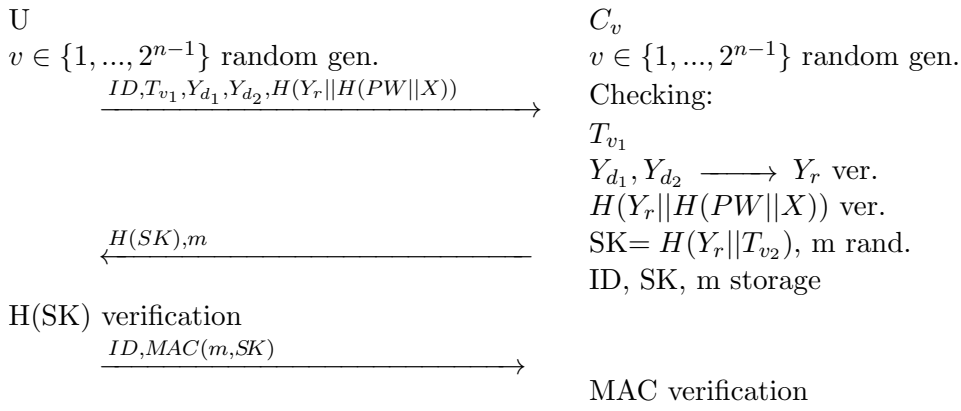


Figure 8.11. Authentication

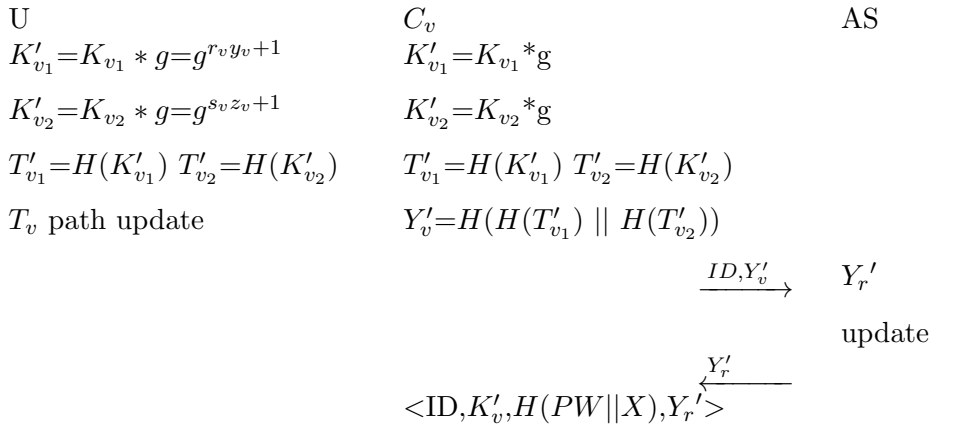


Figure 8.12. Synchronization

1. Authentication of both parties
   (a) Authentication of users: Adversaries must not be able to impersonate a legal user and achieve illegal access to the user data.
   (b) Authentication of the server: Adversaries must not be able to impersonate a legal cloud server.
2. Secrecy of the MAC key: During the key exchange the newly generated key is a confidential datum and an adversary must not have any information about the new key.
3. Key freshness: During a protocol run a new, randomly chosen key must be exchanged so that a protocol execution could not be successfully finished with an old, already used key exchanged.
4. Both parties must verify that the other party knows and is able to use the new MAC key.

We apply injective correspondences for the security analysis of the user and server authentication. All the queries return with the value true, which means that user and server authentications and key secrecy hold in our model and ProVerif do not find an attack. Assuming a successful mutual authentication, *key freshness and key confirmation* hold, as well.

Scalable Distributed Authentication for Cloud Services

In Chapter 4.2, we propose a multi-server password-based authenticated key exchange scheme (Figures 8.13 - 8.15). In contrast to other threshold password-based protocols applying secret-sharing algorithms ([9, 16, 27, 28, 99, 65, 63, 86, 68]), even if we share the password information among the servers, it is not reconstructed from the shares to verify it. To show that the proposed protocol is provably secure, we introduce the threshold hybrid corruption model. Unlike [27, 47] we provide a detailed security analysis based on the Bellare and Rogaway model. Compared to other schemes, we also consider the scalability property, which is one of the main requirements for clouds. We demonstrate a new way of generating a strong secret (e.g. long-lived key) from a password, which

is also suitable for scalability. In the IoT environment, an authenticated key exchange (AKE) protocol is presented ([105]) on wireless sensor networks. They focus on the key shares and establish the authenticated key between Wireless sensor networks and the cloud server, which performs a centralized authentication. Another variant of AKE is demonstrated in [117] which includes a permanent Control Server and cloud servers on 5G network. Our solution differs from these papers ([117, 105]) since we can scale the generated long-lived keys on the user's and the provider's sides as well. Compared to our earlier proposed protocol ([58]), we use secret splitting technique and we also achieve the scalable property. The results of this chapter are contained in our paper ([59]). This paper is a joint work with Andrea Huszti.

Our protocol results in a session key, which provides the confidentiality of the subsequent messages between the participants. The protocol has two phases. During registration, the client sets password-based long-lived keys with all the n servers. We propose a simple solution in which the client accesses the long-lived keys by using a password. We assume that a client software is running on the client device (e.g. smartcard, mobile phone etc.) that requires a password from the user to initiate the authentication process. After the client gives the password, the client software generates the long-lived keys and the execution of authentication begins. The correctness of the password is verified by the servers and not by the client software, hence a client device does not store any information about the password. The client randomly chooses k servers out of n servers for authentication. During authentication, a server is only able to calculate the challenge value w given by the client with the knowledge of the symmetric, long-lived key K_i ($i \in \{1, \ldots, k\}$) which is generated from the client password. *KKDF* denotes a Keyed Key Derivation Function that for a message m and a *key* generates a secret key K. The authentication server (J_v) authenticates the client by verifying the correctness of all the k challenge values received from the participating servers.

In our proposed protocol, servers communicate on secure channels. We prefer one randomly chosen server that communicates with the client, hence the client does not need to communicate with all the k servers in parallel and build secure channels. During the design of the protocol, the efficiency of authentication is ensured by MAC and other fast cryp-

tographic algorithms (hash, xor operation, symmetric encryption). The protocol is provably secure and the necessary adversary model and the formal proof are given. We assume that $\mathcal{A}$ is allowed to make the `Send, Reveal, Corrupt, Test` queries.

We apply distributed authentication, thus we extend the model with the concept of threshold hybrid corruption. We assume that the participants can be corrupt in our proposition. A model is a *strong corruption model* ([13]) if long-lived keys $K_{I,J}$ and all the values stored by the participant I (e.g. randomly chosen secret values) are transferred to $\mathcal{A}$ during the protocol run. In the case of the *weak corruption model*, only the long-lived keys $K_{I,J}$ are transferred or replaced, the adversary does not completely compromise the machine. Other values generated and stored during the protocol run are not revealed. We introduce a new threshold hybrid corruption model.

8.0.1 Definition. We call a model *threshold hybrid corruption model* if the client is uncorrupted, there are at least $n - k + 1$ uncorrupted servers

I J_v

$(K_1, \ldots, K_k)$, G K_v, G

$K_i = KKDF_{key}^{c+i}(psw)$, where $key = H(salt||psw)$

$K_n = KKDF_{key}(psw) \oplus \cdots \oplus KKDF^{c+n-2}{}_{key}(psw)$

$t_1, \ldots, t_{k-1}, t_v$; $r_1, \ldots, r_{k-1}$, r_v, x random

$w_1 = H(t_1), \ldots, w_v = H(t_v)$

$w = H(w_1|| \ldots ||w_{k-1}||w_v)$

$m_0 = H(w)$

$m_i = (MAC_{K_i}(r_i \oplus J_i) \oplus w_i)||r_i$

$m_v = (MAC_{K_v}(r_v \oplus xG \oplus J_v) \oplus w_v)||r_v||xG$

$$\xrightarrow[\text{public channel}]{M_1 = I||J_1||...||J_k||m_0||...||m_k}$$

Figure 8.13. Authentication - Client process

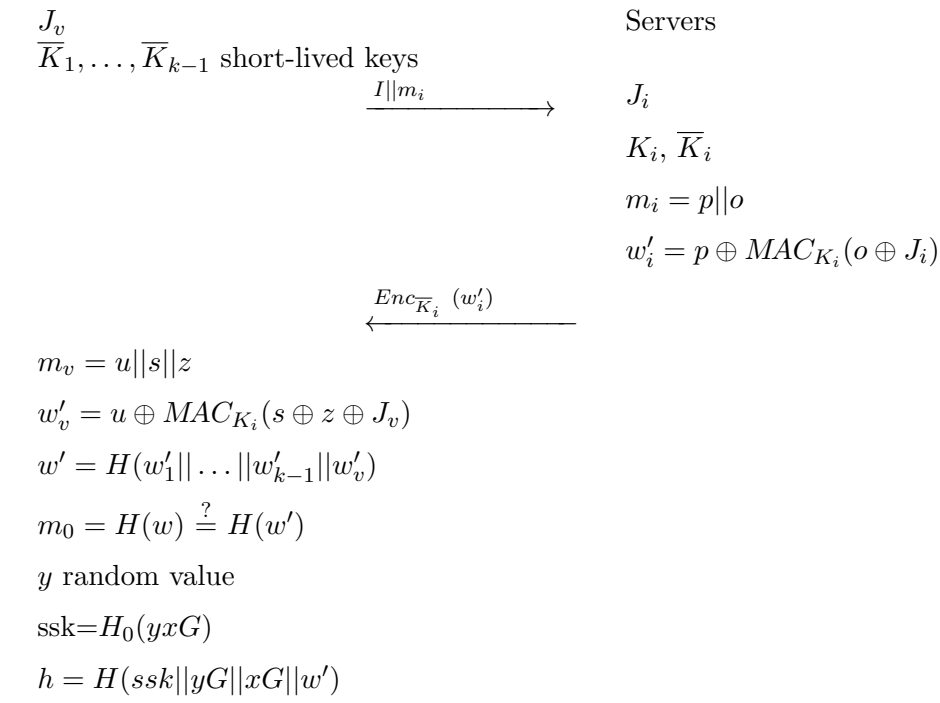


Figure 8.14. Authentication - Cloud servers communication

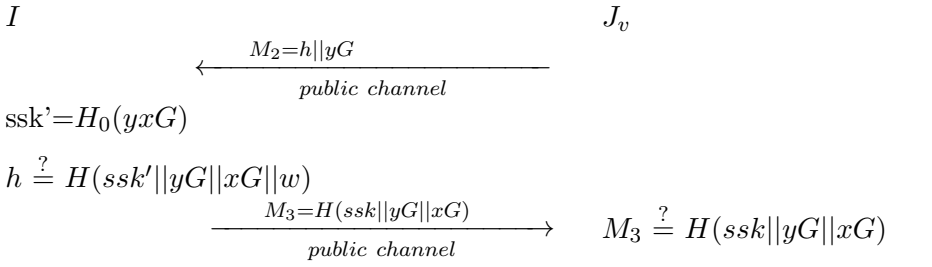


Figure 8.15. Authentication - Final process

out of the n servers and k servers are chosen randomly for authenticated key exchange with key confirmation protocol (AKC). Moreover, the server chosen to communicate with the client is

1. uncorrupted, or
2. corrupted weakly and among the remaining servers there is at least one uncorrupted.

In order to give the definition of a secure AKC protocol, we need to review the definitions of conversation and matching conversation from [19].

Matching conversation formalizes real-time communication between entities I and J, it is also necessary to be specified for the authentication property of an AKC protocol. We give the definition of the event $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ that is a modified version of the definition given in [19]. We leave out the requirement that $J \in Server$ is uncorrupted. In our multi-server setting, each client communicates with a server that can be weakly corrupted if there is at least one uncorrupted server from the k servers.

8.0.2 Definition. $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ denotes an event when in a protocol P in the presence of an adversary $\mathcal{A}$ assuming a threshold hybrid corruption model, there exists

1. a client oracle $\prod_{I,J}^{s}$ which is accepted, but there is no server oracle $\prod_{J,I}^{t}$ having a matching conversation with $\prod_{I,J}^{s}$, or
2. a server oracle $\prod_{I,J}^{s}$ which is uncorrupted and accepted, but there is no client oracle $\prod_{J,I}^{t}$ having a matching conversation with $\prod_{I,J}^{s}$, or
3. a server oracle $\prod_{I,J}^{s}$ which is weakly corrupted and accepted, but there is no client or no uncorrupted server oracle having a matching conversation with $\prod_{I,J}^{s}$.

In order to give the definition of a secure AKC, it is essential to define the notion of *freshness* and redefine the *benign adversary*.

8.0.3 Definition. A $k+1$-tuple of oracles containing one client and k server oracles is fresh if in the threshold hybrid corruption model the client

oracle and the server oracle with which it has had a matching conversation are unopened. We call an oracle *fresh* if it is an element of a fresh $k+1$-tuple.

8.0.4 Definition. An adversary is called *benign* if it is deterministic, and restricts its action to choosing a $k+1$ tuple of oracles containing one client and k server oracles, and then faithfully conveying each flow from one oracle to the other, with the client oracle beginning first.

8.0.5 Definition. We introduce that a protocol is a *secure AKC protocol* if

1. In the presence of the benign adversary the client oracle and the server oracle communicating with the client oracle always accept holding the same session key ssk, and this key is distributed uniformly at random on $\{0,1\}^{\kappa}$.

and if for every adversary $\mathcal{A}$

2. If in a threshold hybrid corruption model there is a server oracle $\prod_{I,J}^{l}$ having matching conversations with a client oracle and if $\prod_{I,J}^{l}$ is weakly corrupted, $\prod_{I,J}^{l}$ has matching conversation with an uncorrupted server oracle, then the client oracle and oracle $\prod_{I,J}^{l}$ both accept and hold the same session key ssk.

3. The probability of $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ is negligible.

4. If the tested oracle is fresh, then $Adv^{\mathcal{A}}(\kappa)$ is negligible.

8.0.6 Theorem. *The proposed protocol is a secure AKC protocol in the random oracle model, assuming MAC is* universally unforgeable under an adaptive chosen-message attack *and symmetric encryption scheme is* indistinguishable under chosen plaintext attack*, moreover,* ECCDH assumption holds *in the elliptic curve group.*

Efficiency is an important aspect during the design of our protocol. In the protocol, the session key is generated by ECDH key exchange, and the other operations are hash and xor operations, which are extremely fast.

Provably Secure Identity-Based Remote Password Registration

In Chapter 5, we demonstrate the Certificate-Less Secure Blind Registration Protocol (CLS-BPR) on the Identity-Based Cryptography, *i.e.* both the user and the service provider are authenticated by their short-lived identity-based secret key. The proposed protocol suits our smart home user authentication scheme where the values of the bilinear map are stored on the IoT devices. Our cloud scheme ([58]) can also be easily modified with the proper long-lived key setting to be compatible with our registration scheme.

For secure storage of the password, a bilinear map with a salt is applied, therefore in case of an offline attack the adversary is forced to calculate a computationally expensive bilinear map for each password candidate and salt, which slows down the attack. In contrast to traditional registration schemes, our solution does not require a Transport Layer Security (TLS) channel and can also omit the associated certificate management, which can be efficiently implemented in a corporate or educational institution. According to our implementation, our protocol is more cost-effective than the TLS-based and the other blind solutions ([71, 72]). It is not necessary to manage certificates or execute costly zero knowledge (ZK) proof. Unlike other schemes ([71, 72]) besides the password hashing scheme we also consider the interactions, when the password information is sent securely. Consequently, we prove that our solution is secure against online attacks as well. We introduce the definition of a secure password registration scheme, provide an adversarial model and show that our scheme is provably secure. Our registration is flexible, which is optimal for Single Sign-On (SSO) and Kerberos, but it is also suitable for systems where different passwords must be applied for each service. The bilinear map of the password and the salt can be used as a long-lived symmetric key and applied for entity authentication or session key generation. The results of this chapter are contained in our paper ([17]). This paper is a joint work with Andrea Huszti, Csanád Bertók and Szabolcs Kovács.

The protocol consists of a Setup and a Registration phase (Figure 8.16, Figure 8.17). During the Setup, system parameters and keys are generated for the participants. Let P be a generator of $\mathbb{G}$, where $\mathbb{G}$ additive group of order q for some large prime q. Choose a random $\alpha \in \mathbb{Z}_q^*$ and generate

parameters $P, \alpha P$. The master secret key for the system is α. Identities denoted by ID_C and ID_S and public keys are derived, *i.e.* $PK_C = Q_C = tr(ID_C)$ and $PK_S = Q_S = tr(ID_S)$. Since our password hashing scheme uses bilinear pairings ($\hat{e}$) on elliptic curves, we need an efficient way to map passwords first into $\mathbb{Z}_p$, where p is a large prime, then these points of $\mathbb{Z}_p$ into a point on the curve. Let denote tr this function. The Private Key Generator calculates the participants' secret keys $SK_C = \alpha Q_C$ and $SK_S = \alpha Q_S$. In the Registration phase, the clients send their password information to the server and confirm that the server has received the verification value. The protocol meets all the necessary requirements, including password secrecy, mutual authentication and resistance against offline attacks.

We provide a security model that considers resistance against online attacks in addition to offline attacks. Our proposed model take the whole registration process into account unlike [72] and [71]. We have regard to all communication messages between the client and the server as well. Hence mutual authentication of the participants and password secrecy are also studied during transmission. Adversary $\mathcal{A}$ is allowed to make the queries that model adversarial attacks. These queries are `Send, Corrupt, Reveal, Test, Execute` and `Finalise`.

We define the security goals for password registration protocols and consider the whole registration process assuming the minimum requirements. We introduce the definition of secure registration:

8.0.7 Definition. A protocol is a *secure registration protocol* if

- Online resistance:

    1. In the presence of the *benign adversary* the client oracle and the

Client (C)	PKG	Server (S)
	$\alpha \in \mathbb{Z}_q^*$ (msk)	$x \in \mathbb{Z}_q^*$ secret key
	Public information: $P, \alpha P, x\alpha P$	
$Q_C = tr(ID_C)$ (PK_C)		$Q_S = tr(ID_S)(PK_S)$
αQ_C (SK_C)		αQ_S (SK_S)

Figure 8.16. Setup

server oracle communicating with the client oracle are always accepted. The server stores the password verification value confirmed by the client.

and for every adversary $\mathcal{A}$

2. If there is an uncorrupted client oracle having matching conversations with an uncorrupted server oracle, then they are always accepted. The server stores the password verification value confirmed by the client;
3. For uncorrupted server and client oracles the probability of $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ is negligible;
4. For the tested oracle $Adv^{\mathcal{A}}(\kappa)$ is negligible. If it is a client oracle, then it is unopened;

- Offline resistance:

5. If for all dictionaries D_n adversary $\mathcal{A}$ generates at most t tuples

Client (C)		Server (S)
$z \in \mathbb{Z}_q^*$ random, psw password		
$R = tr(psw)$		
$m = \hat{e}(Q_S, zx\alpha P + \alpha Q_C) \cdot \hat{e}(zP, R)$		
$K = H(\hat{e}(zP, R))$		
$V = H(\hat{e}(Q_S, zx\alpha P + \alpha Q_C)\|\|K)$		
	$\xrightarrow{Q_C, zP, m, V}$	
		$x \cdot zP$
		$\overline{K} = m \cdot \hat{e}(\alpha Q_S, xzP + Q_C)$
		$K' = H(\overline{K})$
		$V \stackrel{?}{=} H(\hat{e}(\alpha Q_S, xzP + Q_C)\|\|K')$
		$r \in \mathbb{Z}_q^*$ random
		$MAC_{K'}(r)$
	$\xleftarrow{Q_S, MAC_{K'}(r), r}$	
$MAC_{K'}(r) \stackrel{?}{=} MAC_K(r)$		
		Store: $Q_C, \hat{e}(zP, R), zP$

Figure 8.17. Password registration protocol

(C, S, psw), then

$$Pr[\texttt{Finalise}(C, S, psw) = 1] \leq \frac{t}{2^{\beta_{D_n}} \cdot t_{pre}} + \mu(\kappa),$$

where $\mu(\kappa)$ is negligible, $\frac{t}{2^{\beta_{D_n}} \cdot t_{pre}}$ denotes the probability that $\mathcal{A}$ finds psw by trying t number of (C, S, psw) tuples, β_{D_n} is the min-entropy for dictionary D_n and t_{pre} denotes the computational cost to calculate the input value of the one-way function from the password.

Protocol security is considered in the random-oracle model, the hash functions and the bilinear map are supposed as random oracles. We define two security models. In the case of client-server protocols, clients usually are assumed to be malicious, *i.e.* they deviate form the steps of the protocol, they apply any type of strategy to attack. The servers providing some service are usually considered to be honest, meaning they do not launch any attack or honest-but-curious, *i.e.* they initiate only passive attacks, not leaving any trace of the attack. Depending on whether the server is honest or honest-but-curious, we differentiate **honest and honest-but-curious models.** In [72] and [71] honest models are used.

8.0.8 Theorem. *The proposed password registration protocol is resistant against online attacks in the honest-but-curious model, assuming MAC is* existentially unforgeable under an adaptive chosen-message attack*, solving the* Bilinear Diffie-Hellman problem *is computationally infeasible, moreover, the bilinear map is considered in the generic bilinear group model and the hash functions are random oracles.*

8.0.9 Theorem. *The proposed password registration protocol is resistant against offline attacks in the random oracle model if the bilinear map is a one-way pairing and the client is weakly corrupted.*

Our registration protocol achieves better results in term of efficiency and Table 8.1 demonstrates this comparison.

Scheme	Client	Server	Full
BPR- two server	1,4 s	0,68 s	2,76 s
BPR - VPAKE	0,72 s	0,67 s	1,5 s
TLS			0,168 s
Our proposition	0,072 s	0,023s	0,095 s

Table 8.1. The execution time of the protocols

Scalable, Password-Based and Threshold Authentication for Smart Homes

In Chapter 6, a threshold and password-based, distributed, mutual authenticated key agreement with key confirmation protocol for a smart home environment is presented. In our proposed cloud authentication scheme ([59]), we assume that the cloud servers are always available. However, the devices can be of various types in smart home systems, which means some devices are battery-powered or resource-constrained and might not be available. We need to consider this property of smart homes, and we propose a new smart home user authentication scheme with a secret sharing technique, where we require k devices to be available out of n ones, which can be chosen dynamically. The results of this chapter are contained in our paper ([60]). This paper is a joint work with Andrea Huszti and Szabolcs Kovács.

The protocol is designed to achieve the password-only setting, and end-to-end security if the chosen IoT devices are also authenticated besides the user. The proposed protocol is a scalable and robust scheme, which forces the adversary to corrupt $k-1$ smart home devices, where k is the threshold, in order to perform an offline dictionary attack. In the scientific literature, a threshold Password-Protected Secret Sharing (PPSS) scheme was formalized by Bagherzandi et. al. ([9]). Jarecki et. al. ([65]) present the first round-optimal PPSS scheme, requiring just one message from user to server and from server to user, and prove its security in the challenging password-only setting where users do not have access to an authenticated public key. However, it is not scalable. These recommendations ([63, 62]) considered similar properties to our proposition (scalability, robustness,

password usage, etc.). However, these are more suitable in the cloud environment and their protocols contain storage providers. Our solution is recommended typically for a smart home environment and provided a better result for $n \geq 10$ number of devices and for $o \geq 5$ thresholds.

There are two participants in our protocol. One of them is the *IoT system* including the manager device and the IoT devices $(J_1, \ldots, J_n)$ and the other one is the *user* (I), who queries services and data. We apply secret sharing where we use a (k, n) threshold scheme. A secret S can be divided into n shares in a way that $k \leq n$ will be the threshold number of the shares which we need to be able to compute S. Thus $k-1$ or fewer shares leave S completely undetermined. We apply Shamir's secret sharing threshold scheme for the IoT devices to construct the password.

During the setup phase, the user chooses a password psw, the client software generates and securely stores a random salt value z and a random polynomial for the Shamir secret sharing of psw. The secret shares s_i, where $i = 1, \ldots, n$ are generated and values $\hat{e}(P, Q_z)$ and $\hat{e}(s_i P, Q_z)$ are sent and stored by the devices, where $\hat{e}(,)$ denotes the bilinear map and $Q_z = H(psw||z)$. The password share-based long-lived symmetric secret keys $K_i = H(\hat{e}(s_i P, Q_z))$ are calculated for the authentication during the authentication phase. If a user wants to set new devices to the smart home system, they need to give the password to the client software, which generates new extra shares s_i for the same polynomial, where $i > n$. This way the construction includes the property of scalability. Let E denotes an elliptic curve defined over a finite field $\mathbb{F}$ and $G \in E(\mathbb{F})$ be a generator element. Each IoT devices possess symmetric encryption keys $(\overline{K}_1, \ldots, \overline{K}_n)$ for authenticated encryption of the messages sent to the manager device.

The authentication phase consists of three main subphases. The first subphase (Figure 8.18) is carried out by the client software. A secret, random authentication value w is chosen and split into shares with Shamir secret sharing. These shares and a hash value m_0 based on the authentication value w, the password and the salt value are transferred securely to the manager device.

During the second subphase (Figure 8.19), randomly chosen smart home devices calculate their password share-based long-lived symmetric secret keys $K_i = H(\hat{e}(s_i P, Q_z))$, construct and also verify the user's knowl-

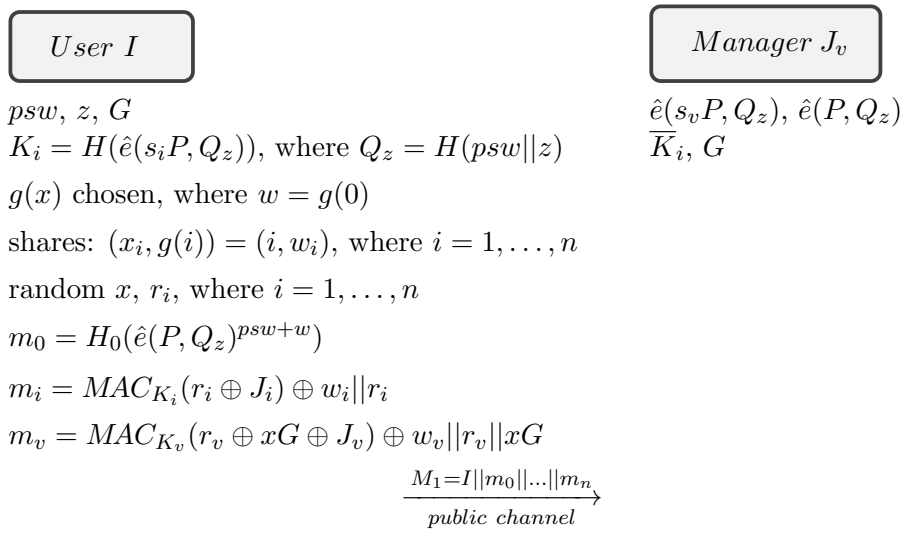


Figure 8.18. Authentication - Client process

edge of the value $\hat{e}(P, Q_z)^{w+psw}$, which is based on the password, the salt and the secret, random authentication value w.

In the third subphase (Figure 8.20), a secret symmetric key is exchanged between the user and the manager device and the user checks whether the smart home system is able to calculate $\hat{e}(P, Q_z)^{w+psw}$, *i.e.,* whether the devices possess the password shares and the salt.

A detailed security analysis of the proposed AKC protocol is provided. One of the indispensable security requirements is the mutual authentication of the participants. The secure mutual authentication of participants prevents adversaries from impersonating a legal user or the device manager and gaining illegal access to sensitive data. Another security goal is key secrecy, *i.e.,* an adversary must not possess any information about the new key. During a protocol run, a new randomly chosen session key should be exchanged between the participants, a protocol execution cannot be successfully completed with an old key exchanged before. At the end, parties should be able to verify that the other party knows and is able to use the new session key. Known-key security and forward secrecy properties are also considered. Known-key security preserves the security

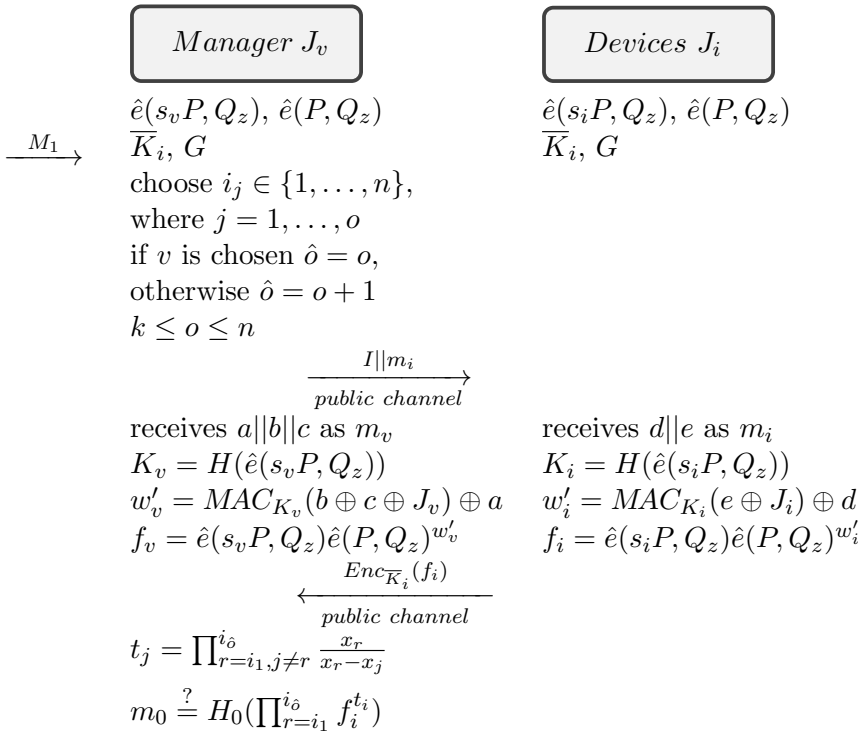


Figure 8.19. Authentication - Devices' process

of other session keys after disclosure of a session key. Disclosure of a session key should not jeopardize the security of other session keys. Forward secrecy holds if the long-term secrets of one or more entities are compromised but the secrecy of previous session keys is not affected. The user's role including the user's steps in the protocol was formalized in AVISPA. We apply the OFMC and CL-AtSe and executed the attacker simulation. The results of the security analysis show that the attacker is not able to impersonate the legal participants or obtain the session key. We have selected a threshold authentication system [63], which is similar to our system. We compare their runtime results with ours for the different number of devices and thresholds. According to [53], on average 500 devices will be connected per household in 2022, hence a large number of devices and thresholds should be considered. Our proposition provides a better result

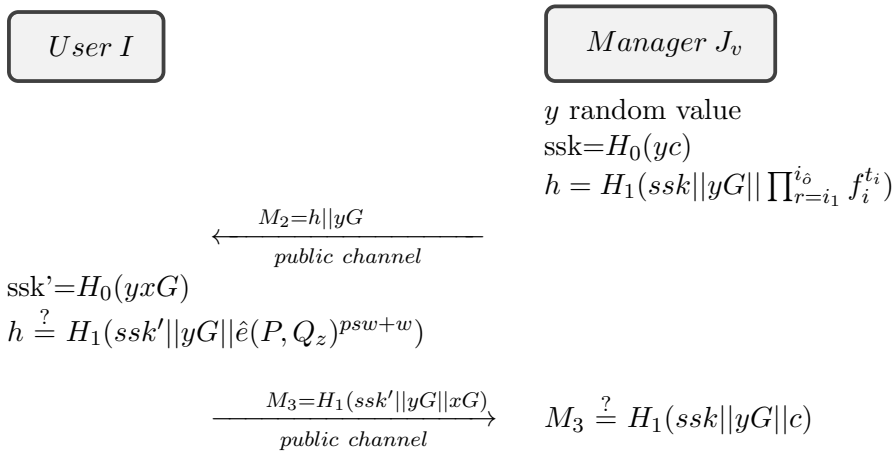


Figure 8.20. Authentication - Final process

for $n \geq 10$ number of devices and for $o \geq 5$ threshold (Table 8.2). Today, achieving computing capacity and adequate security are important considerations for IoT devices. The cost of manufacturing affects the capabilities of these devices, however, we need to ensure security. These aspects are also taken into account during the design of our protocol.

Threshold	**2-5**	**3-6**	**5-10**
Işler, Küpçü - DSPP	0,00806	0,01171	0,01833
Our proposition	0,0150602	0,0150648	0,0150766

Table 8.2. Performance comparison (in seconds).

Összefoglaló

Az online kommunikáció során az egyik alapvető kérdés a résztvevők biztonságos hitelesítése. Ha a hitelesítés megfelelően működik, akkor elkerülhetőek a különböző támadások (pl. megszemélyesítéses támadás), ellenben az autentikáció helytelen működése esetén nem biztosított a felhasználó hozzáférés-ellenőrzés, illetve a felhasználói adatok bizalmassága és sértetlensége. A felhasználó hitelesítési sémák esetén számos biztonsági követelményt kell figyelembe venni, amelyek függenek az alkalmazott környezet jellemzőitől. Az egyik leggyakrabban használt hitelesítési módszer rövid titkokon, például jelszavakon alapul. Az első entitás hitelesítési fázis előtt mindig szükséges egy regisztrációs folyamat végrehajtása, mely a tudományos irodalomban kevés figyelmet kap.

A jelen disszertáció három új felhasználó hitelesítési protokollt, illetve egy felhasználói regisztrációs protokollt mutat be. Az autentikáció végrehajtása osztott, vagyis több résztvevő által történik a felhőalapú számítástechnikai szolgáltatások és az okos otthon környezetek magasabb biztonsági szintjének elérése érdekében. Formális elemzéssel bizonyítjuk, hogy a protokollok teljesítik a szükséges biztonsági követelményeket. Megoldásaink hatékonyabbak, mint a jelenlegi gyakorlati és elméleti sémák.

Az első fejezet a felhasználói hitelesítési rendszerek és megoldások tudományos hátterét tartalmazza.

A második fejezetben részletezzük a protokolljainkban alkalmazott kriptográfiai primitíveket és megadjuk a szükséges definíciókat.

A 3. fejezet az automatizált biztonságelemző eszközökkel foglalkozik, és bemutatja a bizonyítható biztonság fogalmának részleteit.

A 4. fejezetben két, felhő környezetben alkalmazható elosztott felhasználó hitelesítési rendszert mutatunk be.

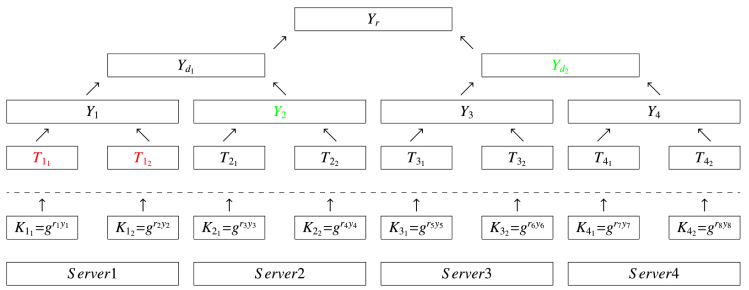
Az 5. fejezetben egy Identitás Alapú Kriptográfián és jelszón alapuló regisztrációs sémát ismertetünk, ahol a felhasználót és a szolgáltatót egyaránt hitelesíti a rövid életű, identitás alapú titkos kulcsa.

A 6. fejezetben bemutatunk egy küszöbszámon és jelszón alapuló, elosztott, kölcsönösen hitelesített kulcsmegegyezés és kulcskonfirmáció protokollt az okos otthon környezetekre.

Felhasználó hitelesítési protokoll Merkle-fa használatával

A 4.1. fejezetben bemutatunk egy felhő környezetben alkalmazható, Merkle fa használatán alapuló kétfaktoros hitelesítési sémát ([57]). Az elméleti ([33, 40]) és a gyakorlati megoldások centralizált hitelesítést alkalmaznak, ahol *egyetlen* felhőszerver végzi a felhasználók hitelesítését. A mi megoldásunk több szervert alkalmaz a felhasználók hitelesítésére. A [58] cikkben a séma biztonsági elemzését mutatjuk be applied pi kalkulusban. Protokollunkban a támadás csak akkor lehet sikeres, ha az ellenfél rendelkezik a szerverek által ismert összes jelszórésszel. A hitelesítési fázisunk hatékonyságát a [33, 40] munkájával összehasonlítva azt találjuk, hogy a mi sémánk hatékonyabb, mivel a résztvevő felek főleg csak hash számításokat végeznek. A fejezet eredményeit a Huszti Anderával közös cikkek ([57, 58]) tartalmazzák.

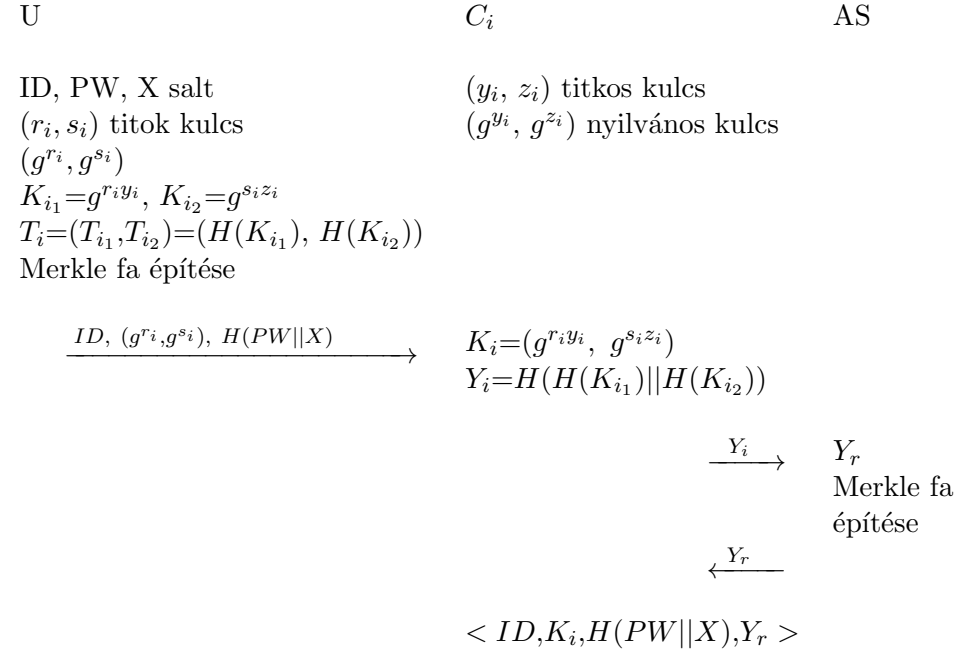
A felhasználó hitelesítése a szolgáltató oldalán egy statikus és egy egyszer használatos jelszóval történik egy véletlenszerűen kiválasztott szerver segítségével. A kiválasztott szerver a Merkle fát (9.21. ábra) alkalmazva az egyszer használatos jelszó helyességét tudja ellenőrizni. A Merkle fa levele egy jelszórész hash értéke, és a fa gyökérelemével, illetve a hozzá tartozó Merkle fa útvonallal megtörténik a teljes egyszer használatos jelszó helyességének ellenőrzése.



9.21. ábra. Merkle-fa 8 levélelemmel

A protokoll három fázisból áll: regisztráció, hitelesítés és szinkronizálás. A regisztrációs fázisban (9.22. ábra) a titkos kulcsok cseréje során

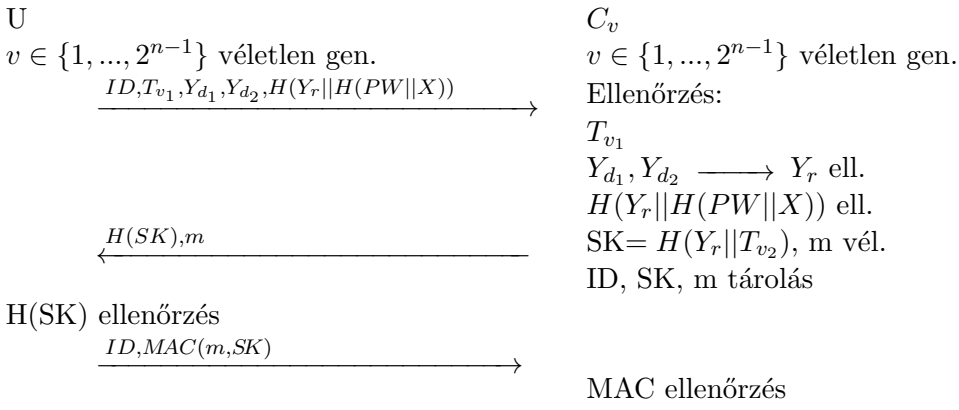
nagy mennyiségű egyszer használatos jelszó generálódik a felhasználó és a felhőszerverek között. Minden felhőszerver (C_i) rendelkezik egy aszimmetrikus kulcspárral: $SK_{C_i} = (y_i, z_i)$, $PK_{C_i} = (g^{y_i}, g^{z_i})$, ahol g egy ciklikus csoport generátoreleme, és a $y_i, z_i \in \mathbb{Z}_q$ véletlen értékek, ahol q egy nagy prím.



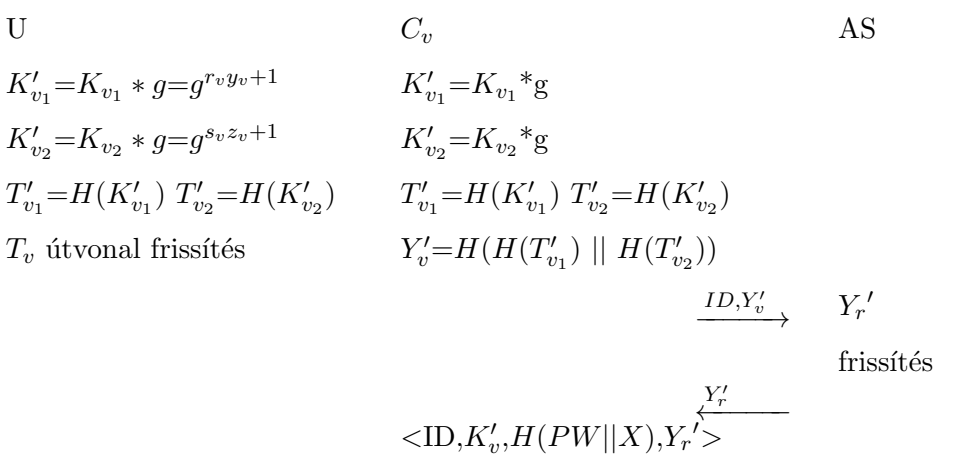
9.22. ábra. Regisztráció

A hitelesítési fázisban (9.23. ábra) megtörténik a felhasználó és egy véletlenszerűen kiválasztott felhőszerver (C_v) kölcsönös hitelesítése, valamint végrehajtódik egy MAC kulcscsere. Üzenet hitelesítési kulcs (MAC) cseréje garantálja az üzenetek változatlanságát és eredetének integritását a későbbi interaktív kommunikáció során.

A hitelesítés után a szinkronizálás folyamata következik (9.24. ábra), ahol a kiválasztott szerver jelszava frissül a fához tartozó útvonallal együtt. A biztonsági elemzéshez a protokollt ProVerifben formalizáljuk. A ProVerif teszt eredménye azt mutatja, hogy a megadott biztonsági kritériumok teljesülnek. Ezt a protokollt kulcscsere sémaként elemezzük, így figye-



9.23. ábra. Autentikáció



9.24. ábra. Szinkronizáció

lembe vesszük a kölcsönös entitás hitelesítési sémák tipikus biztonsági követelményeit, valamint a kulcsokkal kapcsolatos követelményeket. A következő négy tulajdonságot sikerült igazolni:

1. Kölcsönös hitelesítés

   (a) A felhasználók hitelesítése: A támadók nem adhatják ki magukat legális felhasználónak, és nem férhetnek hozzá a felhasználói

adatokhoz.

(b) A szerver hitelesítése: A támadók nem adhatják ki magukat legális felhőszervernek.

2. A MAC kulcs titkossága: A kulcscsere során az újonnan generált kulcs bizalmas adat, és a támadónak nem szabad információval rendelkeznie az új kulcsról.

3. Kulcs frissessége: A protokoll futása közben egy új, véletlenszerűen kiválasztott kulcsot kell generálni, így a protokoll végrehajtása nem lehet sikeres egy régi, már korábban használt kulccsal.

4. Mindkét félnek ellenőriznie kell, hogy a másik fél ismeri és tudja használni az új MAC-kulcsot.

A felhasználó és a szerver hitelesítésének biztonsági elemzésére injektív lekérdezéseket alkalmazunk. A lekérdezések mindegyike igaz értékkel tér vissza, ami azt jelenti hogy a modellünkben a felhasználó és a szerver kölcsönös hitelesítésének megsértésére, valamint a kulcs titkosságának sérülésére nem talál támadást a ProVerif. A kölcsönös hitelesítés mellett a *kulcs frissessége és kulcskonfirmáció* szempontok is teljesülnek.

Skálázható és elosztott felhasználó hitelesítés felhő szolgáltatásokhoz

A 4.2. fejezetben egy többszerveres jelszó alapú hitelesített kulcscsere sémát (9.25. - 9.27. ábrák) javasolunk. Más, küszöbszámon alapuló titokmegosztási algoritmusokat alkalmazó és jelszó alapú protokollokkal [9, 16, 27, 28, 99, 65, 63, 86, 68] ellentétben, habár a jelszóinformációkat megosztjuk a szerverek között, a titkot nem kell rekonstruálni a titokrészekből, hogy ellenőrizze a felhasználó hitelességét. Annak bizonyítására, hogy a javasolt protokoll bizonyíthatóan biztonságos, bevezetjük a küszöbszám alapú hibrid korrupciós modellt. A [27, 47]-tól eltérően részletes biztonsági elemzést adunk a Bellare és Rogaway modell alapján. Más sémákkal összehasonlítva figyelembe vesszük a skálázhatósági tulajdonságot is, amely az egyik fő követelmény a felhőkkel szemben és bemutatunk egy új módot arra, hogy a jelszóból skálázható erős titkot állítsunk elő (pl. hosszú élettartamú kulcsot). A [105] cikkben

a szerzők IoT környezetbeli vezeték nélküli szenzorhálózatokra tervezett hitelesített kulcscsere (AKE) protokollt mutatnak be. A szerzők a kulcsmegosztásokra összpontosítottak, és egy hitelesített kulcscserét javasolnak a vezeték nélküli szenzorhálózat (WSN) és a központi hitelesítést végző felhőszerver között. Az AKE protokoll egy módosított változatát a [117] publikációban ismertetik, mely 5G hálózatra tervezett, és a felhőszerverek mellett egy rögzített vezérlőszervert tételez fel. A javaslatunk eltér ezektől a megoldásoktól ([117, 105]), mivel a generált hosszú élettartamú kulcsokat a felhasználói és a szolgáltatói oldalon is skálázhatjuk. A korábban javasolt protokollunkhoz ([58]) képest a skálázhatóság mellett titokmegosztási technikát alkalmazunk. A fejezet eredményeit Huszti Andreával közös [59] cikkünk tartalmazza.

Az általunk javasolt protokoll sikeres lefutása egy munkamenetkulcsot eredményez, amely biztosítja a résztvevők közötti későbbi üzenetek bizalmasságát. A protokollnak két fázisa van. A regisztráció során a kliens jelszó alapú, hosszú élettartamú kulcsokat cserél ki az összes (n darab) szerverrel. A kliens oldalon egy egyszerű megoldást javasolunk, amelyben a kliens jelszóval fér hozzá a hosszú élettartamú kulcsokhoz. Feltételezzük, hogy a kliens egy klienseszközzel rendelkezik, melyen (pl. intelligens kártya, mobiltelefon stb.) fut egy kliensszoftver, amely jelszót kér a felhasználótól a hitelesítési folyamat elindításához. Miután a kliens megadta a jelszót, a kliensszoftver legenerálja a hosszú élettartamú kulcsokat, és megkezdődik a hitelesítés végrehajtása. A jelszó helyességét nem a kliensszoftver, hanem a szerver ellenőrzi, így a klienseszköz nem tárol semmilyen információt a jelszóról. A felhasználó n szerverből véletlenszerűen kiválaszt k szervert a hitelesítéshez. A hitelesítés során a szerver csak a szimmetrikus, hosszú élettartamú K_i kulcs ($i \in \{1, \ldots, k\}$) ismeretében tudja kiszámítani a kliens által generált w kihívásértéket. A *KKDF* egy Keyed Key Derivation Function-t jelöl, amely egy m és egy key üzenethez egy K titkos kulcsot generál. A hitelesítési szerver (J_v) a résztvevő szerverektől kapott összes k darab kihívás érték helyességének ellenőrzésével hitelesíti a klienst.

A javasolt protokollban a szerverek biztonságos csatornákon kommunikálnak egymással. Egy véletlenszerűen kiválasztott szerver kommunikál a klienssel, így a kliensnek nem kell párhuzamosan kommunikálnia az összes k szerverrel és biztonságos csatornákat kiépíteni. A protokoll ter-

vezése során a hitelesítés hatékonyságát MAC és egyéb gyorsnak számító kriptográfiai algoritmusok (hash, xor művelet, szimmetrikus titkosítás) biztosítják. A protokoll bizonyíthatóan biztonságos. Feltételezzük, hogy a támadó ($\mathcal{A}$) számára engedélyezett a `Send, Reveal, Corrupt, Test` lekérdezések végrehajtása.

Az elosztott hitelesítés elemzésére az alapmodellt a küszöbszám alapú hibrid korrupciós modellel bővítjük. Feltételezzük, hogy a résztvevők korruptak lehetnek. A modell *erős korrupciós modell* ([13]), ha a hosszú életű kulcsok $K_{I,J}$ és a résztvevő I által tárolt összes érték (pl. véletlenszerűen kiválasztott titkos értékek) a protokoll futása során az $\mathcal{A}$ támadó tudomására jut. A *gyenge korrupciós modell* esetén csak a $K_{I,J}$ hosszú életű kulcsok módosulnak vagy kerülnek ki, a támadó nem kompromittálja teljesen a gépet. A protokollfutás során létrehozott és tárolt egyéb értékek nem kerülnek nyilvánosságra.

1. Definíció Egy modellt *küszöbszámon alapuló hibrid korrupciós modellnek* nevezünk, ha feltételezzük, hogy az autentikált kulcscsere

I $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ J_v

$(K_1,\ldots,K_k)$, G $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ K_v, G

$K_i = KKDF_{key}^{c+i}(psw)$, ahol $key = H(salt||psw)$

$K_n = KKDF_{key}(psw) \oplus \cdots \oplus KKDF^{c+n-2}{}_{key}(psw)$

$t_1,\ldots,t_{k-1},t_v$; $r_1,\ldots,r_{k-1}$, r_v, x véletlen

$w_1 = H(t_1),\ldots,w_v = H(t_v)$

$w = H(w_1||\ldots||w_{k-1}||w_v)$

$m_0 = H(w)$

$m_i = (MAC_{K_i}(r_i \oplus J_i) \oplus w_i)||r_i$

$m_v = (MAC_{K_v}(r_v \oplus xG \oplus J_v) \oplus w_v)||r_v||xG$

$$\xrightarrow[nyilv.csatorna]{M_1=I||J_1||\ldots||J_k||m_0||\ldots||m_k}$$

9.25. ábra. Hitelesítés - Kliens folyamat

J_v Szerverek

$\overline{K}_1, \ldots, \overline{K}_{k-1}$ rövid életű kulcsok

$\xrightarrow{I||m_i}$ J_i

K_i, $\overline{K}_i$

$m_i = p||o$

$w'_i = p \oplus MAC_{K_i}(o \oplus J_i)$

$\xleftarrow{Enc_{\overline{K}_i}\ (w'_i)}$

$m_v = u||s||z$

$w'_v = u \oplus MAC_{K_i}(s \oplus z \oplus J_v)$

$w' = H(w'_1|| \ldots ||w'_{k-1}||w'_v)$

$m_0 = H(w) \stackrel{?}{=} H(w')$

y véletlen

ssk=$H_0(yxG)$

$h = H(ssk||yG||xG||w')$

9.26. ábra. Hitelesítés - Felhő szerverek közötti kommunikáció

I J_v

$\xleftarrow[nyilv.csatorna]{M_2=h||yG}$

ssk'=$H_0(yxG)$

$h \stackrel{?}{=} H(ssk'||yG||xG||w)$

$\xrightarrow[nyilv.csatorna]{M_3=H(ssk||yG||xG)}$ $M_3 \stackrel{?}{=} H(ssk||yG||xG)$

9.27. ábra. Hitelesítés - Végső folyamat

kulcskonfirmációval protokoll (AKC) során n szerverből véletlenszerűen kiválasztunk k szervert, valamint a kliens nem korrupt, n szerver közül legalább $n - k + 1$ szerver nem korrupt. Ezen kívül a klienssel való kommunikációnál a kiválasztott szerver

1. nem korrupt, vagy

2. gyengén korrupt, és a fennmaradó szerverek között van legalább egy nem korrupt.

A biztonságos AKC protokoll definíciójának megadásához át kell tekintenünk a [19] alapján a beszélgetés és az illeszkedő beszélgetés definícióját.

Az illeszkedő beszélgetés az I és a J entitások közötti valós idejű kommunikációt formalizálja. A $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ esemény definíciója a [19] dolgozatban megadott definíció módosított változata. Többszerveres beállításunkban minden kliens kommunikálhat gyengén korrupt szerverrel feltéve, hogy van legalább egy nem korrupt szerver a k szerverek között.

2. Definíció A P protokollban a $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ egy olyan esemény, ahol egy $\mathcal{A}$ támadó jelenlétében küszöbszám alapú hibrid korrupciós modellt tételezünk fel és létezik

1. egy $\prod_{I,J}^{s}$ kliens orákulum, mely elfogadott állapotban van, de nincs $\prod_{J,I}^{t}$ szerver orákulum, amely illeszkedő beszélgetést folytatna $\prod_{I,J}^{s}$ orákulummal, vagy

2. egy $\prod_{I,J}^{s}$ szerver orákulum, amely nem korrupt és elfogadott, de nincs olyan kliens orákulum, amelyik $\prod_{J,I}^{t}$ illeszkedő beszélgetést folytatna a $\prod_{I,J}^{s}$-vel, vagy

3. egy $\prod_{I,J}^{s}$ szerver orákulum, amely gyengén korrupt és elfogadott, de nincs kliens vagy nem korrupt szerver orákulum, amely illeszkedő beszélgetést folytatna a $\prod_{I,J}^{s}$-val.

A biztonságos AKC meghatározásához szükséges a *frissesség* fogalmának meghatározása és az *jóindulatú támadó* újradefiniálása.

3. Definíció Egy klienst és k szerver orákulumot tartalmazó elem $k+1$-es friss, ha a küszöbszámon alapuló hibrid korrupciós modellben a kliens

orákulum és a szerver orákulum, amellyel illeszkedő beszélgetést folytatott, nem nyitott (unopened). Az orákulumot *frissnek* nevezzük, ha eleme egy friss elem $k+1$-esnek.

4. Definíció Egy támadót *jóindulatúnak* nevezünk, ha determinisztikus, és tevékenységét arra korlátozza, hogy választ egy elem $k+1$-es orákulumot, amely egy klienst és k szerver orákulumot tartalmaz, majd minden üzenetet tisztességesen továbbít egyik orákulumtól a másikig, a kliens orákulumtól indulva.

5. Definíció A protokoll egy *biztonságos AKC protokoll*, ha

1. A jóindulatú támadó jelenlétében a kliens és a klienssel kommunikáló szerver orákulum mindig elfogadja ugyanazt az ssk munkamenetkulcsot, mely egyenletes eloszlással generált a $\{0,1\}^\kappa$ halmazon.

minden $\mathcal{A}$ támadó jelenlétében

2. Egy küszöbszám alapú hibrid korrupciós modellben van egy kiválasztott $\prod_{I,J}^{l}$ szerver orákulum, amely illeszkedő beszélgetést folytat a kliens orákulummal, és ha ez a $\prod_{I,J}^{l}$ szerver orákulum gyengén korrupt, akkor a $\prod_{I,J}^{l}$ szerver orákulum illeszkedő beszélgetést kell folytatnia egy nem korrupt szerver orákulummal. A kliens orákulum és a $\prod_{I,J}^{l}$ szerver orákulum elfogadja és ugyanazt ssk munkamenetkulcsot használja.

3. A $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ valószínűsége elhanyagolható.

4. Ha a tesztelt orákulum friss, akkor $Adv^{\mathcal{A}}(\kappa)$ elhanyagolható.

9.0.10 Tétel. A javasolt protokoll egy biztonságos AKC protokoll a véletlenszerű orákulum modellben, feltételezve, hogy a MAC *univerzálisan hamisíthatatlan adaptív, választott üzenet alapú támadás esetén*, a szimmetrikus titkosítási séma *megkülönböztethetetlen a választott nyílt szöveg alapú támadásnál*, és az *Elliptikus görbe Diffie-Hellman kiszámíthatósági (ECCDH) probléma nehéz* az elliptikus görbe csoportban.

Protokollunk tervezése során fontos szempont volt a hatékonyság. A protokollban a munkamenetkulcsot az Elliptikus görbe Diffie-Hellman

(ECDH) kulcscsere állítja elő, a többi művelet pedig a hash és xor műveletek, amelyek rendkívül gyorsak.

Bizonyíthatóan biztonságos identitás alapú távoli jelszóregisztráció

Az 5. fejezetben egy Identitás Alapú Kriptográfián és jelszón alapuló regisztrációs sémát mutatunk be, ahol a felhasználót és a szolgáltatót egyaránt hitelesíti a rövid életű, identitás alapú titkos kulcsa. A javasolt protokoll illeszkedik az okos otthon környezetben alkalmazott felhasználó hitelesítési sémánkhoz, ahol a bilineáris leképezés értékeit az IoT-eszközökön tárolják. A javasolt felhősémánk ([58]) is könnyen módosítható a megfelelő hosszú élettartamú kulcsbeállítással, hogy kompatibilis legyen regisztrációs sémánkkal.

A biztonságos tároláshoz egy salt-tal ellátott bilineáris leképezést mutatunk be, ahol a salt egy rövid (12–48 bites) véletlenszerű adat, amelyet a hashelés előtt összefűznek a jelszóval. Így offline támadás esetén a támadó minden lehetséges jelszójelölthöz és salt-hoz kénytelen számításigényes bilineáris leképezést számolni, ami lassítja a támadást. A megoldásunk a hagyományos regisztrációs megoldásokkal ellentétben nem igényel Transport Layer Security (TLS) csatornát és mellőzi a hozzá tartozó tanúsítványkezelést is. Ez vállalati vagy oktatási intézményekben hatékonyabb működést tesz lehetővé, ahol jellemzően az egyedi azonosítók használata miatt ideális az Identitás Alapú Kriptográfia alkalmazása. A protokollunk hatékonyabb, mint a fent említett TLS-alapú és a többi vak regisztráció [71, 72], mivel nincs szükség tanúsítványok kezelésére vagy költséges nulla ismeretű bizonyítás végrehajtására. A többi rendszerrel ellentétben ([71, 72]) a jelszó hash-elő séma mellett figyelembe vettük az interakciókat is a protokoll résztvevői között és a jelszót ellenőrző információt biztonságosan küldjük. Bebizonyítottuk, hogy megoldásunk az online támadások ellen is biztonságos. Bevezetjük a biztonságos jelszóregisztációs rendszer definícióját, illetve megadjuk a támadói modellt és megmutatjuk, hogy a rendszerünk bizonyíthatóan biztonságos. A regisztrációnk rugalmas, ami optimális a föderációs bejelentkezésnél (SSO) vagy a Kerberos hitelesítéseknél, de olyan rendszereknél is alkalmas, ahol minden egyes szolgáltatáshoz más-más jelszót kell alkalmazni. A jelszó

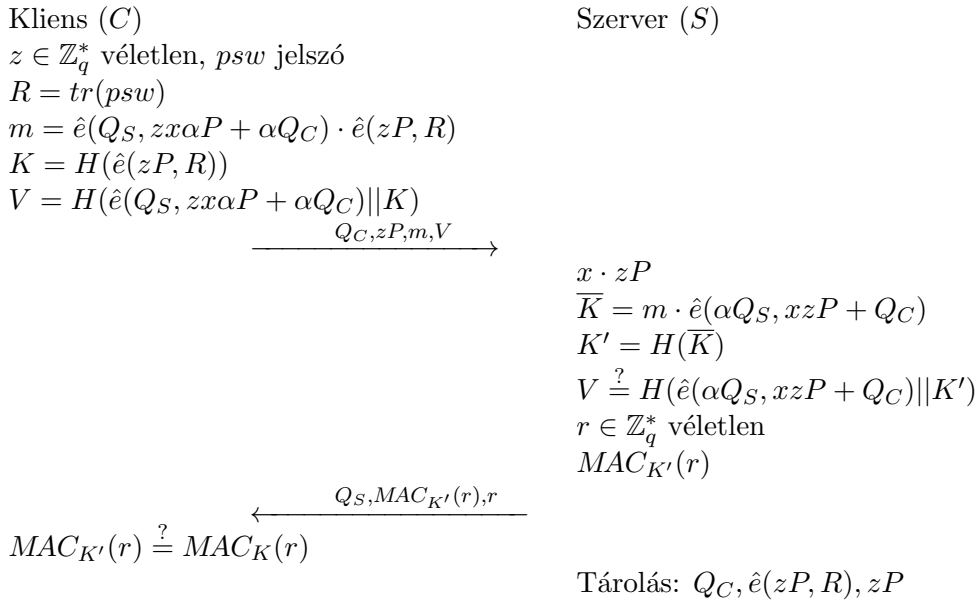
és a salt bilineáris leképezése hosszú élettartamú szimmetrikus kulcsként használható, és alkalmazható entitás hitelesítésre vagy munkamenetkulcs generálásra. A fejezet eredményeit az elfogadott cikkünk tartalmazza ([17]), amely Huszti Andreával, Bertók Csanáddal és Kovács Szabolccsal közös munka.

A protokoll egy beállítási és egy regisztrációs fázisból áll (9.28. ábra, 9.29. ábra). A Beállítás folyamata során legeneráljuk a rendszerparamétereket és a kulcsokat a résztvevők számára. Legyen P a $\mathbb{G}$ egy generátora, ahol $\mathbb{G}$ egy q-adrendű additív csoport, ahol q egy nagy prím. Válasszunk egy véletlen $\alpha \in \mathbb{Z}_q^*$ értéket, és generáljuk le a $P, \alpha P$ paramétereket. A rendszer mester titkos kulcsa az α. A ID_C, ID_S azonosítók, a $PK_C = Q_C = tr(ID_C)$ és $PK_S = Q_S = tr(ID_S)$ nyilvános kulcsok. Mivel a jelszó hash sémánk elliptikus görbén alapuló bilineáris párosításokat ($\hat{e}$) használ, hatékony módszerre van szükségünk ahhoz, hogy a jelszavakat először egy $\mathbb{Z}_p$-beli elemre képezzük le, ahol p egy nagy prím, majd a $\mathbb{Z}_p$-beli elemet a görbe egy pontjához rendeljük hozzá. Jelöljük ezt a függvényt tr-rel. A privát kulcsgenerátor (PKG) kiszámítja a résztvevők titkos kulcsait ($SK_C = \alpha Q_C$ és $SK_S = \alpha Q_S$). A regisztrációs fázisban az ügyfelek elküldik jelszóadataikat a szervernek, és meggyőződnek arról, hogy a szerver megkapta a jelszóellenőrző értéket. A protokoll minden szükséges követelménynek megfelel, beleértve a jelszó titkosságával, a felek kölcsönös hitelesítésével és az offline támadásokkal szembeni ellenállást.

Kliens (C)	PKG	Szerver (S)
	$\alpha \in \mathbb{Z}_q^*$ (msk)	$x \in \mathbb{Z}_q^*$ titkos kulcs
	nyilvános információk: $P, \alpha P, x\alpha P$	
$Q_C = tr(ID_C)$ (PK_C)		$Q_S = tr(ID_S)(PK_S)$
αQ_C (SK_C)		αQ_S (SK_S)

9.28. ábra. Beállítás

Olyan biztonsági modellt adunk meg, amely az offline támadások mellett az online támadásokkal szembeni ellenállást is tekinti. A javasolt modellünk a teljes regisztrációs folyamatot figyelembe veszi, ellentétben a [72] és [71] modellekkel. Tekintetbe veszi a kliens és a szerver közötti



9.29. ábra. Jelszó regisztrációs protokoll

összes kommunikációs üzenetet. Ezért a résztvevők kölcsönös hitelesítését és a jelszó titkosságát is vizsgáljuk az átvitel során. Az $\mathcal{A}$ támadó olyan lekérdezéseket végezhet, amelyek modellezik a támadásait. Ezek a lekérdezések a következők: `Send, Corrupt, Reveal, Test, Execute` és `Finalise`.

Meghatározzuk a jelszóregisztrációs protokollok biztonsági céljait a teljes regisztrációs folyamatra vonatkozóan. Bevezetjük a biztonságos regisztráció definícióját:

6. Definíció A protokoll egy *biztonságos regisztrációs protokoll* ha

1. A *jóindulatú támadó* jelenlétében a kliens és a vele kommunikáló szerver orákulum mindig elfogadott állapotba kerül. A szerver tárolja az ügyfél által megerősített jelszó ellenőrzési értéket.

és minden $\mathcal{A}$ támadóra

2. Ha van egy nem korrupt kliens orákulum, amely illeszkedő beszélgetéseket folytat egy nem korrupt szerver orákummal, akkor mindig elfogadott. A szerver tárolja a kliens által megerősített jelszó ellenőrzési értéket;

3. Nem korrupt szerver és kliens orákulum esetén a $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ valószínűsége elhanyagolható;

4. A tesztelt orákulumban a $Adv^{\mathcal{A}}(\kappa)$ elhanyagolható. Ha ez egy kliens orákulum, akkor nem nyitott;

5. Ha az összes D_n szótárnál az $\mathcal{A}$ támadó legfeljebb t darab (C, S, psw) elemhármast generál, akkor

$$Pr[\texttt{Finalise}(C, S, psw) = 1] \leq \frac{t}{2^{\beta_{D_n}} \cdot t_{pre}} + \mu(\kappa),$$

ahol $\mu(\kappa)$ elhanyagolható, a t_{pre} pedig az egyirányú függvény bemeneti értékének kiszámításához szükséges számítási költséget jelöli.

A protokoll biztonságát véletlen orákulum modellben vizsgáljuk, ahol két biztonsági modellt különböztetünk meg. A kliens-szerver protokollok esetében a kliensekről általában feltételezhető, hogy rosszindulatúak, azaz eltérnek a protokoll lépéseitől és bármilyen típusú stratégiát alkalmazhatnak a támadás során. A szolgáltatást nyújtó szerverek általában *becsületesnek* számítanak, vagyis nem indítanak támadást, vagy *becsületes, de kíváncsiak*, azaz csak passzív támadásokat kezdeményeznek, nem hagyva nyomot a támadás során. Attól függően, hogy a szerver becsületes vagy becsületes, de kíváncsi, megkülönböztetünk **becsületes és becsületes, de kíváncsi modelleket**. A [72] és [71] becsületes modelleket használnak. A javasolt protokollban becsületes, de kíváncsi modellt tételezünk fel.

9.0.11 Tétel. A javasolt jelszóregisztrációs protokoll ellenáll az online támadásoknak a becsületes, de kíváncsi modellben, feltételezve, hogy a MAC *egzisztenciálisan hamisíthatatlan egy adaptív választott üzenet alapú támadás során*, *Bilineáris Diffie-Hellman* nehéz probléma, továbbá a bilineáris leképezéseket az általános bilineáris csoport modellben, illetve a hash függvényeket véletlen orákulumnak tekintjük.

9.0.12 Tétel. A javasolt jelszóregisztrációs protokoll ellenáll az offline támadásoknak a véletlen orákulum modellben, ha a bilineáris leképezés egyirányú leképezés és a kliens gyengén korrupt.

Összehasonlítva a hatékonyságot más regisztrációs protokollokkal (9.3. táblazat) az eredmény azt mutatja, hogy az általunk javasolt regisztrációs protokoll hatékonyabb a többi javaslathoz képest.

Sémák	**Kliens**	**Szerver**	**Teljes**
BPR- 2 szerveres	1,4 s	0,68 s	2,76 s
BPR - VPAKE	0,72 s	0,67 s	1,5 s
TLS			0,168 s
Mi javaslatunk	0,072 s	0,023s	0,095 s

9.3. táblázat. A protokollok végrehajtási ideje (másodpercben)

Skálázható, jelszó és küszöbszámon alapuló hitelesítés okos otthonokhoz

A 6. fejezetben bemutatunk egy küszöbszámon és jelszón alapuló, elosztott, kölcsönösen hitelesített kulcsmegegyezés és kulcskonfirmáció protokollt egy okos otthon környezetben. A javasolt felhőalapú hitelesítési sémánkban ([59]) feltételezzük, hogy a felhőkiszolgálók mindig elérhetőek. Az okos otthoni rendszerekben azonban az eszközök különféle típusúak lehetnek, ami azt jelenti, hogy egyes eszközök akkumulátorról működnek, míg mások korlátozott erőforrásokkal rendelkeznek, és előfordulhat, hogy nem elérhetőek a felhasználó számára. Figyelembe véve az okos otthonok ezen tulajdonságát egy új titokmegosztási technikával működő felhasználói hitelesítési sémát javaslunk, ahol megköveteljük, hogy a dinamikusan választható n darab készülék közül k legyen elérhető, ahol $k \leq n$. A fejezet eredményeit a Huszti Andreával és Kovács Szabolccsal közös cikkünk [60] tartalmazza.

A protokoll tervezése során fontos a megfelelő jelszóhasználat beállítása és a végpontok közötti biztonságos kommunikáció elérése. A javasolt protokoll egy méretezhető és robusztus séma, ahol a sikeres szótártámadáshoz $k - 1$ darab okos otthoni eszközt (k a jelszó küszöbszám) kell kompro-

mittálnia a támadónak. A tudományos irodalomban Bagherzandi a [9] dolgozatban egy jelszóval védett titokmegosztási (PPSS) és küszöbszámon alapuló megoldást mutat be. Jarecki a [65] publikációban javasol egy egykörös optimális PPSS-sémát, amely mindösszesen két üzenetküldést tartalmaz. Ezek a megoldások azonban nem skálázhatóak. Işler és Küpçü a [62, 63] publikációiban hasonló szempontokat vesznek figyelembe (skálázhatóság, robusztusság, jelszóhasználat stb.), viszont jobban alkalmazhatóak felhő környezetben, és protokolljaik tartalmaznak tárolószolgáltatókat. A mi megoldásunk okos otthon környezetre lett kialakítva, ahol $n \geq 10$ eszköz és $o \geq 5$ küszöbszám esetén (o a hitelesítéshez szükséges IoT eszközök száma) jobb hatékonysági eredményt érünk el.

Két résztvevője van a protokollunknak. Az egyik az *IoT rendszer*, amely tartalmazza a eszközkezelőt és az IoT-eszközöket $(J_1, \ldots, J_n)$. A másik résztvevő a *felhasználó* (I), amely kéri a szolgáltatásokat és az adatokat. A protokollban titokmegosztást alkalmazunk, ahol (k, n) küszöbszám sémát használunk. Egy titkos S egész felosztható n részre oly módon, hogy $k \leq n$ lesz a titokrészek küszöbszáma, amellyel ki kell tudjuk számítani az S egészt. Így $k-1$ vagy annál kevesebb titokrésszel nem lehet meghatározni az S egészt. A jelszó létrehozásához Shamir-féle titokmegosztást alkalmazunk az IoT-eszközökön.

A beállítási fázis során a felhasználó kiválaszt egy psw jelszót, majd a kliens szoftver generál és biztonságosan tárol egy véletlenszerű z salt értéket és egy véletlenszerű polinomot a psw Shamir-féle titokmegosztásához. Legeneráljuk az s_i titokrészeket, ahol $i = 1, \ldots, n$ és az eszközök elküldik és tárolják az $\hat{e}(P, Q_z)$ és $\hat{e}(s_i P, Q_z)$ értékeket, ahol $\hat{e}(,)$ a bilineáris leképezést jelöli, $Q_z = H(psw||z)$ és P a $\mathbb{G}$ egy generátora, ahol $\mathbb{G}$ egy q-adrendű additív csoport, ahol q egy nagy prím. A hitelesítési fázisban a kliensszoftver kiszámolja a jelszómegosztáson alapuló, hosszú élettartamú szimmetrikus titkos kulcsokat $K_i = H(\hat{e}(s_i P, Q_z))$. Ha a felhasználó új eszközöket akar beállítani az okos otthon rendszerbe, akkor meg kell adnia a jelszót a kliensszoftvernek, amely új extra s_i megosztásokat generál ugyanarra a polinomra, ahol $i > n$. Így a konstrukció tartalmazza a skálázhatóság tulajdonságát. Legyen E egy véges $\mathbb{F}$ test felett definiált elliptikus görbe, $G \in E(\mathbb{F})$ pedig egy generátorelem. Minden IoT-eszköz rendelkezik egy szimmetrikus titkosítási kulccsal $(\overline{K}_1, \ldots, \overline{K}_n)$, amely a menedzsereszköznek küldött üzenetek bi-

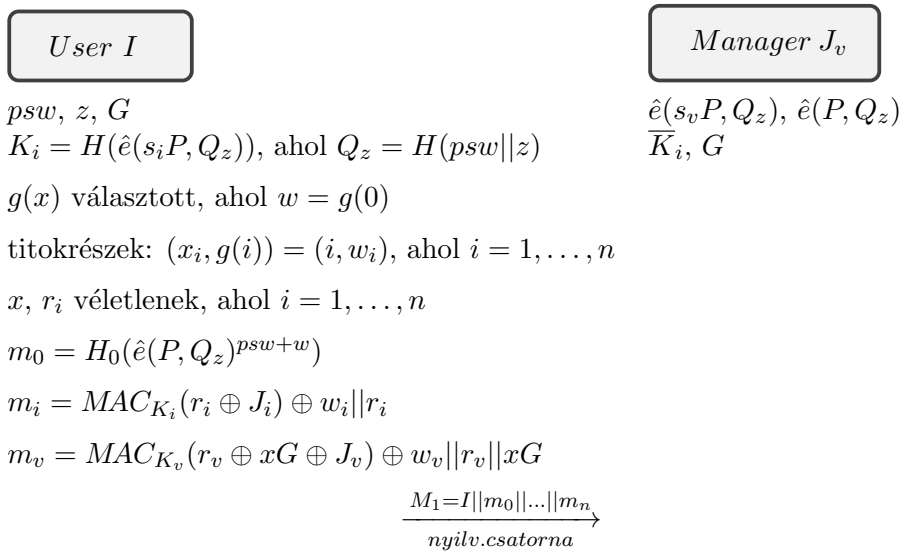
zalmasságát és hitelesítését biztosítja.

A hitelesítési szakasz három fő szakaszból áll. Az első fázist (9.30. ábra) a kliens szoftver hajtja végre. A rendszer egy titkos, véletlenszerű w hitelesítési értéket választ, és a Shamir-féle titokmegosztással felosztja. Ezek a titokrészek, a w, a jelszó és a salt-on alapuló m_0 hash érték biztonságosan átkerül az eszközkezelőhöz.

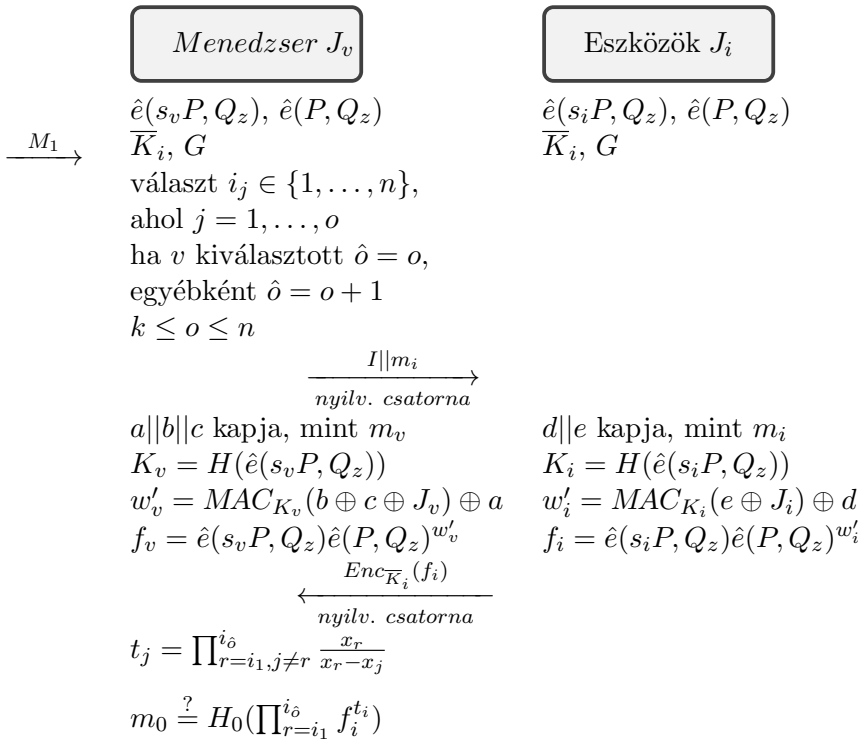
A hitelesítés második fázisában (9.31. ábra) a véletlenszerűen kiválasztott okos otthoni eszközök kiszámolják jelszó titokrészeiken alapuló hosszú élettartamú szimmetrikus titkos kulcsukat $K_i = H(\hat{e}(s_i P, Q_z))$, összeállítják és ellenőrzik az $\hat{e}(P, Q_z)^{w+psw}$ értéket, amely a jelszó, a salt és a w titkos, véletlenszerű hitelesítési értéken alapszik.

A harmadik fázisban (9.32. ábra) egy titkos szimmetrikus kulcsot cserél a felhasználó és az eszközkezelő, majd a felhasználó ellenőrzi, hogy az okos otthon rendszere képes-e kiszámítani az $\hat{e}(P, Q_z)^{w+psw}$ értéket, tehát az eszközök rendelkeznek-e a megfelelő jelszó titkokkal és salt-tal.

Részletes biztonsági elemzést nyújtunk a javasolt AKC protokollról. Az egyik alapvető biztonsági követelmény a résztvevők kölcsönös hitelesítése, amely megakadályozza, hogy a támadók érvényes felhasználónak

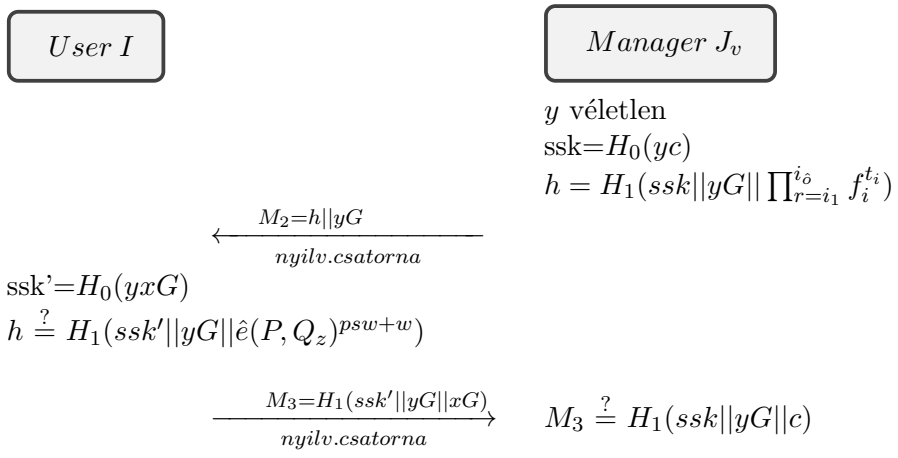


9.30. ábra. Hitelesítés - Kliens folyamat



9.31. ábra. Hitelesítés - Eszközök folyamat

vagy eszközkezelőnek adják ki magukat, és illegálisan hozzáférhessenek az érzékeny adatokhoz. Egy másik biztonsági cél a generált kulcs titkossága, azaz a támadónak nem szabad semmilyen információval rendelkeznie az új munkamenetkulcsról. Protokollfuttatás során egy véletlenszerűen kiválasztott új munkamenetkulcsot kell kicserélni a résztvevők között, és fontos, hogy a protokoll végrehajtását ne lehessen sikeresen befejezni egy korábban kicserélt kulccsal. A feleknek képesnek kell lenniük ellenőrizni, hogy a másik fél ismeri-e és képes-e használni az új munkamenetkulcsot. Figyelembe vesszük az ismert kulcs biztonságot és forward secrecy tulajdonságokat is. Az ismert kulcs biztonság lényege, hogy megőrzi a munkamenetkulcsok biztonságát abban az esetben is, ha egy munkamenetkulcsot felfedtek. Tehát egy munkamenetkulcs nyilvánosságra hozatala nem veszélyeztetheti más munkamenetkulcsok biztonságát. A forward secrecy tulajdonság fennáll, ha egy vagy több entitás hosszú távú kulcsai sérülnek és ez nincs hatással a korábbi munkamenetkulcsok titkosságára. A felhasználó szerepét vagyis a felhasználó lépéseit a protokollban AVISPA eszközzel formalizáltuk. Alkalmaztuk az OFMC és a CL-AtSe modellt, és végrehajtottuk a támadószimulációt. A biztonsági elemzés az mutatja, hogy kölcsönös hitelesítés megsértésére, illetve a munkamenetkulcs titkosságának sérülésére nem talált támadást az AVISPA.



9.32. ábra. Hitelesítés - Végső folyamat

A hatékonysági elemzéshez kiválasztottunk egy küszöbszám hitelesítési rendszert [63], amely leginkább hasonló a mi rendszerünkhöz. A futási időket összehasonlítottuk a két rendszernél különböző számú eszköz és küszöbszám esetén. A [53] szerint 2022-ben háztartásonként átlagosan 500 IoT eszköz lesz csatlakoztatva, ezért az eszközök nagy száma és a küszöbszámok figyelembe vétele kiemelt szempont. Az általunk javasolt rendszer jobb eredményt ad $n \geq 10$ számú eszköz és $o \geq 5$ küszöbszám esetén (9.4. táblazat).

Küszöbszám	**2-5**	**3-6**	**5-10**
Işler, Küpçü - DSPP	0,00806	0,01171	0,01833
Javasolt megoldás	0,0150602	0,0150648	0,0150766

9.4. táblázat. Teljesítmény összehasonlítás (másodpercben).

Manapság a számítási kapacitás optimalizálása és a megfelelő biztonság fontos szempont az IoT eszközöknél. A gyártási költség befolyásolja ezen eszközök képességeit, azonban gondoskodnunk kell a biztonságról. Ezeket a szempontokat is figyelembe vettük a protokollunk kialakítása során.

Irodalomjegyzék

[1] M. Abadi, C. Fournet, *Mobile Values, New Names, and Secure Communication*, 28th ACM Symposium on Principles of Programming Languages (POPL 2001), (2001), pp. 104–115.

[2] T. Alladi, V. Chamola, B. Sikdar, *Consumer IoT: Security vulnerability case studies and solutions.* IEEE Consumer Electronics Magazine, 9.2, (2020), pp. 17–25.

[3] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P. C. Héam, O. Kouchnarenko, J. Mantovani , et. al., *The avispa tool for the automated validation of internet security protocols and applications.* International conference on computer aided verification (Springer), (2005), pp. 281–285.

[4] A. Armando, L. Compagna, *Automatic SAT-Compilation of Protocol Insecurity Problems via Reduction to Planning.*, In Proc. FORTE 2002, LNCS 2529, Springer, (2002), pp. 210–225.

[5] A. Armando, L. Compagna, *Abstraction-driven SAT-based Analysis of Security Protocols.*, In Proc. SAT'03, LNCS 2919, Springer, (2003), pp. 257–271.

[6] A. Armando, L. Compagna, *SATMC: a SAT-based Model Checker for Security Protocols.*, In Proc. JELIA'04, LNAI 3229, Springer, (2004), pp. 617–627.

[7] A. Armando, L. Compagna, *An Optimized Intruder Model for SAT-based Model-Checking of Security Protocols.*, In Proc. ARSPA'04. Electronic Notes in Theoretical Computer Science 125(1), (2005), pp. 91–108.

[8] AVISPA Team, *AVISPA v1.0 User manual. Information society technologies programme*, http://people.irisa.fr/Thomas.Genet/Crypt/AVISPA_manual.pdf, (2006), Accessed: 30/01/2022.

[9] A. Bagherzandi, S. Jarecki, N. Saxena, Y. Lu, *Password-protected secret sharing.* In: ACM Conference on Computer and Communications Security (2011), pp. 433–444.

[10] D. Basin, S. Mödersheim, and L. Viganó. *Constraint Differentiation: A New Reduction Technique for Constraint-Based Analysis of Security Protocols.*, In Proc. CCS'03, ACM Press, (2003), pp. 335–344.

[11] D. Basin, S. Mödersheim, L. Viganóm *Algebraic intruder deductions.*, In Proc. LPAR'05, LNAI 3835, Springer, (2005), pp. 549–564.

[12] D. Basin, S. Mödersheim, L. Viganó. *OFMC: A symbolic model checker for security protocols.*, International Journal of Information Security, 4(3), (2005), pp. 181–208.

[13] M. Bellare, D. Pointcheval, P. Rogaway, *Authenticated key exchange secure against dictionary attacks.*, In Bart Preneel, editor, Advances in Cryptology - EUROCRYPT2000, volume 1807 of Lecture Notes in Computer Science, Springer, (2000), pp. 139–155.

[14] M. Bellare, P. Rogaway, *Entity authentication and key distribution.*, In: Stinson D.R. (eds) Advances in Cryptology - CRYPTO' 93. CRYPTO 1993. Lecture Notes in Computer Science, vol 773. Springer, Berlin, Heidelberg, (1993), pp. 232–249.

[15] M. Bellare, P. Rogaway, *Provably secure session key distribution: the three party case.*, Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, (1995), pp. 57–66

[16] S. M. Bellovin, M. Merritt, *Encrypted key exchange: Password-based protocols secure against dictionary attacks.*, In Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on. IEEE, (1992), pp. 72–84.

[17] C. Bertok, A. Huszti, S. Kovacs, N. Olah, *Provably Secure Identity-Based Remote Password Registration.*, Cryptology ePrint Archive, (2022). pp.

[18] Bitdefender Whitepaper, *Severe Vulnerability in iBaby Monitor M6S Camera Leads to Remote Access to Video Storage Bucket* https://www.bitdefender.com /files/News/CaseStudies/study/315/Bitdefender-Whitepaper-Severe-Vulnerability-in-iBaby-Monitor-M6S-Camera-interactive.pdf, (2019), Accessed: 30/01/2022.

[19] S. Blake-Wilson, D. Johnson, A. Menezes, *Key agreement protocols and their security analysis*, Proceedings of the sixth IMA International Conference on Cryptography and Coding, LNCS 1355, (1997), pp. 30–45.

[20] B. Blanchet, *Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif*, Foundations and Trends in Privacy and Security, vol. 1, no. 1-2, (2016), pp. 1–135.

[21] B. Blanchet, B. Smyth, V. Cheval, M. Sylvestre, ProVerif 2.00: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial [online] http://prosecco.gforge.inria.fr /personal/ bblanche/ proverif/ manual.pdf, Accessed: 30/01/2022.

[22] Y. Boichut, P.-C. Héam, O. Kouchnarenko, *Automatic verification of security protocols using approximations.*, Technical Report RR-5727, INRIA, (2005)

[23] Y. Boichut, P.-C. Heam, O. Kouchnarenko, and F. Oehl. *Improvements on the Genet and Klay Technique to Automatically Verify Security Protocols.*, In Proc. International Workshop on Automated Verification of Infinite-State Systems (AVIS'2004), (2004), pp. 1–11.

[24] D. Boneh, M. Franklin, *Identity based encryption from the Weil pairing*, Advances in Cryptography - Proceedings of Crypto 2001, Vol 2139 of LNCS, (2001), pp. 213–229.

[25] D. Boneh, X. Boyen, E. J. Goh, *Hierarchical identity-based encryption with constant size ciphertext*, in Proc. EUROCRYPT, in Lecture Notes in Computer Science, vol. 3494. Berlin, Germany: Springer, (2005), pp. 440–456.

[26] J. Bonneau, *The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords,* in IEEE S & P. IEEE Computer Society, (2012), pp. 538–552.

[27] X. Boyen, *Hidden credential retrieval from a reusable password.*, In: Proceedings of the 4th International Symposium on Information, ACM, (2009), pp. 228–238.

[28] V. Boyko, P. MacKenzie, S. Patel, *Provably secure password-authenticated key exchange using diffie-hellman*, In EUROCRYPT'00, volume 1807 of LNCS, Springer, (2000), pp. 156–171.

[29] M. Broz, V. Matyás, *Selecting a new key derivation function for disk encryption*, International Workshop on Security and Trust Management. Springer, Cham, (2015), pp. 185–199.

[30] M Burrows, M. Abadi, R. M. Needham, *A logic of authentication*, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 426, 1871, (1989), pp. 233–271.

[31] J. Camenisch, A. Lehmann, G. Neven, *Optimal distributed password verification.* In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, (2015), pp. 182–194.

[32] J. Cao, L. Xu, R. Abdallah, W. Shi, *EdgeOS_H: A home operating system for Internet of everything*, In Proc. 37th Int. Conf. Distrib. Comput. Syst., (2017), pp. 1756–1764.

[33] N. Chen, R. Jiang, *Security Analysis and Improvement of User Authentication Framework for Cloud Computing*, Journal of Networks, **9(1)**, (2014), pp. 198–203.

[34] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, *Deciding the Security of Protocols with Commuting Public Key Encryption.*, In Proc. ARSPA'04. Electronic Notes in Theoretical Computer Science 125(1), (2005), pp. 55–66.

[35] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, L. Vigneron, *Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents.* In Proc. FST TCS'2003, LNCS 2914 Springer, (2003), pp. 124–135.

[36] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, L. Vigneron. *Extending the Dolev-Yao Intruder for Analyzing an Unbounded Number of Sessions.* In Proc. CSL'2003, LNCS 2803 Springer, (2003), pp. 128–141.

[37] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani. *An NP Decision Procedure for Protocol Insecurity with XOR.* Theoretical Computer Science 338(1-3), (2005), pp. 247–274.

[38] Y. Chevalier, M. Rusinowitch, *Combining Intruder Theories.*, In Proc. ICALP 2005, LNCS 3580 Springer, (2005), pp. 639–651.

[39] H. Y. Chien, J. K. Jan, Y. M. Tseng, *An efficient and practical solution to remote authentication smart card*, Computers & Security, **21(4)**, (2002), pp. 372–375.

[40] A. J. Choudhury, P. Kumar, M. Sain, *A Strong User Authentication Framework for Cloud Computing*, Proceedings of IEEE Asia -Pacific Services Computing Conference,(2011), pp. 110–115.

[41] W. Diffie, M. Hellman. *New directions in cryptography*, IEEE Transactions on Information Society, 22(6), (1976), pp. 644–654.

[42] D. Dolev, A. Yao. *On the security of public key protocols*, IEEE Transactions on information theory, 29(2), (1983), pp.198–208.

[43] P. H. Drielsma, S. Mödersheim, *The ASW Protocol Revisited: A Unified View.*, In Proc. ARSPA'04. Electronic Notes in Theoretical Computer Science 125(1), (2005), pp. 141–156.

[44] M. Dürmuth, T. Kranz, *On Password Guessing with GPUs and FPGAs,* in PASSWORDS'14, (2014), pp. 19–38.

[45] A. Escobar, C. Meadows, J. Meseguer, *A rewriting-based inference system for the NRL Protocol Analyzer and its meta-logical properties*, Theoretical Computer Science, **367(1-2)**, (2006), pp. 162–202.

[46] J. Folláth, A. Huszti, A. Pethő, *Designin asymmetric authentication system*, In: Proceedings of the 7th ICAI Conference, Eger, (2007), pp. 53–61.

[47] W. Ford, B.S. Kaliski, *Server-Assisted Generation of a Strong Secret from a Password*, Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, (2000), pp. 176–180.

[48] P. A. Fouque, M. Tibouchi, *Deterministic Encoding and Hashing to Odd Hyperelliptic Curves.* In: Pairing-Based Cryptography - Pairing 2010., (2010), pp. 265–277.

[49] M. Fránik, M. Cermák, *Serious flaws found in multiple smart home hubs: Is your device among them?*, https://www.welivesecurity.com/2020/04/22/serious-flaws-smart-home-hubs-is-your-device-among-them/, (2020)

[50] D. Freeman, *Pairing-based identification schemes*, Hewlett-Packard Laboratories, Report no. HPL-2005-154, (2005)

[51] T. Genet, F. Klay, *Rewriting for cryptographic protocol verification.*, In Proc. CADE'00, LNCS 1831, Springer, (2000), pp. 271–290.

[52] T. Genet, V. Viet Triem Tong, *Reachability Analysis of Term Rewriting Systems with Timbuk.*, In Proc. LPAR'01, LNCS 2250, (2001), pp. 695–706.

[53] T. Gu, P. Mohapatra, *Bf-iot: Securing the iot networks via fingerprinting-based device authentication,* In: 2018 IEEE 15Th international conference on mobile ad hoc and sensor systems (MASS). IEEE, (2018), pp. 254–262.

[54] Hashcat, *hashcat - advanced password recovery*, http://hashcat.net/, Accessed: 30/01/2022.

[55] V. H. Hoang, E. Lehtihet, Y. Ghamri-Doudane, *Password-based authenticated key exchange based on signcryption for the Internet of Things.*, In 2019 Wireless Days (WD) IEEE., (2019), pp. 1–8.

[56] https://docs.microsoft.com/en-us/azure/virtual-machines/acu Accessed: 30/01/2022

[57] A. Huszti, N. Olah, *A simple authentication scheme for clouds*, Proceedings of IEEE Conference on Communications and Network Security (CNS), (2016), pp. 565–569.

[58] A. Huszti, N. Olah, *Security analysis of a cloud authentication protocol using applied pi calculus.*, International Journal of Internet Protocol Technology, 12(1), (2019), pp. 16–25.

[59] A. Huszti, N. Olah, *Provably Secure Scalable Distributed Authentication for Clouds.*, International Conference on Cryptology and Network Security,Springer, Cham, (2020), pp. 188–210.

[60] A. Huszti, Sz. Kovács, N. Olah, *Scalable, password-based and threshold authentication for smart homes.*, Int. J. Inf. Secur. 21, https://doi.org/10.1007/s10207-022-00578-7, (2022), pp. 707— 723.

[61] M. S. Hwang, L. H. Li, *A new remote user authentication scheme using smart cards*, IEEE Transactions on Consumer Electronics, 46(1), (2000), pp. 28–30.

[62] D. Işler, A. Küpçü, *Threshold single password authentication.*, ESORICS Data Privacy Management, Cryptocurrencies and Blockchain Technology, (2017), pp. 143–162.

[63] D. Işler, A. Küpçü, *Distributed Single Password Protocol Framework.*, IACR Cryptol. ePrint Arch., 976., (2018),

[64] W. Iqbal, H. Abbas, B. Rauf, Y. Abbas, F. Amjad, A. Hemani, *PCSS: Privacy Preserving Communication Scheme for SDN Enabled Smart Homes.* IEEE Sensors Journal, (2021)

[65] S. Jarecki, A. Kiayias, H. Krawczyk, *Round-optimal password-protected secret sharing and T-PAKE in the password-only model*, In International Conference on the Theory and Application of Cryptology and Information Security, Springer, Berlin, Heidelberg. (2014), pp. 233–253.

[66] Y. Jiang, Q. Huang, C. Zhu, Y. Wang, J. Shen, *An Efficient Key Agreement Protocol for Secure Group Communications Using Periodic Array.*, In 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID) IEEE., (2019), pp. 36–40.

[67] J. Katz, P. MacKenzie, G. Taban,V. Gligor, *Two-server password-only authenticated key exchange*, In International Conference on Applied Cryptography and Network Security, Springer, Berlin, Heidelberg. (2005), pp. 1–16.

[68] J. Katz, R. Ostrovsky, M. Yung, *Efficient password-authenticated key exchange using human-memorable passwords.*, In EUROCRYPT 2001. Springer, (2001), pp. 475–494.

[69] J. Kelsey, B. Schneier, C. Hall, D. Wagner, *Secure applications of low-entropy keys*, In: Okamoto E., Davida G., Mambo M. (eds) Information Security. ISW 1997. Lecture Notes in Computer Science, vol 1396. (1998), pp. 121–134.

[70] M. A. Khan, K. Salah, *IoT security: Review, blockchain solutions, and open challenges*, Future Generation Computer Systems, 82, (2018), pp. 395–411.

[71] F. Kiefer, M. Manulis, *Blind password registration for verifier-based PAKE.*, In: Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography. (2016), pp. 39–48.

[72] F. Kiefer, M. Manulis, *Blind password registration for two-server password authenticated key exchange and secret sharing protocols.*, In: International Conference on Information Security. Springer, Cham, (2016), pp. 95–114.

[73] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, *DDoS in the IoT: Mirai and other botnets*, Computer, 50(7) , (2017), pp. 80–84.

[74] H. Krawczyk, P. Eronen, *HMac-based extract-and-expand key derivation function (HKDF)*, RFC 5869, (2010)

[75] H. Krawczyk, *The opaque asymmetric pake protocol.* Internet-Draft, (2020), https://datatracker.ietf.org/doc/pdf/draft-irtf-cfrg-opaque-08 Accessed: 30/07/2022.

[76] W. C. Ku, S. M. Chen, *Weaknesses and improvements of an efficient password based remote user authentication scheme using smart cards*, IEEE Transactions on Consumer Electronics, **50(1)**, (2004), pp. 204–207.

[77] G. Kurtz, D. Alperovitch, E. Zaitsev, *Hacking exposed: Beyond the Malware*, RSA 2015 (slide deck), https://www.rsaconference.com/writable/presentations/ file_upload/expt10_hackingexposedbeyond-themalware.pdf, (2015), Accessed: 30/01/2022

[78] R. Küsters, T. Truderung, *Reducing Protocol analysis with XOR to the XOR-free case in the Horn theory based approach*, 15th ACM conference on Computer and communications security (CCS'08), (2008), pp. 129–138.

[79] Y. Lee, S. Kim, D. Won, Enhancement of two-factor authenticated key exchange protocols in public wireless LANs. Computers & electrical engineering 36.1, (2010), pp. 213–223.

[80] LHN: Zipato Smart Home Hub Exploits Discovered That Could Allow An Attacker to Open iOT Door Locks *https://latesthackingnews.com/2019/07/05/hackers-can-exploit-zipato-smart-home-hub-flaws-to-break-in/, (2019), Accessed: 30/01/2022.*

[81] L. Lin, X. Liao, H. Jin, P. Li, *Computation Offloading Toward Edge Computing*, In Proceedings of the IEEE, vol. 107, no. 8, doi: 10.1109/JPROC.2019.2922285, (2019), pp. 1584–1607.

[82] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, X. Fu, *Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System*, In IEEE Internet of Things Journal, vol. 4, no. 6, doi: 10.1109/JIOT.2017.2707465, (2017), pp. 1899–1909.

[83] G. Lowe, *Breaking and fixing the Needham-Schroeder public key protocol using FDR.*, In Tools and Algorithms for the Construction and Analysis of Systems, Springer-Verlag, (1996), pp. 147–166.

[84] J. Ma, W. Yang, M. Luo, N. Li, *A Study of Probabilistic Password Models*, in IEEE S&P, (2014), pp. 689–704.

[85] P. MacKenzie, S. Patel, R. Swaminathan, *Password-authenticated key exchange based on RSA*, International conference on the theory and application of cryptology and information security. Springer, Berlin, Heidelberg, (2000), pp. 599–613.

[86] P. MacKenzie, T. Shrimpton, M. Jakobsson, *Threshold password-authenticated key exchange*, In CRYPTO 2002. Springer, (2002), pp. 385–400.

[87] S. Marksteiner, V. J. Exposito Jimenez, H. Vallant, H. Zeiner, *An Overview of Wireless IoT Protocol Security in the Smart Home Domain*, In: 2017 Joint 13th CTTE and 10th CMI Conference on Internet of Things Business Models, Users, and Networks, Copenhagen, (2017), pp. 1–8.

[88] C. A. Meadows, *The NRL Protocol Analyzer: An Overview*, Journal of Logic Programming, **26(2)**, (1996), pp. 113–131.

[89] A. J. Menezes, T. Okamoto, S. A. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Transactions on information Theory, **39(5)**, (1993), pp. 1639–1646.

[90] R. C. Merkle, *A Digital Signature Based on a Conventional Encryption Function*, Advances in Cryptology - CRYPTO '87, Lecture Notes in Computer Science, **293**, (1987), pp. 369–378.

[91] P. L. Montgomery, *Speeding the Pollard and Elliptic Curve Methods of Factorization*, Mathematics of Computation, **48(177)**, (1987), pp. 243—264.

[92] R. Needham, M. Schroeder, *Using encryption for authentication in large networks of computers*, Communications of the ACM, 21 (12), (1978), pp. 993–999.

[93] D. M. Nessett, *A Critique of the Burrows, Abadi, and Needham Logic,* Operating System Review, v. 20, n. 2, (1990), pp. 35-–38

[94] OpenStack Security Guide [online] https://docs.openstack.org/security-guide/index.html, Accessed: 30/01/2022.

[95] Openwall, *John the Ripper password cracker*, http://www.openwall.com/john/, Accessed: 30/01/2022.

[96] Y. M. Park, S. K. Park, *Two factor authenticated key exchange (TAKE) protocol in public wireless LANs,* IEICE Transactions on Communications, E87-B(5), (2004), pp. 1382–1385.

[97] C. Percival, S. Josefsson, *The scrypt password-based key derivation function*, IETF Draft URL: http://tools. ietf. org/html/josefsson-scrypt-kdf-00. txt, (2016), Accessed: 30/01/2022

[98] D. Pointcheval, S. Zimmer, *Multi-factor authenticated key exchange,* In Steven M. Bellovin and Rosario Gennaro, editors, Applied Cryptography and Network Security (ACNS) 2008, LNCS, volume 5037 Springer, (2008), pp. 277–295.

[99] M. D. Raimondo, R. Gennaro, *Provably secure threshold password-authenticated key exchange*, International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, (2003), pp. 507–523.

[100] M. M. Rathore, E. Bentafat, S. Bakiras, *Smart Home Security: A Distributed Identity-Based Security Protocol for Authentication and Key Exchange*, In: 2019 28th International Conference on Computer Communication and Networks (ICCCN). IEEE, (2019), pp. 1–9.

[101] RFC 4120: The Kerberos Network Authentication Service (V5) [online] https://tools.ietf.org/html/rfc4120, Accessed: 30/01/2022.

[102] RFC 4422: Simple Authentication and Security Layer [online] https://tools.ietf.org/html/rfc4422, Accessed: 30/01/2022.

[103] A. W. Roscoe, *The Theory and Practice of Concurrency.* Prentice Hall, (1998)

[104] P. Ryan, S. Schneider, *Modelling and Analysis of Security Protocols.* Addison-Wesley, (2001)

[105] M. E. S. Saeed, Q. Y. Liu, G. Y. Tian, B. Gao, F. Li, *AKAIoTs: authenticated key agreement for Internet of Things.* Wireless Netw 25, (2019), pp. 3081–3101.

[106] Samsung SmartCam. *https://www.exploitee.rs/index.php/%20Samsung%20SmartCam%E2%80%8B#Fixing%20Password%20Reset%20.22Pre-Auth.22, August 2014.*

[107] B. Schmidt, S. Meier, C. Cremers, D. Basin, *Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties*, 25th IEEE Computer Security Foundations Symposium(CSF'12), (2012), pp. 78–94.

[108] A. Shamir, *Identity-based cryptosystems and signature schemes*, Workshop on the theory and application of cryptographic techniques. Springer, Berlin, Heidelberg, (1984), pp. 47–53.

[109] J. Shen, S. Chang, J. Shen, Q. Liu, X. Sun, *A lightweight multilayer authentication protocol for wireless body area networks,* Future Generation Computer Systems, 78, (2018), pp. 956–963.

[110] M. Soria-Machado, D. Abolins, C. Boldea, K. Socha, *Kerberos Golden Ticket Protection, Mitigating Pass-the-Ticket on Active Directory*, CERT-EU Security Whitepaper 2014-007, (2016), Accessed: 30/01/2022

[111] M. V. Steen, A. Tanenbaum, *Distributed systems principles and paradigms.*, Network, 2, 28., (2002)

[112] R. Trimananda, A. Younis, B. Wang, B. Xu, B. Demsky, G. Xu, *Vigilia: Securing Smart Home Edge Computing,* 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, (2018), pp. 74–89.

[113] N. Provos, D. Mazieres, *Bcrypt algorithm.*, In USENIX, (1999)

[114] Á. Vécsi, A. Bagossy, A. Pethő, *Cross-platform Identity-based Cryptography using WebAssembly,* Infocommunications, 31., (2019), pp. 31–38.

[115] Á. Vécsi, A. Pethő, *Formal language identity-based cryptography*, Rad Hrvatske akademije znanosti i umjetnosti: Matematičke znanosti, (25), (2021), pp. 143 – 159

[116] G. Wassermann, *ZyXEL NBG-418N, PMG5318-B20A and P-660HW-T1 routers contain multiple vulnerabilities.* http://www.kb.cert.org/vuls/id/870744, (2015)

[117] T. Wu, Z. Lee, M. S. Obaidat, S. Kumari, S. Kumar, C. Chen, *An authenticated key exchange protocol for multi-server architecture in 5G networks.* IEEE Access 8, https://doi.org/10.1109/ACCESS.2020.2969986, (2020), pp. 28096–28108.

[118] F. F. Yao, Y. L. Yin, *Design and analysis of password-based key derivation functions*, Cryptographers' Track at the RSA Conference. Springer, Berlin, Heidelberg, (2005), pp. 245–261.

[119] I. Zavalyshyn, N. O. Duarte, N. Santos, *HomePad: A privacy-aware smart hub for home environments*, In Proc. IEEE/ACM Symp. Edge Comput., (2018), pp. 58–73.

A. függelék

List of papers of the author and citations to them

1. A. Huszti, N. Oláh, *A simple authentication scheme for clouds.*, In proceedings of 2016 IEEE Conference on Communications and Network Security (CNS), IEEE, (2016), pp. 565–569.

    - M. Howlader, M. Rahman,*User attribute aware multi-factor authentication framework for cloud based systems*, (2018).
    - N. Kumar, J. K. Samriya, *EAAP: Efficient Authentication Agreement Protocol Policy for Cloud Environment.*, In International Conference on Next Generation Computing Technologies, Springer, Singapore, (2018), pp. 311 – 320
    - H. Al-Refai, K. Batiha, A. M. Al-Refai, *An Enhanced User Authentication Framework in Cloud Computing.*, International Journal of Network Security & Its Applications (IJNSA), Vol, 12., (2020), pp. 59 – 75
    - X. Yin, J. He, Y. Guo, D. Han, K. C. Li, A. Castiglione,*An efficient two-factor authentication scheme based on the Merkle tree*, Sensors, 20(20), 5735., (2020).
    - Á. Vécsi, A. Pethő, *Formal language identity-based cryptography*, Rad Hrvatske akademije znanosti i umjetnosti: Matematičke znanosti, (25), (2021), pp. 143 – 159

2. A. Huszti, N. Oláh, *Identity-Based Cloud Authentication Protocol*, In The 11th Conference of PhD Students in Computer Science, Volume of short papers (2018), pp. 33–36.

3. A. Huszti, N. Oláh, *Security analysis of a cloud authentication protocol using applied pi calculus*, International Journal of Internet Protocol Technology, vol. 12, no. 1, (2019), pp. 16 – 25. SJR: Q4

    - X. Chen, H. Deng, *Efficient Verification of Cryptographic Protocols with Dynamic Epistemic Logic*, Applied Sciences, 10(18), 6577 (2020)
    - Á. Vécsi, A. Pethő, *Formal language identity-based cryptography*, Rad Hrvatske akademije znanosti i umjetnosti: Matematičke znanosti, (546= 25) (2021), pp. 143 – 159
    - F., Dang, W., Zhang, H. Liu, *Analysis of Key Technologies of Cloud Computing Security Based on Trust Model*, International Conference on Cognitive based Information Processing and Applications (CIPA 2021), Springer, Singapore. (2022), pp. 404 – 412
    - Q. Fan, Y. Chen, Y. Wen, M. Luo, *Eland: An Efficient Lightweight Anonymous Authentication Protocol Applied to Digital Rights Management System*, Journal of Internet Technology, 23(2), (2022), pp. 267–278
    - U. Verma, D. Bhardwaj, *Centralised and distributed authentication scheme in internet of things: review and outlook*, International Journal of Internet Technology and Secured Transactions, 12(2), (2022), pp. 127–160

4. A. Huszti, N. Oláh, *Provably Secure Authenticated Key Agreement with Key Confirmation for Distributed Systems*, In WCST Proceedings of the 14th International Conference for Internet Technology and Secured Transactions, (ICITST - 2019), Infonomics Society (2019), pp. 69 – 74.

5. A. Huszti, N. Oláh, *Provably Secure Scalable Distributed Authentication for Clouds*, Lecture Notes In Computer Science 12579., Chapter 10, (2020), pp. 188–210. CORE: B

6. A. Huszti, Sz. Kovács, N. Oláh, *Hybrid anonymous message broadcast for VANETs*, In 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE, Piscataway (2021), pp. 103 – 108. CORE: B

7. A. Huszti, Sz. Kovács, N. Oláh, *Scalable, password-based and threshold authentication for Smart Homes*, Int. J. Inf. Secur. 21, https://doi.org/10.1007/s10207-022-00578-7, (2022), pp. 707 – 723. SJR: Q2, IF: 2.067

8. Cs. Bertók, A. Huszti, Sz. Kovács, N. Oláh, *Provably Secure Identity-Based Remote Password Registration*, to appear in Publicationes Mathematicae Debrecen. SJR: Q2, IF: 0.636

B. függelék

List of talks of the author

1. Securing cloud authentication, *International Conference on Applied Informatics,* Eger, Hungary, 2017.

2. DECAP-Distributed Extensible Cloud Authentication Protocol, *Cryptacus: Workshop & MC meeting,* Nijmegen, Netherlands 2017.

3. Provably Secure Authenticated Key Agreement with Key Confirmation for Distributed Systems, *Central European Conference on Cryptology* , Smolenice, Slovakia, 2018.

4. Security Analysis of Identity-based Password Registration for Distributed Systems, *OGIK 2019,* Budapest, Hungary, 2019.

5. Provably Secure Authenticated Key Agreement with Key Confirmation for Distributed Systems, *Central European Conference on Cryptology,* Telc, Czech Republic, 2019.

6. Provably Secure Authenticated Key Agreement with Key Confirmation for Distributed Systems, *International Conference for Internet Technology and Secured Transaction (ICITST-2019)* , London, United Kingdom, 2019.

7. Identity-based Password Registration for Clouds, *The 11th International Conference on Applied Informatics,* Eger, Hungary, 2020.

8. Provably Secure Scalable Distributed Authentication for Clouds, *19th International Conference on Cryptology and Network Security*, Online, 2020.

9. Biztonságos autentikáció okos otthonokra *ADA 2020* Online, 2020.

10. Provably secure authentication for smart homes, The 1st Conference on Information Technology and Data Science Debrecen, Hungary, 2021.

11. Scalable, password-based and threshold authentication for Smart Homes, *Central European Conference on Cryptology* , Online, 2021.

12. Secure Blind Password Registration, *Central European Conference on Cryptology* , Smolenice, Slovakia, 2022.

C. függelék

Acknowledgements

I would like to thank Dr. Andrea Pintér-Huszti for being an excellent supervisor. I am grateful for the excellent guidance, a lot of help and ideas, and a lot of care.

In addition, I want to thank Dr. Attila Pethő for always inspiring me in my research.

Thanks Dr. Csanád Bertók and Szabolcs Kovács for the collaborative work on research papers that are included in this dissertation.

Finally, I thank my father, my mother, my siblings, and last but not least, my wife for their love and patience throughout working on my thesis.

D. függelék

Köszönetnyilvánítások

Köszönöm Dr. Pintér-Huszti Andreának, hogy kiváló témavezetőm volt. Hálás vagyok a kiváló útmutatásért, a sok segítségért és ötletért, illetve a rengeteg törődésért.

Emellett szeretném megköszönni Dr. Pethő Attilának, hogy mindig inspirált a kutatásom során.

Köszönöm Dr. Bertók Csanád és Kovács Szabolcs társszerzőim kooperatív munkáját, hogy segítettek a disszertációban szereplő cikkek elkészítésében.

Végül, de nem utolsó sorban, köszönöm édesapám, édesanyám, a testvéreim és a feleségem végtelen türelmét és szeretetét, támogatásukkal lehetővé tették e disszertáció elkészülését.