Securing data access and exchanges in a heterogeneous ecosystem : An adaptive and context-sensitive approach

Van-Hoan Hoang

PhD Thesis

Van-Hoan HOANG

January 27 2022

THÈSE DE DOCTORAT

en vue de l'obtention du grade de

DOCTEUR DE LA ROCHELLE UNIVERSITÉ

Spécialité: Informatique et Applications

École Doctorale: EUCLIDE

Présentée par

Van-Hoan HOANG

Securing data access and exchanges in a heterogeneous ecosystem: An adaptive and context-sensitive approach

---

Devant la commission d'examen formée de :

Frédéric Cuppens	*Professeur à Polytechnique Montréal*	Rapporteur
Akka Zemmari	*Professeur à l'Université de Bordeaux*	Rapporteur
Hakima Chaouchi	*Professeure à Télécom SudParis*	Examinatrice
Samia Bouzefrane	*Professeure à CNAM*	Examinatrice
Ahmed Serhrouchni	*Professeur à Télécom Paris*	Examinateur
Damien Sauveron	*Maître de Conférences à l'Université de Limoges*	Examinateur
Elyes Lehtihet	*Innovation Manager à Oodrive*	Encadrant
Yacine Ghamri-Doudane	*Professeur à La Rochelle Université*	Directeur de thèse

Abstract

Cloud-based data storage and sharing services have been proven a successful paradigm for both individual and organizational purposes since the last decades. The underlying business model of such services helps users not to expensively spend on hardware to store data while they are still able to access data anywhere and whenever they desire in possession of any devices with an Internet connection. In addition, service providers can attract an increasing number of users due to the low cost of highly convenient services, resulting in a growth in revenue. All the provided advantages, however, come at the expense of the security that plays a vital role in protecting both the users and their resources.

In terms of outsourced data, service providers responsibly take security measures to ensure data availability and data confidentiality. While data availability serves the main purpose of the service, data confidentiality protection is also highly needed to protect sensitive data from any potential leakage. Generally, data should be encrypted while being stored and only available to authorized users. In fact, data confidentiality can be achieved in different ways depending on the chosen encryption approach. Each encryption approach may offer different security guarantees and user experience. The most adopted approach is that service providers encrypt data and hold decryption keys in secret which will be used on user requests for data retrieval. This relieves users from the key management. To reduce reliance on the service provider, it is recommended encrypting the data on the user side before outsourcing. This, however, causes the complexity in the key management in the collaborative scenario in which a data owner needs to securely distribute the decryption key to a group of authorized users. This process gets more complex when the group is dynamic. That is, users can join and depart from the group at any point in time.

Regarding user privacy, they are required to prove who they are to access their own resources hosted by service providers. This can be achieved via a secure authentication protocol. The authorization of a user is then verified by the service provider based on the user's credential and the predefined access control policy associated to the requested data. However, showing credentials enables the service provider to detect who shares data with whom or even build a profile for each user based on traced information. In some contexts, i.e., data censorship, the service provider can block access or remove all the data pertaining to some specific users. For that reason, user privacy should be an optional requirement when designing the appropriate authentication protocols used in the context of data storage and data sharing.

In this thesis, we aim at constructing efficient and secure data sharing platforms. For this purpose, we not only build authentication protocols and encryption schemes which are the main components composing such platforms but also create a privacy-preserving decentralized

data sharing platform as an independent and complete work. Moreover, the efficiency and effectiveness of all proposals are taken into account when compared to existing solutions which may not support some of the functionalities which our proposals offer.

First of all, conventional secure authentication protocols are mainly based on two categories that are the symmetric setting and the asymmetric setting. In the former one, two or many communicating participants authenticate one another based on a pre-shared secret, which could be a password or a strong cryptographic key. Due to the low entropy, password is human-memorable and not necessarily stored on user devices. Designing a secure password-based authentication protocol is, however, much more complex than a cryptographic key-based protocol. In the asymmetric setting, widely-adopted protocols are mainly built on Public Key Infrastructure (PKI), which falls into two approaches: Certificate Authority (CA)-based and Web of Trust (WoT)-based. The two approaches diverge in the way that trust is established between users. In practice, the architectural simplicity of the CA-based approach eases the design of authentication protocols. Nevertheless, CA-based PKIs have two inherent shortcomings, namely centralized trust and expensive operation cost. On the other hand, the WoT-based approach requires users to obtain signatures on their identities from existing users in the system, thus removing the centralized trust existing in CA-based PKIs. However, in such system, users must have technical knowledge about the underlying system to correctly use it, which restricts its practical adoption. To address the problems in the existing solutions, we propose the first contribution, namely Password-based Signcryption Authenticated Key Exchange, a lightweight and provably secure authentication protocol allowing two parties to mutually authenticate one another based on a human-memorable password and then establish a high entropy session key. The second contribution is a Privacy-Enhancing Decentralized Public Key Infrastructure (DPKI) which is purposely designed to mitigate the centralized trust of CA-based PKIs and give the control over identities to owners. The practicability of the proposed DPKI is demonstrated by constructions of standard authentication and key exchange protocols especially in the contexts where user privacy is needed.

As mentioned above, data should be encrypted on the user side before being outsourced to protect data confidentiality against third parties. In this regard, Ciphertext-Policy Attribute Based Encryption (CP-ABE) has been emerging as a promising outsourcing solution due to its rich set of features including data confidentiality protection, fine-grained data access management, and effective key management. CP-ABE schemes enable a data owner to encrypt data under a desired access structure before outsourcing. Only those who own a certified attribute set which satisfies the access structure can decrypt the encrypted data. Therefore, we propose Forward-Secure Data Outsourcing Based on Attribute-Based Encryption as the third contribution in this thesis. Compared to the existing CP-ABE schemes, our proposal allows efficiently revoking users and their attributes at will. Moreover, it also guarantees the forward-secrecy requirement in which revoked users are no longer able to decrypt the data shared in the past.

As the fourth contribution, we propose a Privacy-Preserving Blockchain-based Data Sharing Platform based on Decentralized Storage Systems as a complete solution. The platform is designed to simultaneously guarantee data confidentiality, data access control policy, data availability, and user privacy. In addition to data confidentiality and data access control which are generally basic guarantees offered by any centralized data sharing services, we also aim at enhancing data availability and user privacy. While the former can be achieved by building a platform on effective decentralized storage systems, the latter is obtained by our proposed cryptographic mechanisms.

Acknowledgments

My Ph.D. journey was a challenging and enjoyable adventure that couldn't have been completed without the support of all those to whom I want to give my heartfelt thanks.

Firstly, I would like to express my deep sense of gratitude to my advisor Dr.Elyes Lehtihet for the opportunity he gave to me to do this research, for his continuous support of my Ph.D. research, for his guidance, encouragement, and immense knowledge as well. He always does his best to help me regarding both technical and financial resources. Thanks to his wholehearted supports, I could have chances to meet technical partners as well as attend scientific conferences. I could not have imagined having a better advisor and mentor than him for my doctorate study.

Apart from my advisor, I owe a debt of gratitude to my thesis director Yacine Ghamri-Doudance for his patience, and for his scientific approach with vast knowledge, insightful suggestions, and detailed feedback in all the time of research and writing this thesis. It has been a great honor for me to be his Ph.D. student. His careful and endless guidance is hard to forget throughout my life.

From bottom of my heart, I would also like to acknowledge my colleagues at Oodrive for their wonderful collaboration. I thank them all, one by one, for the great time we share during and after the work. In particular, my sincere thanks go to Frédéric, Saoussen, Kateryna, Amdjed, Abderrazak, David, Sara, and Saly, who have been my inspiration to go to the office during the hardest time of my thesis and share a great amount of enjoyment with me.

I also would like to express my deepest appreciation to the members of my PhD committee who patiently devoted their efforts in reading and reviewing this dissertation. My sincere thanks goes to Prof. Frédéric Cuppens, Prof. Akka Zemmari for being reviewers and members of the jury. Their insightful comments and encouragement inspired me to widen my research from various perspectives. I would like to thank the rest of my thesis committee: Prof. Hakima Chaouchi, Prof. Samia Bouzefrane, Prof. Ahmed Serhrouchni, Dr. Damien Sauveron for their interest, involvement and time.

Last but not least, words cannot express how grateful I am to my beloved family: my father, my mother, my elder sister, my brother-in-law, my younger brother and my girlfriend: Cu, Hue, Huyen, Phong, Hung, and Thanh-Huyen. They always give me the unconditional love, and tremendous supports throughout this thesis and my life in general. They are always there for me.

List of Figures

Contents

Chapter 1

Introduction

Contents

Cloud-based services have witnessed a growing interest due to the business model that benefits both end users and service providers since the last decades. Such operational benefits continually accelerate the growth and the development of the market. Indeed, a great number of end users and organizations have been moving towards cloud-based storage solutions to avoid spending on expensive hardware that is in charge of locally storing a huge volume of data. Traditionally, in addition to the cost of hardware, users are usually responsible for all maintenance and security costs. Indeed, storage hardware needs to be securely protected against any potential risks not to leak the data stored on it to the outside world. However, if hardware has been corrupted due to any internal hardware-related issues, users have no way of recovering the stored data. By contrast, cloud-based solutions allow users to use any devices with the Internet connection to access their own data hosted by service providers. Moreover, they can also share data with other users in the system without fear of data loss. That is, service providers often deploy internally distributed systems to replicate users' data on multiple nodes physically located at different locations to offer them a high level of data redundancy at a reduced cost. All these conveniences, however, come at the cost of data security and user privacy as service providers are in charge of taking security measures to ensure the safety of data and users. Intuitively, user identity verification is performed by the service provider before the user having access to the requested data. If the verification succeeds, the user can retrieve the data at least in the encrypted form. More clearly, the verification is often realized via a secure mutual authentication and key exchange protocol. The latter allows two communicating entities to authenticate each other then negotiate a secret key which is used to secure then subsequent communications. Afterwards, the authorized user is granted access to the data. If the data has been encrypted before being outsourced, the retrieved data is evidently still encrypted. However, if the data encryption mechanism is carried out by the service provider, the received data is in the plain from transferred via the established secure channel. With that said, when designing an effective data storage and sharing platform, there is an obvious trade-off between data confidentiality and user experience to deal with. Most traditional platforms

take an approach in which service providers are responsible for encrypting and decrypting data. This approach will relieve users from dealing with complex key management especially when data is shared between a dynamic group of users. However, this also requires users to fully trust the service provider for data confidentiality.

In this regard, we envision two necessary modular components for building a secure data storage and sharing platform which is the main objective of the thesis. These components are authentication and key exchange protocols and secure data sharing schemes. Conceptually, the authentication and key exchange protocols help verify user identities as well as establish a secure communication channel between users and service providers over public environments. On the other hand, the data sharing schemes aim at facilitating specifying data access policies over outsourced data and protecting data confidentiality against unauthorized parties so that only authorized users are able to read the plain data. Optionally, these two components could be designed to support user privacy as well.

1.1 Towards Trusted and Privacy-Preserving Data Sharing Platform

1.1.1 Authenticated Key Exchange Protocols

Authenticated key exchange protocols represent the most important building block to ensure the safety of communications between remote users. Such protocols allow users to know with whom they are communicating before engaging in exchanging information. To effectively integrate such protocols in any larger infrastructure, i.e, a data storage and sharing platform, one needs to assess them basing on different factors, including *security*, *scalability*, and *user experience*. In practice, the widely-used authentication protocols mainly fall into two different categories, those using a symmetric setting and those using an asymmetric setting [1–3]. We briefly present functional models, advantages, and shortcomings of each of these categories:

Technically, the symmetric setting indicates scenarios where two or many participants agree upon a shared secret before running a protocol execution. The secret is used for the participants to authenticate one another then negotiate a session key which will be used to secure communications between them. Clearly, the protocols in this setting may offer *security* and *user experience* but have a difficulty satisfying *scalability*. The reason is that the participants in the protocols must know each other beforehand to negotiate a secret necessary to run the protocols. The shared secret can simply be a password, a PIN code, or a strong cryptographic key.

- Password-based authenticated key exchange protocols: Despite the underlying weaknesses pertaining to the low entropy of chosen passwords, these protocols still remain as one of the most practical protocols. These are convenient for both ordinary users and service providers due to their natural simplicity in use and in implementation. As such, in order to make these protocols a standard approach, more security investigations are needed to strengthen the security model to capture all the potential attacks. For this purpose, many security models [4, 5] have been rigorously investigated and proposed in the literature to offer the standards to password-based protocol designs. The complexity of the models, however, often results in decreasing performance, thus affecting *user experi-*

ence. Therefore, an effective protocol design also needs to take protocol performance into consideration.

- Authenticated key exchange protocols based on strong cryptographic keys [6–8]: Protocols in this category generally benefit from the simplicity in design and high security level without fear of dealing with low entropy keys. However, unlike password-based protocols, users are required to securely store keys in local hardware which will be used at the time of running the protocols. That is, the hardware needs to be securely designed to prevent an adversary from easily extracting the stored key. While this requirement is affordable in some specific contexts, it is not in the others when ordinary users do not want to carry such devices to everywhere they tend to perform an authentication.

On the contrary, the asymmetric setting can generally offer *security* and *scalability* while having issues when it comes to *user experience*. Practically, the protocols in this setting are commonly built on top of Public Key Infrastructures (PKI) [9–11], which are classified into two different approaches: the Certificate Authority (CA)-based approach and the Web of Trust (WoT)-based approach. The common role of all PKIs is to issue certificates which attest to the bindings between users' identity data and their corresponding public keys. The possession of the legitimate certificates facilitates the design of authenticated key exchange protocols without the communicating parties having to agree upon a secret beforehand.

- In CA-based PKIs, CAs act as trusted parties to issue and manage user certificates. To prove its identity to other parties, a user simply proves the ownership of the certificate by using the associated private key. Due to the centralization property of such systems, users are required to fully rely on the CAs for all security concerns. Thus, if any of the CAs gets compromised, the whole system will fall apart as the attacker can use the CA's private key to sign fraudulent certificates to impersonate others. For example, a high-profile Dutch corporation, named DigiNotar CA, had a security breach in 2011. This resulted in fraudulent issuing of hundreds of certificates, including a **.google.com* certificate [12]. Besides, the expensive processes in issuing, managing, and revoking certificates are also not convenient for ordinary users.

- Pretty Good Privacy (PGP) [13] is a prominent standard leveraging the Web of Trust (WoT) model [14] to build an entirely decentralized PKI. In this model, users are able to designate others as trustworthy by signing their public key certificates. By doing so, a user accumulates digital signatures for its certificate from entities that have been deemed to be trustworthy. The certificate is then trusted by a third party if the certificate has been signed by a person that the third party entrusts. Clearly, this model offers the decentralization feature in certificate issuance which, however, comes at the cost of user-friendliness. For a user to be widely visible (trusted) on the network, it needs to accumulate signatures given by a large number of well-known entities. In addition, certificate revocation is also a critical issue to be properly solved in such model. In practice, all issued certificates are uploaded onto the key servers which are supposed to play the information role only. If anyone is interested in verifying the public key of a user, they must send a lookup request to one of the key servers to retrieve the desired certificate. Similarly, in order to revoke a certificate, the certificate owner can publish a revocation certificate on the key servers as a request to stop broadcasting its certificate. The log servers are voluntarily run by the PGP community which does not have any economic

incentive. However, even under the assumption that the key servers act honestly and do not hide revocation information from users, delay in certificate synchronization between them may already cause a huge damage to the system. These reasons therefore hinder the adoption of the model in practice which is actually limited to secure email exchanges.

1.1.2 Secure Data Sharing Schemes

Secure data sharing schemes are composed of two tasks: (1) allowing data owners to freely specify access policies as well as authorized users to retrieve the shared data, and (2) protecting data confidentiality against unauthorized users. In practice, most service providers take the simple yet effective approach in which they are in charge of both tasks, thus relieving users from the complexity of key management and technical knowledge requirements. That is, they do not need to handle a complex decryption key distribution while having a convenient user interface to determine with whom they want to share data. There exist, however, two notions that we should clarify and enhance from traditional systems. We define these notions as follows:

- Data Transferability: This means that authorized users are enabled to retrieve the shared data at least in the encrypted form. This notion differs from the data availability in the sense that data could be always available on the cloud but the service provider stops sending data for some unexpected reasons that are out of control of authorized users. This notion seems to be quite obvious and should be satisfied once user identity verification has been finalized by the authenticated key exchange protocols. However, it is not always guaranteed in sensitive contexts where user privacy matters. Indeed, we observe that outsourcing data to a cloud-based storage service partially deprives a data owner of the ownership. Users must rely on the service provider maintaining the consistency and operations in accordance with user-defined access control policy. However, due to legal reasons, such as data censorship, the service can block access to data of an individual or a group of users. Even worse, the service can remove all data related to some specific users. Moreover, a user has to provide its identity to the service for verification before being granted access to the requested data. This leads to the user being traceable in the system. The lack of user privacy enables the service provider to trace all activities of any targeted users or discover the relationship between data stakeholders, i.e., who shares data with whom. This is not desired in many scenarios, for example, in healthcare systems, in which patients prefer concealing their identities while sharing their sensitive Electronic Medical Records (EMR) to healthcare institutions. Despite its importance, user privacy is rarely taken into account in the context of data sharing.

- Data Confidentiality: For security reasons, data is often stored on the cloud in encrypted form. To simplify the complex key management, many service providers usually adopt a strategy in which they control decryption keys and keep them safe in the system. That is, every time an authorized user requests the shared data, the service provider decrypts the data and sends it to the user. Thus, the user does not need to decrypt the data nor store the decryption key on its own side. The communication between two parties is generally secured by the authenticated key exchange protocols. The actual approach works well under the assumption that service providers are fully trusted and do not leak users data and harm user privacy. This is not, however, always guaranteed in real life. In an attempt to reduce the reliance on service providers with respect to data confidentiality, one may

suggest encrypting data on the user side before outsourcing it onto the cloud [15]. This, however, has limitations regarding key management when data is shared with a growing number of users [16]. Intuitively, a data owner must securely generate a private key on its own side every time it wants to share data with different groups. This approach is clearly not suitable for dynamic groups of users in which users can join and leave at anytime as illustrated in Figure 1.1. Therefore, it raises the need for designing a secure data sharing scheme which provides efficient key management for users.

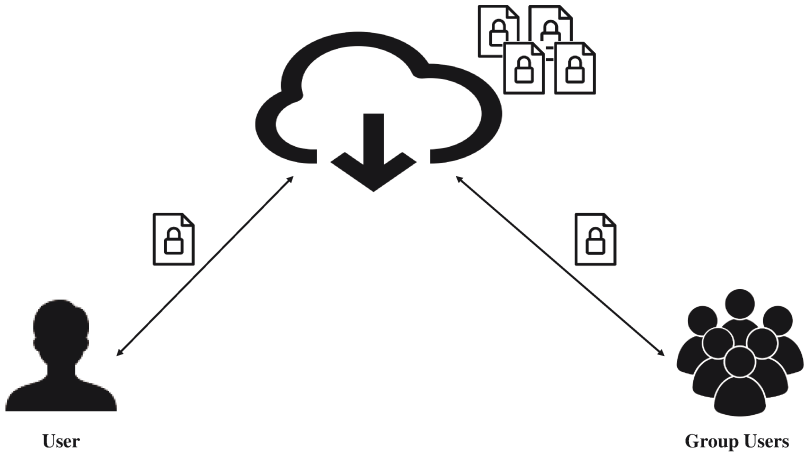


Figure 1.1: Generic group-based data sharing scenario.

1.2 Motivations and Contributions

Architectural designs of data storage and sharing platforms are highly dependent on the security, the privacy, and the data availability that service providers can offer to their users. Such platforms are generally composed on two modular components which are authentication and key exchange protocols and secure data sharing schemes. In this dissertation, we aim to propose the solutions for each of these two components while addressing their inherent drawbacks. Even though they are purposely designed to target the data storage and sharing context, one can still use them in other environments when authentication and key exchange protocols and encryption are needed. Specifically, the proposals aim at tackling the following questions: *How to design an effective and secure password-based authenticated key exchange protocol? How to overcome the centralized trust in the traditional PKIs to implement secure authentication and key exchange protocol? How to design an encryption scheme with underlying effective key management? How to optionally impose user privacy requirement in the construction of a data storage and sharing platform?* And *how to move beyond the centralized approach of traditional services to offer higher levels of data availability?*

First of all, conventional password authentication protocols are generally implemented upon the Transport Layer Security (TLS) protocol [11] which mainly relies on the Certificate Authority (CA)-based PKIs. Specifically, TLS allows a user's application to authenticate with a server and negotiate a session key. Then, the password of the user is sent over the channel which is secured by the session key to the server for verification. The two inherent shortcomings of traditional PKIs are centralized trust and expensive execution cost. First, the security of a PKI is

totally put on the Certificate Authorities (CAs), which are structured in a hierarchical manner. The compromise of any of these CAs results in the collapse of the whole system. Second, the PKI requires huge resources for construction and maintenance over time. Moreover, the size of a certificate and computationally expensive operations of a PKI-based authentication protocol are not favorable for resource-constrained devices. To address these challenges, we propose Password-based Signcryption Authenticated Key Exchange, a lightweight protocol allowing two parties to mutually authenticate one another based on a human-memorable password and then establish a high entropy session key. The latter will be used to secure communications between them. This is a response to the first question *How to design an effective and secure password-based authenticated key exchange protocol* ? The proposed protocol is then provably secure in the Find-then-Guess model, a standard security model.

Second, the next contribution relates to the questions of *How to overcome the centralized trust in the traditional PKIs to implement secure authenticated key exchange protocol*? And *How to optionally impose user privacy requirements in the construction of a data storage and sharing platform*? For this purpose, we propose a Privacy-Enhancing Decentralized Public Key Infrastructure (DPKI) to mitigate the centralized trust of the traditional PKI design and put the control over identities to the identity owners. The proposal allows a user to register a digital identity with a public blockchain, namely Ethereum. This provides users with a way of managing their identities, such as revocation and update, in a transparent and publicly verifiable manner. This is far more different than traditional PKIs in which all these operations are centrally handled by CAs. By contrast, the transparency of the Ethereum blockchain may make it a central point to correlate all activities of a user. The proposed DPKI incorporates key decoupling and anonymous identity registration mechanisms to offer a strong user privacy guarantee, preventing others from correlating Internet activities to registered user identities.

Third, in centralized storage systems, in order to protect data confidentiality and integrity, data should be encrypted on the user side before being outsourced into the cloud. In this sense, Ciphertext-Policy Attribute Based Encryption (CP-ABE) has been emerging as a promising outsourcing solution due to its rich feature set including data confidentiality protection and fine-grained data access management without relying on the cloud. CP-ABE schemes enable a data owner to encrypt data under a desired access structure before outsourcing it. Only those who own a certified attribute set which matches with the access structure, can decrypt the encrypted data. Despite the advantages over conventional encryption algorithms, attribute and user revocations of CP-ABE schemes remain challenging. Firstly, existing revocation mechanisms cannot thoroughly solve the problem as they raise security issues and increase the computational and communication overheads. Secondly, forward secrecy, in which revoked users are unable to decrypt data shared in the past, is not rigorously taken into account. We investigate these problems and design two CP-ABE schemes that allow both efficient user and attribute revocations. The forward secrecy requirement of the proposed schemes is guaranteed by integrating re-encryption techniques. As such, this contribution answers the question of *How to design an encryption scheme with underlying effective key management*? To demonstrate the feasibility of the proposals, both performance evaluation and security analysis are conducted and provided.

Fourth, we propose a Privacy-Preserving Blockchain-based Data Sharing Platform based on Decentralized Storage System to enhance data confidentiality, data access control, data availability, and user privacy. In fact, the contribution departs from an entirely different approach which leverage the blockchain technology to build the proposed platform on the InterPlanetary

File System (IPFS), a content-addressable peer-to-peer storage system. We consider this contribution as a complete solution for data sharing which incorporates the two above-mentioned components while offering a higher level of data availability. As such, it is the succinct answer to all of the five raised questions.

1.3 Thesis Organization

This dissertation shows how to tackle all the identified issues on secure data sharing on the cloud-based context by improving existing solutions and introducing completely different approaches. The organization of the thesis is as follows:

Chapter 2 introduces the first contribution, named Password-based Signcryption Authenticated Key Exchange (PSKE). The contribution not only aims at improving existing password-based protocols in terms of computational performance and communication but also creating a new paradigm to design such protocols by applying a signcryption scheme. The security of the proposed protocol is then validated in the Find-then-Guess model under standard computational hardness assumptions.

Chapter 3 presents the second contribution, namely Privacy-Enhancing Decentralized Public Key Infrastructure, which purposely removes the centralized trust put on Certificate Authorities in traditional Public Key Infrastructure and mitigates technical problems encountered in the Web of Trust model. In addition, user privacy is taken into account in the construction of the DPKI to facilitate the designs of authentication protocols that also guarantee user privacy requirements. This idea is materialized by a discussion about the adoption of the proposal in different contexts.

Chapter 4 describes Forward-Secure Data Outsourcing Based on Revocable Attribute-Based Encryptionn as the third contribution. The main purposes of this contribution is to strengthen the security of centralized data outsourcing solutions and provide an efficient key management in the group-based sharing. It also features an effective revocation mechanism to handle the dynamics of the group in which members can join and leave at anytime. More specifically, the forward secrecy requirement which prevents revoked users from reading data shared in the past, is guaranteed in the design of the contribution.

Chapter 5 presents a completely different approach to design a secure data storage and sharing platform. Besides data confidentiality and secure data access control, we look to provide high data availability and user privacy in the fourth contribution. Data availability could be obtained by building a platform on a decentralized storage network where nodes are financially incentivized and punished according to their behavior in the system. On the other hand, user privacy in the context of data sharing, to the best of our knowledge, has never been thoroughly solved. In this contribution, we leverage the blockchain technology along with advanced cryptographic mechanisms, such as ring signature, hidden access control policy, and predicate encryption, to provide this privacy requirement without having negative impacts on other factors.

Chapter 2

Password-based Signcryption Authenticated Key Exchange

Contents

2.1 Foreword

Along with the explosive growth of Internet applications during the last decade, the security plays a vital role in protecting users against data leakage. More specifically, before engaging in any exchanges with others, users must ensure that they are actually communicating with the right entities, i.e. other individual users or remote servers, and that their communication must be secured against data alteration, modification, and leakage. To this end, there are many effective approaches relying on various advanced technologies, i.e., Public Key Infrastructure (PKI), Identity-Based Cryptography (IBC) [17], or symmetric key exchange. These technologies can be used as standalone solutions as well as being part of hybrid solutions in combination with other authentication factors for reinforced security. For example, a secure channel is first built based on these technologies over which passwords or biometric information are transmitted between communicating parties. The common shortcoming of these solutions is the need for additional physical equipment to store strong cryptographic secrets. Regarding PKI and IBC, the costs of deployment and maintenance likely make them impractical in some constrained environments.

In this context, Bellovin et al. [18] put forward a new direction of developing authentication protocols only based on low entropy secrets, such as code pin or passwords to achieve a better trade-off between security and user experience. These are generally called Password-based Authenticated Key Exchange (PAKE). By definition, PAKE protocols allow two remote entities to authenticate each other over an adversarially-controlled environment then negotiate a strong cryptographic key to protect their subsequent communication. Arguably, due to the low entropy of passwords, PAKE protocols could be highly vulnerable to classical attacks such as dictionary attacks. If the protocols are not securely designed, any partial information leakage about users' passwords helps adversaries significantly reduce the time of successfully mounting these attacks [19–21]. We state that using cryptographic keys obviously facilitates the design of protocols to eliminate these attacks as they are required to fulfill weaker security models. Moreover, storing a cryptographic key implicitly provides an additional authentication factor as users are required to possess the hardware at the time of executing the protocol. In some cases, these greatly fit user expectations about security but they do not in other cases where users are still satisfied with password-based protocols and feel uncomfortable to physically carry small hardware to anywhere they want to run the protocol. Therefore, to take advantage of the compelling positives of PAKE, there have been many variants proposed in the literature to enhance the security and performance of PAKE. However, as described in 2.2, none of these is fully satisfactory, thus calling for improvements.

In the first contribution, we create a new paradigm to design PAKE-like protocols by leveraging the signcryption technique. The reason for building such a paradigm is to further optimize the protocol execution cost while still keeping the protocol provably secure. Concretely, compared to existing protocols, the proposed Password-based Signcryption Authenticated Key Exchange (PSKE) is also more efficient in terms of communication and computation while being provably secure in the random oracle model. The security and efficiency of PSKE also make it more suitable for constrained environments such as the Internet of Things.

2.2 Related Works

In the literature, the PAKE protocols could be categorized into two classes: the Balanced PAKE (B-PAKE) and the Augmented PAKE (A-PAKE). These are discussed in the following.

2.2.1 Balanced Password Authenticated Key Exchange

B-PAKE takes place in the context that user and intended server share the same password-derived value H(PW) that could be computed by performing a secure one-way function. The parties use this value to authenticate each other. However, when the server is compromised and the password file is leaked, an attacker who knows H(PW) will be able to impersonate the user without knowing the plain-text password of the user.

The first B-PAKE construction protocol is the password-based Encrypted Key Exchange (EKE) that was introduced by Bellovin et al. in [22] with some informal security analyses. The authors covered three different variants including: RSA-EKE, Elgamal-EKE, and DH-EKE. Even though these solutions were found insecure in [23], they became a basis for many following researches.

Jablon et al. [24] derived a new scheme called Simple Password Exponential Key Exchange (SPEKE), which uses a password to generate the base for exponentiation over a cyclic group.

This solution subsequently was also pointed out to be insecure in [25, 26] as it allows an active adversary to test multiple passwords in one dishonest execution. In fact, there is literally no way to prevent active attackers from trying to impersonate a legitimate user with adaptively selected passwords. However, a secure PAKE protocol requires that such an attack only allows the adversary to test one password per session. Moreover, one can mitigate active attacks by putting restrictions on the number of consecutive failed authentication attempts pertaining to a user. In order to formally analyze the security of PAKE protocols, Bellare et al. [4] presented a formal security model for the first time. Then, they also presented a modified EKE protocol, thus called EKE2, and claimed that EKE2 is secure in the ideal-cipher model. Additionally, the authors sketched a generic way to add mutual authentication to all PAKE schemes. In fact, Ballare et al.'s model is extended from [8] which defines a security model for mutual authentication and authenticated key exchange in a two-party symmetric setting. The extension is to deal with attacks which are generally tied to password-based authentication protocols.

Since then, many other studies have been continuously presented which were proved secure in different security models, thus resulting in different protocol performances. For example, Boyko et al. [27] introduced the PAK protocol which is provably secure in the random oracle model under the Decisional Diffie-Hellman assumption. In an attempt to construct secure PAKE protocols in the standard model, the works in [28–33] have made use of the common string model (CRS), which assumes that the communicating parties agree on a securely generated set of public parameters even before running the protocol executions. While the CRS model facilitates the design of the PAKE protocols, the assumption about the secure generation of public parameters seems too strong and is not practically easy to deal with. Indeed, such parameter generation is usually carried out via complex multiparty computation protocols [34] which are expensive in terms of communication and computation.

A practical scheme, called J-PAKE, is presented in [35] by leveraging the zero knowledge proof technique. The protocol is then proved secure in the random oracle model by Abdalla et al. [36] and is added into the OpenSSL library for use. From the computation point of view, J-PAKE requires each party to perform up to 14 modular exponentiations. Our scheme PSKE in this contribution aims at providing better communication and computation costs while keeping the protocol provably secure.

2.2.2 Augmented Password Authenticated Key Exchange

As an enhanced version of B-PAKE, A-PAKE requires servers to keep password verifiers which are computed from users' passwords. The leakage of the verifiers in case of server compromise will not allow an adversary to impersonate the users. Arguably, there is nothing preventing the attacker from conducting time-consuming brute-force attacks based on the verifiers to retrieve the corresponding passwords. However, the main purpose of B-PAKE is to give the servers enough time to react and inform the users about the attacks. Bellovin and Merritt [37] are the first ones to present an A-PAKE scheme with some informal analyses. Briefly, the proposed protocol, called A-DHEKE, requires a server to store both a password-derived value $\mathrm{H}(PW)$ and a password verifier, which is in the form of a public key PK generated from PW for each user. In the first phase of the protocol, $\mathrm{H}(PW)$ is first used to run a B-PAKE execution, which is precisely the DH-EKE protocol as described in [22] so that the two parties agree on a negotiated key K. In the second phase, the user additionally proves the knowledge of PW

by applying a verifiable one-way function F that takes PW and K as inputs. Concretely, the user sends out $\text{Enc}_K(F(K, PW))$ which can only be validated by the password verifier stored on the server. Many other works [24, 30, 37–42] have followed this strategy to propose different protocols proven secure in either the random oracle or standard model. For those secure in the standard models, i.e. [30, 39], they have adapted the CRS model, thus being impractical for use. A work which does not follow this strategy is secure remote password (SRP). The first version of the protocol is SRP-3, which was firstly published in [43]. Unfortunately, this version suffers from various potential attacks including the two-for-one guessing attack. After many changes, the author finally introduced SRP-6, the version that fixes these security concerns. This version nevertheless does not enable an instantiation over elliptic curve (EC) and therefore keeps a high computational complexity.

The PSKE protocol, which we propose in this contribution, does not also follow this two-phase strategy. We encapsulate both key exchange and mutual authentication in one phase to provide more efficient protocol in terms of communication and computation. Later, in the section 2.7, we will demonstrate the efficiency of the PSKE protocol over the presented protocols above as well as its adaptability in resource-constrained contexts.

2.3 Security Model

In this section, we present the security assumptions and the security model which we use to prove the security of PSKE.

2.3.1 Security Assumptions

Considering a multiplicative cyclic group $\mathbb{G}$ =<g> of prime order q where g is its generator, we recall the following security assumptions:

Definition 2.1 (Computational Diffie-Hellman (CDH) Assumption)**.** *Given an instance* (g, g^a, g^b) *where* a *and* b *are randomly picked over* $\mathbb{Z}_q^*$*, the probability* $Succ_{g,\mathbb{G}}^{cdh}(t)$ *that a probabilistic polynomial time adversary can find out* g^{ab} *is negligible within time* t*.*

Definition 2.2 (Decisional Diffie-Hellman (DDH) Assumption)**.** *Given an instance* (g, g^a, g^b, g^c) *where* a, b, c *are randomly selected over* $\mathbb{Z}_q^*$*, the probability* $Succ_{g,\mathbb{G}}^{ddh}(t)$ *that a probabilistic polynomial time adversary can decide whether* $g^{ab} = g^c$ *is negligible within time* t*.*

Definition 2.3 (Gap Diffie-Hellman (GDH) Assumption)**.** *Given the DDH problem is solvable in* $\mathbb{G}$*, the CDH problem is still hard for a probabilistic polynomial time adversary. The GDH assumption states that* $Succ_{g,\mathbb{G}}^{gdh}(t) \leq \varepsilon$ *with* t/ε *not too high.*

2.3.2 Security Model

Due to the widespread use of secure authentication and key exchange protocols in network security standards [1–3], it is necessary to have an extensively investigated security model to capture all the potential attacks. Bellare and Rogaway [8] presented the first formal security model for entity authentication and key exchange in the symmetric two-party setting. Later, they proposed an extension to this model to capture the three-party setting [44]. Afterwards,

Blake-Wilson et al. [45] further extended the model to cover the asymmetric setting. Independently, the work in [4] presented extensions to the model to allow for password authentication. The works in [46, 47] introduced models which include both passwords and asymmetric keys. That is, the targeted protocols rely on both user's password and server's public key. In this contribution, we use the extensively investigated security model defined in [4] and its extension [38] to handle the forward secrecy requirement in order to prove the security of PSKE.

PSKE Execution: The protocol is message-driven and takes place between two participants denoted by U (i.e., user) and S (i.e., remote servers) in the presence of an adversary. Upon completion of a protocol execution, the participants agree on a secure session key which will be used to secure their consequent communication. Conceptually, U owns a secret password pw and keeps it in private while S only holds some values derived from pw. Practically, pw is uniformly drawn from a Dictionary of size N. In order to prove the security of the protocol which implies the privacy of the negotiated session key we use a simulation in which the capabilities of an adversary in real attacks are exactly modeled via some oracle queries. At the end of the simulation, the protocol is deemed secure if the adversary only has a negligible advantage in distinguishing the real session key from a random key. Let U^i and S^j respectively denote an instance of U and S during a protocol execution, we detail the oracle queries as follows:

- $Execute(U^i, S^j)$: This query models passive attacks where the adversary is able to eavesdrop exchanges between the user instance U^i and the server instance S^i. The output of this query is the messages exchanged during the protocol execution.

- $Send(A^i, m)$: This query models active attacks by allowing the adversary to send a message m to an instance A^i and then get a response generated appropriately according to the PSKE design. The query $Send(S^j, start)$ allows to initialize an execution of PSKE.

- $Reveal(A^i)$: This query models the misuse of the agreed session key which has been generated from an honest execution between A^i and its communicating partner. If no session key is defined for A^i or a *Test* query, which is defined below, has been sent to A^i or its partner, the output of the query is $\perp$. Otherwise, the output is the session key held by A^i.

PSKE-Security: To prove the semantic security of a session key generated by PSKE, we use the query type called $Test(A^i)$:

- $Test(A^i)$: This query models the adversary's capability of distinguishing a session key from a random one. This query is only asked to a *fresh* instance A^i. In order to respond to the query, the oracle flips a private coin b and then forwards the session key to the adversary if $b = 1$. Otherwise ($b = 0$), the oracle replies with a random key. We note that a *Test* query can only be made at most once by the adversary to a *fresh* instance of a participant. Given a communication between an instance A^i and its partner $A^{i'}$, the instance A^i is considered as *fresh* if the two following conditions are satisfied: (1) A^i had accepted a session key with $A^{i'}$; (2) The adversary has never asked $Reveal(A^i)$ and $Reveal(A^{i'})$ queries.

At the end of the simulation, the objective of the adversary is to correctly guess the hidden bit b in the *Test* query by outputting the bit b'. If $b = b'$, we say that the adversary is capable of

breaking the security of the protocol. We denote by Pr[Succ] the probability of such an event, we measure the advantage of the adversary **PSKE-advantage** as follows:

$$\mathrm{Adv}^{\mathrm{pske}} = 2\mathrm{Pr}[\mathrm{Succ}] - 1.$$

PSKE-Forward Secrecy: The forward secrecy guarantee of PSKE aims at protecting past communications between a pair of participants even in case that their long-term keys will be leaked in the future. That is, there is a negligible probability for the adversary to recover the session keys generated in the past sessions by using the compromised long-term keys. In order to prove this security guarantee, we make use of an additional query, namely *Corrupt*:

- *Corrupt*(A): This query models the physical attack on a participant A and then the adversary is returned with A's long-term key. In case of PSKE, the query receives A's password as the response. We assume that the internal states of all instances of A are not revealed to the adversary.

This new query type, however, requires an extension to the definition of the *freshness* notion. Concretely, a third condition to be satisfied for an instance of a participant to be considered as *fresh*: (3) There is no *Corrupt* query asked by the adversary since the start of the experiment. In other words, the adversary can only make a *Test* query to a *fresh* instance A^i before A is corrupted.

We denote by Pr[Succ] the probability of the event in which the adversary with additional access to the *Corrupt* oracle correctly guesses the hidden bit b used by the *Test* oracle. The **FS-PSKE advantage** of the adversary is defined as:

$$\mathrm{Adv}^{\text{fs-pske}} = 2\mathrm{Pr}[\mathrm{Succ}] - 1.$$

The PSKE protocol is considered as (t, ε)-secure if the advantage of any adversary which is running with time t is smaller than ε.

2.4 Signcryption overview, proposed extension, and security analysis

2.4.1 Signcryption overview

The most widespread application of cryptography is to build a secure channel which offers data confidentiality and authenticity between remote communicating parties over public environments. Basically, these two security requirements can be achieved by composing encryption and signature or Message Authentication Coding (MAC) functions in different ways. Practically, there have been two proposed approaches based on different assumptions pertaining to communicating parties.

The first approach is implemented in the symmetric setting in which two parties trust each other and share a common secret. The realizations of this approach are compositions of an encryption and a MAC function which were usually referred to as authenticated encryption and were extensively studied in the works of [48–51]. More specifically, they introduced different security notions for authenticated encryption schemes and proved whether the proposed

compositions, which are Encrypt-then-MAC (EtM), MAC-then-Encrypt (MtE), and Encrypt-and-MAC (EaM), meet these notions. Moreover, they also studied the amplification of confidentiality achieved in these compositions. From this perspective, EtM is secure against a chosen-ciphertext attack (CCA2) even with the base encryption only secure against chosen-plaintext attack (CPA). Thus, this composition is generically preferable to the others in practice.

The second approach is constructed in the asymmetric setting in which each party has a pair of keys, a public and a private key. The natural analogues to the above compositions in this setting are Encrypt-then-Sign (EtS) and Sign-then-Encrypt (StE). The security of these compositions were also investigated by An et al. [52]. The definition of signcryption was first introduced by Zheng in [53], which is a variant of the Encrypt-and-Sign (EaS) composition. This scheme was originally constructed to achieve better performance compared to the simple sequential compositions like EtS and StE. According to Zheng, the signcryption constructions in [53] may save 50% in computational cost and 85% in communication cost when compared to traditional StE or EtS schemes. Because of the necessity of confidentiality and authenticity in general communication, it is reasonable to design tailored schemes which yield gains in performance. However, due to the complexity of EaS composition, the need for new security notions was raised in [52,54]. In fact, the security of Zheng's scheme was only proven secure ten years after its publication by Baek et al. [55] in the oracle model under the Gap Diffie-Hellman assumption [56]. Since then, signcryption has rapidly become a new paradigm in the field of public key cryptography, leading to many variants built on different computational hardness assumptions [57–66]. In this chapter, signcryption is considered as a building block to construct the PSKE protocol.

Signcryption security: The security of signcryption is composed of two distinct notions, including indistinguishability against chosen-ciphertext attack (IND-CCA2) and existential unforgeability against chosen message attack (UF-CMA). These notions are differently considered in the two-user setting and the multi-user setting under the insider and outsider security models [52, 67]. The multi-user setting describes a context where a sender sends a signcrypted message to different users, creating challenges pertaining to users' identities and requiring more intricate security model. In the context of PSKE, we, however, only consider the two-user setting in which the aim of two communicating parties is to authenticate each other. We explain more clearly about the insider and outsider security models:

- **Outsider security model**: The primary aim of this security model is to capture network attacks. That is, the security of signcryption is protected against an adversarial outsider to the system who does not know either the sender's private key or the recipient's private key with respect to the above-mentioned security notions, which are guaranteed against the adversary.

- **Insider security model**: The primary aim of this model is to capture attacks based on the corrupted party which is either the sender or the recipient of a communication. That is, an adversary manages to corrupt either party and then uses the corrupted one's private key to break the security of the signcryption scheme. In this model, a signcryption scheme needs to guarantee two requirements. On one hand (1), knowing the sender's private key does not allow the adversary to decrypt the messages previously signcrypted by the sender. This requirement is needed in providing forward-secrecy. On the other hand (2), knowing the receiver's private key does not allow the adversary to signcrypt a random message

and prove that the message has been signcrypted by the sender unless the adversary also knows the sender's private key.

Obviously, the insider security model is appealing and actually becomes mainstream when analyzing signcryption schemes. The signcryption scheme used to design the PSKE protocol meets this security model. We remark that non-repudiation which enables any party to publicly verify the validity of a signature on a message to anyone does not exist by default in signcryption design. More specifically, a signcrypted message can be verified by the targeted recipient but this cannot be generally done by a third party that does not directly participate in the communication. We call this feature internal non-repudiation. For example, in the construction proposed in [53], a third party needs the receiver's private key to be able to verify the authenticity of the underlying message. There have been many works in [68, 69] which aim at enabling public non-repudiation. That is, a third party can verify the validity of a signcrypted message without the private key of the receiver. In the context of PSKE, we argue that internal non-repudiation is sufficient as the validity of signcrypted messages from one party only needs to be verified by its communicating partner but not a third party. Therefore, the signcryption scheme used in designing the PSKE protocol does not support this feature, thus avoiding unnecessary overhead in the protocol.

2.4.2 Proposed signcryption

A signcryption scheme is composed of two primary algorithms, namely *signcrypt* and *unsigncrypt*. Figure 2.1 illustrates the system model in which a sender makes use of signcryption to simultaneously add confidentiality and authenticity to data which is then transmitted to a recipient. Conceptually, *signcrypt* involves taking the sender's private key SK_s and the recipient's public key PK_r as inputs to produce signcrypted data. On the other hand, *unsigncrypt* involves unsigncrypting the signcrypted data to obtain the plain data and verify its authenticity by using the sender's public key PK_s and the recipient's private key SK_r.

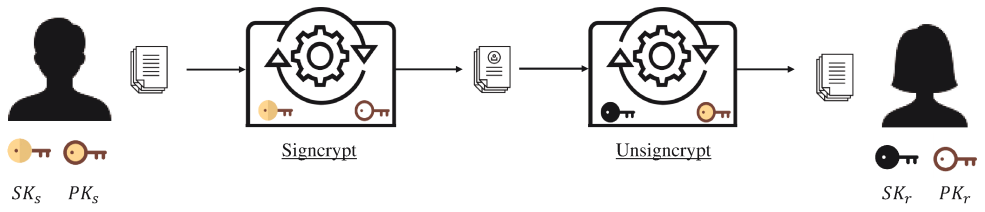


Figure 2.1: The overview of signcryption schemes

In this contribution, we first enhance the scheme proposed in [53] to design the PSKE protocol. This is due to the compatibility of the scheme with our conceptual design of PSKE. From the security perspective, the scheme, however, does not provide forward secrecy which is discussed in the insider security model. Informally, forward secrecy means that even if the sender's long-term private key is disclosed, the attacker cannot decrypt the past exchanges. Given a modification to Zheng's first scheme, we slightly modify the scheme to make it forward-secure. We denote by MSE the modified signcryption scheme which is also defined as a tuple of 4 probabilistic polynomial time algorithms:

$PP \leftarrow \mathbf{Setup}(1^l)$: Given a security level parameter l, the system generates a list of public parameters: $\mathbb{G} = <g>$, which is a multiplicative cyclic group of a prime order q with a generator

g over $\mathbb{Z}_p$ so that p is also a prime number and $q|(p-1)$. Enc_K, Dec_K, $\text{KDF}(K)$, H_K and KH_K are, respectively, the symmetric encryption/decryption functions, a secure key derivation function, the key-ed hash functions on the input of a private key K.

$(SK, PK) \leftarrow$ **Keygen**(PP): Sender ($\mathcal{S}$) and Recipient ($\mathcal{R}$) respectively generate their own long-term key pairs:

$$SK_S = x \xleftarrow{R} \mathbb{Z}_q^* \qquad PK_S = g^x$$
$$SK_R = y \xleftarrow{R} \mathbb{Z}_q^* \qquad PK_R = g^y$$

$(C, R, S) \leftarrow$ **Signcrypt**(SK_S, PK_S, PK_R, M): $\mathcal{S}$ takes his key pair (SK_S and PK_S), the public key PK_R of $\mathcal{R}$ and a message M as input to generate a signcrypted message. $\mathcal{S}$ first randomly chooses $z \xleftarrow{R} \mathbb{Z}_q^*$, then computes:

Step 1: $K = (g^y)^z$ and $(k_1, k_2) = \text{KDF}(K)$

Step 2: $r_1 = \text{KH}_{k_1}(M)$ and $r = g^{r_1}$, $t = \text{H}_{k_1}(M)$

Step 3: $s = z/(r_1 + x)$

Step 4: $c = \text{Enc}_{k_2}(M)$

Then, $\mathcal{S}$ transmits (c, s, r, t) to $\mathcal{R}$.

$M \leftarrow$ **Unsigncrypt**$(SK_R, PK_R, PK_S, c, s, r, t)$: On input of its key pair and $\mathcal{S}$'s public key, $\mathcal{R}$ unsigncrypts (c, s, r, t) to obtain the original message:

Step 1: $K = (g^x.r)^{y.s} = g^{yz}$

Step 2: $(k_1, k_2) = \text{KDF}(K)$

Step 3: $M = \text{Dec}_{k_2}(c)$

Step 4: Verify $t = \text{H}_{k_1}(M)$. If the condition is satisfied, $\mathcal{R}$ obviously accepts M. In contrast, M is considered invalid.

2.4.3 Security analysis

2.4.3.1 Security Proof

We briefly show that the modified scheme MSE guarantees the security requirements achieved by Zheng's original scheme [53], including the authenticity and the confidentiality. We first clarify the main differences between these two schemes:

- In our modified scheme: a signcrypted message is in form of $S_{\text{MSE}} = (c, s, r, t)$.
- In Zheng's scheme, a signcrypted message is in form of $S_{\text{Zheng}} = (c, s, r_1)$.

We prove the security of MSE by reduction without relying on any additional computational hardness assumption. With regards to these two mentioned above security requirements, Zheng's scheme was formally demonstrated secure [53], we reduce the security of Zheng's scheme to that of MSE. We achieve this goal by comparing S_{Zheng} to S_{MSE}. Indeed, with the input of $S_{\text{Zheng}} = (c, s, r_1)$, one easily computes $r = g^{r_1}$. Now, the difference between the two schemes comes from that between $r_1 = \text{KH}_{k_1}(M)$ in Zheng's scheme and $t = \text{H}_{k_1}(M)$ in MSE. These two values serve as the validation codes. We note that both two key-ed hash functions (KH_k and H_k) are collision-resistant, an attacker can thus only deduce the same information from both $\text{H}_{k_1}(M)$ and $\text{KH}_{k_1}(M)$ in the random oracle model. Therefore, the attacker can break the security of MSE if and only if it can break Zheng's scheme's security with the same probability.

2.4.3.2 Forward Secrecy Proof

With regard to the forward secrecy of MSE, we handle passive attacks in which adversaries manage to know the long-term private key of the sender and try to deduce the session keys of the past protocol executions. We formally prove that such an attack is at least as hard as breaking the DDH assumption in the random oracle model. That is, if an adversary $\mathcal{A}$ has an advantage δ of breaking the forward secrecy of MSE, we can construct an adversary $\mathcal{B}$ to successfully break the DDH assumption with the same advantage δ. We remind a DDH attack wherein on input of a tuple (g^a, g^b, g^c), $\mathcal{B}$ has to successfully guess whether $g^{ab} = g^c$ with a non-negligible probability. To do so, we construct an interaction between $\mathcal{A}$ and $\mathcal{B}$, in which the latter plays a role of a challenger, as follows:

Assume that $\mathcal{B}$'s and Recipient's key pairs are $(SK_S = x, PK_S = g^x)$ and $(SK_R = b, PK_R = g^b)$, respectively. $\mathcal{B}$ signcrypts a random message M under its private key x and Recipient's public key g^b as follows:

Step 1: $\mathcal{B}$ computes $K = g^c$ and $(k_1, k_2) = \mathrm{KDF}(K)$.

Step 2: $t = \mathrm{H}_{k_1}(M)$.

Step 3: $\mathcal{B}$ randomly selects $s \in \mathbb{Z}_q^*$ and computes: $r = g^{r_1} = g^{\frac{a}{s}} g^{-x}$.

Step 4: $cp = \mathrm{Enc}_{k_2}(M)$.

The adversary $\mathcal{A}$ eavesdrops the signcrypted message $\mathrm{T} = (cp, s, r, t)$ in the communication between $\mathcal{B}$ and the Recipient. Note that $\mathcal{A}$'s view in the model is identical to its view in the real attack model. In the context of forward security, we suppose that $\mathcal{A}$ somehow knows $\mathcal{B}$'s private key x. Given (T, x, g^y) to unsigncrypt T, $\mathcal{A}$ has to retrieve $K = g^{ab}$. Therefore, if $\mathcal{A}$ can break MSE (successfully unsigncrypt T), $\mathcal{B}$ can determine that $g^c = g^{ab}$, thus breaking the DDH assumption. Therefore, the forward secrecy of MSE is proven.

2.5 Efficient Password-Based Authenticated Key Exchange

After having presented the modified signcryption scheme, we now describe in detail the PSKE protocol, which consists of 3 phases as follows:

Setup Phase: During the setup phase, the server is responsible for distributing all the public system parameters to users. Given the security parameter l, the server generates a cyclic group $\mathbb{G} = <g>$ of prime order q. Let g be a generator of $\mathbb{G}$. The cryptographic hash functions are defined as: H_0: $\{0,1\}^* \to \mathbb{G}$, H_1: $\{0,1\}^* \to \mathbb{Z}_q^*$, H_2: $\{0,1\}^* \to \{0,1\}^l$, H_3: $\{0,1\}^* \to \{0,1\}^l$, and KDF: $\{0,1\}^* \to \mathbb{Z}_q^* \times \mathbb{Z}_q^*$. Precisely, H_0, H_1, H_2, H_3 and KDF are respectively the hash functions and the key derivation function used to generate authentication codes and private keys in the protocol.

Registration Phase: To be part of the system, the user must register with the server. The user sends a registration request to the server that replies with the supported cipher suite and a secure *salt*. Next, the user computes a password verifier $P = g^p$ where $p = \mathrm{H}_1(pw||salt)$ along with a password-derived value $D = \mathrm{H}_0(ID_u||ID_s||pw)$. Lastly, the server inserts the tuple of (*salt*, identity, P, and D) of the user into its database.

Authentication Phase: This essential phase is illustrated in Figure 2.2. This one describes the workflow of our protocol that permits the user and the server to authenticate each other and establish a shared session key over an insecure network; as explained hereafter:

Message 1 (User → Server): At the start of a new session, the user prepares a new authentication request and sends it to the server. This request only contains the user's identity.

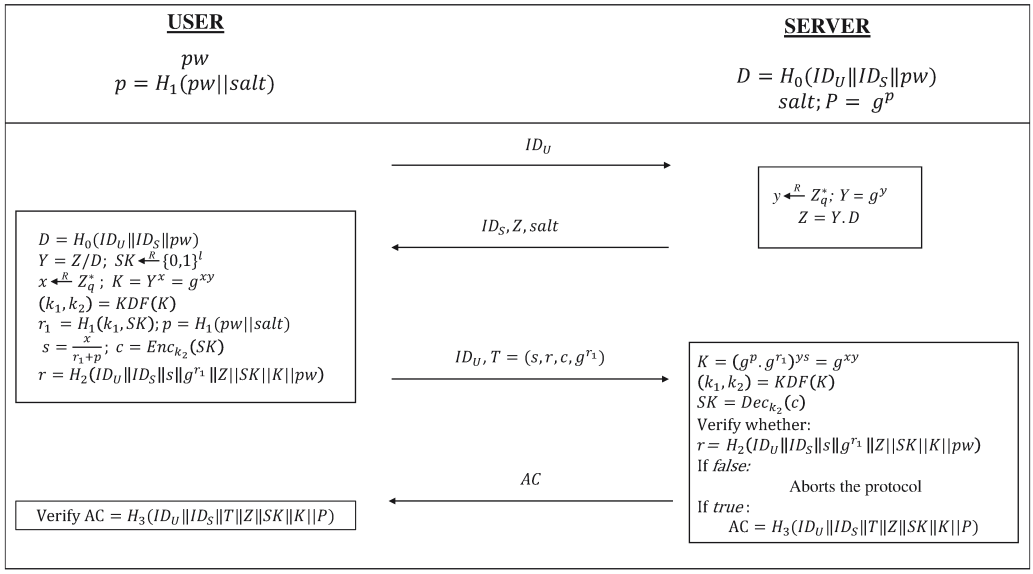


Figure 2.2: PSKE workflow

Message 2 (Server → User): Upon receipt of an authentication request, the server looks up a corresponding tuple of values including the corresponding *salt*, identity, D and P of the user. Then, the server generates randomly a private key $y \xleftarrow{R} \mathbb{Z}_q^*$. Next, it calculates $Z = g^y D$ and sends Z and the *salt* to the user.
Message 3 (User → Server): Once Z is received, the user takes a random session key $SK \xleftarrow{R} \mathbb{Z}_l^*$, then signcrypts SK with $p = \mathrm{H}_1(pw||salt)$ and sends the signcrypted message T to the server.
Message 4 (Server → User): The server uses y to unsigncrypt T. If the signature verification leads to an error, the server aborts immediately the authentication process. By contrast, the server generates an authentication code calculated from hashing a set of values (ID_U, ID_S, T, Z, SK, K, and P) and then transmits it to the user.
Message 5 (User): The user recomputes the authentication code and compares it with the one the server sent to. If the verification holds, the process is successfully terminated. On the contrary, the session is dropped.

2.6 Security Proof

In this section, we formally prove the security of PSKE. We first demonstrate that PSKE securely distributes session keys that are semantically secure over an adversarially controlled network. Then, we enhance the security of PSKE by making it forward-secure. The forward secrecy protects the past session keys against adversaries even if participants are corrupted and leak their password. In order to achieve this requirement, we thus necessarily make the simulation completely independent of any password. The security of PSKE is stated in Theorem 2.1.

Theorem 2.1 (PSKE Security). *Considering the PSKE protocol in Figure 2.2 where a password is randomly selected from a Dictionary of size N. We define the advantage of an adversary A who is allowed to make less than* q_{send} *Send-queries,* $q_{\mathrm{execution}}$ *Execute-queries,* q_{KDF} *key derivation*

queries and q_{H_i} *hash-queries where* $i = 0, 1, 2, 3$ *within the time* t*:*

$$\begin{aligned}\mathrm{Adv}^{\mathrm{fs-pske}}(A) \leq\ & 8N \times \mathrm{Succ}^{\mathrm{gdh}}_{g,\mathbb{G}}(t + 3\tau_e) \\ & + \frac{4(q_{\mathrm{send}_U} + q_{\mathrm{send}_S})}{N} + \frac{(q_{\mathrm{execute}} + q_{\mathrm{send}})^2}{q} \\ & + \frac{2q_{\mathrm{H}_0}}{q} + \frac{q^2_{\mathrm{KDF}} + (q_{H_0} + q_{\mathrm{H}_1} + q_{\mathrm{H}_2} + q_{\mathrm{H}_3})^2}{q}\end{aligned} \tag{2.1}$$

where τ_e *is the computation time of an exponentiation on* $\mathbb{G}$ and q_{send_S}, q_{send_U} *are the number of Send-queries to S and U, respectively.*

We reach our goal by using a sequence of games, starting from the game $\mathbf{G_0}$ and ending up at the game $\mathbf{G_5}$. The probability of each event in these games is bounded according to the Shoup's difference lemma [70]. We denote by S_i an event where an adversary correctly guesses the hidden bit b in a *Test*-query in the game i.

Game $\mathbf{G_0}$: This game describes the original protocol where the event S_0 means that the adversary successfully guesses the hidden bit b in the *Test*-query:

$$\mathrm{Adv}^{\mathrm{fs-pske}}(A) = 2\mathrm{Pr}[S_0] - 1. \tag{2.2}$$

Game $\mathbf{G_1}$: In this game, we simulate all the hash oracles KDF and H_j (for $j = 0, 1, 2, 3$) and the corresponding private oracles H'_j (for $j = 2, 3$) by maintaining all the hash lists ($\mathrm{L}_{\mathrm{KDF}}$, $\mathrm{L}_{\mathrm{H}_j}$) and ($\mathrm{L}_{\mathrm{H}'_j}$). We will introduce in more detail the private oracles in the game $\mathbf{G_3}$.

- $\mathrm{KDF}(K)$: If there is a record (K, k_1, k_2) in the list $\mathrm{L}_{\mathrm{KDF}}$, the oracle returns (k_1, k_2) as the answer. Otherwise, it chooses two random numbers $k_1, k_2 \in \mathbb{Z}^*_q$ and adds a new record (K, k_1, k_2) to $\mathrm{L}_{\mathrm{KDF}}$. Finally, the oracle answers with (k_1, k_2).
- $\mathrm{H}_0(q)$: If there is a record (q, r) in the list $\mathrm{L}_{\mathrm{H}_0}$, the oracle returns r as the answer. Otherwise, it chooses a random number $r \in \mathbb{G}$ and adds a new record (q, r) to $\mathrm{L}_{\mathrm{H}_0}$. Finally, the oracle answers with r.
- $\mathrm{H}_1(q)$: If there is a record (q, r) in the list $\mathrm{L}_{\mathrm{H}_1}$, the oracle returns r as the answer. Otherwise, it chooses a random number $r \in \mathbb{Z}^*_q$ and adds a record (q, r) to $\mathrm{L}_{\mathrm{H}_1}$. Finally, the oracle answers with r.
- $\mathrm{H}_2(q)$, $\mathrm{H}_3(q)$: If there is a record (q, r) in the lists $\mathrm{L}_{\mathrm{H}_2}$, $\mathrm{L}_{\mathrm{H}_3}$, the oracle returns r as the answer. Otherwise, it chooses a random number $r \in \{0, 1\}^l$ and adds a new record (q, r) to $\mathrm{L}_{\mathrm{H}_2}$, $\mathrm{L}_{\mathrm{H}_3}$. Finally, the oracle answers with r.
- $\mathrm{H}'_2(q)$, $\mathrm{H}'_3(q)$: If there is a record (q, r) in the lists $\mathrm{L}_{\mathrm{H}'_2}$, $\mathrm{L}_{\mathrm{H}'_3}$, the oracle returns r as the answer. Otherwise, it chooses a random number $r \in \{0, 1\}^l$ and adds a new record (q, r) to $\mathrm{L}_{\mathrm{H}'_2}$, $\mathrm{L}_{\mathrm{H}'_3}$. Finally, the oracle answers with r.

In addition, we also simulate all the instances of the participants, which we enable the adversary to make the queries defined in the section 2.3.2. Thus, the simulation is completely indistinguishable from the real attack.

$$\mathrm{Pr}[S_0] = \mathrm{Pr}[S_1]. \tag{2.3}$$

Game $\mathbf{G_2}$: In this game, we cancel all the games where the collisions may appear:

- Collision on the transcript ((ID$_U$, T), (ID$_S$, Z)).
- Collision on the output of KDF and H$_j$.

Let Coll$_2$ denote the event where any of the above collisions happens. We bound the probability of these collisions using the birthday attack:

$$\Pr[\text{Coll}_2] = \frac{(q_{\text{execute}} + q_{\text{send}})^2}{2q} + \frac{q_{\text{KDF}}^2 + (q_{\text{H}_0} + q_{\text{H}_1} + q_{\text{H}_2} + q_{\text{H}_3})^2}{2q}. \tag{2.4}$$

Game G$_3$: In this game, we make some changes to the authentication codes r, AC (see Figure 2.2) by using the private oracles H$'_2$ and H$'_3$:

- Hash function H$_2$(ID$_U$||ID$_S$||s||g^{r_1}||Z||SK||K||pw): On input of (ID$_U$||ID$_S$||s||g^{r_1}||Z||SK||K ||pw), the oracle returns the result of H$'_2$(ID$_U$||ID$_S$||s||g^{r_1}||Z).
- Hash function H$_3$(ID$_U$||ID$_S$||T||Z||SK||K||P): On input of (ID$_U$||ID$_S$||T||Z||SK||K||P), the oracle returns the result of H$'_3$(ID$_U$||ID$_S$||T||Z).

We note that the private hash functions H$'_2$ and H$'_3$ are not available to the simulation. By using H$'_2$ and H$'_3$, the authentication codes r, AC are independent from not only the hash functions (H$_2$, H$_3$) but also the password pw and the session key SK. The games **G$_2$** and **G$_3$** are now indistinguishable unless the adversary has already asked respectively H$_2$, H$_3$ on (ID$_U$||ID$_S$||s||g^{r_1}||Z||SK||K||pw) and (ID$_U$||ID$_S$||T||Z||SK||K||P) in accepted sessions. We need to prove that the probability of such an event is negligible. Unfortunately, this goal can only be achieved if the password has not been revealed to the adversary before the protocol has started. Since in case that one of the participants is corrupted, the adversary may know x or y and then K, SK using the revealed password pw. Now, we need to make H$_2$, H$_3$ return the same answers as the private hash functions H$'_2$, H$'_3$ for sessions accepted after the corruption. We modify the functions of H$_2$ and H$_3$ as follows:

▶ Hash function H$_2$(ID$_U$||ID$_S$||s||g^{r_1}||Z||SK||K||pw):

- Before the corruption, choose $r \in \{0,1\}^l$
- After the corruption, we check:
  - The password pw is correct.
  - ID$_U$, T, ID$_S$, Z corresponds to the session ID of a session accepted after the corruption.
  - Derive $g^x = (g^p g^{r_1})^s$ and $g^y = Z/D$.
  - Verify whether or not $K = \text{CDH}_{g,\mathbb{G}}(g^x, g^y)$ (using the DDH oracle).

  and then let H$'_2$(ID$_U$||ID$_S$||s||g^{r_1}||Z) = r.

▶ Hash function H$_3$(ID$_U$||ID$_S$||T||Z||SK||K||P):

- Before the corruption, choose $r \in \{0,1\}^l$.

- After the corruption, we check:
  - P is correct.
  - $\mathrm{ID_U}$, T, $\mathrm{ID_S}$, Z corresponds to the session ID of an session accepted after the corruption.
  - Derive $g^x = (g^p g^{r_1})^s$ and $g^y = Z/D$.
  - Verify whether or not $K = \mathrm{CDH}_{g,\mathbb{G}}(g^x, g^y)$ (using the DDH oracle).

  and then let $\mathrm{H}'_3(\mathrm{ID_U}||\mathrm{ID_S}||\mathrm{T}||\mathrm{Z}) = r$.

We state that the games $\mathbf{G_2}$ and $\mathbf{G_3}$ are perfectly indistinguishable unless the adversary queried H_2, H_3 on some execution transcript $(\mathrm{ID_U}||\mathrm{ID_S}||s||g^{r_1}||\mathrm{Z}||SK||K||pw)$ and $(\mathrm{ID_U}||\mathrm{ID_S}||T||Z||SK||K||P)$ before the password corruption. We call this event AskH. We then define:

$$\mathrm{AskH}_3 = \mathrm{AskH2}_3 \vee \mathrm{AskH3}_3.$$

- $\mathrm{AskH2}_3$: The adversary queried the oracle H_2 on some transcript $(\mathrm{ID_U}||\mathrm{ID_S}||s||g^{r_1}||\mathrm{Z}||SK||K||pw)$.
- $\mathrm{AskH3}_3$: The adversary queried the oracle H_3 on some transcript $(\mathrm{ID_U}||\mathrm{ID_S}||T||Z||SK||K||P)$ but $\mathrm{AskH2}_3$ has not supposedly happened yet.

$$\Pr[\mathrm{AskH}_3] \leq \Pr[\mathrm{AskH2}_3] + \Pr[\mathrm{AskH3w1}_3]. \tag{2.5}$$

To facilitate the security proof, the event AskH2 can be split into 3 sub-cases:

- $\mathrm{AskH2\text{-}Passive}_3$: The transcript is generated from an execution of PSKE between two legitimate instances U^i and S^j. In this case, both two instances are simulated.

- $\mathrm{AskH2\text{-}ActiveU}_3$: The transcript is generated from an execution between an adversary and an instance U^i. In this case, U^i is simulated.

- $\mathrm{AskH2\text{-}ActiveS}_3$: The transcript is generated from an execution between an adversary and an instance S^j. In this case, S^j is simulated.

We now evaluate the success probability of the adversary in this game. For clarity, we consider both passive and active adversaries. For a passive adversary, the session key SK is randomly selected over $\{0,1\}^l$ and is encrypted by a secure symmetric encryption mechanism, the passive adversary cannot correctly guess the hidden bit used in the *Test*-oracle with a probability greater than 1/2. For an active attack, we consider two scenarios in which either the adversary impersonates the user or the adversary impersonates the server. In the case that the adversary impersonates the user, according to the simulation, the adversary randomly chooses the session key SK over $\{0,1\}^l$. However, the session key can only be approved by the server if the adversary can correctly distinguish r, which is computed by the private hash function H'_2, from a random number in $\{0,1\}^l$. Similarly, in the case that the adversary impersonates the server, it needs to correctly distinguish AC, which is computed by the private hash function H'_3, from a random number in $\{0,1\}^l$. As the private hash functions H'_2, H'_3 are not available to the simulation and the outputs of these functions are independent of the password and the session key, the probability for the active adversary to correctly distinguish r_1, AC from a random number over $\{0,1\}^l$ is not greater than 1/2. We conclude the success probability of the adversary in this game:

$$\Pr[S_3] = \frac{1}{2}. \tag{2.6}$$

Game $\mathbf{G_4}$: In this game, we make use of a Diffie-Hellman instance $(P, Q \in \mathbb{G})$ as follows:

- We modify the function of H_0: Upon input of q, if $q = (\mathrm{ID_U}||\mathrm{ID_S}||*)$, we choose a random number $k \in \mathbb{Z}_q^*$ and return $r = P^{-k}$. The record (q, r) is added in the list $\mathrm{L_{H_0}}$.

- In the third round, in response to the message from S^i, U^i sends out $\mathrm{T} = (s, r, c, Q^{r_1})$ such that s is selected at random from $\mathbb{Z}_q^*$.

As we have excluded the case where $k = 0$, we then have:

$$|\Pr(\mathrm{AskH2}_4) - \Pr(\mathrm{AskH2}_3)| \leq \frac{q_{\mathrm{H}_0}}{q}. \tag{2.7}$$

Game $\mathbf{G_5}$: We now evaluate the probability of the AskH event. To do so, we cancel the games where for some transcript $(\mathrm{ID_U}, T, \mathrm{ID_S}, Z)$, there are two different passwords pw_1 and pw_2 such that $(\mathrm{ID_U}, \mathrm{ID_S}, s, Q^{r_1}, \mathrm{Z}, SK, K, pw)$ lies in the hash list $\mathrm{L_{H_2}}$.

Intuitively, we observe that $\mathrm{T} = (s, r, c, Q^{r_1})$ is a signcrypted message generated by our modified signcryption scheme, namely MSE. As MSE provides the same security guarantees as the original version of Zheng, i.e, internal non-repudiation. That is, it is computationally impossible to generate the same signcrypted message T from two different private keys (pw_1, pw_2) on the same message SK under the Gap Diffie-Hellman assumption. For the sake of completeness, we also prove this in the following.

Lemma 2.1. *For any transcript (ID_U, T, ID_S, Z) involved in an accepted session, there is at most one password such that $(ID_U, ID_S, s, Q^{r_1}, \mathrm{Z}, pw, SK, K = \mathrm{CDH}_{g,\mathbb{G}}[(g^p Q^{r_1})^s, Z.P^k])$ is in the list $\mathrm{L_{H_2}}$, unless one can solve the Diffie-Hellman problem:*

$$\Pr[\mathrm{CollH_5}] \leq N \times \mathrm{Succ}^{\mathrm{gdh}}_{g,\mathbb{G}}(t + 3\tau_e). \tag{2.8}$$

Proof: Assume that there is a tuple (s, Q^{r_1}, Z) involved in an accepted communication such that the two following tuples (for $i = 1, 2$) are in the list $\mathrm{L_{H_2}}$:

$$(\mathrm{ID_U}, \mathrm{ID_S}, s, Q^{r_1}, \mathrm{Z}, pw, SK, K = \mathrm{CDH}_{g,\mathbb{G}}[(g^{p_i} Q^{r_1})^s, Z = Z.P^{k_i}].$$

Since (s, g^{r_1}, Z) involved in an execution of PSKE (from *Send*-queries or *Execute*-queries), we have (s, Q^{r_1}) or Z that have been simulated. Therefore, at least r_1 or y is known. We suppose that r_1 is known:

$$\begin{gathered} K_i = \mathrm{CDH}_{g,\mathbb{G}}[Q^{sr_1}, Z] \times \mathrm{CDH}_{g,\mathbb{G}}[P, Q]^{sr_1 k_i} \times Z^{sp_i} \\ \times P^{k_i p_i s} \\ K_1/K_2 = P^{s(p_1 k_1 - p_2 k_2)} \times \mathrm{CDH}_{g,\mathbb{G}}[P, Q]^{sr_1(k_1 - k_2)} \\ \times Z^{s(p_1 - p_2)} \\ \mathrm{CDH}_{g,\mathbb{G}}[P, Q] = [Z^{u_1} \times P^{u_2} \times K_1/K_2]^{u_3} \end{gathered}$$

where $u_1 = s(p_2 - p_1), u_2 = s(p_2 k_2 - p_1 k_1)$ and u_3 are the inverse of $sr_1(k_1 - k_2)$ in $\mathbb{Z}_q$.

We analyze the three sub-cases $\mathrm{AskH2\text{-}Passive_5}$, $\mathrm{AskH2\text{-}ActiveU_5}$ and $\mathrm{AskH2\text{-}ActiveS_5}$:

- $\mathrm{AskH2\text{-}Passive_5}$: In order to bound this event, we state the Lemma 2.2

Lemma 2.2. *For any transcript (s, Q^{r_1}, Z) involved in an accepted session, there is no password pw such that $(ID_U, ID_S, s, Q^{r_1}, \mathrm{Z}, pw, SK, K = \mathrm{CDH}_{g,\mathbb{G}}[(g^p Q^{r_1})^s, Z.P^k])$ lies in $\mathrm{L_{H_2}}$, unless an adversary can solve the Diffie-Hellman problem:*

$$\Pr[\mathrm{AskH2 - Passive_5}] = N \times \mathrm{Succ}^{\mathrm{gdh}}_{g,\mathbb{G}}(t + 3\tau_e). \tag{2.9}$$

Proof: We suppose that there exist (s, Q^{r_1}, Z) involved in a passive transcript of an accepted session, and $D = P^{-k}$ such that $(\mathrm{ID_U}, \mathrm{ID_S}, s, Q^{r_1}, \mathrm{Z}, pw, SK, K)$ lies in $\mathrm{L_{H_2}}$. Since both r_1 and y are known, we have:

$$K = (g^p Q^{r_1})^{s.y} \times P^{s.k.p} \times \mathrm{CDH}_{g,\mathbb{G}}[P, Q]^{s.k.r_1}$$
$$\mathrm{CDH}_{g,\mathbb{G}}[P, Q] = (K \times (g^p Q^{r_1})^{u_1} \times P^{u_2})^{u_3}$$

where $u_1 = -s.y$, $u_2 = -s.k.p$ and u_3 is the inverse of $s.k.r_1$ in $\mathbb{Z}_q$.

- AskH1-ActiveU_5: As stated above, with each transcript (s, Q^{r_1}, Z) involved in an execution with an instance U^i, there is at most one password so that $(s, Q^{r_1}, \mathrm{Z}, pw, k_1, SK, K)$ lies in $\mathrm{L_{H_2}}$, we thus have:

$$\Pr[\mathrm{AskH2} - \mathrm{ActiveU}] = \frac{q_{\mathrm{send}_S}}{N}. \tag{2.10}$$

- AskH1-ActiveS_5: Similarly, with each transcript (s, Q^{r_1}, Z) involved in an execution with an instance S^j, there is at most one password so that $(s, Q^{r_1}, \mathrm{Z}, pw, k_1, SK, K)$ lies in the list $\mathrm{L_{H_2}}$, we thus have:

$$\Pr[\mathrm{AskH2} - \mathrm{ActiveS}] = \frac{q_{\mathrm{send}_U}}{N}. \tag{2.11}$$

We do exactly the same analysis for AskH3 that should reach the following result:

$$\Pr[\mathrm{AskH3}_5] \leq 2N \times \mathrm{Succ}^{\mathrm{gdh}}_{g,\mathbb{G}}(t + 3\tau_e) + \frac{q_{\mathrm{send}_U}}{N} + \frac{q_{\mathrm{send}_S}}{N}. \tag{2.12}$$

Combining (2.2), (2.3), (2.4), (2.6), (2.7), (2.8), (2.9), (2.10), (2.11), and (2.12), we have the result in (2.1).

2.7 Performance Evaluation

In this section, we perform the relative communication and computational cost measurements of PSKE and then make a comparison with other widely deployed protocols in the field.

2.7.1 Computational Performance Evaluation

The computational evaluation is based on the total cost of all cryptographic operations. From the computational point of view, a modular exponentiation, which is composed of repetitions of modular multiplication, is much more expensive than other operations such as: division, subtraction, multiplication, hash operation, or even symmetric encryption [71]. For simplicity, we only take into account the number of modular exponentiations and use it as a comparison factor among the protocols. The performance of all protocols is evaluated in Table 2.1.

According to Table 2.1, DH-EKE and SPEKE seem to be as efficient as ours which requires only two modular exponentiations for each party. However, the cost of executing a modular exponentiation linearly depends on the exponent length. To be able to resist the partition attack, DH-EKE and SPEKE require the participants to choose large exponents from $\mathbb{Z}_q^*$ where q typically is a large prime number of 3072 bits. On the other hand, PSKE demands the participants to only select exponents distributed from a subgroup $\mathbb{Z}_q^*$ where q is just a number of at least 256 bits. Our protocol is hence computationally much cheaper than DH-EKE and SPEKE and is obviously the most computationally efficient solution in the PAKE family.

With the emergence of the elliptic curve (EC) cryptography [72], we are able to significantly save the cost of protocols based on Discrete Logarithm. Indeed, a protocol built on top of a

Table 2.1: Performance Evaluation of 2-PAKE protocols

Category	PAKE protocol	No. ME at User	No. ME at Server	Supported over EC	No. of Rounds	Mutual Authentication
B-PAKE	DH-EKE [22]	2	2	Yes	5	Yes
	SPEKE [24]	2	2	Yes	4	Yes
	J-PAKE [35]	14	14	Yes	4	No
A-PAKE	AUCPACE [75]	3	4	Yes	8	Yes
	A-DHEKE [37]	3	4	Yes	6	Yes
	B-SPEKE [76]	3	4	Yes	4	Yes
	SRP [43]	3	3	No	4	Yes
	VB-EKE [38]	3	4	Yes	2	No
	VTBPEKE [30]	4	4	Yes	3	No
	PSKE	**2**	**2**	**Yes**	**4**	**Yes**
ME: Modular exponentiation; EC: Elliptic Curve						

multiplicative cyclic group can be easily converted into a variant based on an additive cyclic group on EC over a finite field $\mathbb{F}_p$. According to [73], a system based on EC using a smaller key size can achieve the same security level as the RSA system. For example, a RSA 3072-bits key provides a comparable strength as a 256-bits key over EC. That is why we mark EC support as a comparison factor in Table 2.1. In fact, unlike a widely-used protocol, such as SRP which is not supported on EC, implementation of PSKE over EC is obviously feasible and then makes itself much more efficient.

2.7.2 Communication Performance Evaluation

In terms of the communication cost, we simply count the number of rounds that are necessary to accomplish the mutual authentication and the key exchange phases. As stated in Table 2.1, some protocols such as J-PAKE and VB-EKE do not originally provide mutual authentication. In order to achieve the strictest security, we have to add this feature by requiring each party to send at most one extra round containing an authentication code as suggested in [74].

In practice, a PAKE execution is initiated by a user. If the user is required to perform cryptographic operations as soon as in the first round, the user first needs to know the cipher suites supported by the server. It is the case of all protocols listed in Table 2.1 other than PSKE. To obtain the cipher suite configuration, the mentioned protocols require two extra rounds (one for a request from the user, one for a response from the server). Interestingly, PSKE allows a user to start the protocol by simply sending its identity to the server without knowing the supported cipher suites. The cipher suites configurations could be included in a message sent from the server in the second round. Hence, PSKE does not need any additional round to be adopted. More generally, we conclude that in order for a PAKE protocol to support the mutual authentication and to be deployable in practice, 4 exchange rounds is definitely minimal, as in PSKE.

2.8 Conclusions

In this chapter, we have presented a new lightweight password-based authenticated key exchange protocol to solve the authentication problem in environments where resource constraints exist. Our proposal, PSKE, provides mutual authentication and secure key exchange between remote entities. While PSKE is designed as a standalone solution, it can be integrated into certificate-based authentication protocols, such as TLS, to resist the phishing attacks. We formally proved the security of PSKE in the find-then-guess model, which has been well investigated in the

literature. We also showed that PSKE is more efficient than other widely-used protocols such as EKE, J-PAKE, or SRP in terms of both the computational and the communication costs.

Chapter 3

Privacy-Enhancing Decentralized Public Key Infrastructure

Contents

3.1 Foreword

Digital Identity is the most valuable data of users on the Internet as it allows the users to prove who they are before engaging in any transactions with others. In this regard, Public Key Infrastructure (PKI) has been proven a successful solution which is involved in various applications such as Internet Banking, E-commerce, etc. The two common approaches of PKI are categorized as Certificate Authority (CA)-based and Web of Trust (WoT)-based. A common role of a general PKI is to issue certificates which attest to the bindings between users identity data and their corresponding public keys. In CA-based PKIs, CAs act as trusted parties to issue and manage user certificates. Due to the centralization property, users are required to fully rely on the CAs for all security concerns. In an attempt to improve the trustability of such an approach, a project called Certificate Transparency (CT) [77] was invented by Google to improve the transparency of certificate issuance and help quickly detect rogue certificates. CT consists of a network of log servers and auditors. The log servers usually receive newly issued certificates from CAs and add them into the append-only log. CT is, however, pointed out to

be vulnerable to split-world attack [78] in which an attacker is able to compromise log servers and shows inconsistent views of the log to the users to perform Man-In-The-Middle attacks.

Pretty Good Privacy (PGP) [13] is a prominent standard leveraging the Web of Trust (WoT) model [14] to build an entirely decentralized PKI. In this model, users are able to designate others as trustworthy by signing their public key certificates. By doing so, a user accumulates a certificate containing digital signatures from entities that have deemed him trustworthy. The certificate will be then trusted by a third party if the certificate has been signed by a person that the third party entrusts. Clearly, this model offers the decentralization feature in certificate issuance which comes at the cost of user-friendliness. For a user to be widely visible on the network, it needs to accumulate signatures given by a large set of well-known entities. The user registration is illustrated in Figure 3.1. In practice, all issued certificates are uploaded onto the key servers which are supposed to play the information role only. If anyone wants to get the public key of a user, it must send a lookup request to one of the key servers to retrieve the associated certificate. In addition, certificate revocation is also a critical issue to be properly solved in such a model. In order to revoke a certificate, the certificate owner needs to publish a revocation certificate on the key servers as a request to stop broadcasting its certificate. The key servers are voluntarily run by the PGP community which does not have any economic incentive. However, even under the assumption that the key servers act honestly and do not hide revocation information from the users, a delay in certificate synchronization between them may already cause huge damage to the system. Similarly, a key update which happens when a user wants to change the key pair while retaining all the identity data is also extremely hard to realize in this model. These reasons therefore hinder the WoT's adoption in practice which is actually limited to secure email exchanges.

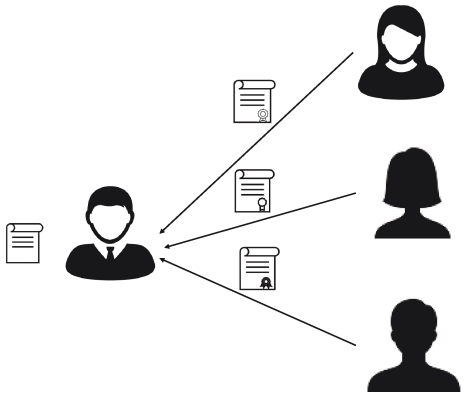


Figure 3.1: User registration in PGP

Due to its decentralization, transparency, and immutability, the blockchain technology inspires a new paradigm to create a global and decentralized digital identity system which is called Decentralized Public Key Infrastructure (DPKI). This long-awaited paradigm necessarily allows users to issue and manage their identities in their own ways. Unlike other approaches, once an identity is issued and registered in a blockchain, i.e., Ethereum, only the owner of the identity can make changes or revoke it in a publicly verifiable manner. We note that third parties still exist in a DPKI design such as blockchain validators but their role is restricted to information-only and users have a means of verifying the consistency of their operations. More specifically, the validators are in charge of verifying and adding new transactions into the blockchain. In order to work correctly, the validators must have enough space to store a

replica of the fast-growing blockchain. For thin clients who have limited storage capacity, they are only required to store block headers which serve verifying information retrieved from the validators. Even though the DPKI approach has been discussed in the blockchain community for a long time, it still has technical challenges to be properly addressed. Indeed, whereas the blockchain is maintained by a large set of economically-incentivized validators, it is a shared place to publicly store all decentralized identities. This makes all operations on users identities transparent, thus making the blockchain a potential correlation point of all activities of users and possibly undermining their privacy. In the well-known DPKI design proposed in [79], each user possesses an identity key pair which is tied to the identity data, and an online key pair which is used to authenticate with others, i.e., service providers. The main purpose of identity keys is to introduce, update, and revoke the online keys. This dual key model guarantees the safety of identity keys by keeping them disconnected from the Internet most of the time and only used in case of necessity. Even though the approach seems to be efficient and secure, the linkings between the online keys and the identity data are, however, visible to everyone, thus violating user privacy. Consider a scenario where a user authenticates itself to a service provider by using its registered public online key. In order to validate the online key, the service provider requests its revocation status from blockchain validators. Clearly, this allows the blockchain validators to trace the identity data and the activities of the user based on its online key, such as which service providers the user is visiting. A costly way to mitigate this privacy vulnerability with regard to the blockchain validators is that each service provider also stores a replica of the blockchain to locally perform the key revocation checking. We argue that it is very unlikely for service providers to store a real-time updated fast-growing replica of the blockchain. Even if it is the case, the user privacy risk still exists because of the possibility that some service providers collude to correlate the online keys of some targeted users, therefore being able to trace their online activities which may be used for data analytics purposes without user explicit permission.

In this chapter, we propose a privacy-enhancing decentralized public key infrastructure which exactly targets these user privacy vulnerabilities. We introduce an architectural design and cryptographic mechanisms to decouple online keys from user identity data and user identity key to avoid the blockchain validators from identifying the identity data associated with an online key. In our proposal, a user uses a different online key pair to authenticate with each service provider. The linking between an online public key and the identity key of a user can only be revealed to the corresponding service provider only if the user wants to do so. However, revealing the linking between a public online key and the identity key does not help adversaries reveal the linking between the identity key and other online keys. To illustrate the applicability of the proposed DPKI, we also discuss some applications in different contexts, including data storage and sharing.

3.2 Related Works

Due to the critical shortcomings of the traditional PKI approaches, we first present the improvements brought to by both academia and industry. We then discuss a variety of DPKI proposals which leverage the blockchain technology.

There have been many works in [80–82], which leverage the blockchain technology to mitigate the split-world attack, which was detected in the Certificate Transparency project [78] as mentioned above. Even though the proposed works use different blockchain platforms to

build their systems, they, however, share similar ideas to achieve the same objectives. In [80] and [81], the common idea is to build their systems on the smart contract-enabled blockchain platforms such as Ethereum and Hyperledger [83] in which each CA controls a dedicated smart contract to store two arrays of hashes. The first one is to store the hashes of all issued certificates and the second one is to store the hashes of all revoked certificates. The conception could be migrated to any other provably secure smart contract platforms such as Cardano [84], Algorand [85] depending on the chosen trade-off between security, decentralization, and scalability. The three factors constitute a well-known blockchain trilemma [86] and lead to different consensus protocol designs [87]. By contrast, Chen et al. [82] worked on improving efficiency by leveraging the Namecoin blockchain [88] for enhancing the transparency of the CA-based PKI. Namecoin is part of the first blockchain generation which does not support smart contracts. Due to the lack of smart contract support, one has no direct access to the newest status of a certificate that is stored and gradually updated on the blockchain. Therefore, in order to audit the operations made on a certificate, an auditor needs to traverse the blockchain to retrieve relevant information. The authors improved the efficiency by defining an X509-like operation format which not only includes the related information and a specific operation, such as revocation or update, made on a certificate, but also encompasses a pointer to the block height where the latest operation on the certificate has been stored. By doing so, the auditor only needs to traverse the blockchain to get the latest operation on the certificate then can have direct access to previous operations, making the auditing process more efficient. Matsumoto et al. [89] leveraged the blockchain technology to automate responses to rogue certificates issued by CAs which, therefore, implicitly incentivized CAs to behave correctly. That is, each domain registers domain certificate policies (DCPs) into the blockchain so that any CA issuing a certificate that does not meet the registered DCPs will be financially sanctioned according to pre-negotiated Reaction Policies (RP). Any entity that triggers the mismatching detection automatically receives a reward from the dishonest CA. To simultaneously reduce the risk of single points of failure and improve the transparency of CAs, Dykcik et al. [90] proposed a blockchain-based architecture which consists of 3 types of smart contract, namely the central contract, the domain contract, and the storage contract. The central contract is deployed once to receive all certificate requests that explicitly specify the authorized CAs which will sign the requested certificates. Once a certificate request is posted on the central contract, a new instance of certificate contract is automatically deployed on the blockchain which is implemented to ensure the execution of the signing procedure. The certificate data and resulting signature are also stored on a newly deployed storage contract. In practice, the authorized CAs run the multi-signature protocol based on the Schnorr signature [91] to sign the certificate. The goal of using the multi-signature protocol is to keep the resulting signature length compact, or precisely equal to the length of an individual signature. We observe that the Schnoor-based multi-signature protocol is interactive. That is, the signers have to interact with one another during the execution of the protocol, thus making the protocol expensive in communication. We suggest using the BLS multi-signature [92] instead, which is based on bilinear pairing over elliptic curves, for efficiency improvement. The utilization of BLS multi-signature protocol is twofold. First, the signing procedure is non-interactive. Second, a BLS multi-signature length is smaller than a Schnorr-based multi-signature length.

In the context of PGP Web-of-Trust, Yakubov et al. [93] proposed a blockchain-based Pretty Good Privacy framework in which they attempted to mitigate the security concerns regarding key servers. The proposed framework requires adding a user's Ethereum (ETH) address into

the Comment Field of its PGP certificate. The rights of a user over its certificate, such as introduction and revocation, within a system are considered based on the embedded ETH address. To introduce a certificate of an entity, the introducer signs and uploads the certificate into the global smart contract. The owner of the certificate has the right to verify and approve the introduction or just refuse it. The user also has the right to revoke the certificate by proving the ownership of the ETH address associated with the certificate. Similarly, Al-Bassam et al. [94] proposed a system in which each user deploys an individual smart contract to manage its identity and attributes. The smart contract is programmed with functions allowing other entities to sign the attributes of the contract owner who can subsequently approve or revoke the signatures.

Certcoin [79] is the first paper to employ the blockchain technology to create a decentralized public key infrastructure. Certcoin allows users to freely register their identities with the blockchain and have full control over them. More specifically, a user initially generates two key pairs, including an online key pair and an offline key pair. While the online key pair is used as an identifier to access services on the Internet, the offline key pair is kept secret and only be used to revoke or update online key pairs. However, Certcoin does not take user privacy into account, making user identity a central correlation point of all activities. Improved upon Certcoin, Qin et al. [95] add an identity verification via a challenge-response procedure during the identity registration. For instance, a user who wants to prove the ownership of a domain receives a challenge which requires posting a document on its domain. Once the verification succeeds, the binding of its public key and identity data will be registered on the blockchain.

With regard to the DPKI designs preserving user privacy, Axon et al. [96] presented a system where the binding of the public key and the identity data of a user which is posted on the blockchain is not recognizable by the public. The idea of the system is that the user first posts the initial public key and the identity data on the blockchain. Subsequently, it posts a new public key which is cryptographically bound to the initial key. While the binding is not visible to the public, the user is still able to prove the fact in case of necessity. The new public key is then used to authenticate the user to other parties. The introduction of a new public key systematically invalidates the old one. Therefore, only one key is valid at a time, thus making the user unable to access different services using different identifiers at the same time. The user repeats the process in case that it wants to introduce a new key. In the system, the auditing process, which aims at proving the binding of the current public key to the identity, i.e., for legal reasons, requires the user to reveal all the old public keys, thus completely destroying user privacy. In [97], Jiang et al. proposed a system based on Certcoin which supports privacy for thin clients. To be more specific, the system allows a thin client to retrieve the public key associated with an identity without the blockchain validators learning the identity nor the public key. This is achieved by using Private Information Retrieve (PIR) [98]. Roughly speaking, PIR is a multi-server protocol in which each server stores a copy of the shared database. If a user wants to retrieve the data associated with an index, the user runs an interactive protocol with the servers so that at the end of the protocol, the user retrieves the desired data and the servers do not know exactly what the data and the index are. In [97], due to the limited storage capacity to store a replica of the blockchain, a thin client makes use of PIR to retrieve the public key of a user identity without the blockchain validators learning the identity of the user. However, PIR requires that all the blockchain validators engaging in the protocol are honest and follow exactly the prescribed protocol. Otherwise, the user ends up receiving the false public key. This assumption seems to be unrealistic in the context of DPKI

as the number of decentralized identities increases exponentially and a huge number of queries make the blockchain validators vulnerable to DDoS attacks [99]. Moreover, the proposed DPKI only deals with user privacy with regard to the blockchain validators and does not provide a mechanism to avoid the collusion between the service providers that the user has accessed by using its decentralized identity. Our proposed solution aims at thoroughly solving these privacy issues.

3.3 System and Threat Models

3.3.1 System Model

The proposed platform includes 2 primary entities, including users and a smart contract-enabled blockchain:

Blockchain: is a growing chain of blocks which are validated and chained together. The blockchain is maintained by a great number of economically-incentivized validators. Since being introduced in 2008 [100] as the underlying technology of Bitcoin, the blockchain technology has attracted great interest from both industry and academia. Later, Ethereum [101] appeared as the next blockchain generation, which allows for deployment of smart contracts. A smart contract is a self-executing computer program stored on the blockchain, and is run by miners in a trustless way. Once being deployed on the blockchain, a smart contract cannot be modified. This provides users with integrity, transparency, and autonomy guarantees. Even though Ethereum is actually only able to process around 15 transactions per second, the number will be improved greatly when Ethereum upgrades the network to Ethereum 2.0, which is known as Serenity. The proposed DPKI leverages the Ethereum blockchain to allow users to use smart contracts for digital identity management.

User: is any entity, i.e., service providers or ordinary users, on the Internet that wants to register decentralized identities on the blockchain. The users are provided with necessary software to interact with the blockchain. Conceptually, each user possesses an identity key pair (iPK, iSK) which is bound to the user identity data and a set of online key pairs (nPK, nSK) used to interact with different service providers. The identity data could be any data which describes the user, i.e., username or a URI pointing to a location where the user data is stored. Moreover, the user data can be updated and modified by the owner.

3.3.2 Threat Model

The main purpose of our work is to propose a DPKI architectural design that ensures the two-level privacy guarantee. That is, we separately investigate privacy vulnerabilities with respect to blockchain validators and to third parties. The latter could be identity verifiers, such as service providers, which authenticate users based on their decentralized identities. We first present the security assumptions which we make use of to prove the privacy guarantees then present in detail the different privacy requirements.

3.3.2.1 Security Assumptions

Considering a multiplicative cyclic group $\mathbb{G}$ =<g> of prime order p where g is its generator, we formalize the following security assumptions:

Definition 3.1 (Computational Diffie-Hellman (CDH) Assumption)**.** *Given an instance* (g, g^a, g^b) *where* a *and* b *are randomly picked over* $\mathbb{Z}_p^*$*, the probability* $Succ_{g,\mathbb{G}}^{cdh}(t)$ *that a probabilistic polynomial time adversary can find out* g^{ab} *is negligible within time* t*.*

Definition 3.2 (Decisional Diffie-Hellman (DDH) Assumption)**.** *Given an instance* (g, g^a, g^b, g^c) *where* a, b, c *are randomly selected over* $\mathbb{Z}_p^*$*, the probability* $Succ_{g,\mathbb{G}}^{ddh}(t)$ *that a probabilistic polynomial time adversary can decide whether* $g^{ab} = g^c$ *is negligible within time* t*.*

3.3.2.2 Privacy Vulnerabilities with respect to Blockchain Validators

Due to the large size of the fast-growing Ethereum blockchain, there is little chance that identity verifiers, such as service providers, store a real-time synchronized replica of the blockchain to locally perform lookups of key or of revocation information associated with certain identities. To alleviate the storage burden, they only need to store block headers which are much more compact but still useful. Indeed, when the identity verifiers retrieve information stored on the blockchain from the blockchain validators, the block headers serve as proofs to check the correctness of retrieved information. Even though the approach is efficient in terms of storage, it, unfortunately, causes privacy risks. Specifically, the inherent transparency characteristic of the blockchain technology potentially makes it a correlation point of users' activities. For example, in order to authenticate a user, a service provider requests revocation information of the user's identity from the blockchain validators. Therefore, the latter obviously gain some information related to the user, i.e., which service the user is actually accessing.

3.3.2.3 Privacy Vulnerabilities with respect to Third Parties

In order to make the proposed DPKI more flexible and become a tool for many scenarios, we also take into account user privacy with respect to third parties.

In our model, each user has one identity key pair tied to the identity data but can have a set of online key pairs. The user uses a different public online key as its identifier to authenticate with a service provider. The privacy requirement must require that the service provider cannot, by default, link a public online key with the public identity key, hence being unable to read the associated identity data. Our decoupling mechanism allows cryptographically hiding the linking between an online key pair with the identity key pair. In practice, there are service providers that require no or the least user personal information, i.e., self-claimed data, to grant access to the provided services. Nevertheless, they still require an identifier for each user so that they can maintain the previous sessions of the same user. This is naturally suitable to our design due to the decoupling of an online key pair and the identity key pair.

In case of necessity, a user is, however, still able to cryptographically disclose the hidden linkings between its online keys and the identity key pair to the service provider if the user wants to do so. In this context, we need to ensure that if a user uses different public online keys to access different services, the service providers are unable to determine whether these online keys belong to the same user. However, if the user has ever shown the linkings between the identity key and some online keys to some service providers, there will be a risk that these service providers can come all together and correlate the activities of the user. Such a risk is still reasonable in many contexts. Typically, a user should be blocked from using the services due to legal reasons or violating some of the privacy and security rules imposed by the service providers. The linking disclosure prevents the user from registering new online key pairs to

regain access to the services. However, we make sure that the knowledge of linkings between the identity key pair and some online key pairs do not help discover new linkings concerning other online keys.

3.4 The Architecture

In this section, we highlight the main architectural building blocks constituting the proposed DPKI, which is depicted in Figure 3.2.

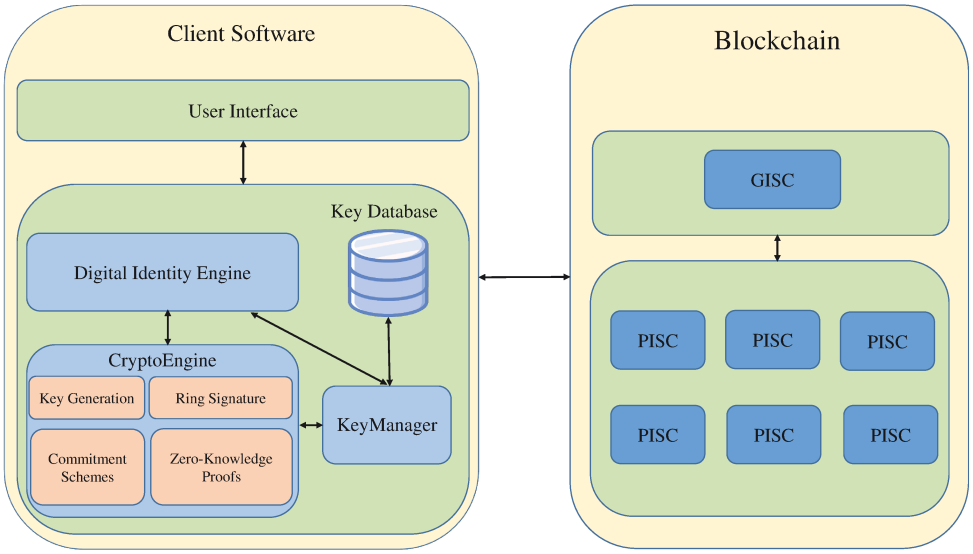


Figure 3.2: The architecture of the proposed DPKI

On the blockchain side, the platform has two main types of smart contract as following:

- **Personal Identity Smart Contract (PISC)**: Each user that wants to register its decentralized identity has to deploy a new instance of PISC on the blockchain. In principle, an identity consists of a public identity key iPK and the identity data of the the identity owner. The binding between the public identity key and the identity data is therefore guaranteed by the unalterability and consistency of the blockchain. The contract is implemented with the necessary operations, such as key/identity data revocation and update for the user to manage the contract appropriately.

- **Global Identity Smart Contract (GISC)**: This type of contract serves as a pool of all public online keys of registered users. A GISC can be deployed by owners of a service or a federation of services, i.e, a country, a university. In practice, the owners of the GISC can define additional security and privacy restrictions which are programmed in the contract. Users need to carefully consider all these restrictions before registering their public online keys. In fact, the users do not store their online keys on their own PISCs but rather using their private identity key iSK to anonymously introduce them on the GISC via our proposed key registration mechanism to avoid being traced by anyone else. We note that the key registration procedure ensures that only registered users can introduce online keys into the system. Similar to PISC, the GISC also supports the common operations allowing key revocation and update.

On the user side, a user is provided with software which allows interacting with 4 following internal components to register and manage its decentralized identity:

- **Digital Identity Engine**: This engine allows making key generation requests to CryptoEngine, specifying the anonymous key registration procedure as well as defining key recovery policies for identity keys in case of loss.
- **Crypto Engine:** is responsible for all cryptographic operations on the user side. This allows generating key pairs, anonymously registering public online keys on the blockchain via a ring signature scheme, committing to secret, and issuing zero-knowledge proofs of possession of keys and identity data while authenticating with other parties. In the proposed DPKI, commitment schemes and zero-knowledge proofs involve many operations on users' identities such as revocation, update.
- **KeyManager**: is responsible for storing and retrieving keys from the database in an authenticated way.
- **Key Database**: is a secure store of keys on the user side.

3.5 The CryptoEngine and KeyManager Components

For ease of understanding the functioning of the proposed DPKI, we describe in detail the CryptoEngine and KeyManager which contain our main cryptographic contributions.

3.5.1 KeyManager

KeyManager involves storing and retrieving keys from the Key Database. In the platform, we classify 2 types of key, each of which has a public key (PK) and a private key (SK):

- **Identity Keys** (iPK, iSK): In order to register with the DPKI, a user deploys an instance of the personal identity smart contract. Then, it generates an identity key pair and sends its identity data along with the public identity key iPK to the smart contract.
- **Online Keys** (nPK, nSK): A user can have many online keys which are anonymously introduced into the system by the private identity key. That is, the online keys and the identity data are cryptographically decoupled with respect to others. Each public online key nPK serves as an identifier to authenticate with a service provider. This mechanism aims to ensure the safety of the private identity key by keeping it disconnected from the Internet and only be used in urgent cases such as loss of online keys, which blocks users from accessing the services. In such cases, the private identity key helps revoke old online keys and introduce new ones, thus allowing the users to regain access without losing their accounts on the services.

3.5.2 CryptoEngine

The CryptoEngine plays a vital role in the platform due to its responsibility for all cryptographic computations. We now describe the functionalities that it supports. The usage of each functionality in the proposed DPKI will be discussed in the next section.

3.5.2.1 Identity, Online Keys Generation

The main difference among online and identity keys lies in their usage purposes as outlined above. These keys are, however, generated in the same way. Given a multiplicative cyclic group $\mathbb{G}$ of prime order p, which is generated by a generator g, Algorithm 3.1 depicts the generation of identity keys. We remark that some systems propose a different approach so that a user derives online keys from the identity key pair so that the user only needs to protect the private identity key. In fact, the user may use a different online key to establish a connection to each service provider. Therefore, if an adversary can steal the private identity key, it can impersonate the user to all services that the user has accessed.

In our approach, online and identity keys are independently generated. Therefore, a compromise of the private identity key does not allow impersonating the user to service providers. In fact, knowing the private identity key only enables an adversary to revoke and update online keys owned by the user as we shall see later. However, such operations can be detected easily due to the transparency of the blockchain technology.

Algorithm 3.1. Generating Identity Key

Input: Given a multiplicative cyclic group $\mathbb{G}$ of order p, generated by g.
Output: A identity key pair iSK, iPK.
1: Chooses a random number $x \in \mathbb{Z}_p^*$ and assigns $iSK = x$.
2: Computes $iPK = g^x$.
3: **return** iSK, iPK

3.5.2.2 Ring Signature

Ring Signature was first introduced by Rivest et al. in [102]. Its formal security model was further studied in [103]. Practically speaking, ring signature is a type of digital signature that allows a user to sign a message on behalf of a group of users, which is arbitrarily formed by the signer at the signing time. A verifier can only verify whether the signature comes from the group, without knowing exactly the identity of the signer. Therefore, ring signature provides users with anonymity guarantee whose extent completely depends on the ring size. More specifically, a signer takes its private key SK_s and all public keys $S = (PK_1, ..PK_s, ..PK_n)$ of all members in a group of size n to sign a message. A verifier checks the validity of the signature to determine whether the message was anonymously signed by a member of S. Due to a variety of applications, such as ad-hoc anonymous identification schemes, there have been many ring signature schemes proposed in the literature, which are based on the intractability of different computational hardness problems [104–106]. In this contribution, we purposely use the ring signature scheme of Herranz et al. [104]. This scheme was proven computationally efficient and is existential unforgeability under adaptive chosen-message attacks in the random oracle model. Second, this scheme is based on Discrete Logarithm (DL) assumption, making it compatible with the current cryptosystem in the Ethereum blockchain and removing the need for deploying another cryptosystem, i.e., RSA-like system, on the user side. The detail of the ring signature scheme is described in detail as following:

Setup(1^l): Given a security parameter l and (p, q) are two large prime numbers so that $q|p-1$, the system generates a multiplicative subgroup $\mathbb{G}$ of order q on $\mathbb{Z}_p$, which is generated by g. Let H denote a collision-resistant hash function H: $\{0,1\}^* \rightarrow \mathbb{Z}_q^*$.

Each user i generates a key pair, including a private key $SK_i = x_i \in \mathbb{Z}_q$ and a public key $PK_i = g^{x_i}$.

Sign(m, R, SK_k): To sign a message m, a signer k forms a ring consisting of n members, including itself. The set of member public keys is denoted by $R = (PK_1,...PK_n)$. Then, the signer follows the singing procedure in Algorithm 3.2 to obtain the signature σ.

Algorithm 3.2. Signing Algorithm

Input: m, SK_k, R.
Output: σ.
1: $\forall i \in \{1..n\}/\{k\}$, choose $a_i \in \mathbb{Z}_q^*$ and compute $R_i = g^{a_i}$.
2: Choose a random number $a \in \mathbb{Z}_q^*$.
3: Compute $R_k = g^a \prod_{i \neq k} g^{-H(m,R_i)x_i}$. If $R_k = 1$ or $R_k = R_i$ for some $i \neq k$ then go back to Step 2.
4: Compute $\alpha = a + \sum_{i \neq k} a_i + x_k H(m, R_k) \mod q$.
5: Construct the signature:
$\sigma = (m, \alpha, R_1, ...R_n, h_1, ...h_n)$.
6: **return** σ

Verify(m, R, σ): In order to check the validity (val) of a ring signature, the verifier follows the procedure in Algorithm 3.3.

Algorithm 3.3. Verifying Algorithm

Input: m, σ, R.
Output: val.
1: Verify whether $\forall i \in 1, .., n$, $h_i = H(m, R_i)$.
2: Verify whether $g^\alpha = \prod_{i=1}^n R_i g^{h_i x_i}$.
3: **if** (1) and (2) are fulfilled **then**
4: val = true.
5: **else**
6: val = false.
7: **end if**
8: **return** val.

3.5.2.3 Zero Knowledge Proofs

Before defining zero-knowledge proofs, we recall the definition of an interactive proof system. An interactive proof system [107] is a two-party game where a prover $\mathcal{P}$ tries to convince a verifier $\mathcal{V}$ about the validity of a statement, i.e., a statement x belongs to a NP-language $\mathcal{L}$. In such a system, $\mathcal{P}$ supposedly has infinite computing power whereas $\mathcal{V}$ is polynomial-time. At the end of the game, $\mathcal{V}$ outputs a result about the proof which is either *accept* or *reject*. Formally, interactive proof systems must satisfy the two following conditions:

- *Completeness*: If $x \in \mathcal{L}$ then the probability that $(\mathcal{P}, \mathcal{V})$ rejects x is negligible.
- *Soundness*: If $x \notin \mathcal{L}$ then for any prover, the probability that $(\mathcal{P}, \mathcal{V})$ accepts x is negligible in the length of x.

Loosely speaking, the two conditions ensure that an honest prover can always convince a verifier about a true statement but there is no proving strategy which allows convincing the verifier about false statements. In the literature, there exist some variants of interactive proof systems such as interactive argument systems, public-coin proof systems, and proofs of knowledge. Introduced by Brassard, Chaum and Crépeau [108], interactive argument systems relaxed the soundness condition by only considering polynomial-time provers. Public-coin proof system, which is also known as Arthur-Merlin game, was introduced by Babai [109]. The system is a special case of interactive proofs in which the verifier must send the outcome of any coin it tosses. With regard to proofs of knowledge systems, a prover claims to know some secret piece of information and tries to convince a verifier about it. A typical example of such a system is a proof of knowledge of discrete logarithm. We refer to the work of Bellare et al. [110] for a formal definition.

Zero-knowledge was formally introduced in the seminal work of Goldwasser, Micali and Rackoff [111] as an extra property for an interactive proof system and its variants:

- *Zero-knowledge*: Using whatever strategy and prior knowledge, a verifier cannot gain more information than the fact about the proved statement, which is *true* or *false*.

The intuition is formalized through a simulation approach: We say that a protocol is zero-knowledge if there exists a simulator which does not interact with the prover but can simulate a malicious verifier's output. In addition, the simulated manuscript is indistinguishable from real transcript between the prover and the verifier. The existence of such a simulator implies that if an adversary succeeds in learning extra information after having communicated with a prover, then the adversary would have reached the same result without the help of the prover.

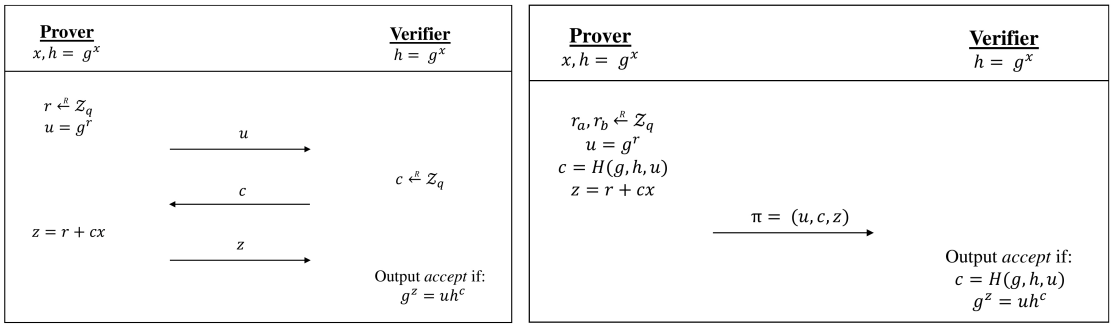
Goldreich, Micali, and Wigderson [112] showed a notable result about zero-knowledge proofs. That is, assuming the existence of commitment schemes, which we will describe in more detail in the next section, one can construct zero-knowledge proofs for any NP-set, thus opening the usability of zero-knowledge proofs in many applications. These range from payment systems [113,114], privacy-preserving identification system [115] to voting system [116]. Zero-knowledge proofs are originally interactive and involve exchanging messages between a prover and a verifier. However, there have been many works [117–121] aiming at turning them into non-interactive protocols. More specifically, a non-interactive zero-knowledge proof (NIZK) only has a single flow which consists of a message prepared and sent by the prover to the verifier in this model. Although NIZK proofs are clearly more practical but, unfortunately, it was shown that NIZK proofs in the plain model only exist for trivial languages in BPP [119]. Then, Blum et al. [117] defined NIZK proofs in the common random string (CRS) model. In this model, a common string is generated by a trusted party and is accessible to both provers and verifiers before running the protocol. Similarly, the works in [120, 121] defined NIZKs in the pre-processing and secret-key models. Fiat-Shamir heuristic [122] can also be used for efficiently transforming public-coin interactive zero-knowledge proofs into NIZK arguments by using a cryptographic hash function to compute the verifier's challenges. This methodology can only be proven in the random oracle model. However, we remark that there are several examples of protocols that are secure in the random oracle model but are insecure when being instantiated with any concrete hash function [106, 123, 124]. With the line of works from Groth, Ostrovsky, and Sahai [125–127], NIZK proofs can be instantiated more efficiently in the standard model using the pairing-based cryptography.

In the proposed DPKI, we mainly use zero-knowledge proofs of knowledge for discrete logarithm or similar statements. The purpose of using these proofs is to disclose the linking

between an online key to an identity key as well as prove the knowledge of private keys. We will detail this in more detail in the next sections. Let $\mathbb{G} =< g >$ be a cyclic group of prime order q, and is generated by g. We reuse a general expression introduced in [128] like:

$$ZKP[(\alpha, \beta, \gamma) : y_1 = g_1^{\alpha} g_2^{\beta} \wedge y_2 = y_1^{\alpha} g_3^{\gamma}] \quad \text{where} \quad (g_1, g_2, g_3) \in \mathbb{G}^3$$

to denote a zero-knowledge proof of knowledge of α, β, γ such that the relations on the right hand side are satisfied. Given $y = g^x \in \mathbb{G}$, we describe two proofs of knowledge of discrete logarithm in Figure 3.3. The first proof in Figure 3.3a was introduced by Schnorr [129] and is interactive. This proof of knowledge system is only known to be honest-verifier zero-knowledge in which the verifier is assumed to behave honestly. The reason is that if the malicious verifier adaptively chooses the challenges c, we do not know whether the verifier can extract more value from x. In other words, we cannot construct an efficient simulation which can anticipate the challenges adaptively chosen by the malicious verifier. In order to make the system zero-knowledge in the standard definition, we can make use of the Fiat-Shamir heuristic technique [122] to transform it to a non-interactive proof system [130] as shown in Figure 3.3b. The main difference between the two proof systems is that the random challenge from the verifier is replaced by a hash computed by the prover.



(a) Proof of knowledge of discrete logarithm

(b) Zero-knowledge proof of knowledge of discrete logarithm

Figure 3.3: Proofs of knowledge of discrete logarithm

3.5.2.4 Commitment Schemes

The notion of commitment scheme was introduced by Blum et al. [131] and Even et al. [132]. Loosely speaking, a commitment scheme allows a user to choose a value from a very large set and commit to this value so that the user cannot change it anymore. Such a scheme also allows the user to hide the chosen value and only reveal it to other parties at a later time. More intuitively, we consider a protocol between two parties, a sender $\mathcal{S}$ and a receiver $\mathcal{R}$ that follow the steps below [133]:

Step 1: $\mathcal{S}$ chooses a value x from a large set, writes it on a piece of paper, then locks it into a box so that no one can read the value.

Step 2: $\mathcal{S}$ sends the box to $\mathcal{R}$.

Step 3: $\mathcal{S}$ opens the commitment and reveals x to $\mathcal{R}$.

In order to present commitment schemes more clearly, we consider a practical example, namely coin flipping by telephone, introduced by Blum et al. [131]. In the underlying model,

two parties Peggy and Victor who do not trust each other try to play the game. To ensure the integrity of the game, the two parties follow the steps below:

Step 1: Peggy first selects his bet on *Heads* or *Tails* which are represented by a bit b of information. If Peggy chooses *Heads*, b is set to be 0. In the other case, b is set to be 1. Then Peggy commits to b and sends the commitment to Victor.

Step 2: Victor tosses a coin then announces the result.

Step 3: Peggy reveals the committed value b.

More formally, a commitment scheme is a two-phase protocol between two players $\mathcal{S}$ and $\mathcal{R}$. The scheme consists of 3 algorithms:

Setup(1^k): Given a security parameter k, the algorithm generates the public parameters.

Commit(m, r): Given a message m, $\mathcal{S}$ takes some random value r to commit to m. The outputs of the algorithm are the commitment c and an open value d.

Decommit(c, m, d): Given a commitment c, a message m, and an open value d, $\mathcal{R}$ rejects or accepts them as a true tuple.

It is not hard to realize that a commitment scheme must satisfy two essential properties which are *binding* and *hiding*. The *binding* property ensures that $\mathcal{S}$ cannot change the committed after the commitment phase. The *hiding* property ensures that $\mathcal{R}$ cannot read the committed value before the decommitment phase. Both properties come in two flavors:

- *Computational binding*: This security notion means that a cheating sender $\mathcal{S}^*$ cannot open the commitment to two different values except with a negligible probability. This can be expressed as the probability of breaking a hardness problem which is computationally hard to be solved.

- *Unconditional binding*: This security notion means that even if a cheating sender $\mathcal{S}^*$ has infinite computing power, it cannot open the commitment in two different ways.

- *Computational hiding*: This security notion means that an adversarial polynomial-time receiver $\mathcal{R}^*$ cannot successfully guess the committed value in a commitment except with a negligible probability.

- *Unconditional hiding*: This security notion means that even if an adversarial receiver $\mathcal{R}^*$ has infinite computing power, it cannot successfully guess the committed value.

For *computational binding* and *computational hiding*, the hardness of computational problems increases in the size of the security parameter k used in the setup phase. It is, however, well-known that a commitment scheme cannot achieve both unconditional binding and unconditional hiding [133].

Since its inception, commitment scheme has become a fundamental primitive for many cryptographic applications especially in secure multi-party computations and privacy-preserving technology. In this chapter, we use the Pedersen commitment scheme [134] for constructing a zero-knowledge proof of knowledge system. In fact, the Pedersen commitment scheme has been proven to be perfectly hiding and computational binding. The binding property relies on the discrete logarithm assumption:

Let us consider $\mathbb{G} =< g >$, a cyclic group of primer order q. Let $g, h \in \mathbb{G}$ be two generators of the group. The Pedersen commitment scheme consists of two following phases:

Commit(x, r): In order to commit to a value $x \in \mathbb{Z}_q$, $\mathcal{S}$ randomly chooses a number $r \in \mathbb{Z}_q^*$ then computes the commitment $c = g^x h^r$.

Decommit(c, x, r): In this phase, in order to open the commitment, $\mathcal{S}$ simply reveals (x, r) to $\mathcal{R}$. The latter verifies whether $c = g^x h^r$ to accept or refuse the opening value.

In the context of the proposed DPKI, the Pedersen commitment scheme is employed to achieve some desired privacy guarantees related to revocation tokens associated with public online keys. For the decommit phase, we do not want $\mathcal{S}$ to directly reveal (x, r) to $\mathcal{R}$. Instead, we create a zero-knowledge proof of knowledge of (x, r) for this purpose. Such a proof allows $\mathcal{S}$ to prove that it knows the committed and opening values without revealing them to $\mathcal{R}$. Our purpose can be described by the following expression:

$$ZKP[(x, r) : y = g^x h^r]$$

In Figure 3.4, $\mathcal{S}$ plays the role of a prover while $\mathcal{R}$ plays the role of a verifier. The proof system is both non-interactive and zero-knowledge.

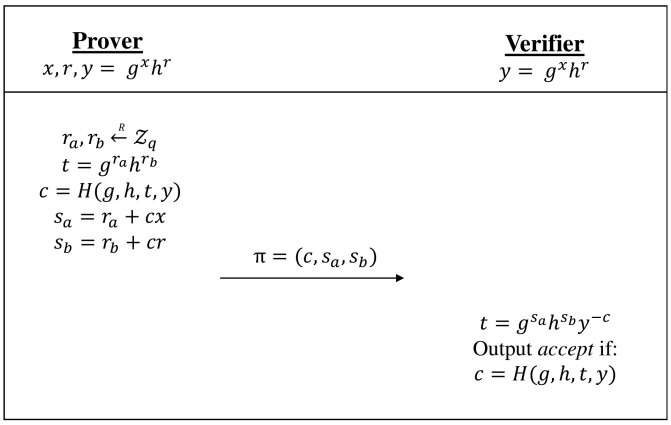


Figure 3.4: Zero-knowledge proof of knowledge of Pedersen commitment

3.5.2.5 Revocation Token

We introduce an additional notion, namely revocation token. In practice, each public online key is stored along with a revocation token computed by the identity owner on a GISC. As the name suggests, a revocation token allows revoking the associated public online key in case of necessity. Specifically, in order to revoke a public online key nPK, a user is provided with two options: either it proves the knowledge of the private online key nSK or it proves the knowledge of its private identity key iSK in a zero-knowledge way. The first option seems to be obvious and can be carried out by following the zero-knowledge proof of knowledge of discrete logarithm in Figure 3.3b. The reason we need the second option is that the private online key is active most of the time, thus facing a huge risk of loss. In such a case, the second option should be used to perform the task of revoking nPK. However, due to the decoupling mechanism between nPK and the identity key iSK, enabling the user to revoke nPK by using iSK seems a bit contradictory. This is where our zero-knowledge proof of knowledge comes into play. More clearly, at the time of registering an online public key in the GISC, the user computes a revocation token according to Algorithm 3.4. In fact, the token is a result of committing to the private identity key iSK. The pair of nPK and T are stored in the GISC. On one hand, due to the hiding property, an adversary cannot extract any information from the token unless it can break the discrete logarithm assumption. On the other hand, the zero-knowledge proof of

knowledge described in Figure 3.4 allows the user to prove the knowledge of (x, r), thus being able to revoke nPK. As we shall see in the next section, revocation token also plays a vital role in helping a user disclose linkings between its public online keys and public identity key.

Algorithm 3.4. Generating Revocation Token

Input: $iSK = x$ and $iPK = g^x$
Public parameters $g, h \in \mathbb{Z}_q$
Output: A revocation token T.
1: Chooses a random number $r \in \mathbb{Z}_q^*$.
2: Computes $T = g^x h^r$.
3: **return** T

3.6 Privacy-Enhancing Decentralized Public Key Infrastructure

In this section, we describe the operations which are necessarily supported in the proposed DPKI, including identity registration, key and identity data update, key revocation as well as key recovery.

3.6.1 Identity Registration

In order to register a decentralized identity with the blockchain, a user generates an identity key pair (iPK, iSK) on its local device. Then, the user deploys an instance of PISC and sends its identity data, denoted by ID, along with the public identity key iPK to the contract.

In fact, ID could be simply a username or a URI pointing to the location where a user's identity data is stored. In fact, storing valuable data, i.e, certified identity data, even in an encrypted form directly on the blockchain is neither secure nor practical. Indeed, with a rapid evolution of the quantum technology, adversaries can decrypt all the encrypted data stored on the blockchain which will never be removed. Moreover, storing data on the Ethereum blockchain is also extremely expensive. In the case of the URI, a URI may point to a container of data records which relates to the user. Each record contains data in one of the different types according to the different levels of trust as follows:

- Self-Assertion: This type involves data which is claimed by the user itself.
- Peer Assertion: This type involves data which is signed by other users in the system. In fact, this approach underlies the Web of Trust model.
- High Trust Assertion: This type involves data which is signed by institutions, i.e., banks, universities, or governments, which can provide a high trust level to the data.

Each data record is signed to bind the data record to the address of the corresponding PISC. This is to ensure that if the user wants to update its identity keys, it does not need to request new signatures for its data records again. Moreover, periodic key rotation is a common practice to reduce the power of cryptographic keys in the long term. In the traditional CA-based PKIs, a user always needs to request new signatures from certificate authorities in this context.

In practice, all the data records are symmetrically encrypted by using different keys. For simplicity, the user can generate a master key msk from which it derives an encryption key sk_{r_i} for each record r_i by applying a secure key derivation function (KDF). Therefore, in order to decrypt any data record, one would need the user to release the corresponding key.

$$sk_{r_i} = KDF(msk, aux)$$

where aux is auxiliary data to the record r, i.e., type of data, index of the record.

In reality, when a user shows some of its data records to a verifier, the latter will decide whether it chooses to believe the signer or not.

3.6.1.1 Online Key Registration

With regard to online keys which serve as a user's identifiers to different service providers, we need a more sophisticated way to register them according to the context. In the proposed DPKI, we aim at protecting user privacy in both open and specific environments.

In an open environment, any user may join the system without eligibility verification. More specifically, a user generates a pair of online keys (nPK, nSK) and a revocation token T. The token is generated according to Algorithm 3.4. The user then signs the token by using the newly generated private online key nSK. The resulting signature is denoted by σ. Thereafter, the user forms a request whose format is depicted in Figure 3.5. The request is broadcast to the blockchain network to include the tuple of nPK and T in the GISC.

nPK	$T = g^x h^r$	σ

Figure 3.5: Online key registration format in open environment

In a specific environment, only users of the environment, i.e., a university, a company, a country, or a federation of services can join the system. For example, for a university which proposes many services such as voting, online courses, or a discussion forum, only students of the university are eligible to register online public keys into the GISC to use these services. A student can also register a different public online key for each service. In practice, the GISC is owned by the owners of services that can define security and privacy restrictions that users need to accept to be able to register online keys. We also note that in this example, the online key registration should not allow the university or any third party to identify the linking between a public online key and the public identity key of a user. To this end, we propose an anonymous online key registration procedure as follows:

The user fetches from the blockchain a list of public identity keys $\mathcal{L} = \{iPK_k\}_{k=1..n}$ of eligible users including its own key which has been registered in the system. Next, the user takes its private identity key iSK and $\mathcal{L}$ as inputs to create a ring signature σ_r over a tuple of 2 elements. These are the public online key nPK and a revocation token T. The latter is simply a Pedersen commitment of its private identity key iSK. The tuple and the signature σ_r compose a request depicted in Figure 3.6 which is sent to the GISC for inclusion into the blockchain.

In reality, the anonymous online key registration procedure can be implemented in two different ways which are off-chain and on-chain. For the off-chain approach, the service owners keep a list $\mathcal{L}$ of public identity keys of eligible users in their own database. In order to register

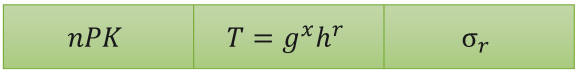


Figure 3.6: Online key registration format in specific environment

a pubic key, the user needs to get $\mathcal{L}$ from the service providers to generate a key registration request. Then, the providers check the validity of the request and insert the online key and the associated revocation token to the GISC. For the on-chain approach, the service owners directly store $\mathcal{L}$ or a hash of $\mathcal{L}$ in the GISC. More specifically, before a key registration, the user still needs to interact with the service owners for eligibility verification. If the verification passes, the service owners will add the user's public identity key to $\mathcal{L}$. In order to register a public online key, the user sends a request to the blockchain validators. The latters will check the validity of the request based on the current list $\mathcal{L}$ in the GISC. The main difference between the two approaches lies in the way that $\mathcal{L}$ is stored and managed. Clearly, the on-chain approach is preferable as it provides transparency to $\mathcal{L}$. We stress that eligibility verification is normally an off-chain process and it is generally insufficient to say that a user is actually using the proposed services. For example, all students of a university are eligible to use services proposed by the university but one can not say whether a certain student is actually using one of these services.

3.6.1.2 Identity Key Update

In order to update a public identity key $iPK = g^x$, which is coupled with an identity data ID on the PISC, a user needs to prove the ownership of iPK. That is, the user must know the corresponding private key iSK. For this purpose, the user first generates a new identity key pair $(iSK' = x', iPK' = g^{x'})$. Next, the user signs iPK' with iSK. The resulting signature σ and iPK' are then sent to the PISC. Once the validators accept the transaction, the public identity key of the user is updated to iPK'. We note that the update operation should be implemented with a timelock function. That is, the new identity key iPK' will only be really valid after a reasonably short period of time from the transaction verification. Such a period has been predetermined by the user. This is to make sure that if the new key is being introduced by an adversary which has managed to compromise the user's private identity key iSK, the user will have enough time to invalidate it by using iSK. A successful update process explicitly

Figure 3.7: Public identity key update format

invalidates the old public identity key because the system only allows one identity key pair for each user at any time.

3.6.1.3 Online Key Update

The update of a public online key nPK can also be done in a similar way to the identity key update by proving the knowledge of the corresponding private online key nSK. As a nSK needs to be connected most of the time to the Internet for interacting with the corresponding service

providers, there are substantial risks that it gets lost or compromised, making the update of the associated nPK impossible.

In this context, we have introduced the notation of revocation token which is computed as in Algorithm 3.4. The revocation token stored along with nPK on the GISC serves as an additional way to achieve the objective. More specifically, the user follows the zero-knowledge proof of knowledge system that we described in Figure 3.4 to generate a proof π. The latter explicitly indicates that the user knows the committed value in the revocation token. Then, π is signed with a newly generated online private key nSK'. Finally, the user broadcasts an update request consisting of π, the signature σ, and the associated public nPK' to the blockchain network. The blockchain validators will check for the validity of π as well as the associated signature. If they accept the request, the public online key will be changed to nPK'.

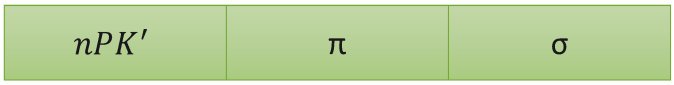


Figure 3.8: Public online key update format

3.6.2 Key Revocation

One can think of a key revocation mechanism as a key update without introducing new keys. The revocation mechanism in the proposed DPKI is realized via the zero-knowledge proof of knowledge systems that we introduced earlier.

Specifically, in order to revoke a public identity key, the user must prove the knowledge of the associated private identity key. For this purpose, it can simply follow the proof system in Figure 3.3b to generate a witness π. The latter is then broadcast to the blockchain network for validation.

With regard to a public online key, the user has two options: it generates a witness π to state that it knows either the associated private online key or the committed value in the associated revocation token by using the private identity key. The former one can be done by following the proof system in Figure 3.3b whereas the latter one can be realized by following the proof system in Figure 3.4. The resulting witness is also then broadcast to the blockchain network for validation.

3.6.3 Key Recovery

In order to reduce the key management complexity, we only propose a key recovery mechanism for identity keys. The reason is that even if a user has lost a private online key, it still can revoke the key and then introduce a new one by using the private identity key to prove the knowledge of the committed value in the associated revocation token. As the old and new public online keys are technically bound together in the GISC, the user can prove this binding to the service provider without the fear of losing its account.

The key recovery mechanism is a social off-blockchain solution, and is based on the social backup mechanism introduced by Shamir in [135]. Specifically, a user divides its private identity key iSK into n shards so that the knowledge of k out of n shards allows reconstructing the private key. This is achieved by using the Lagrange polynomial interpolation. Practically, n shards are securely sent to n trusted entities. We also recommend that the designed entities

should not know each other to avoid that k entities out of them curiously reconstruct iSK without an explicit permission from the user.

3.6.4 Revealing hidden linking between online and identity keys

In the proposed DPKI, linkings between a user's public online keys $\{nPK_i\}$ and public identity key iPK is hidden by default. In practice, a user may only have one identity key pair but it can have many online key pairs at a time. Revealing the hidden linking between a public online key nPK_i and iPK does not allow discovering the linkings between iPK and the other online keys. In order words, if a user discloses the linking between its identity key and one of the online keys to a service provider, namely $\mathcal{A}$, other service providers cannot gain additional knowledge from it, even with the help of $\mathcal{A}$.

For this purpose, it is sufficient that the user can prove the hidden relation between iPK and the revocation token T which is stored along with nPK_i in the GISC. More specifically, if the user can somehow prove that T and iPK are owned by the same user, i.e, they are generated from the same identity private key iSK, this linking disclosure process succeeds. Practically, this process can be realized via a zero-knowledge proof of knowledge system which is described by the following expression:

$$ZKP[(x, r) : C = g^x \wedge V = g^x h^r]$$

where C is the public identity key iPK, V is the revocation token associated with the given public online key, x is the user's private identity key iSK, and r is an opening value of V.

In practice, we can efficiently transform the above proof system into another which can be realized more easily via the following expression:

$$ZKP[(x, r) : C = g^x \wedge VC^{-1} = h^r]$$

We now remark that the transformed proof system consists of two zero-knowledge proof of discrete logarithm systems, which we described in Figure 3.3b.

3.7 Discussion about the applicability of the proposed DPKI

3.7.1 Construction of a mutual authentication and key exchange protocol

In this section, we discuss some applications of the proposed DPKI in data storage and sharing platforms and other contexts. The first use case that one can think of is building an authentication and key exchange protocol (MAKE) between users and service providers. Practically, users can authenticate themselves with a cloud-based service provider by using their public online keys. Moreover, depending on the context, a user may choose to reveal the linking between its online and identity keys. In practice, MAKE protocols are integral to secure communication between entities over insecure channels. Simply put, MAKE protocols allow two parties to authenticate each other over public channels and then mutually agree upon a session key which will be used to secure their communications. Due to the great importance of such protocols,

there have been many proposals of MAKE protocols built upon different underlying cryptosystems such as PKI [45], Certificateless PKI [136], Identity-Based Cryptography [137, 138], and Password-based MAKE protocols [18, 31, 139]. There are also works extending the original idea to deal with multi-party settings [5, 140, 141] in which a group of users engages in MAKE protocols. Similar approaches work well for the proposed DPKI. Figure 3.9 shows a Diffie-Hellman based authentication and key exchange protocol between a user and a service provider. In this case, the server is a well-known entity which is represented by its public identity key. We argue that preserving the privacy of such an entity does not make sense. Therefore, the service provider conducts the protocol by directly using the identity keys. On the contrary, the user only uses one of its online key pairs to run the protocol. We detail the workflow of the protocol as follows:

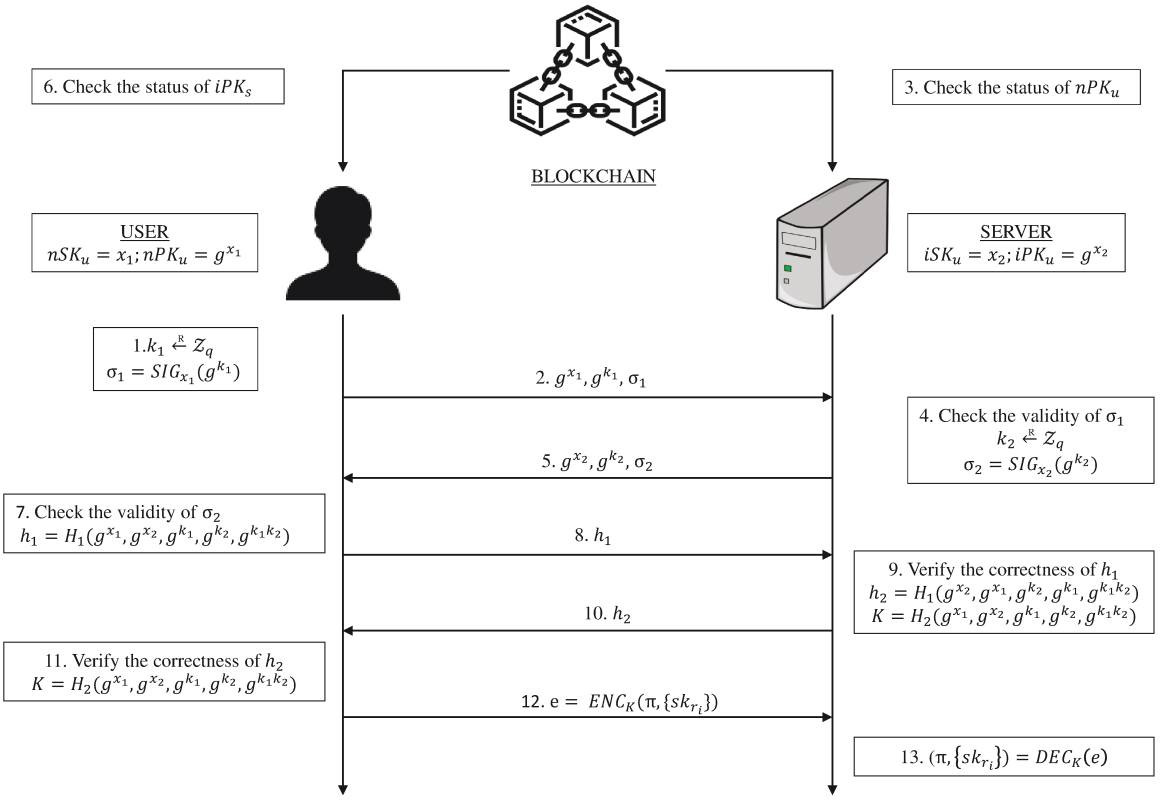


Figure 3.9: The workflow of a Diffie-Hellman based authentication and key exchange protocol

Given a security parameter l, the system chooses two different collision-resistant hash functions $H_1, H_2 : \{0,1\}^* \rightarrow \{0,1\}^l$. Let $\mathrm{ENC}_k(m)$ and $\mathrm{DEC}_k(m)$ respectively denote the symmetric encryption and decryption on m using a secret key k. Let $\mathrm{SIG}_{sk}(m)$, $\mathrm{VER}_{pk}(m)$ denote the signing and verifying operations on the message m with a signing key sk and a verifying key pk.

Step 1: In order to start the protocol, the user $\mathcal{U}$ picks a random number k_1 over $\mathbb{Z}_p$ and signs it with the private online key nSK_u. Conceptually, k_1 serves as an ingredient in the composition of the session key.

Step 2: $\mathcal{U}$ sends its public online key nPK_u, g^{k_1}, and the resulting signature σ_1 to the server $\mathcal{S}$ for verification. The message is implicitly considered as an authentication request.

Step 3: Upon receipt of the authentication request, $\mathcal{S}$ retrieves the revocation information

of nPK_u from the blockchain validators. If nPK_u has been revoked, $\mathcal{S}$ aborts the protocol. We note that $\mathcal{S}$ needs to sample a subset of available blockchain validators for information request. This is to prevent some dishonest validators from trying to manipulate the final result. If the results from the selected validators are identical, $\mathcal{U}$ can believe in the accuracy of the result. Otherwise, $\mathcal{U}$ samples another subset of validators to get the accurate and coherent result.

Step 4: $\mathcal{S}$ checks the validity of σ_1. If the latter is not valid, $\mathcal{S}$ aborts the protocol. Otherwise, $\mathcal{S}$ also picks a random number k_2 over $\mathbb{Z}_p$. Similar to k_1, the main role of k_2 is to contribute to the composition of the session key.

Step 5: Then, $\mathcal{S}$ signs g^{k_2} by using its private identity key iSK_s. The resulting signature σ_2 is sent to $\mathcal{U}$ for verification.

Step 6: Upon receiving the message from $\mathcal{S}$, $\mathcal{U}$ verifies the revocation status of iPK_s. If it has been revoked, $\mathcal{U}$ aborts the protocol. Otherwise, it proceeds to the next step.

Step 7: $\mathcal{U}$ verifies the validity of σ_2. If the latter is not valid, $\mathcal{U}$ aborts the protocol. Otherwise, $\mathcal{U}$ continues computing h_1 as an authentication code by using the hash function H_1.

Step 8: $\mathcal{U}$ sends h_1 to $\mathcal{S}$ for verification.

Step 9: $\mathcal{S}$ recomputes h_1 and compares it with the value received in Step 8. If they are not equal, $\mathcal{S}$ aborts the protocol. Otherwise, $\mathcal{S}$ computes another authentication code h_2 using the hash function H_2.

Step 10: $\mathcal{S}$ sends the authentication code h_2 to $\mathcal{S}$ for verification.

Step 11: $\mathcal{U}$ checks the validity of h_2. At this stage, if both h_1 and h_2 have been correctly computed, the two parties know that they are mutually authenticated and agree on the same session key K. The communication between the two parties is subsequently encrypted by the session key.

Step 12(optional): This step is optional and dependent on the context in which $\mathcal{U}$ wants to disclose the linking between nPK_u and its public identity key as well as some data records $\{r_i\}$. For this purpose, $\mathcal{U}$ generates a witness π for the linking and sends it with the corresponding private keys sk_{r_i} which are used to symmetrically encrypt the related data records.

Step 13(optional): $\mathcal{S}$ uses the session key to decrypt the received message, resulting the witness and the private keys.

3.7.2 Construction of a mutual authentication and key exchange protocol for a group of users

Beyond such a simple authentication and key exchange protocol, we may also think of a multi-party setting in which a group of users may also directly use their public online keys to build a private channel which is secure against any outsider adversaries and even the service provider. The channel is then used to securely exchange data between the involved parties. This approach has been broadly deployed in many real-life applications, i.e., instant messaging, video conference, with the common name: end-to-end encryption. We refer to [142–144] for some typical examples of such an approach.

3.7.3 Applications in contexts where user privacy is a necessity

Besides, the proposed DPKI can be adopted in different contexts where user privacy is a desirable property. We re-examine the use case in which a university offers different services to its students. In reality, a student still needs to show some of its identity data, such as data

indicating that the student belongs to the university, to be eligible to join the system via the anonymous key registration procedure and benefit from the proposed services. However, the university cannot learn who is using which service as each student has multiple public online keys as their identifiers to these services. In this case, all public online keys of all the students should be managed by only one instance of GISC. However, if a student has violated some of the rules predefined by one of the services, the user can still be banned and cannot register a new public online key for that service. We observe that in the proposed DPKI, user privacy guarantee is a priority and unless the students chose to reveal their identity public keys, there is no way for the university to know such information. To mitigate the problem, we consider two different approaches which can be programmed in the GISC owned by the university. The first one is based on a trusted party, namely an inspector. More specifically, at the time of the online key registration, a student also needs to generate a proof π for the linking disclosure. This proof is then encrypted and sent to the inspector such that only the inspector can decrypt and verify the proof. The inspector is supposed to work honestly. That is, upon receipt of a proof, the inspector checks its validity and sends the result to the GISC. If the result is *true*, the registration is completed. Otherwise, it failed and the student's online key is not included in the GISC. Moreover, in case that the student violates some predetermined rules, the inspector will reveal the linking to the GISC to prevent it from registering a new online key. The other approach is more heuristic but does not involve a trusted party. Intuitively, the university can encode an economic model in the GISC so that at the time of key registration, the student also needs to lock some ETHs, the cryptocurrency underlying the Ethereum blockchain. In the case that the student violates the rules, the service provider can trigger a function in the GISC so that the student is given some time to send a valid proof of the linking. If the student fails to do so, the service provider will take the locked ETHs from the student. Besides, the student can also unlock its ETHs in case that it has not violated any rules and does not want to continue using these services.

3.8 Privacy Analysis

In this section, we examine the security and privacy guarantees of the proposed DPKI with regard to blockchain validators and third parties.

3.8.1 Privacy Vulnerabilities with respect to Blockchain Validators

We prove that the blockchain validators ($\mathcal{BV}$s), which are responsible for maintaining the blockchain, are unable to correlate users activities based on user information publicly available on the blockchain.

In the proposed infrastructure, we employed a dual key model in which each user has an identity key pair and a set of online key pairs. In order to ensure the cryptographically hidden linking between a public online key and a public identity key, we used two cryptographic mechanisms, namely ring signature and token revocation. The former one allows an eligible user to anonymously register a new public online key into a deployed GISC without revealing its real identity. The latter one allows for key revocation, update, and especially revealing the hidden linking in case that the user decides to do so. In fact, we used the ring signature scheme proposed by Herranz et al. [104] which was proven to be existential unforgeability under adaptive chosen-message attacks in the random oracle. With regard to revocation token, which

is technically a Pedersen commitment [134] to the user's private identity key. In fact, the scheme is also perfectly hiding, thus making it impossible to extract private information from it even for adversaries which have infinite computing power.

Moreover, a user uses a different online key pair to authenticate with each service provider. $\mathcal{BV}$s are responsible for replying to revocation information requests concerning the public online keys from the service providers. Due to the proposed decoupling mechanism between the public identity key and public online keys, $\mathcal{BV}$s have no way of correlating the user's activities such as where the user has visited.

3.8.2 Privacy Vulnerabilities with respect to Third Parties

As a user can use different online key pairs (nPK_i, nSK_i) to authenticate with each service provider, we observe that the latter gains no advantage over the blockchain validators when it comes to correlating the user's activities based on information publicly available on the blockchain. In addition, eligible users register their public online keys via the anonymous key registration procedure. This prevents the service provider from identifying users who are using the service while being ensured that they are eligible.

However, in the case that the user has disclosed linkings between the public identity key iPK and some of the public online keys nPK_i, this potentially offers the service providers a means of correlating the user's activities. We stress that the disclosure can only be realized if the user wants to do so. In some contexts, it is also needed to prevent the user from violating the rules imposed by the service providers. In practice, the disclosure process is realized via a zero-knowledge proof of knowledge system described in the section 3.6.4. The security of the proof relies on the discrete logarithm assumption. At a higher level, the proof is based on the fact that the user owns both the revocation token and the public identity key. We remark that each online public key is stored with a different revocation token which is generated by the user by committing to its private identity key using a different opening value. Therefore, revealing the linking between iPK and a public online key does not help discover the linkings between iPK and other public online keys.

3.9 Performance Evaluation

In this section, we experimentally evaluate the performance of the proposed DPKI by measuring the computational costs of the computationally heavyweight operations performed by users as well as the costs of deploying the smart contracts on the Ethereum blockchain.

3.9.1 Evaluation of the Smart Contract Deployment Cost

The Ethereum blockchain is maintained by an increasing number of miners that are incentivized by the underlying economic model to verify and add transactions to the blockchain. Therefore, users have to pay the miners to execute all types of operations, including smart contract deployment and executing functions in the deployed smart contracts. The more complex the operation we want to execute, the more fees we have to pay. In fact, these operations are initiated by broadcasting the corresponding transactions to the network. Depending on the type of transaction, the miners will act accordingly. In the Ethereum blockchain, *gas* is used as a unit of measurement that defines the amount of computational effort needed to execute an

Table 3.1: Deployment costs of smart contracts

Contracts	Transaction Cost (gas)	Execution Cost (gas)	Total Deployment Cost (gas)
GISC	1734598	1586541	3321139
PISC	1489091	1267811	2756902
Ring Signature Contract	2442294	1802498	4244792

operation. Generally, to initiate a transaction, a sender must specify two variables: *gas limit* and *gas price*. The former indicates the maximum amount of *gas* that the sender is willing to pay for the transaction while the latter indicates the price of *gas*. For example, a standard ETH transfer requires a *gas limit* of $21,000$ units of *gas*. The *gas price* is set based on ETH, the underlying cryptocurrency of the Ethereum blockchain. More specifically, 1 ETH is divided into smaller units, namely *wei* (1 ETH = 10^{18}*wei*). In fact, some wallets and service providers which facilitate interactions between users and some specific smart contracts automatically set *gas limits* and *gas prices*. However, users can still adjust these values. We remark that if we set the *gas price* too low, the miner could ignore our transaction and make it stuck for a long time.

In the platform, there are 2 main types of contract, namely PISC and GISC, which are written in the Solidity language. In the proposed DPKI, one instance of GISC can be used to globally manage all users for a federation of services. With regard to PISC, each user has to deploy one instance of PISC to manage their identity key and identity data. The identity management involves operations such as registration, revocation, update. We also deploy one instance of the Ring Signature contract which is called by the GISC whenever it wants to check the validity of a ring signature made by a user. Table 3.1 shows the deployment cost of these smart contracts.

3.9.2 Evaluation of Computationally Heavyweight Operations

On the user side, the main computational burden is caused by the cryptographic operations which are performed on their devices such as PC or mobile. We observe that the generation of keys or revocation tokens requires a small number of lightweight operations, including additions and multiplications. Therefore, we concentrate on the computation costs of the heavyweight operations which can also vary and depend on the context: the ring signature. With regard to the ring signature scheme, it consists of signing and verifying algorithms. The cost of these algorithms is linearly dependent on the number of public keys that the signer takes to create a ring signature. In the proposed DPKI, these operations are needed whenever a user wants to introduce a new public online key in the GISC. More specifically, the signing operation is performed on the user side whereas the verifying operation is performed by the blockchain miners and is implemented in the Ring Signature contract. Due to the huge computation capacity of the blockchain miners, the verifying operation can be done nearly instantly. However, for the completeness of the demonstration, we show the costs of both operations when being carried out on the experimental computer.

We implement the ring signature scheme on a computer with Core i7 2.81GHz processor and 8Gb RAM. The ring signature scheme is implemented on the curve ECSecp256k. As illustrated in Figure 3.10, the ring signature scheme is practical as it takes less than 2.5s to create a signature on behalf of a large group of 100 members. Moreover, the signature verifying algorithm respectively only requires 1s for such a signature. This scheme is, therefore, efficient

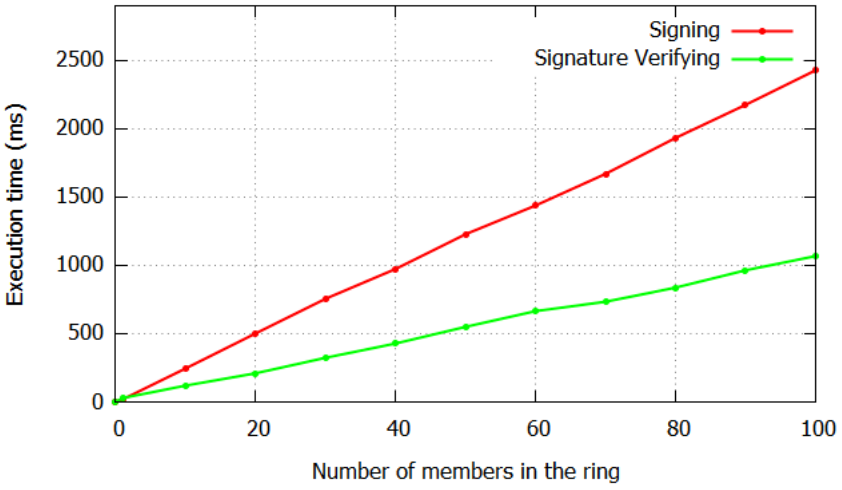


Figure 3.10: Time execution measurement of the ring signature

to be adopted in the proposed DPKI.

3.10 Conclusions

In this chapter, we proposed a privacy-enhancing decentralized public key infrastructure to overcome the security and privacy problems that we encounter in the traditional PKI approaches as well as in the existing DPKIs in the literature. Like other existing DPKIs proposals, we constructed our DPKI on a smart contract platform which is the Ethereum blockchain in our case. Whereas the blockchain facilitates the deployment of a user-centric identity management system, it potentially turns into a central point of correlation, thus allowing unauthorized parties to correlate activities related to a targeted user. Therefore, we proposed a key decoupling mechanism to prevent such attacks. In practice, the application of the proposed DPKI in the authentication protocols allows preserving user privacy with respect to blockchain validators as well as third parties by default. However, users are also allowed to choose to reveal its identity key to some of the third parties if they want to do so. In order to demonstrate the applicability of the proposed DPKI, we also discuss some applications in different contexts. We also experimentally measure the computation cost of heavyweight operations as well as the deployment cost of the global identity smart contract and the individual identity smart contract, which are the two primary components of the proposed DPKI, in the Ethereum blockchain.

Chapter 4

Forward-Secure Data Outsourcing Based on Revocable Attribute-Based Encryption

Contents

4.1 Foreword

Cloud-based storage has become an increasingly popular solution providing data availability and data access management for both personal and organizational purposes. For the sake of users' protection, many widely-used data storage platforms often take the responsibility to ensure data confidentiality. From the security perspective, data owners have to rely on the service providers, which are generally not fully trusted. To address this issue, it is highly recommended to encrypt data on the user side before outsourcing. This, however, leads to an important complexity in the key management when it comes to group-based data sharing in which group members change dynamically over time. Indeed, anytime a data owner initiates a sharing, it generates a secret key to encrypt the shared data. To enable authorized users to decrypt the data, the data owner has to securely communicate the key to the corresponding group members in which some of them might not be always reachable at some point in time. Moreover, the group dynamicity significantly increases the complexity of key management as revoked users might lose the capacity of decrypting the shared data. In this context, Bethencourt et al. introduced the first Ciphertext-Policy Attribute Based Encryption (CP-ABE) scheme [145], a user-centric approach, to efficiently handle both the data confidentiality and fine-grained access control.

They achieved this result by moving beyond the traditional public key crytosystems in which each user holds a key pair and which are more suited for two-party communication.

In a CP-ABE system, there exists a trusted authority called Key Distribution Center (KDC) which uses a randomly chosen master key to distribute private keys to users according to their corresponding certified representative attribute sets. Before being outsourced to the cloud, data is encrypted by a data owner under its desired access structure. Upon receiving the ciphertext from the cloud, any user who possesses a certified attribute set satisfying the defined access structure can decrypt data using its own private key without interactions with the data owner. An illustrative example of the scheme is shown in Figure 4.1. In possession of a private key associated with a set of attributes, including *Manager* and *IT dept*, Bob is capable of decrypting data encrypted with the access structure: *Marketing* ***AND*** *(Manager* ***OR*** *IT Dept).*

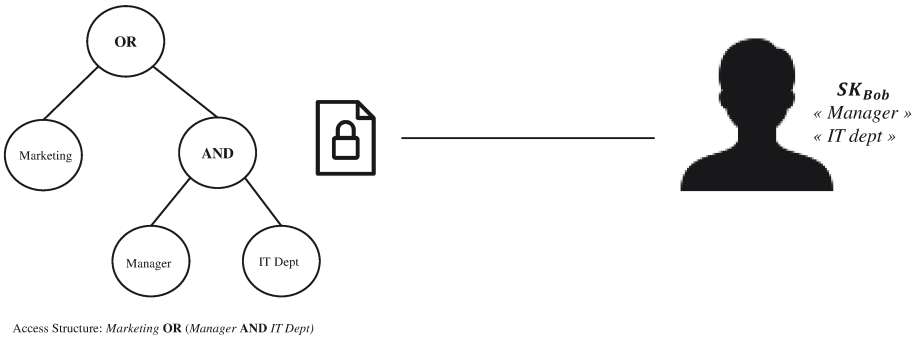


Figure 4.1: An illustrative example of CP-ABE scheme

Despite the novel and positive features it brings, the CP-ABE adoption is restricted due to the lack of efficient user and attribute revocation mechanisms. Intuitively, users may share some common attributes which could change over time. Therefore, without an effective revocation mechanism, revoking a certain user or some of its attributes often results in revocation of others in the system. Moreover, forward secrecy requirement which prevents revoked users from decrypting past shared data is not originally taken into account in CP-ABE schemes. In order to address the revocation issue, there exist two approaches: indirect revocation and direct revocation. Indirect revocation requires the KDC to periodically distribute a key update to all unrevoked users [145]. Even if a user is revoked, its private key is still valid until the end of the current time interval. Alternatively, direct revocation allows revoking users as soon as necessary [146, 147], and fits into systems with high security requirements. The existing CP-ABE schemes require unrevoked users to locally keep a long and up-to-date list of revoked users which will be embedded in ciphertexts. This makes the computational costs of encryption and decryption algorithms increase linearly to the number of revoked users on the list. As the system grows rapidly over time, users may incur a huge computational overhead. With regard to forward secrecy, the natural solution is to require unrevoked users to re-encrypt all the shared data. The solution is, however, not computationally efficient especially in the case that the volume of data is huge. On the other hand, if the data is re-encrypted by the storage service using its own secret key, the service will take full control of revocation [148], thus removing the original user-centric characteristic of CP-ABE.

The third contribution of this thesis is to construct two secure CP-ABE schemes with direct revocation capability. The first scheme, U-CPABE, allows user revocation whereas the second one, A-CPABE, supports attribute revocation. Firstly, unlike the existing schemes, the

decryption and encryption computational costs of our proposals are independent of the number of revoked users in the system, making them suitable for resource-constrained environments. Secondly, we integrate proxy re-encryption techniques to make the proposed schemes forward-secure and keep them user-centric.

4.2 Related Work

In this section, we first present the design principles of the existing provably secure attribute-based encryption schemes and their underlying security models. We then describe the methodologies for integrating revocation mechanisms into these schemes. Finally, we present some concrete realizations of the methodologies proposed in the literature as well as a high level intuition of our proposals.

4.2.1 Attribute-Based Encryption

Attribute-Based Encryption was first introduced by Sahai and Waters [149] in which user private key and ciphertext are independently associated with two different sets of attributes S and S'. During the key request, a threshold parameter k is also embedded into the user private key via a k-degree polynomial. For a user to be able to decrypt a ciphertext, there must be at least k attributes overlapped between two attributes sets ($|S \bigcap S'| \geq k$), which are associated with the ciphertext and the user private key. This construction is often referred to as Threshold ABE. Consequently, there have been two approaches developed to improve the expressiveness of Threshold ABE. The first one is Key Policy Attribute-Based Encryption (KP-ABE) introduced by Goyal et al. [150]. In this model, a user key is associated with an access structure whereas a ciphertext is associated with a set of attributes. If a user's attributes satisfy an access structure, it will be able to decrypt the associated ciphertext. The second approach introduced by Bethencourt et al. [145], which is Ciphertext-Policy Attribute Based Encryption (CP-ABE), follows the inverse direction. That is, a ciphertext is associated with an access structure and a user key is associated with a set of attributes. Clearly, CP-ABE better fits the Attribute-Based Access Control (ABAC) model, which is practically adopted by most organizations. According to Gartner's research [151], there will be 70% of all organizations migrating to the ABAC model by 2020. Due to the high interest in the CP-ABE approach, there have been many other variants proposed in the literature which aim at improving security as well as adding more features into their schemes.

As observed by Waters in [152], the main difficulty in designing CP-ABE schemes lies in programming the system's public parameters in the challenge Game between an adversary and a programmed simulator. That is, the system public parameters must be appropriately programmed so that the simulator can correctly reply to all key queries from the adversary. This challenge leads to different choices of security models when designing CP-ABE schemes. More concretely, Bethencourt et al. [145] proved their scheme in the generic group model [153]. This model implies the impossibility of extracting the special structure of the group which the scheme is implemented on. In the simulation language, the adversary needs to access an oracle to perform any operations on the group.

Waters developed three different CP-ABE schemes in [152] based on standard assumptions in the selective model, which was introduced by Canetti et al. in [154, 155]. The selective model is a well-known efficient methodology to prove the security of related attribute-based

encryption systems [17, 149, 152, 156]. More specifically, in CP-ABE systems, the adversary is required to declare the access structure it wants to attack even before receiving the system public parameters. The knowledge of the access structure helps the simulator to program the system public parameters so that it can correctly reply to all private key queries except those that can decrypt the challenge message. From the view of the adversary, it is impossible to distinguish the simulation from the real system. Without selectivity, this requirement could not be achieved because the simulator cannot anticipate which keys the adversary will ask for, thus resulting in the wide use of the selective model.

In order to remove the limitations of traditional security proofs in prior works, the dual system encryption methodology is proposed in [157, 158] to enhance the traditional security models. The aim of this methodology is to provide a convenient way to achieve a fully secure attribute based encryption system. In his work [157], Waters refers to the traditional methodology as the partition reduction in which the simulator partitions the keys it can produce based on the access structure declared by the adversary and the keys that it cannot produce. The latter are those that can trivially decrypt the challenge message. The advantage of the adversary in this Game is equivalent to the advantage of the simulator to break the underlying hardness assumptions. Moving beyond this traditional approach, the proposed dual system encryption methodology consists of a series of Games. We briefly explain the high level intuition of the methodology as follows. In the system, there are two types of keys and ciphertexts, called *normal* and *semi-functional*. Both *normal* and *semi-functional* keys can successfully decrypt *normal* ciphertexts. *Semi-functional* ciphertexts are only decryptable with *normal* keys. There is a negligible probability that a *semi-functional* key can decrypt a *semi-functional* ciphertext. In contrast to the partition reduction, the simulator in this new methodology is so powerful that it can reply to all key and ciphertext queries without the need for selectivity. At the initial Game of the methodology, all keys and ciphertexts are *normal* and are distributed according to the real system. Next, the ciphertext is changed from *normal* to *semi-functional*. Then, all the private keys are gradually changed to *semi-functional* one by one. At the end of the sequence of Games, all keys and ciphertexts are *semi-functional* and the security proof is straightforward. Under underlying hardness assumptions, the probability for an adversary to distinguish the modifications between Games is negligible. Lewko et al. [159] proposed the first fully secure CP-ABE scheme by leveraging the dual system methodology. The complexity of the security proof methodology often results in poorer system performance. Indeed, in order to achieve a fully secure CP-ABE scheme, Lewko et al. constructed it on groups of composite orders. It is well-known that group operations and pairing over groups of composite order are very expensive and sometimes they are not even worth using.

The huge potential of CP-ABE in practice leads to construction of many variants. In [160–162], the authors tried to extend the original setting of CP-ABE to a multi-authority setting in which there were multiple Key Distribution Centers (KDCs). In this extended system, a user can decrypt a ciphertext if its attributes distributed by multiple authorities match the underlying access structure. Matthew et al. [163] worked on improving the efficiency of a decrypting ciphertext which grows linearly to the complexity of the associated access structure. The idea of the proposed technique is to allow a third party, i.e, a computationally powerful storage provider, to blindly translate an ABE ciphertext to a constant-sized Elgamal-like ciphertext [164]. While the third party cannot extract any useful information about the plaintext, the method helps to significantly reduce the decryption workload on the user side.

4.2.2 Attribute-Based Encryption With Revocation

The common issue of all variants of CP-ABE schemes is the revocation as raised in [145, 152], there have been two primary revocation approaches emerging in designing CP-ABE schemes, namely direct and indirect revocations. In the indirect revocation approach [145, 165, 166], the KDC periodically runs a key renewal algorithm and distributes new private keys to unrevoked users. Despite its simplicity in design, this approach encounters a tricky tradeoff between security and system performance. Indeed, if the time interval is set too large, the revoked users are still able to use their private keys to decrypt data until a predetermined key expiration date. On the other hand, if the time interval is too small, it will cause a key distribution workload to the KDC. Besides, the time-based revocation approach makes CP-ABE schemes inappropriate for high security requirement systems.

In the direct revocation class, the works in [146, 167, 168] are designed with the aim of instantly revoking users without any key renewals from the KDC. Specifically, the authors have employed two different broadcast encryption techniques [169, 170] in conjunction with the CP-ABE scheme of Waters [152] to construct two revocable CP-ABE schemes. The first scheme requires an encryptor to locally keep a list of unrevoked users which would be subsequently embedded into ciphertexts. However, the scheme is no longer solely attribute-based as all the unrevoked users are explicitly identified in the system due to the built-in broadcast technique. On the other hand, the second scheme requires an encryptor to embed an up-to-date list of revoked users into ciphertext so that only unrevoked users can decrypt it. As the system grows over time, keeping such a long list in local could be a burden to users. In addition, in this scheme, encryptors and decryptors have to perform an extra number of group operations linearly to the number of revoked users on the list. In [171], Yu et al. presented a new CP-ABE scheme with efficient direct revocation capability. However, the scheme is less expressive as it only supports access structures consisting of AND operators.

The works in [148, 172–175] have recently leveraged cloud-assisted revocation mechanisms. The underlying idea of these solutions relies on the trustworthiness of the service provider. To revoke a set of users, the service provider is responsible for re-encrypting all shared data using a newly generated private key. The latter is only used to decrypt the shared data for unrevoked users. Nevertheless, if the service provider is compromised and the re-encryption key is leaked, the revoked users can decrypt all data that has even been outsourced after their revocations.

4.2.3 Proxy Re-Encryption

In order to prevent revoked users from decrypting data previously shared with them, the simplest strategy is to require an unrevoked user to download then re-encrypt all the shared data before re-outsourcing the data to the cloud. However, the approach seems not to be computationally efficient as the amount of shared data could be huge. The alternative solutions in [148, 172–174] rely on the service provider to re-encrypt data using its own private key. Thereafter, the service provider has to authenticate users and only partially decrypts the re-encrypted data to unrevoked users. These solutions have some inherent drawbacks. First, the service provider manages revocation, removing the user-centric characteristic of CP-ABE. Second, re-encryption keys are persistently kept by the service provider for partial data decryption. If the provider is attacked by an adversary (i.e., revoked users), it can decrypt all the re-encrypted data. In an independent line of work, Liang et al. [176] have taken the initiative to integrate proxy re-encryption techniques into CP-ABE schemes. Roughly speaking, proxy

re-encryption allows translating a ciphertext encrypted with a public key to another ciphertext encrypted with another public key. In the context of CP-ABE, the intuition is to delegate to a semi-trusted proxy to blindly translate a ciphertext under a certain access structure to one under another access structure using a re-encryption key. This key is computed by a data owner and is sent to the proxy. After the re-encryption operation, the re-encryption key can be removed since it is not necessary for decrypting data afterward. This approach allows saving the data owner from the communication overhead of downloading data from the cloud for the re-encryption and avoids the risk of re-encryption key leaking. However, the cost of re-encryption key computation in the proposed scheme is even more expensive than directly re-encrypting data with the second access structure. This contradicts our aim of reducing both computation and communication overheads for the data owner.

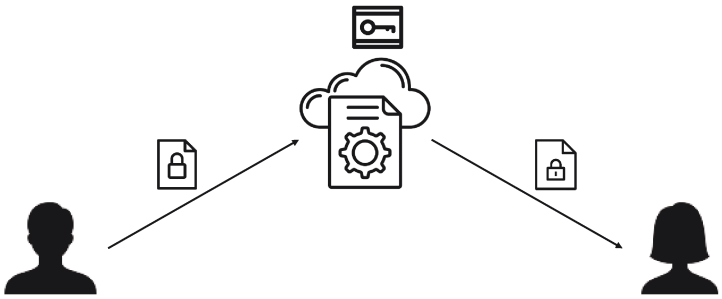


Figure 4.2: Generic cloud-based proxy re-encryption scheme

Blaze et al. [177] introduced an informal concept and the first construction of proxy re-encryption. Technically, proxy re-encryption allows a proxy to blindly translate a ciphertext encrypted with a public key k_1 to another ciphertext with another public key k_2 without the proxy learning any information about the plaintext and the related private keys. The concept was formally improved by Ateniese et al. [178], in which a set of desired characteristics is included to further extend the adoption of proxy re-encryption. Due to the widespread adoption ranging from digital right management [179,180], encrypted mail forwarding [177], to group key management [181], there have been many constructions in the literature [177, 182, 183]. These respectively rely on the underlying public key cryptosystems, including ElGamal encryption [164], identity-based encryption [184], and certificateless public key encryption [136]. Figure 4.2 illustrates the abstraction of how the adoption of proxy re-encryption in cloud-based data storage looks like. In this chapter, we introduce two different proxy re-encryption techniques which are purposely designed to be securely compatible with two proposed revocable CP-ABE schemes as part of this contribution.

4.3 Mathematical Background

In this section, we present the mathematical definitions which are necessary for constructing our proposals.

4.3.1 Monotone Access Structure

As indicated above, the common intuition in the design of CP-ABE schemes is that a user private key associates with a set of attributes whereas a ciphertext associates with an access structure composed of a set of attributes. In this contribution, we only deal with the CP-ABE schemes supporting monotone access structure which includes the attributes of positive literals.

Definition 4.1 (Monotone Access Structure)**.** *Let $\{P_1, P_2..P_n\}$ be a set of parties. A collection of $\mathcal{A} \subseteq 2^{\{P_1,P_2,..P_n\}}$ is monotone if $\forall \mathcal{B}, \mathcal{C}$: if $\mathcal{B} \in \mathcal{A}$ and $\mathcal{B} \subseteq \mathcal{C}$ then $\mathcal{C} \in \mathcal{A}$. A monotone access structure $\mathcal{A}$ is a monotone collection $\mathcal{A}$ of non-empty subsets of $\{P_1, P_2, ..P_n\}$, i.e., $\mathcal{A} \subseteq 2^{\{P_1,P_2,..P_n\}}$. The sets in $\mathcal{A}$ are called the authorized sets, the sets not in $\mathcal{A}$ are called the unauthorized sets.*

Ito et al. [185] showed that every monotone access structure or a monotone boolean formula has a Linear Secret Sharing Scheme (LSSS) that realizes the access structure. The model of LSSS is described as follows. We consider a system consisting of N parties $\{P_i, P_2, ..P_n\}$ and a dealer D. The latter has a secret input x to be shared among N other parties. LSSS allows D to generate N pieces of information from x so that a linear combination of a subset of N enables it to reconstruct x. The cardinality of the subset is parameterized as one of the public system parameters. The formal definition of LSSS is more specifically described in Definition 4.2.

Definition 4.2 (Linear Secret Sharing Scheme)**.** *Let K be a finite field, and $\prod$ be a secret sharing scheme with the domain of secrets $S \subseteq K$ realizing an access structure A. We say that $\prod$ is a linear secret sharing scheme over K if:*

- *The piece of each party is a vector over K. That is, for every i, there exists a constant d_i such that the piece of P_i is taken from K^{d_i}. We denote by $\prod_{i,j}(s, r)$ the $j-th$ coordinate in the piece of P_i (where $s \in S$ is a secret and $r \in R$ is the dealer's random input).*
- *For every authorized set, the reconstruction function of the secret from the pieces is linear. That is, for every $G \in A$, there exist constants $\{\alpha_{i,j} : P_i \in G, 1 \leq j \leq d_i\}$, such that for every secret $s \in S$ and every choice of random inputs $r \in R$:*

$$s = \sum_{P_i \in G} \sum_{1 \leq i \leq d_j} \alpha_{i,j} \prod_{i,j}(s, r). \tag{4.1}$$

The existence of a LSSS is equivalent to the existence of a span program [186], an efficient algebra computation model. The definition of span programs is formally presented in [187].

Definition 4.3 (Span Program over Monotone Boolean Function)**.** *Let $\{P_1, P_2..P_n\}$ be a set of parties. A collection of $\mathcal{A} \subseteq 2^{\{P_1,P_2,..P_n\}}$ is monotone if $\forall \mathcal{B}, \mathcal{C}$: if $\mathcal{B} \in \mathcal{A}$ and $\mathcal{B} \subseteq \mathcal{C}$ then $\mathcal{C} \in \mathcal{A}$. A monotone access structure $\mathcal{A}$ is monotone collection $\mathcal{A}$ of non-empty subsets of $\{P_1, P_2, ..P_n\}$, i.e., $\mathcal{A} \subseteq 2^{\{P_1,P_2,..P_n\}}$. The sets in $\mathcal{A}$ are called the authorized sets, the sets not in $\mathcal{A}$ are called the unauthorized sets.*

Most practical CP-ABE schemes [145,152,159] make use of LSSS and span programs in one way or another in the design.

4.3.2 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_{\mathrm{T}}$ denote two cyclic groups of prime order q, where g is a generator of $\mathbb{G}$. We say that e is a bilinear map such that $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathrm{T}}$ if e is efficiently computed over $\mathbb{G}$ and satisfies the two following properties:

- Bilinearity: for all $(u, v) \in \mathbb{G}^2$ and $(a, b) \in \mathbb{Z}_q^2$, we have: $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degeneracy: $e(g, g) \neq 1$.

We also present two standard security assumptions that we use to prove the security of the proposed CP-ABE schemes:

Definition 4.4 (Decisional Diffie-Hellman Assumption (DDH))**.** *Given a tuple $(g, g^a, g^b) \in \mathbb{G}$, the probability of distinguishing g^{ab} from a random element on $\mathbb{G}$ is negligible.*

Definition 4.5 (Decisional Bilinear Diffie-Hellman Assumption (DBDH))**.** *Given (g, g^a, g^b, g^c) for all $(a, b, c) \in (\mathbb{Z}_q^*)^3$, the probability that an adversary succeeds in distinguishing $e(g, g)^{abc}$ from a random element of $\mathbb{G}_{\mathrm{T}}$ is negligible.*

4.3.3 Lagrange Interpolation

The Lagrange interpolation is a simple yet important mathematical component employed by Shamir to design a t-out-of-n threshold secret sharing scheme [135]. By definition, the technique allows reconstructing a polynomial $\mathrm{P}(x)$ of degree t from a set of $(t + 1)$ points $S = \{x_i, \mathrm{P}(x_i)\}_{i=0..t}$ by the following formula:

$$\begin{aligned} \mathrm{P}(x) &= \sum_{i=0}^{t} \mathrm{P}(x_i)\Delta_{i,S(x)} \\ \textit{where:} \quad \Delta_{i,S(x)} &= \prod_{i \neq j; j \in [0,t]} \frac{x - x_j}{x_i - x_j} \end{aligned} \tag{4.2}$$

4.4 System and Threat Models

In this section, we detail the system model of our proposed CP-ABE schemes, called U-CPABE and A-CPABE, and the adversarial model we consider in the system.

4.4.1 System Model

The system model is outlined in Figure 4.3. this one involves the 4 following entities:

- **Key Distribution Center (KDC)**: is responsible for private keys generation. Upon receiving the attributes of a user, the KDC first verifies the validity of the attributes and then issues a corresponding private key and an initial unrevocation proof to the user. The valid revocation proof enables the user to prove its current status, which is *revoked* or *unrevoked*.

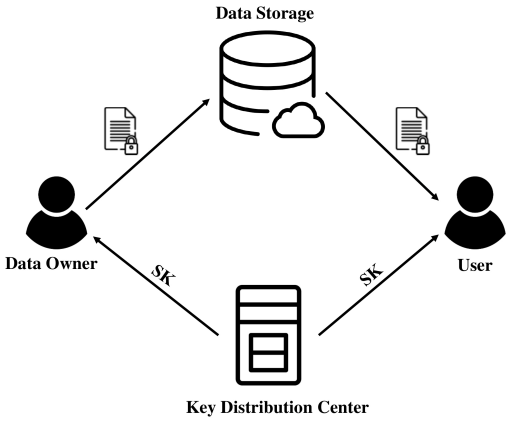


Figure 4.3: A practical deployment scenario of the proposed CP-ABE schemes

- **Cloud-Based Data Storage Provider**: Serving as an intermediate party, which is responsible for data storage, the service provider takes the responsibility to store and distribute encrypted data to authorized users. In principle, the provider can send encrypted data to any users that require it because, without valid private keys, unauthorized users have no way to decrypt the data. The approach removes the identity verification process carried out by the service provider, thus avoiding leaking user personal information to a third party. However, without restrictions on data retrievals, it makes the solution vulnerable to Distributed Denial of Service (DDoS), in which malicious actors send massively data retrieval requests to the provider, making it unable to handle the requests. In order to have a better trade-off between security and system performance, we adopt a strategy in which users need to provide a set of attributes that satisfies the access structure, which is used to encrypt the requested data, to the service provider for verification. We argue that a user may have many attribute sets which meet this requirement and sometimes a set of attributes is not sufficient to identify the real identity of the user. Besides, in case of attributes or user revocations, the provider is also in charge of re-encrypting previously shared data such that revoked users no longer have the ability to decrypt the data.

- **Data Owner**: In the system, a data owner is allowed to freely define access structures over data before encrypting and outsourcing it to the cloud. In the case of revocation, it may be in charge of generating a re-encryption key which will be then sent to the service provider for re-encrypting the data. Conceptually, re-encryption generation could be performed by either the data owner or authorized users with whom the data owner shares the data.

- **User**: The term refers to any entities in the system. A user is only able to decrypt data whose associated access structures are met by the user's attributes.

The two proposed CP-ABE schemes are constructed based on the practical scheme proposed by Bethencourt et al. [145], which is provably secure in the generic group model. The reason we choose this scheme for a detailed construction over others [152, 188] is due to its pioneering role and high performance compared to the others. However, one can still integrate our revocation and re-encryption mechanisms into similar schemes to achieve forward secrecy.

The entities in the system model are in charge of the following algorithms:

(PK, MSK, MRK, PRK, URP) ← **Setup**(1^l): Given a security parameter l, the KDC outputs a system public key PK, a public revocation key PRK, a master key MSK, a master revocation key MRK, and an initial unrevocation proof URP_{init}.

(SK, urk, URP_{init}) ← **KeyGen**(MSK, $\mathcal{S}$): User submits its attribute set S to the KDC which verifies and then securely delivers a corresponding private key SK, the initial unrevocation proof URP_{init} along with an unrevocation key urk to the user. In the case of revocation, only unrevoked users can generate a new URP based on their own urk and a public unrevocation proof update delivered from the KDC.

CT ← **Encrypt**($\mathcal{M}$, PK, $\mathcal{T}$, PRK): The data owner defines an access structure $\mathcal{T}$ to encrypt data $\mathcal{M}$ before outsourcing it to the cloud-based storage service.

PU ← **SKeyUpdate**(MRK, $\mathcal{L}$): Upon receiving a request to revoke a list of users $\mathcal{L}$, the KDC generates an unrevocation proof update PU and publicly broadcasts it to unrevoked users.

URP_{new} ← **UKeyUpdate**(PU, urk): Given an update PU, unrevoked users privately update their unrevocation proof.

k_{re} ← **RKeyGen**(URP_{new}, URP_{old}): Given an updated unrevocation proof URP_{new}, an unrevoked user computes a re-encryption key k_{re} and securely sends it to the cloud.

CT_{re} ← **Re-encryption**(CT, k_{re}): Upon a re-encryption request, the service provider translates the ciphertext CT to another version which all revoked users no longer have the ability to decrypt. Then, k_{re} is immediately removed for security purposes.

$\mathcal{M}$ ← **Decryption**(CT/CT_{re}, URP, SK): Given a private key SK and an unrevocation proof URP, a user decrypts a ciphertext CT/CT_{re} to retrieve the underlying plaintext.

4.4.2 Threat Model

In the following, we formalize the security requirements that the proposed schemes must fulfill to be secure in practice.

Data Confidentiality: In order to ensure data confidentiality, our aim is to prove the security of the proposed schemes against Chosen-Plaintext Attacks (CPA) and then refer to [189] for a generic way to convert these schemes into that semantically secure against Chosen-Ciphertext Attacks (CCA). The CPA security of the proposed schemes is modeled by a challenge Game in which an Adversary ($\mathcal{A}$) and a Challenger ($\mathcal{C}$) interact with one another as follows [145]:

Setup: Given a security parameter l, $\mathcal{C}$ generates system public parameters and makes them available to the system.

Phase 1: $\mathcal{A}$ is allowed to make private key extraction queries according to its selected attribute sets $(S_1, S_2..S_{q_1})$ to $\mathcal{C}$.

Challenge: Using the system public parameters, $\mathcal{A}$ outputs two different messages $\mathcal{M}_1$ and $\mathcal{M}_2$ and gives them to the challenger. Next, $\mathcal{C}$ flips a coin in order to decide on which message to encrypt and then returns an encryption of $\mathcal{M}_x$ where $x \in \{0, 1\}$ to $\mathcal{A}$.

Phase 2: Phase 1 is repeated so that $\mathcal{A}$ continues to adaptively make private key extraction queries for the sets of attributes according to its selected attribute sets $(S_{q_1+1}, S_{q_1+2}..S_q)$ to $\mathcal{C}$. We impose that none of the issued private keys successfully decrypts the challenge ciphertext.

Guess: Given encryption of $\mathcal{M}_x$, $\mathcal{A}$ has to guess the underlying plaintext $\mathcal{M}_{x'}$. If $\mathcal{M}_x = \mathcal{M}_{x'}$, $\mathcal{A}$ wins the game.

The proposed schemes are said to be semantically secure if $\mathcal{A}$ has a negligible advantage to win the game in this Challenge.

Secure Re-encryption: Given that re-encryption is performed by the service provider. In order to avoid unnecessary security incidents, the prescribed protocol must allow the service provider to remove the re-encryption key after the re-encryption operation. The aim of the requirement is not to give adversaries more power even in the case that the service provider is compromised. That is, the adversary has no means of recovering the re-encryption key in such a case.

Collusion Resistance: Suppose that the service provider does not follow the prescribed protocol and intentionally retains the re-encryption key after a re-encryption operation. In this case, we ensure that if the service provider maliciously collaborates with revoked users, they are only capable of decrypting data outsourced in advance but not after their revocations. We argue that the capability of decrypting data outsourced before their revocations in such case is trivial as the service provider could simply keep a copy of the original data that is decryptable by the private keys of the revoked users.

4.5 Our CP-ABE schemes

In this section, we describe our two proposed CP-ABE schemes with user and attribute revocation capability.

4.5.1 CP-ABE with user revocation capability

For the sake of clarity, we describe the special case, which allows revoking up to t users, where t is predetermined during the **Setup** phase of the scheme. Then, the general case is presented to support a revocation of an unbounded number of users.

4.5.1.1 U-CPABE with t users revocation capability: A special case

Let $\mathbb{G}$ and $\mathbb{G}_{\mathrm{T}}$ be two bilinear groups of prime order q. We denote by g a generator of $\mathbb{G}$. Let e be a pairing function such that e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathrm{T}}$ is efficiently computed over $\mathbb{G}$. Let H denote a collusion-resistant hash function such that H: $\{0,1\}^* \rightarrow \mathbb{G}$. The construction involves the following algorithms:

Setup(1^l): The algorithm is executed by the KDC. Given the security level l, the KDC randomly picks four exponents α, β, a, and z over $\mathbb{Z}_q^*$, and calculates a system public key:

$$\mathrm{PK} = (\mathbb{G}, g, h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha}) \tag{4.3}$$

Next, the KDC chooses a maximal number t of revoked users and a random polynomial of degree t:

$$P(x) = \sum_{i=0}^{t} p_i x^i \quad \text{where} \quad P(0) = p_0 = z \tag{4.4}$$

The KDC calculates a master revocation key MRK, a public revocation key PRK, and an initial unrevocation proof URP_{init}:

$$\mathrm{MRK} = z \quad \mathrm{PRK} = (h^z, h^a) \quad \text{and} \quad \mathrm{URP}_{init} = h^{az} \tag{4.5}$$

The KDC is equipped with an incremental counter $\mathcal{CT}$ that reflects the number of users in the system. The $\mathcal{CT}$ initially starts at zero and increases by 1 when a new user comes to the KDC for a private key request. We denote by $\mathcal{CT}_i$ the value assigned to the user i.

KeyGen(MSK, $\mathcal{S}$): The KDC takes an attribute set S of a user as input to issue a corresponding private key SK. To do so, the KDC chooses a random number r and a random number r_j for each attribute $j \in S$ over $\mathbb{Z}_q^*$. Then, it computes:

$$\begin{aligned} \mathrm{SK} =& (D = g^{(\alpha+r)/\beta}, \\ & \forall j \in S : D_j = g^r H(j)^{r_j}, D'_j = g^{r_j}) \end{aligned} \tag{4.6}$$

In addition, the user i is delivered with the initial unrevocation proof URP_{init} and an unrevocation key urk:

$$urk = P(\mathcal{CT}_i) \tag{4.7}$$

Encryption($\mathcal{M}$, PK, $\mathcal{T}$, PRK): The encrypting algorithm starts by converting an access structure $\mathcal{T}$ into a monotonic access tree. The algorithm continues to choose a random number s associated with the root node. For every node in the tree, the algorithm chooses a random polynomial $q(x) = \prod_{i=0}^{t} a_i x^i$ of the degree t which corresponds to the threshold of the node minus 1. The algorithm parses the tree in a top-down manner. For the root node, $q_r(0) = s$. For the children nodes, $q_x(0) = q_{parent(x)}(index(x))$. We note that the function $parent(x)$ returns the parent node of the node x, and $index(x)$ returns the index of x in the children list of its parent node. All the other coefficients of the polynomials are randomly selected over $\mathbb{Z}_q^*$. Let Y be the leaf node list, and $att(y)$ represent the attribute tied to the node $y \in Y$, the algorithm ends up generating a ciphertext CT:

$$\begin{aligned} \mathrm{CT} =& (\mathcal{T}, \tilde{C} = \mathcal{M} e(g,g)^{\alpha s} e(h^a, h^z)^s, C = h^s \\ & \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}) \end{aligned} \tag{4.8}$$

SKeyUpdate(MRK, $\mathcal{L}$): The algorithm is executed by the KDC which is requested to revoke a list $\mathcal{L}$ of k users. Given that these k users are associated with a list of k corresponding unrevocation keys $\mathcal{L}_{urk} = \{urk_i | i = 1..k\}$, the KDC computes an unrevocation proof update as follows:

If $(k < t)$, the KDC picks $(t - k)$ random numbers j, and then computes $urk_j = P(j)$ such that j is never generated by the $\mathcal{CT}$. Since then, the $\mathcal{CT}$ necessarily skips over all these values in its operation. Finally, the KDC chooses a random exponent $\tilde{a} \in \mathbb{Z}_q^*$ and completes the calculation of PU:

$$\mathrm{PU} = (h^{\tilde{a}} \quad \text{and} \quad \{i, h^{\tilde{a}.urk_i}\}_{i=1..t}) \tag{4.9}$$

PU is publicly broadcasted to all unrevoked users.

UKeyUpdate(PU, urk): The algorithm is executed by all unrevoked users. An unrevoked user takes its unrevocation key (urk_x) and the unrevocation proof update as input to interpolate a new unrevocation proof URP :

$$\mathrm{URP}_{new} = h^{\tilde{a}z} = h^{\Delta_{x,\mathrm{PU}(0)}.\tilde{a}.urk_x} \prod_{i=1}^{t} h^{\Delta_{i,\mathrm{PU}(0)}.\tilde{a}.urk_i} \tag{4.10}$$

Observe that revoked users do not have enough $(t + 1)$ points $(i, h^{\tilde{a}.urk_i})$ to interpolate the URP_{new}. Our mechanism is thus secure against collusion of up to t revoked users.

RKeyGen(URP_{new}, URP_{old}): The algorithm is executed by an unrevoked user. The user randomly selects $X \in \mathbb{G}_\mathrm{T}$, $r \in \mathbb{Z}_q^*$. It computes k_{re} and sends it to the cloud:

$$k_{re} = (h^{-az} H(X), h^r, X e(h^{\tilde{a}}, h^z)^r) \tag{4.11}$$

Re-encryption(CT, k_{re}): Upon receiving a re-encryption request, the service provider transforms the first element $\tilde{C}$ of the ciphertext into $\tilde{C}_{re}$ as follows:

$$\tilde{C}_{re} = \tilde{C}e(h^{-az}H(X), h^s) \tag{4.12}$$

The re-encrypted ciphertext contains the following elements:

$$\begin{aligned} \mathrm{CT}_{re} =& (\mathcal{T}, \tilde{C}_{re} = \mathcal{M}e(g,g)^{\alpha s}e(H(X), h^s), \\ & C = h^s, C_r = h^r, C_x = Xe(h^{\tilde{a}}, h^z)^r, \\ & \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}) \end{aligned} \tag{4.13}$$

Then, the service provider removes k_{re} to make sure that even if the provider is subsequently controlled by an adversary, it is unable to decrypt the re-encrypted data.

Decryption(CT/CT_{re}, URP, SK): The decryption combines two steps:

The first step: This step is applied to decrypt both an original ciphertext CT and a re-encrypted ciphertext CT_{re}. Given a private key SK which satisfies the access structure embedded into the ciphertext, the algorithm parses the access tree $\mathcal{T}$ in a bottom-up manner. For every leaf node x, let $i = att(x)$ represent the attribute of the node, the algorithm proceeds as follows:

$$\begin{aligned} \text{DecryptNode(CT, SK, } x) = F_x &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= e(g,g)^{rq_x(0)} \end{aligned} \tag{4.14}$$

For a non-leaf node x, let S_x be its set of child nodes, the algorithm continues as follows:

$$\text{DecryptNode(CT, SK, } x) = F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x(0)}} \tag{4.15}$$

where $i = index(z)$ and $S'_x = \{index(z) : z \in S_x\}$.

Likewise, at the root node of the tree, the algorithm achieves:

$$A = e(g,g)^{rs} \tag{4.16}$$

The second step: As the original ciphertext differs from the re-encrypted one, the user chooses appropriately one of the following formulas:

- Ciphertext without re-encryption CT:

$$\mathcal{M} = \frac{\tilde{C}A}{e(C,D)e(C,\mathrm{URP})} \tag{4.17}$$

- Ciphertext with re-encryption CT_{re}:

First, the user must achieve X by the following formula:

$$X = \frac{C_x}{e(C_r, \mathrm{URP}_{new})} = \frac{Xe(h^{\tilde{a}}, h^z)^r}{e(h^r, h^{\tilde{a}z})} \tag{4.18}$$

Then, the user completes the decryption process by computing:

$$\mathcal{M} = \frac{\tilde{C}_{re}A}{e(C,D)e(H(X),C)} \tag{4.19}$$

4.5.1.2 U-CPABE with unbounded users revocation capability: a general case

We now present the general case that allows revoking an unbounded number of users. Suppose that there are N users in the system, and t is a maximum of users tied to a polynomial, where $t \ll N$. We express N by $N = A.t + B$, where $0 \leq B < t$. We define $\lceil \frac{N}{t} \rceil = A + 1$ if $B > 0$, and $\lceil \frac{N}{t} \rceil = A$ if $B = 0$. The KDC selects $\lceil \frac{N}{t} \rceil$ numbers $z_j \in \mathbb{Z}_q$ for $\lceil \frac{N}{t} \rceil$ following random polynomials such that:

$$P_j(x) = \sum_{i=0}^{t} p_i x^i, \quad \forall j \in \{1, .., \lceil \frac{N}{t} \rceil\} \quad \text{where} \quad P_j(0) = p_0 = z_j \tag{4.20}$$

Then, the KDC computes the sets of corresponding master revocation keys MRK_j, public revocation keys PRK_j, and initial unrevocation proofs URP^j_{init}:

$$\text{MRK}_j = z_j \quad \text{PRK}_j = (h^{z_j}, h^{a_j}) \quad \text{URP}^j_{init} = h^{a_j z_j} \quad \forall j \in \{1, .., \lceil \frac{N}{t} \rceil\} \tag{4.21}$$

The incremental counter $\mathcal{CT}$ held by the KDC increases by 1 for every new incoming user U_i. The user i, assigned to the current value of $\mathcal{CT}_i$, is delivered with the private key SK as in (4.6), a corresponding initial unrevocation proof, and an unrevocation key as follows:

$$\text{URP}^j_{init}; \quad urk = P_j(\mathcal{CT}_i) \quad \text{where} \quad j = \lceil \frac{\mathcal{CT}_i}{t} \rceil \tag{4.22}$$

In the encryption phase, to encrypt a message $\mathcal{M}$, a data owner is required to perform $\lceil \frac{N}{t} \rceil - 1$ additional pairings:

$$\begin{aligned} \text{CT} = (\mathcal{T}, C = h^s \\ \tilde{C}_j = \mathcal{M} e(g,g)^{\alpha s} e(h^{a_j}, h^{z_j})^s, \quad \forall j \in \{1, .., \lceil \frac{N}{t} \rceil\} \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}) \end{aligned} \tag{4.23}$$

In the decryption phase, the user picks the corresponding value $\tilde{C}_j$ which it can decrypt using its unrevocation proof URP^j. Observe that this phase does not require any extra computation when compared to the special case. Regarding forward secrecy, we apply the same proxy re-encryption technique to prevent revoked users from decrypting the data shared in the past.

4.5.2 CP-ABE with attribute revocation capability

If some of the attributes of a user have changed over time while the remaining attributes keep valid, we need a mechanism to make the corresponding part of the private key, i.e. the one related to the revoked attributes, useless in decrypting data. We first propose a scheme, namely A-CPABE, which allows revoking attributes shared by at most t users. Furthermore, we apply the same strategy proposed above such that the scheme allows revoking attributes shared by an unbounded number of users. This scheme is highly required in cases where some common attributes of many users change at the same time.

In order to design A-CPABE, we make some changes to the first scheme supporting user revocation capability as follows:
Setup(1^l): Given the security level l, the KDC randomly picks two exponents α and β over $\mathbb{Z}_q^*$ and calculates a system public key as:

$$\text{PK} = (\mathbb{G}_0, g, h = g^\beta, f = g^{1/\beta}, e(g,g)^\alpha) \tag{4.24}$$

For each attribute j in the attribute set Y used in the system, the KDC chooses a maximal number t of revocations and generates a random polynomial of degree t:

$$P_j(x) = \sum_{i=0}^{t} p_i x^i \quad \text{where} \quad P_j(0) = p_0 = z_j \tag{4.25}$$

Then, the KDC computes a master revocation key MRK^j, a public revocation key PRK^j, and an initial unrevocation proof URP^j_{init}:

$$\begin{aligned} \forall j \in Y : \text{MRK}^j &= z_j \\ \text{PRK}^j &= (h^{z_j}, h^{a_j}); \text{URP}^j_{init} = h^{a_j z_j} \end{aligned} \tag{4.26}$$

Now we change the way the incremental counter $\mathcal{CT}$ of the KDC works. Upon a key request from a new user, $\mathcal{CT}$ increases by 1 for each of its attributes. As such, each attribute of each user corresponds to a different value of $\mathcal{CT}$. We denote by $\mathcal{CT}_{i,j}$ the value tied to the j^{th} attribute of the i^{th} user.

Let Enc_k and Dec_k denote respectively a symmetric encryption and decryption under a private key k. Let $H_1 : \{0,1\}^* \to \{0,1\}^l$ denote a collision-resistant hash function.
KeyGen(MSK, S): Given an attribute set S of a user i, the KDC issues a private key SK, a set of secret unrevocation keys urk, and a set of initial unrevocation proofs URP:

$$\begin{aligned} &\text{SK} = (D = g^{(\alpha+r)/\beta}, \\ &\forall j \in S : D_j = g^r.H(j)^{r_j}, D'_j = g^{r_j}) \\ &\forall j \in S : urk_{i,j} = P_j(\mathcal{CT}_{i,j}); \text{URP}^j_{init} \end{aligned} \tag{4.27}$$

Encryption($\mathcal{M}, \text{PK}, \mathcal{T}, \text{URP}$): Unlike the encryption procedure of the first user revocable scheme presented above, the algorithm binds C_y with $k_y = H_1(\text{URP}^j)$ such that $j = att(y)$:

$$\begin{aligned} \text{CT} &= (\mathcal{T}, \tilde{C} = \mathcal{M}e(g,g)^{\alpha s}, C = h^s, \\ \forall y \in Y : C_y &= g^{q_y(0)k_y}, C'_y = H(att(y))^{q_y(0)}) \end{aligned} \tag{4.28}$$

SKeyUpdate(MRK, $\mathcal{L}$): If a shared attribute y of k users, which are related to an unrevocation proof list $\mathcal{L}_{urk} = \{urk_{i,y} | i = 1..k\}$, needs to be revoked, the KDC generates an unrevocation proof update PU_k and publicly broadcasts it to unrevoked users. If $(k < t)$, the KDC picks $(t-k)$ random numbers j then computes $urk_{j,k} = P_k(j)$. Note that the randomly selected values j will be skipped over in the operation of $\mathcal{CT}$. The KDC selects $\tilde{a} \in \mathbb{Z}_q^*$ and computes:

$$\text{PU}_y = (h^{\tilde{a_y}} \quad \text{and} \quad \{i, h^{\tilde{a_y}.urk_{i,y}}\}_{i=1..t}) \tag{4.29}$$

UKeyUpdate(PU_y, $urk_{i,y}$): The unrevoked user i whose attribute y is still valid can update its unrevocation proof using PU_y:

$$\begin{aligned} \text{URP}^y_{new} &= h^{\Delta_{i,\text{P}_y(0)}.\tilde{a}_y.urk_{i,y}} \prod_{j=1}^{t} h^{\Delta_{j,\text{P}_y(0)}.\tilde{a}_y.urk_{j,y}} \\ &= h^{\tilde{a}_y z_y} \end{aligned} \tag{4.30}$$

For revoked users, in possession of PU_y, they do not have enough $(t+1)$ points $(i, h^{\tilde{a_y}.urk_{i,y}})$ to interpolate URP^y_{new}.
RKeyGen(URP^y_{new}, URP^y_{old}): The algorithm is executed by one of the users whose attribute y is still valid in the system. The user randomly chooses $p \in \{0,1\}^l$ and computes:

$$\begin{aligned} \tilde{k}_y = H_1(\mathrm{URP}^y_{new}), k_y = H_1(\mathrm{URP}^y_{old}) \\ k_{re} = (k_{re1}, k_{re2}) = (\mathrm{Enc}_{\tilde{k}_y}(p), p\frac{\tilde{k}_y}{k_y}) \end{aligned} \tag{4.31}$$

The re-encryption key is securely sent to the cloud for a re-encryption phase.
Re-encryption(CT, k_{re}): The algorithm is executed by the service provider which transforms C_y of the ciphertext CT into $\tilde{C}_y$:

$$\tilde{C}_y = C_y^{k_{re1}} = g^{q_y(0).p.\tilde{k}_y} \tag{4.32}$$

Let R denote a set of leaf nodes related to revoked attributes, the algorithm outputs the re-encrypted ciphertext:

$$\begin{aligned} &\mathrm{CT}_{\mathrm{re}} = (\mathcal{T}, \tilde{C} = \mathcal{M}e(g,g)^{\alpha s}, C = h^s, \\ &\forall y \in R : C_{p,y} = \mathrm{Enc}_{\tilde{k}_y}(p), \tilde{C}_y = g^{q_y(0).p.\tilde{k}_y}, \\ &C'_y = H(att(y))^{q_y(0)} \\ &\forall y \in Y \backslash R : C_y = g^{q_y(0).k_y}, C'_y = H(att(y))^{q_y(0)})) \end{aligned} \tag{4.33}$$

Then, the service provider removes the re-encryption key k_{re}.
Decryption($\mathrm{CT}/\mathrm{CT}_{re}$, URP^j, SK): The algorithm involves two steps:

The first step: There are two subcases according to whether the ciphertext is re-encrypted or not.

- Ciphertext without re-encryption, CT:

$$\forall y \in Y : g^{q_y(0)} = C_y^{k_y^{-1}} \tag{4.34}$$

- Ciphertext with re-encryption, CT_{re}: The user computes $p = \mathrm{Dec}_{\tilde{k}_y}(C_{p,y})$. Then, it computes:

$$\begin{aligned} \forall y \in R : g^{q_y(0)} = \tilde{C}_y^{(p.\tilde{k}_y)^{-1}} \\ \forall y \in Y \backslash R : g^{q_y(0)} = C_y^{k_y^{-1}} \end{aligned} \tag{4.35}$$

The second step: In both cases, the user obtains $g^{q_y(0)}, \forall y \in Y$. Applying the same procedure in the first step of U-CPABE, the user is returned with $A = e(g,g)^{rs}$, then computes the original message as follows:

$$\mathcal{M} = \frac{\tilde{C}A}{e(C,D)} = \frac{\mathcal{M}e(g,g)^{\alpha s}e(g,g)^{rs}}{e(h^s, g^{(\alpha+r)/\beta})} \tag{4.36}$$

4.6 Security Analysis

In the following, we sequentially analyze the security requirements which are mentioned in the section 4.4.2, of both U-CPABE and A-CPABE.

4.6.1 CP-ABE with user revocation capability

4.6.1.1 Data confidentiality

We prove the semantic security of the U-CPABE scheme against an indistinguishable chosen-plaintext attack. To facilitate the security proof, we first define the Basic Encryption scheme as follows:

setup: Given a security parameter l, the KDC generates a public revocation key PRK and an unrevocation proof URP:

$$\text{PRK} = (h, h^z, h^a) \quad \text{and} \quad \text{URP} = h^{az} \tag{4.37}$$

encrypt: One who wants to encrypt a message $\mathcal{M}$ can simply select a random number $r \in \mathbb{Z}_q^*$ and computes:

$$\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2) = (h^r, \mathcal{M}e(h^z, h^a)^r) \tag{4.38}$$

decrypt: Upon receiving a ciphertext $\mathcal{C}$, one makes use of the unrevocation proof to retrieve the underlying message $\mathcal{M}$:

$$\mathcal{M} = \frac{\mathcal{C}_2}{e(\text{URP}, \mathcal{C}_1)} = \frac{\mathcal{M}e(h^z, h^a)^r}{e(h^{az}, h^r)} \tag{4.39}$$

The Basic Encryption scheme should be regarded as a revocation mechanism integrated into the U-CPABE scheme. Indeed, U-CPABE is conceptually a combination of the Basic Encryption scheme and Bethencourt et al.'s scheme [145]. To be able to decrypt the ciphertext, in addition to the private keys of [145], authorized users also need to know the unrevocation proof URP $= h^{az}$. The latter is adaptively updated based on the revocations in the U-CPABE system. More intuitively, one can say that encrypting data with U-CPABE is like making it go through the double encryption system composed of [145] and the Basic Encryption scheme.

Lemma 4.1. *Suppose that a probabilistic polynomial time algorithm $\mathcal{A}$ has an advantage of $\epsilon_2(k)$ against the Basic Encryption scheme, there is an adversary $\mathcal{B}$ who can break the DBDH assumption with an advantage of $\epsilon_2(k)$.*

Proof: Given the system public parameters $\langle \mathbb{G}, h, e \rangle$ and a DBDH tuple $\langle h^a, h^b, h^c \rangle$, where a, b, c are randomly selected over $\mathbb{Z}_q^*$, the adversary $\mathcal{B}$ has to find $e(h, h)^{abc}$ with a non-negligible probability. In order for $\mathcal{B}$ to achieve the purpose, we construct interactions between $\mathcal{B}$ and $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ publishes PRK $= (h^a, h^b)$ as a public revocation key to $\mathcal{A}$.

Challenge: $\mathcal{A}$ chooses two random messages $\mathcal{M}_0$ and $\mathcal{M}_1$ and sends them to $\mathcal{B}$. The latter flips a coin to select the message $\mathcal{M}_b$ with $x \in \{0, 1\}$ that will be encrypted. Next, $\mathcal{B}$ returns a ciphertext $C = (h^c, \mathcal{M}_x e(h, h)^{abc}) = (C_1, C_2)$ to $\mathcal{A}$. As such, $\mathcal{A}$'s view is exactly its view in the real attack model. As soon as $\mathcal{A}$ successfully guesses the underlying plaintext $\mathcal{M}_{x'} = \mathcal{M}_x$, $\mathcal{B}$ can resolve DBDH as $e(h, h)^{abc} = C_2 / \mathcal{M}_{x'}$.

We say that the semantic security of U-CPABE is completely dependent on that of [145] and the Basic Encryption scheme. The former is provably secure in the generic group model [145] whereas the Basic Revocation scheme, as demonstrated above, is secure under DBDH assumption. Our first construction U-CPABE is thus semantically secure.

4.6.1.2 Secure Re-encryption

When a re-encryption operation is completed, the private re-encryption key is immediately removed. Hence, a revoked user, who wants to decrypt the re-encrypted data, has to obtain $e(H(X), h^s)$ or the new unrevocation proof $\text{URP}_{new} = h^{\tilde{a}z}$. The former is unachievable because the re-encryption key is removed by the service provider, thus remaining unknown to the revoked user. Meanwhile, the latter cannot be computationally derived from the public parameters $(h, h^{\tilde{a}}, h^z)$ due to the DDH assumption. The confidentiality of the re-encrypted data is thus protected against revoked users.

4.6.1.3 Collusion Resistance

We ensure that collusion between the service provider and revoked users does not allow learning the new unrevocation proof URP_{new} $(h^{\tilde{a}z})$ to decrypt newly outsourced data. Observe that the only additional information for the revoked users is the re-encryption key which is not removed by the dishonest provider:

$$k_{re} = (h^{-az}H(X), h^r, Xe(h^{\tilde{a}}, h^z)^r) \tag{4.40}$$

Given the expired unrevocation proof h^{az}, k_{re} and the public parameters $(h^{\tilde{a}}, h^z, h^r)$, both revoked users and dishonest service provider cannot extract the new unrevocation proof $h^{\tilde{a}z}$ due to the DBDH assumption. Therefore, our construction is secure against collusion.

4.6.2 CP-ABE with attribute revocation capability

4.6.2.1 Data confidentiality

Similarly to U-CPABE, we employ the same underlying idea to provide a CP-ABE scheme to support attribute-level revocations. Based on [145], we propose to bind each ciphertext component C_y to a corresponding unrevocation proof $URP = k_y$ so that only unrevoked users know this value. In addition, URP is also gradually updated based on the revocations in the system.

$$C_y = g^{q_y(0).k_y} \tag{4.41}$$

Anyone who wants to break the A-CPABE scheme needs to derive $g^{q_y(0)}$ from C_y, and then break the scheme of [145] to obtain the underlying plaintext. However, computing $g^{q_y(0)}$ from C_y is not computationally feasible under the Discrete Logarithm assumption. Moreover, Bethencourt's scheme is provably secure. From the two latter, we derive that A-CPABE is thus semantically secure.

4.6.2.2 Secure Re-encryption

After the service provider has finished a re-encryption operation, the private key component related to the revoked attribute of the user is useless in participating in decrypting the data shared in the past. Indeed, to do so, the user needs to know the re-encryption key to induce $g^{q_y(0)}$ from $g^{q_y(0).p.\tilde{k}_y}$. As the re-encryption key is supposed to be removed by the service provider, this task is computationally impossible under the Discrete Logarithm problem. Therefore, A-CPABE has a secure re-encryption mechanism.

Table 4.1: Relative communication and computational costs comparison

CP-ABE schemes	Type of revocation	Additional encryption cost		Additional decryption cost		Additional ciphertext length
		EXP	P	EXP	P	
BCP-ABE2 [146]	User	2R	0	R	2R	2R|$\mathbb{G}$|
DUR-CP-ABE [147]	User	2R|Y|+R-2|Y|	0	R|Y|-|Y|	2|Y|(R-1)	(2R|Y|-2R)|$\mathbb{G}$|
Liu's scheme [168]	User	R+K+2	0	R	3	2|$\mathbb{G}$|
U-CPABE	**User**	**1**	**1**	**0**	**1**	**0**
A-CPABE	**Attribute**	**0**	**0**	|**Y**|	**0**	**0**

EXP: exponentiation operation; P: pairing operation; R: number of users in the revocation list $\mathcal{L}$; |Y| denotes the number of leaf nodes; |$\mathbb{G}$| denotes the length of an element in $\mathbb{G}$; K is the number of elements representing for a time period.

4.6.2.3 Collusion resistance

The proposed scheme prevents collusion between the service provider and users sharing a revoked attribute from learning the newly updated unrevocation proof URP^{y}_{new} or $\tilde{k}_y = H_1(\text{URP}^{y}_{new})$, which is needed to decrypt any data outsourced after the revocation of their attribute. Indeed, the additional available information for these revoked users in this scenario is the re-encryption key k_{re}, which the provider has been retaining after the re-encryption phase:

$$k_{re} = (k_{re1}, k_{re2}) = (\text{Enc}_{\tilde{k}_y}(p), p\frac{\tilde{k}_y}{k_y}) \tag{4.42}$$

To obtain $\tilde{k}_y$ from k_{re2}, they need to derive p. However, this one is symmetrically encrypted under the hash of the new unrevocation proof and thus cannot be retrieved. Therefore, collusion is failed to achieve its goal.

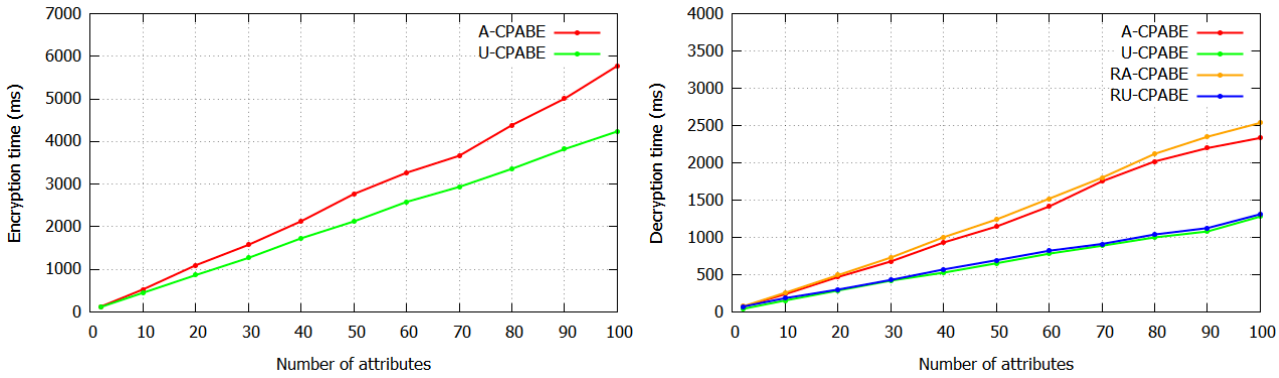
4.7 Performance Evaluation

To adopt CP-ABE schemes in practice, besides the necessary security guarantees, the computational and communication overheads of these schemes should be taken into account. Due to the complexity of access structures along with the costly cryptographic operations, the implementation of these schemes could be a critical issue. Indeed, the most expensive functions in the CP-ABE schemes are encryption and decryption. These functions are performed by users and data owners who can use any resource-constrained devices, i.e., smartphone or pc, on their own. Therefore, we concentrate on evaluating the performance of these functions. We first theoretically compare our proposals to the well-known schemes with regards to computational and communication overheads. To be clear, we only use the schemes which provide instant revocation without the aid of an additional trusted party. Also, we have conducted an implementation of the proposals on a computer to exhibit their performance.

4.7.1 Theoretical performance evaluation

The theoretical comparison between the proposed schemes and other related schemes is illustrated in Table 4.1. Note that the schemes used in the comparison are primarily developed on the inefficiently revocable ABE schemes [145,152]. In order to have a clear comparison, we only count the additional computational costs in the encryption and decryption algorithms caused by the implemented revocation mechanisms. These are approximately measured based on the number of expensive cryptographic operations such as exponentiation and pairing. Observe that our first proposal, U-CPABE, computationally outperforms the well-known schemes [146,147,168].

In addition, unlike the others, the costs of the encryption and decryption algorithms in our proposals are completely independent of the number of revoked users. On the other hand, our second scheme, A-CPABE, is the only one supporting the direct attribute revocation. When compared to the basic scheme [145], A-CPABE requires no additional exponentiation or pairing in the encryption phase, and only adds |Y| extra exponentiations in the decryption phase, where |Y| is the number of attributes in the access structures. In terms of communication overhead, unlike the other schemes, our proposals do not cause any extra length to the original ciphertext in [145].



(a) Encryption time of the proposed schemes (b) Decryption time of the proposed schemes

Figure 4.4: Time execution measurement of the proposed CP-ABE schemes.

4.7.2 Performance evaluation on computer

The widespread adoption of a security solution not only depends on the provided security level but also the underlying efficiency. Indeed, the re-encryption techniques introduced in the proposed schemes also attempt to achieve better efficiency compared to traditional solutions. To experimentally evaluate the performance of the proposals, we have implemented them on a computer with an Intel(R) Core(TM) i7-6600CPU 2.81GHz processor and 8Gb RAM. We made use of the JPBC library [190] to carry out the cryptographic operations in the proposals. With regards to pairing operation, we used the pairing of type A which is constructed on the curve $y^2 = x^3 + x$ over the finite field $\mathbb{F}_q$ for some $q = 3 \mod 4$ [191]. The security level of the curve is defined by the lengths of the prime order r and the field q. In this experiment, the length of q is 512 bits whereas the length of r is 160 bits.

Let RA-CPABE and RU-CPABE respectively denote the cases where attribute and user revocations occur, and the ciphertexts are subsequently re-encrypted. Figure 4.4 illustrates the execution time of the encryption and decryption algorithms in the proposals according to the number of attributes in access structures. As shown in Figure 4.4, we vary the number of attributes from 1 to 100. In fact, 100 attributes are very likely to be enough to describe most access structures in any practical application. Observe that even in the worst case (100 attributes), the encryption and decryption algorithms only take around $5.5s$ and $2.5s$, respectively. These low values demonstrate the feasibility of our proposals on the given computer. With regard to the re-encryption operation, which is delegated to the service provider, the execution cost is highly efficient in both proposals since it is completely independent of the number of attributes in access structures. Indeed, in U-CPABE, the re-encryption operation

only involves one group multiplication and one pairing operation. On the other hand, in A-CPABE, the service provider is only required to carry out one modular exponentiation. We argue that making re-encryption low cost is indispensable in this context because the task is assumed by a centralized party, the service provider, which has to serve many requests from many users in the whole system. Therefore, our solutions fulfill this efficiency requirement.

4.8 Conclusions

The enforcement of user and attribute revocations is a challenging problem of Ciphertext-Policy Attribute-Based Encryption. In this chapter, we have designed two novel CP-ABE schemes with the direct user and attribute revocation capability. The proposed schemes efficiently overcome the constraints of the existing solutions in the literature in terms of efficiency and security. Besides, we integrated the proxy re-encryption techniques into our schemes to securely relieve unrevoked users of the re-encryption workload. The security analysis and performance evaluation demonstrate the feasibility of these schemes in practice.

Chapter 5

Privacy-Preserving Blockchain-Based Data Sharing Platform for Decentralized Storage Systems

Contents

5.1 Foreword

Cloud-based storage services have been widely adopted for facilitating data storage and data sharing between remote users. Since data is stored on a platform governed by a centralized third party which is not fully trusted, there are generally four critical issues to be solved: (1) data confidentiality, (2) data access control, (3) user privacy, and (4) data availability. The advent of advanced encryption schemes, such as attribute-based encryption [145, 152], provides a user-centric model, which can be used to encrypt data before outsourcing to ensure data confidentiality. However, the other issues remain unsolved. First, outsourcing data to a cloud-based storage service partially deprives a data owner of the ownership. Users must rely on the service maintaining consistency and operations in accordance with a user-defined access control policy (2). However, due to legal reasons, for example, data censorship, the service can block access to data of an individual or a group of users. Even worse, the service can remove all data

related to some specific users. Second, regarding user privacy (3), a user has to provide its identity to the service for verification before being granted access to the requested data. This, however, leads to the user being traceable in the system. The lack of user privacy enables the service provider to trace all activities of any targeted user or discover the relationship between data stakeholders, i.e., who shares data with whom. This is not desired in many scenarios, for example, in healthcare systems, in which patients prefer concealing their identities while sharing their sensitive Electronic Medical Records (EMR) to healthcare institutions. Observe that simultaneously preserving data confidentiality (1) and user anonymity (3) is also an efficient way of resisting data censorship. Indeed, from a philosophical point of view, we state that if storage nodes have knowledge about neither the content of data nor the real identities of data stakeholders, they will have no incentives to deny authorized parties access or accept unauthorized ones access to the data. Hence, the only way for a storage service provider to remove data pertaining to a certain user is to destroy all data of the whole system that is, in most cases, not worth doing. The user anonymity is, however, not a trivial work in the context of data sharing where the data owner has to specify users with whom it wants to share data, and authorized users need to prove their right before accessing the data. Third, with the proliferation of applications, such as Internet of Things applications, where a huge daily volume of data is transferred, stored, and accessed by different actors, such a centralized approach seems insufficient to handle all data-related requests at peak times, thus decreasing data availability (4) or potentially leading to a single point of failure (SPOF).

With that being said, we need to rethink the way of efficiently storing, sharing, and processing data to give back data ownership in the hands of users as well as protect both user and data security. This implies a powerful data censorship resistance mechanism. The emergence of the blockchain technology, which has offered many disruptive solutions in various industries, makes it a prominent solution in this context. In this chapter, we propose a privacy-preserving blockchain-based data sharing platform for the InterPlanetary File System (IPFS). The deployment of IPFS [192], which inherits the advantages of many peer-to-peer systems, allows simultaneously retrieving data from multiple storage nodes, thus removing the risk of SPOF and improving data availability. The proposed platform is built on top of the public blockchain and includes three main novel components:

- We propose a revocable predicate encryption scheme for a data owner to ensure data confidentiality while sharing data with other users. While the private key of each user is separately computed and given by the data owner to decrypt the data, the data owner is able to revoke the private keys at will. The scheme incorporates the proxy re-encryption technique allowing delegating re-encryption tasks to the storage nodes in case that the data owner wants to revoke some users. This makes the revoked users unable to decrypt the re-encrypted data.

- We propose a novel mechanism of hiding auditable data access control lists that are stored on the blockchain and fully managed by the data owners. A user can anonymously prove its access right over the shared data, without revealing its identity, to request a data retrieval.

- We present the notation of hidden transaction that prevents adversaries from analyzing the sender and the recipient that are involved in this one.

5.2 Related Work

There have been several works targeting user anonymity in the field of peer-to-peer content distribution networks, such as Freenet [193] and Free Haven [194]. While Freenet allows data owners to encrypt data with their own names, Free Haven does not provide such an encryption mechanism to protect data confidentiality against storage hosts. Users in Freenet and Free Haven can query the networks to retrieve the desired data. The routing protocols in these systems, which are used to pass requests to data hosts and return the data to requesters, allow protecting the identities of the data owners and the data retriever at the network layer. However, these systems do not support any data access control mechanism, making the data owners unable to restrict access to specific users. Therefore, user revocation, which happens when the data owners want to update access control lists, cannot be featured in such systems.

In the context of cloud-based storage, Shen et al. [195] attempt to solve user anonymity by using the group signature technique. In group signature, the group manager computes a master key (MK) and a system public key (PK). Each user joining the group receives a private key generated based on MK by the group manager. To access the shared data, a member creates a signature δ on behalf of the group using its private key. The storage provider verifies the validity of δ by using PK before granting the user access to the requested data. Verifying δ only allows the storage provider to determine whether it has been created by a group member, without revealing the real identity of the signer. This approach, however, has two main drawbacks. First, group members need to directly interact with the group manager to receive their private keys before being able to prove their data retrieval right. Second, in this case, the data access policy is simplified by PK that is used to verify group signatures. Storing access control policy requires to fully rely on the storage service for enforcement and further updates.

The works in [196, 197] aim at blockchain-based access control solutions, but none of them rigorously takes user privacy into account. These simply solve data access auditability, meaning that the blockchain engages in recording access control lists and data access activities in chronological order so that the data owner can audit this information later. However, trivially storing access control lists on a blockchain, which is accessible to everyone, obviously violates the privacy of the data stakeholders. In [198, 199], the authors make use of Ciphertext-Policy Attribute-Based Encryption (CP-ABE), which allows directly enforcing access control lists into ciphertexts, to protect data confidentiality. The utilization of CP-ABE is promising but still falls short in preserving user privacy. Indeed, in the CP-ABE schemes, a private key is associated with a set of attributes while a ciphertext is associated with an access policy. If a user's attributes match the access policy, the user is able to decrypt the data by using its private key. However, the ciphertext must be stored along with the associated access policy which indicates to the data stakeholders the right way of decrypting the data. In storage systems, access policies, however, allow deducing sensitive information about the data stakeholders, thus putting user privacy at risk.

5.3 System and Threat Models

5.3.1 System Model

Our proposed privacy-preserving blockchain-based data sharing platform includes 4 entities illustrated in Figure 5.1.

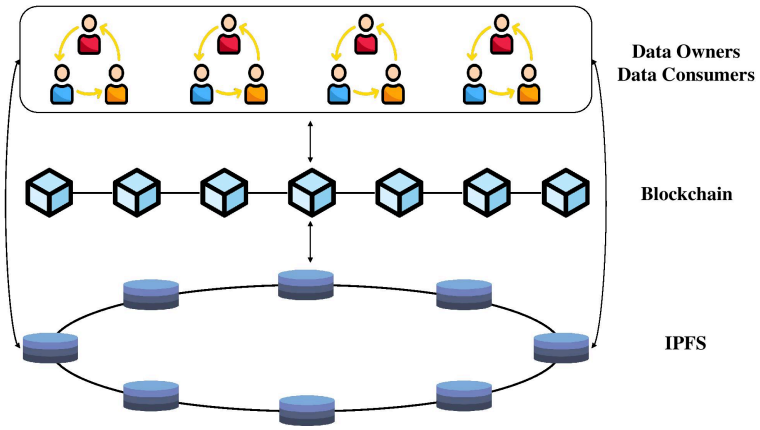


Figure 5.1: The system model of the proposed platform

- **Blockchain:** is a growing chain of blocks that are validated and chained together. Since being introduced in 2008 [100] as the underlying technology of Bitcoin, the blockchain paradigm has attracted great interest from both industry and academia. Later, Ethereum [101] appeared as the next blockchain generation, which allows for the deployment of smart contracts. A smart contract is a self-executing computer program stored on the blockchain and is run by the miners in a trustless way. Once being deployed on the blockchain, a smart contract cannot be modified. The blockchain and smart contracts provide the users with integrity, transparency, and autonomy guarantees. Our proposed platform leverages the use of the Ethereum blockchain to allow users to create smart contracts for data sharing management.

- **InterPlanetary File System (IPFS):** is a peer-to-peer protocol [192] using a content-based addressing technique to enable people worldwide to upload, download, and share data together in a fast and safe way. IPFS combines the advantages from the successful peer-to-peer systems such as Kademlia Distributed Hash Table [200], Bittorent [201], Git, and Self-Certified File system [202,203], thus making it a robust decentralized storage platform, providing quick data block retrieval. Recently, Filecoin [204] has been developed to expand IPFS to global decentralized storage. Specifically, Filecoin integrates an economic incentive mechanism into IPFS, allowing users to set a price and rent out unused disks for storing data of others.

- **Data Owner:** is any user that stores data on the IPFS. It has the ability to encrypt, anonymously delegate access rights, and revoke access over the data. Once some users are revoked, the data owner computes a re-encryption key and sends it to the storage nodes for re-encrypting the data. Thus, the revoked users are no longer able to decrypt the data.

- **Data Consumer**: is anyone authorized to use data shared by data owners.

5.3.2 Threat Model

We formalize the security and privacy requirements that the proposed data sharing platform is supposed to achieve in the following:

5.3.2.1 Data Confidentiality

Our model assumes that data is encrypted before outsourcing, making adversaries unable to decrypt it without the decryption key that is only known to authorized users. In case of revocation, the shared data is re-encrypted by the proposed proxy re-encryption technique, preventing the revoked users from decrypting the data. This is also known as forward-secrecy requirement. It represents the data confidentiality requirement that we aim at protecting.

5.3.2.2 User Privacy

Data owners and consumers anonymously share data in the system without anyone learning any personal information about them. This also disallows detecting whether two specific users have ever shared data. This is the targeted user privacy requirement in the design of the proposed platform.

5.3.2.3 User Unlinkability

User unlinkability implies the impossibility of linking activities made by the same user. For example, one cannot say whether a user has accessed shared data several times or how many data sharings that a data owner has made.

5.3.2.4 User-Data Unlinkability

The user-data unlinkability avoids linking any user with any data shared on the proposed platform.

5.4 The Architecture

This section highlights the main architectural building blocks constituting the platform as depicted in Figure 5.2.

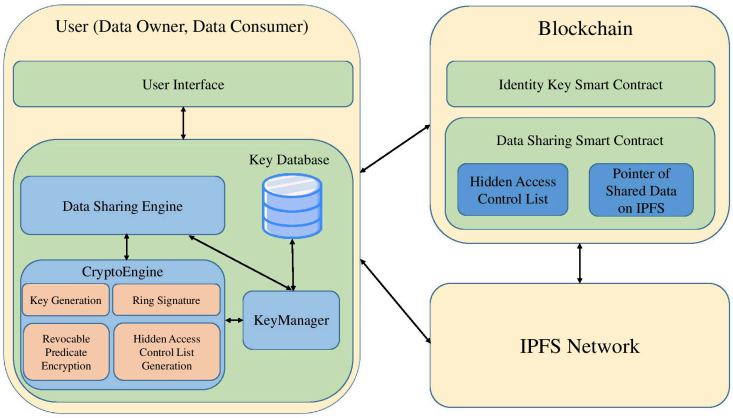


Figure 5.2: The architecture of the proposed data sharing platform

On the user side, a user interacts with the 4 following internal components to anonymously share data with others:

- **Data Sharing Engine**: This component allows a data owner to specify the data consumers with whom it wants to share the data. It also allows the data owner to construct as well as update the corresponding auditable hidden access control list $\mathcal{L}$, which is then stored on the blockchain. The platform is designed as an asynchronous model, meaning that the data consumers are not required to be active (online) when the data owner shares the data. This component helps the data consumers to detect new data sharing from others using $\mathcal{L}$.
- **CryptoEngine**: This component is responsible for all cryptographic operations in the platform. It computes hidden access control lists following the requests of the Data Sharing Engine and allows encrypting as well as decrypting data to protect data confidentiality. For data consumers, the component is used to issue proofs of eligibility to access shared data. Moreover, it also allows constructing hidden transactions among users to protect their identities.
- **KeyManager**: This component is responsible for storing and retrieving keys from the database in an authenticated way.
- **Key Database**: This component is a secure store of keys on the user side.

On the blockchain side, the platform has two main types of smart contracts:

- **Identity Key Smart Contract**: The platform has a common contract for users to register their identity public keys that are anonymously bound to their real identities. This reassures data owners about the existence of data consumers in the platform before sharing data with them. In practice, users exchange their public identity keys over a secure out-of-band channel before sharing data. In case that a user wants to change its own identity keys, it is required to sign the new public identity key with the old private identity key and send both the new public key and the resulting signature to the contract. The mining nodes verify the validity of the signature to update the user's new public key.
- **Data Sharing Smart Contract**: A data sharing smart contract is created by a data owner to manage the location of the shared data on the IPFS and the hidden access control list that anonymously indicates who has the right to access the data. The data owner can audit and make changes to the hidden access control list over time. Data retrieval requests by the data consumers are registered into the contract. These ensure the transparency and auditability of the activities over the shared data.

5.5 The CryptoEngine and KeyManager Components

For ease of understanding the functioning of the platform, we describe in detail the CryptoEngine and KeyManager. Indeed, these two components contain our main cryptographic contributions.

5.5.1 KeyManager

The KeyManager component aims at storing and retrieving keys from the Key Database. In the platform, we classify 4 types of key pairs, each of which has a public key (PK) and a private key (SK):

- **Identity Keys** (iPK, iSK): Each user has one iPK and one iSK which are bound to its identity in the system. Users exchange their public identity keys in a secure out-of-band channel to avoid violating their privacy.
- **Smart Contract Keys** (sPK, sSK): A user can have many smart contract keys which are used to deploy and manage data sharing contracts. However, one private smart contract key sSK is allowed to deploy only one smart contract.
- **Ephemeral Keys** (ePK, eSK): An ephemeral key is a one-time key used to sign all kinds of transactions, except transactions needing to be signed by sSK.
- **Hidden Keys** (hPK_{s-r}, hSK_{s-r}): The aim of hidden keys is to obscure the identities of participants in a transaction with respect to third parties.

5.5.2 CryptoEngine

The CryptoEngine plays a vital role in the platform due to its responsibility for all cryptographic computations. So, let us describe the functionalities that it supports:

5.5.2.1 Key Generation, Hidden Transaction, and Hidden Auditable Access Control List

a) Identity, Smart Contract, and Ephemeral Keys Generation:

These three types of keys are generated as depicted in Algorithm 5.1, which depicts the case of identity keys. The main difference among them lies in their usage purposes as outlined above.

Algorithm 5.1. Identity Key Generating Algorithm

Input: Given a cyclic group $\mathbb{G}$ of order p, generated by G.
Output: iPK, iSK

1: Chooses at random a number $x \in \mathbb{Z}_p^*$.
2: Sets $iSK = x$ and $iPK = xG$.
3: **return** iPK, iSK

b) Hidden Key and Hidden Transaction:

In any transaction, the platform aims to protect the identities of both the Sender $(\mathcal{S})$ and the Recipient $(\mathcal{R})$. Observe that it is straightforward to protect the identity of $\mathcal{S}$ by allowing it to sign the transaction with an ephemeral key. However, preserving the identity of $\mathcal{R}$ is non-trivial work. Intuitively, the transaction needs to be sent to a *random* address so that only $\mathcal{R}$ can determine that the transaction is intended for it. In our context, $\mathcal{R}$ also needs to know the identity of the sender, thus making the problem more challenging. For example, upon receipt of a shared data, a data consumer needs to know exactly the identity of the data owner to determine the corresponding PRE private key for decryption. Therefore, we make use of hidden keys and hidden transactions to effectively overcome the problem. We assume that a Sender $(\mathcal{S})$ wants to make a hidden transaction with a Receiver $(\mathcal{R})$. To this end, $\mathcal{S}$ first generates an ephemeral key pair ($eSK_s = e$ and $ePK_s = eG$). Next, $\mathcal{S}$ takes as input the public identity key iPK_r of $\mathcal{R}$ and eSK_s, then follows Algorithm 5.2 to obtain a public hidden

key hPK_{s-r}. Next, $\mathcal{S}$ creates a transaction, signs it with the private ephemeral key eSK_s, and sends it to the address of the resulting public hidden key hPK_{s-r}.

Algorithm 5.2. Public Hidden Key Generating Algorithm

Input: $eSK_s = e$ and $iPK_r = x_rG$
Let H be a collision-resistant hash function: $\{0,1\}^* \rightarrow \mathbb{Z}_p^*$
Output: hPK_{s-r}
1: Computes $z = H(e.x_r.G)$
2: Computes $hPK_{s-r} = zG + x_rG$
3: **return** hPK_{s-r}

$\mathcal{R}$ needs to scan the blockchain to determine whether a new hidden transaction inserted into the blockchain is intended for it. That is, $\mathcal{R}$ must be able to correctly compute the corresponding private hidden key hSK_{s-r}. Given the public ephemeral key ePK_s of the hidden transaction, $\mathcal{R}$ uses its private identity key iSK_r to compute $hPK'_{s-r} = H(ePK_s.iSK_r)G + iPK_r = H(e.x_r.G)G + x_rG$. If $(hPK'_{s-r} = hPK_{s-r})$, the hidden transaction is intended for $\mathcal{R}$, who can then compute the corresponding private hidden key, $hSK_{s-r} = H(e.x_r.G) + x_r$. In contrast, third parties cannot recompute hSK_{s-r} without iSK_r, thus they cannot identify the receiver of the transaction.

In order for $\mathcal{R}$ to know the sender of the hidden transaction, $\mathcal{S}$ needs to encrypt its public identity key iPK_s with the public hidden key and embed it into the transaction. Hence, $\mathcal{R}$ computes the private hidden key to decrypt iPK_s.

c) Auditable Hidden Access Control List:

An auditable hidden access control list $\mathcal{L}$ is composed of a set of hidden public keys of all the data consumers. The data owner adds or revokes data consumers by adding or removing their public hidden keys to or from $\mathcal{L}$, respectively. A data consumer wanting to access the shared data has to issue a proof stating the existence of its hidden key in the list so that the proof reveals nothing about the consumer's identity. We achieve this by using a ring signature scheme which is presented in the following.

5.5.2.2 Ring Signature

Ring signature is a type of digital signature that allows a user to sign a message on behalf of a group of users, which is arbitrarily formed by the signer at the signing time [103]. A verifier can only verify whether the signature comes from the group, without knowing exactly the identity of the signer. Therefore, ring signature provides users with an anonymity guarantee whose extent completely depends on the ring size. More specifically, a signer takes its private key SK_s and all public keys $S = (PK_1, ..PK_s, ..PK_N)$ of all members in a group of size N to sign a message. A verifier checks the validity of the signature to determine whether the message was anonymously signed by a member of S. In this contribution, we purposely use the ring signature scheme of Herranz et al. [104]. This scheme has been proven computationally efficient and secure in the random oracle model. Second, this scheme is based on Discrete Logarithm (DL) assumption, making it compatible with the current cryptosystem in the Ethereum blockchain and removing the need of deploying another cryptosystem on the user side.

In the platform, the ring signature scheme is used by data consumers to issue proofs of eligibility to anonymously access the shared data. The intuition behind the proof is that given an auditable hidden access control list consisting of hidden public keys $(R = \{hPK_i\})$, an

authorized data consumer issues a ring signature on a random nonce on behalf of R. While the resulting signature allows verifying the eligibility of the consumer, it allows neither learning the consumer's identity nor linking its sessions. The latter is achieved due to the randomness of the ring signature, meaning that the signatures of the same consumer are uniformly random and unlinkable.

5.5.2.3 Predicate Encryption

Ciphertext-Policy Attribute Based Encryption (CP-ABE) is an advanced cryptographic primitive deployed in many data sharing platforms [198, 199], and our work described in Chapter 4. Despite its natural suitability for data sharing systems, CP-ABE incurs privacy issues which could be a challenge hindering the wide adoption of this encryption approach. Specifically, in CP-ABE, user private key is associated with its attributes set (S), i.e., age, name, while data is encrypted with a monotone access structure (P) realizing a boolean formula that is composed of attributes and logical operators. Any user with an attribute set matching with P is able to decrypt the ciphertext. This mechanism thus allows for fine-grained access control over encrypted data. However, such an encryption scheme requires storing P along with the ciphertext to help the user to decrypt it correctly. However, storing P that contains user attributes on a dishonest storage node obviously violates the privacy of data stakeholders. To overcome the privacy issue of CP-ABE while still benefiting from its advantages, we build a CP-ABE scheme supporting hidden access policies from revocable predicate encryption.

a) Introduction of Predicate Encryption:

Predicate Encryption (PE) is generalizing many advanced cryptographic primitives, such as Identity-Based Encryption (IBE) and CP-ABE. We refer to [205] for its formal definition. Generally, in a PE scheme, a user private key SK_f is associated with a predicate f, whereas a ciphertext is associated with a set of attributes S. A user can decrypt the ciphertext by using its private key if S matches the predicate f, i.e, $f(S) = 1$. In [206], Katz et al. introduce a PE scheme supporting conjunctions, disjunctions, and inner product, which enables the constructions supporting the evaluations of polynomials, CNF/DNF formulas.

The formal definition of predicate encryption is described as follows [206]:

Definition 5.1. *Predicate encryption for the class of predicate $\mathcal{F}$ over the set of attributes $\sum$ consists of 4 following Probabilistic Polynomial Time (PPT) algorithms:*

- *Setup(1^n): The algorithm takes as input the security parameter 1^n and outputs a system public key PK and a master secret key MSK.*
- *GenKey(MSK): The algorithm takes as input the master secret key SK and a predicate $f \in \mathcal{F}$ to output a corresponding key SK_f.*
- *Enc(M, I): The algorithm takes as input the public key PK, an attribute $I \in \sum$, and a message M in some associated message space. It returns a ciphertext $C \leftarrow Enc_{PK}(I, M)$.*
- *Dec(C, SK_f): The algorithm takes as input a secret key SK_f and a ciphertext C. It outputs either a message M or $\perp$ depending on whether the predicate accepts, i.e., f(I) = 1.*

In addition to the payload-hiding, which means data confidentiality protection, the authors also introduce a new security notion, called attribute-hiding. In fact, this notion was informally

identified in the context of IBE, which is a subcase of PE. That is, upon receiving a ciphertext C, the adversary cannot identify the identity for which the message has been encrypted. The first practical IBE construction proposed by Boneh et al. achieves this notion, which is not however observed in the original paper. Later, this security notion is described in [207, 208] which explore the relation between Searchable Encryption and Anonymous IBE (or Attribute-Hiding IBE). They propose a generic way to transfer an Anonymous IBE into a Searchable Encryption. Katz et al. further generalize this security notion in PE by the following game between an adversary and a simulator in the selective model.

- *Init*: The adversary submits two attributes $I_0, I_1 \in \sum$, which it wants to attack.
- *Setup*(1^n): The simulator runs the setup algorithm to generate the master key MSK and the system public key PK. The latter is made available to the public.
- *Phase 1*: The adversary adaptively sends key queries according to the predicate $f_1, f_2, .., f_n$ under the restriction that $\forall i f_i(I_0) = f_i(I_1)$. The simulator computes the corresponding private keys $SK_{f_i} = \text{KeyGen}(MSK, f_i)$.
- *Challenge*: A sends two different messages of equal length M_0, M_1 to the simulator. The simulator verifies whether there exists an i such that $f_i(I_0) = f_i(I_1)$ then $M_0 = M_1$. Otherwise, the simulator chooses a random bit b and sends an encryption of M_b back to the adversary $C_b = \text{Enc}_P K(I_b, M_b)$.
- *Phase 2*: The adversary repeats the phase 1.
- *Guess*: The adversary guesses the message $M_{b'}$ was encrypted. If $M_b = M_{b'}$, the adversary wins the game.

  The high level intuition behind the security game is that the polynomial bounded adversary is unable to guess which message/attribute has been used to construct C_b.

  The above security game can be slightly modified to achieve payload-hiding notion. More specifically, in the *Init* phase, the adversary only submits one attribute, i.e., $I_0 = I_1$. And in the key query *Phase* 1 and 2, the adversary is not allowed to ask for private keys corresponding to a predicate f so that $f(I) = 1$.

Clearly, developing a CP-ABE scheme from the PE scheme [206] helps solve the privacy issue. However, such an extended scheme lacks an efficient revocation mechanism to revoke users' private keys while necessary. This feature is indispensable in data sharing. In this contribution, we develop a new PE scheme, called Revocable Predicate Encryption (RPE), as an extension to [206]. The integrated revocation mechanism allows revoking private keys of users. We achieve it by adapting an efficient proxy re-encryption technique into the scheme. Moreover, RPE also fulfills the forward secrecy requirement. That is, when a data owner revokes some users, it computes a re-encryption key and securely sends it to the storage nodes for re-encrypting the shared data. Thus, the revoked users are no longer able to decrypt the re-encrypted data even if the storage nodes are physically corrupted afterward. The computation of a re-encryption key is computationally lightweight. Therefore, this approach relieves the data owner from the workload of downloading and re-encrypting the shared data. We now detail our RPE scheme and then introduce a way of building a CP-ABE scheme supporting hidden access policies from RPE.

b) Revocable Predicate Encryption (RPE):

Definition 5.2. *The Lagrange interpolation allows rebuilding a polynomial P(x) of degree t from a set of* $(t+1)$ *points* $S = \{x_i, P(x_i)\}_{i=0..t}$ *through the following formula:*

$$P(x) = \sum_{i=0}^{t} P(x_i)\Delta_{i,S(x)}; \quad \Delta_{i,S(x)} = \prod_{i \neq j; j \in [0,t]} \frac{x - x_j}{x_i - x_j}$$

The RPE scheme consists of the following functions:

Setup(1^k): Given a security parameter k, the system generates a cyclic group $\mathbb{G}$ of composite order $N = p.q.r$ where p, q, r are large prime numbers. Let $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r$ be groups generated by g_p, g_q, g_r, respectively. Let $\hat{e}$ be an efficient pairing function over $\mathbb{G}$ such that $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathrm{T}}$. Let H be a collison-resistant hash function such that $H : \{0,1\}^* \to \mathbb{G}$.

To generate the master key (MSK) and pubic key (PK), the system randomly chooses $R_{1,i}, R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}, h_{2,i} \in \mathbb{G}_p$. Next, it chooses $\gamma \in \mathbb{Z}_p^*$, $h \in \mathbb{G}_p$ and computes:

$$\begin{aligned} MSK =& (p, q, r, g_q, h^{-\gamma}, \{h_{1,i}, h_{2,i}\}_{i=1}^n) \\ PK =& (g_p, g_r, Q = g_q R_0, P = \hat{e}(g_p, h)^{\gamma}, \\ & \{H_{1,i} = h_{1,i} R_{1,i}, H_{2,i} = h_{2,i} R_{2,i}\}_{i=1}^n) \end{aligned} \tag{5.1}$$

To support revocation, the system first chooses a random number $z \in \mathbb{Z}_p^*$, then forms a random polynomial P of degree t, which allows simultaneously revoking t users:

$$P(x) = \sum_{i=0}^{t} p_i x^i \text{ where: } P(0) = z. \tag{5.2}$$

The system chooses a random number $a \in \mathbb{Z}_p^*$ then computes a master revocation key MRK, an initial unrevocation proof URP, a public revocation key PRK:

$$\text{MRK} = z;\ \text{PRK} = (g_p^z, g_p^a);\ \text{URP} = g_p^{az} \tag{5.3}$$

Encrypt(x, M, PRK): To encrypt a message m with a vector $x = (x_1, x_2, .., x_n)$, where $x_i \in \mathbb{Z}_N^*$, the algorithm randomly selects $s, \alpha, \beta \in \mathbb{Z}_N^*$ and $R_{3,i}, R_{4,i} \in \mathbb{G}_r$, computes:

$$\begin{aligned} C =& (C' = m P^s \hat{e}(g_p^z, g_p^a)^s, C_1 = g_p^s, \\ & \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i}\}_{i=1}^n) \end{aligned} \tag{5.4}$$

KeyGen(v, MSK): To generate a private key for a user associated with a vector $v = (v_1, v_2, .., v_n)$, the algorithm chooses random numbers $r_{1,i}, r_{2,i} \in \mathbb{Z}_p^*$, where $i = 1..n$. Then, it continues picking randomly $f_1, f_2 \in \mathbb{Z}_q^*$, $R_5 \in \mathbb{G}_r$ and $Q_6 \in \mathbb{G}_q$. It generates the private key for the user:

$$\begin{aligned} SK_v =& (K = R_5 Q_6 h^{-\gamma} \prod_{i=1}^{n} h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}}, \\ & \{K_{1,i} = g_p^{r_{1,i}} g_q^{f_1 v_i}, K_{2,i} = g_p^{r_{2,i}} g_q^{f_2 v_i}\}_{i=1}^n) \end{aligned} \tag{5.5}$$

The algorithm associates each user with a unique number num. It also sends to the user the initial unrevocation proof URP_{init} and an unrevocation key urk. The unrevocation proof URP is a complementary element used to decrypt data, and will be securely updated by the user using urk as soon as a new revocation happens. The urk of the user is computed:

$$urk = P(num) \tag{5.6}$$

SKeyUpdate(MRK, L): Suppose that the system wants to revoke n users associated to a list of n unrevocation keys $\mathrm{L} = \{urk_i\}_{i=1..n}$. If $(n < t)$, the system picks $(t-n)$ random numbers j, such these numbers were and will be never used again, and computes $urk_j = P(j)$. Next, the system chooses $\tilde{a} \in \mathbb{Z}_p^*$ and computes an unrevocation update PU:

$$\mathrm{PU} = (h^{\tilde{a}}, \{i, h^{\tilde{a} urk_i}\}_{i=1..t}) \tag{5.7}$$

UKeyUpdate(PU, urk_x): Given PU, an unrevoked user x uses its unrevocation key urk_x to update the unrevocation proof according to the Lagrange Interpolation. This function does not work for revoked users.

$$\mathrm{URP}_{\mathrm{new}} = g_p^{\tilde{a}z} = g_p^{\Delta_{x,\mathrm{PU}(0)} \cdot \tilde{a} . urk_x} \prod_{i=1}^{t} g_p^{\Delta_{i,\mathrm{PU}(0)} \cdot \tilde{a} . urk_i} \tag{5.8}$$

RKeyGen($\mathrm{URP}_{\mathrm{old}} = g_p^{az}$, $\mathrm{URP}_{\mathrm{new}} = g_p^{\tilde{a}z}$): To compute a re-encryption key, an unrevoked user takes $\mathrm{URP}_{\mathrm{old}}$ and $\mathrm{URP}_{\mathrm{new}}$ as inputs, then selects random numbers $X \in \mathbb{G}_{\mathrm{T}}, r \in \mathbb{Z}_N^*$ and computes a re-encryption key:

$$k_{re} = (g_p^{-az} H(X), g_p^r, X \hat{e}(g_p^{\tilde{a}}, g_p^z)^r) \tag{5.9}$$

Re-encryption(C, k_{re}): Using a re-encryption key, the storage node re-encrypts the ciphertext by converting the element C' in the original ciphetext C into C'_{re}:

$$C'_{re} = C' \hat{e}(g_p^{-az} H(X), C_1) = m P^s \hat{e}(H(X), g_p^s) \tag{5.10}$$

The re-encrypted ciphertext would be:

$$\begin{aligned} C_{re} = (&C'_{re} = m P^s \hat{e}(H(X), g_p^s), \\ &C_1 = g_p^s, C_r = g_p^r, C_x = X \hat{e}(g_p^{\tilde{a}}, g_p^z)^r, \\ &\{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i}\}_{i=1}^n) \end{aligned} \tag{5.11}$$

Decryption($C/C_{re}, SK_v$, URP): Depending on the type of ciphertext, a user decrypts it by using its private key.

- To decrypt an original ciphertext C, the user follows the below formula:

$$\begin{aligned} \tilde{m} &= \frac{C' \hat{e}(C_1, K) \prod_{i=1}^n \hat{e}(C_{1,i}, K_{1,i}) \hat{e}(C_{2,i}, K_{2,i})}{\hat{e}(\mathrm{URP}, C_1)} \\ &= m \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) < x, v >} \end{aligned} \tag{5.12}$$

- To decrypt a re-encrypted ciphertext C_{re}, an unrevoked user uses the updated unrevocation proof $\mathrm{URP}_{\mathrm{new}}$ to compute:

$$X = \frac{C_x}{\hat{e}(\mathrm{URP}_{\mathrm{new}}, C_r)} = \frac{X \hat{e}(g_p^{\tilde{a}}, g_p^z)^r}{\hat{e}(g_p^{\tilde{a}z}, g_p^r)} \tag{5.13}$$

Then, the user derives the message from the formula:

$$\begin{aligned} \tilde{m} &= \frac{C'_{re} \hat{e}(C_1, K) \prod_{i=1}^n \hat{e}(C_{1,i}, K_{1,i}) \hat{e}(C_{2,i}, K_{2,i})}{\hat{e}(H(X), C_1)} \\ &= m \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) < x, v >} \end{aligned} \tag{5.14}$$

We observe that if $< x, v >= 0$, the user obtains $\tilde{m} = m$, the exact plaintext.

c) Constructing CP-ABE scheme supporting hidden policy:

In a CP-ABE scheme, the ciphertext is associated with an access policy while the private key of a user is associated with its identifying attributes. Therefore, to construct a CP-ABE scheme supporting hidden policies from RPE, we only need a preprocessing stage to efficiently convert an access policy into an encryption vector x, and convert a set of attributes into a key vector v. If the set of attributes matches the access policy, implying that $< x, v >= 0$, the user with the private key related to v can decrypt the data encrypted with x. To this end, we adopt the conversion strategy proposed in [209] for converting arbitrary boolean formulas and sets of attributes into 2^d elements vectors over $\mathbb{Z}_N$ in which d is the number of attribute categories such as companies, departments, job positions, and seniority. Each attribute category can be assigned to different values. An example of such boolean formula is:
$\mathcal{A}$ = (Department = *Research*) OR (Job Position = *IT Engineer* AND Seniority = *Senior*)

We now briefly describe the way of converting $\mathcal{A}$ composed of 3 attribute categories into a 2^3-element vector. Suppose that H is a collision-resistant hash function which maps an arbitrary input to $\mathbb{Z}_N$: $H : \{0,1\}^* \rightarrow \mathbb{Z}_N$. The multivariate polynomial is constructed as follows:

$$\begin{aligned} p(x_1, x_2, x_3) &= r(x_1 - a_1) + (x_2 - a_2)(x_3 - a_3). \\ &= 0x_1x_2x_3 + 0x_1x_2 + 0x_1x_3 + x_2x_3 + rx_1 - a_3x_2 - a_2x_3 - ra_1 + a_2a_3. \end{aligned} \tag{5.15}$$

where $r \in \mathbb{Z}_N$, $a_1 = H(\text{Department}||\textit{Research})$, $a_2 = H(\text{Job Position}||\ \textit{IT Engineer})$ and $a_3 = H(\text{Seniority}||\textit{Senior})$.

The reason that we multiply $(x_1 - a_1)$ with a randomly chosen number r is to avoid the fact that $p(x_1, x_2, x_3) = 0$ while neither $(x_1 - a_1)$ nor $(x_2 - a_2)(x_3 - a_3)$ is equal to 0. By uniformly choosing r over $\mathbb{Z}_N$, such an event only happens with a negligible probability $\mathcal{O}(\frac{1}{N})$.

$$v = [0, 0, 0, 1, r, -a_3, -a_2, a_2a_3 - ra_1]. \tag{5.16}$$

We suppose that a user has a set S of attributes, including *Research*, *IT Engineer*, and *Senior*, we convert S into an 8-element vector as follows:

$$x = (a_1a_2a_3, a_1a_2, a_1a_3, a_2a_3, a_1, a_2, a_3, 1). \tag{5.17}$$

From now, $< x, v >= 0$ which also means that the attribute set S satisfies the boolean formula, thus enabling the user to decrypt the message encrypted with $\mathcal{A}$.

5.6 Privacy-Preserving Blockchain-based Data Sharing Platform

In this section, we detail the operations during data sharing in the platform.

5.6.1 User Registration

To use the platform, a user first needs to register with the public identity key smart contract deployed on the blockchain. To this end, it creates a wallet containing a public and private identity key (iPK, iSK) according to Algorithm 5.1. Then, the user sends iPK to the contract for registration. We emphasize that the identity keys represent the user identity and as such they should be carefully used not to violate user privacy.

5.6.2 Sharing Key Distribution

A data owner uses RPE to encrypt the data shared with data consumers. Therefore, it needs to share decryption keys with the data consumers for decryption. To this end, the data owner specifies a set of attributes representing each of them. Then, for each attribute set, it runs a **KeyGen** operation, as described in our RPE scheme, to obtain a RPE private key for the consumer. Simply sending a transaction which includes the RPE private key to the public identity key iPK_c of the consumer will obviously violate its privacy. Moreover, there might be many other data owners sending RPE private keys to the same consumer, making it an appealing target to attackers. In order to deal with this concern, the data owner creates a hidden transaction as described in Algorithm 5.3 and sends it to the blockchain. By scanning the blockchain, the intended consumer knows the arrival of a hidden transaction. It then computes the private hidden key hSK_{o-c} which is used to decrypt σ_1 and σ_2 to obtain the RPE private key and the public identity key iPK_o of the data owner. Then, the consumer inserts a pair of iPK_o and the RPE private key into its local storage for future usage. By doing so, even if third parties intensively analyze the hidden transaction, they cannot determine the users involved in the transaction.

Algorithm 5.3. Constructing a key distribution transaction

Input: iPK_c and a RPE private key.
Output: A hidden transaction.

1: Computes a public hidden key hPK_{o-c} with the consumer.
2: Creates an encryption σ_1 of iPK_o under hPK_{o-c}.
3: Creates an encryption σ_2 of the RPE private key under hPK_{o-c}.
4: Creates a transaction including (σ_1, σ_2) and signs it with a newly generated private ephemeral key eSK.
5: **return** The resulting hidden transaction.

5.6.3 Data Uploading

Before uploading data to the storage network, a data owner first generates a smart contract key pair (sPK and sSK). It then creates and deploys a new smart contract by using sSK on the blockchain. In this contract, sPK is declared as the public key of the contract owner who has all rights over it. In principle, one smart contract only manages one upload. An upload is referred to as a set of data that is simultaneously encrypted under the same access policy using RPE. Then, the data owner can start uploading the encrypted data onto the IPFS and, in turn, receives the address where the data is stored. The address is a hash of the encrypted data and can also be used to verify the integrity of the data. Finally, the data owner sends a transaction signed by sSK to the smart contract to declare the address for future data retrieval.

5.6.4 Data Sharing

To share data with a set of data consumers, a data owner specifies a hidden access control list $\mathcal{L}_{hPK}$ corresponding to these consumers as illustrated in Figure 5.3 . Each element in the list contains two objects. The first is a public hidden key hPK_i corresponding to a consumer i, computed from Algorithm 5.2. The second is an encryption of the public identity key iPK_o of the data owner with hPK_i. This object is important for the data consumers to determine

which RPE private key should be used to decrypt the data if they are in collaboration with many data owners. Then, the data owner creates a transaction to declare $\mathcal{L}_{hPK}$ in the contract. The data consumers scan the blockchain and analyze the public hidden keys to be aware of any new data shared with them and who the owner is.

hPK_1	$Enc_{hPK_1}(PK_o)$
hPK_2	$Enc_{hPK_2}(PK_o)$
hPK_3	$Enc_{hPK_3}(PK_o)$
...	...

Figure 5.3: The format of hidden access control list $\mathcal{L}_{hPK}$

5.6.5 Data Retrieval

If a data consumer wants to access data associated with a smart contract, it needs to fulfill the two following requirements:

- Proving that it is part of the associated hidden access control list $\mathcal{L}_{hPK}$ of the smart contract, without revealing its identity.
- Possessing a valid RPE private key to decrypt the data.

For the first requirement, an authorized consumer uses its private hidden key hSK_i, which the corresponding hidden public key is included in $\mathcal{L}_{hPK}$, to create a ring signature with all the public hidden keys in $\mathcal{L}_{hPK}$, then sends the signature to the storage nodes. In parallel, a transaction containing a data retrieval request and the ring signature is also made and sent to the blockchain miners that verify and insert the transaction into the blockchain. The storage nodes verify the validity of the ring signature, and send the encrypted data to the data consumer. Finally, these storage nodes create a transaction, and then send it to the blockchain miners for confirming that the data retrieval has been served.

5.6.6 User Revocation and Data Re-encryption

If a data owner wants to revoke a set of data consumers, it first updates all hidden address lists in its smart contracts by excluding the addresses related to the revoked consumers. Afterward, the consumers are no longer able to provide valid ring signatures corresponding to the updated hidden address lists for requesting data from storage nodes.

The platform also ensures that the revoked consumers are incapable of decrypting the encrypted data. To this end, the data owner runs a **SKeyUpdate** operation to obtain an unrevocation update PU that is signed with the private smart contract key sSK, and is sent to the contract. Upon receipt of PU, the legitimate consumers update their RPE private keys. The **UKeyUpdate** operation only works for legitimate consumers. The **UKeyUpdate** operation performed by the revoked consumers results in a failure. This procedure is to make the RPE private keys of the revoked users useless in decrypting the data shared after their revocation.

Next, a re-encryption key k_{re} is generated by one of the legitimate consumers by carrying out a **RKeyGen** operation. This key is encrypted with the public keys of the storage nodes and is then sent to them for the re-encryption phase. The revoked consumers are then unable to decrypt the re-encrypted data.

5.7 Security and Privacy Analysis

This section aims at proving that the proposed platform fulfills all the security and privacy requirements defined in the section 5.3.2.

5.7.1 Data Confidentiality

5.7.1.1 Security Proof of the Revocable Predicate Encryption (PRE)

As we apply the same revocation mechanism as the one introduced earlier in Chapter 4, the same security proof also applies in the context of PRE. This also shows how flexible our revocation mechanism is to make it compatible with many attribute-based encryption schemes.

To ease the security proof of the RPE scheme, we first define a Basic Encryption scheme as follows:

setup: Let $\mathbb{G}$ be a cyclic group of order p, generated by a generator g_p. The system chooses two random numbers $a, z \in \mathbb{Z}_p^*$, then computes a master revocation key MRK $= z$, a public revocation key PRK $= (g_p^z, g_p^a)$, and an unrevocation proof URP $= g_p^{az}$. Users knowing URP will be considered legitimate and able to decrypt shared data whereas the others are considered revoked.

encrypt: To encrypt a message m, one first picks at random $s \in \mathbb{Z}_p^*$ and computes the ciphertext

$$C = (g^s, m\hat{e}(g_p^a, g_p^z)^s). \tag{5.18}$$

decrypt: To decrypt the ciphertext, an authorized user uses its URP and computes:

$$m = \frac{m\hat{e}(g_p^a, g_p^z)^s}{\hat{e}(g_p^s, g_p^{az})} \tag{5.19}$$

Revoked users who do not know URP need to compute $\hat{e}(g_p^a, g_p^z)^s$ from a tuple of $(\hat{e}, g_p^z, g_p^a, g_p^s)$. However, the possibility of achieving that is negligible under the Decisional Bilinear Diffie-Hellman (DBDH) assumption [210]. Thus, the Basic Encryption scheme is provably secure under the DBDH assumption.

The RPE scheme is developed on the PE scheme in [206], and can be thought of as a combination of two separate encryption schemes including the Basic Encryption scheme and the PE scheme. The two systems are separate in design and all random secrets are independently chosen at random for each scheme. Both schemes are securely proven under the standard assumption, making RPE secure, too.

5.7.1.2 Forward Secrecy

This security requirement guarantees that revoked users are unable to decrypt the data shared with them in the past even in the case that they can compromise the storage nodes and retrieve

the encrypted data. In the platform, once some users are revoked access to encrypted data, the data owner generates a re-encryption key and sends it to the storage nodes to re-encrypt the data that only un-revoked users can decrypt by using their private keys. For revoked users to decrypt the re-encrypted data, they need to compute X from the element $C_x = X\hat{e}(g_p^{\tilde{a}}, g_p^z)^r$ by using related available information, a tuple of $(g_p, g_p^{\tilde{a}}, g_p^z, g_p^r)$. This is, however, unfeasible under the DBDH assumption.

5.7.2 User Privacy

To share data with a set of data consumers, the data owner $\mathcal{S}$ deploys a new data sharing smart contract with a random smart contract key pair (sPK and sSK). The privacy of the data owner is thus protected. To anonymously include a data consumer ($\mathcal{R}$) into the hidden access control list ($\mathcal{L}$), $\mathcal{S}$ generates a public hidden key (hPK_{s-r}) to $\mathcal{R}$. Hereby, $\mathcal{L}$ is composed of the public hidden keys of all the authorized data consumers, and is stored on the blockchain. We now prove that hPK_{s-r} does reveal any information about neither $\mathcal{S}$ nor $\mathcal{R}$. As described in Algorithm 5.2, hPK_{s-r} is computed based on an ephemeral key pair ($eSK_r = e; ePK_r = eG$) and the public identity key ($iPK_r = x_rG$) of $\mathcal{R}$ so that $hPK_{s-r} = H(e.x_r.G)+x_rG$. Therefore, in order to identify the data consumer who has been included into $\mathcal{L}$ by hPK_{s-r}, an adversary has to try the public identity keys $\{iPK_k\}_{k=1..N}$ of all N candidates to recompute the corresponding public hidden key hPK_{s-k} and compare it with hPK_{s-r}. However, given the available information ($ePK_r = eG, hPK_{s-r}$) and $\{iPK_k = x_kG\}_{k=1..N}$, it is computationally infeasible to compute hPK_{s-k}. Indeed, given ($ePK = eG$ and $iPK_k = x_kG$), the adversary cannot obtain $e.x_k.G$, which is required in Step 1 of Algorithm 5.2, due to the Discrete Logarithm Complexity assumption.

5.7.3 User Linkability

We prove that an adversary cannot link activities done by a specific user. For data sharing, a data owner uses a different smart contract key pair (sPK, sSK), which are one-time keys, to deploy a smart contract. This is, thus, impossible to link two contracts made by the same data owner. Similarly, in each data sharing, a data consumer i is represented by a hidden key pair (hPK_i and hSK_i), in which the public hidden key is stored on the blockchain and used to prove the right to access the shared data. The hidden key pair is generated randomly for every data consumer in every data sharing. In other words, if a data consumer is involved in N data sharing contracts, it will have N hidden key pairs, which are computationally indistinguishable, thus making data consumers anonymous across all the smart contracts. Within one smart contract, the adversary cannot distinguish the access sessions of the same user due to the randomness of the ring signature scheme. Hence, we conclude that it is impossible to link activities of the same user over the whole system.

5.7.4 User and Data Linkability

This is the result of the combination of the above security and privacy guarantees. Indeed, both the confidentiality of data and user privacy is protected, making it impossible to determine which user has accessed which data.

Table 5.1: Deployment cost measurement of smart contracts

Contracts	Transaction Cost (gas)	Execution Cost (gas)	Total Deployment Cost (gas)
Identity Key Contract	641176	446080	1087256
Data Sharing Contract	1318303	951831	2270134
Ring Signature Contract	2442294	1802498	4244792

5.8 Performance Evaluation

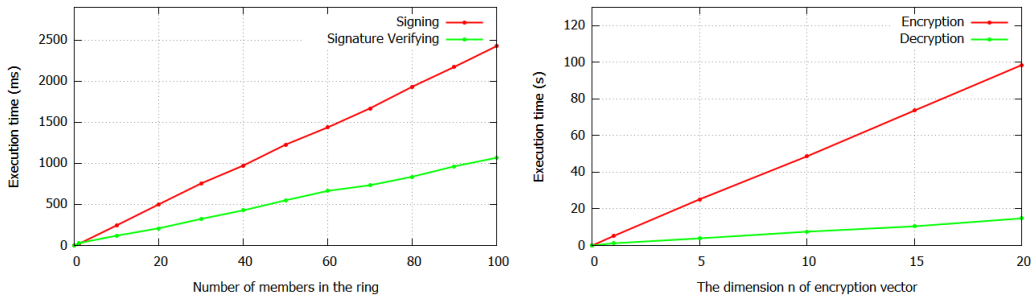
In this section, we evaluate the performance of the platform by measuring the computational costs of the computationally heavyweight operations performed by the users as well as the costs of deploying the smart contracts on the blockchain.

5.8.1 Evaluation of the Smart Contract Deployment Cost

The Ethereum blockchain is maintained by miners that are responsible for verifying and adding transactions into the blockchain. Therefore, all operations must be paid to the miners to be executed on the blockchain. Specifically, *gas* is the unit of measurement that defines the amount of computational effort needed to execute an operation. To initiate a transaction, a sender must specify two variables: *gas limit* and *gas price*. The former indicates the maximum amount of *gas* that the sender is willing to pay for the transaction while the latter indicates the price of *gas*. The *gas* price is set based on ETH, the underlying cryptocurrency of the Ethereum blockchain.

In the platform, there are 3 main types of contract: the Identity Key contract, the Data Sharing contract, and the Ring Signature contract. All these are written in the Solidity language. As explained above, the Identity Key contract allows new users to register to join the sharing system. To outsource an upload, which could be a set of data, the data owner deploys a Data Sharing contract that allows managing anonymous and auditable access control as well as registering the location of the data on the IPFS network. We separately deploy a Ring Signature contract which can be called by the Data Sharing contracts to verify the eligibility of data consumers. The ring signature scheme is implemented on the curve ECSecp256k1, which is standardized by SECG [211]. We also open the source code of the smart contracts in [212]. The deployment costs of the smart contracts are illustrated in Table 5.1.

5.8.2 Evaluation of Computationally Heavyweight Operations



(a) Execution time of the ring signature (b) Execution time of RPE

Figure 5.4: Time execution measurement of the ring signature and RPE schemes.

On the user side, the main computational burden is caused by the cryptographic operations which are performed on their devices such as PC or mobile. We observe that the generation of keys or hidden access control lists requires a small number of lightweight operations, including additions and multiplications. Therefore, we concentrate on the computation costs of the heavyweight operations which can also vary and depend on the context: the ring signature and revocable predicate encryption schemes. With regard to the ring signature scheme, it consists of signing and verifying operations. While the former is performed by data consumers, the latter is performed by the blockchain miners and is implemented in the Ring Signature contract given in [212]. Due to the huge computation capacity of the blockchain miners, the verifying operation can be done nearly instantly. However, for the completeness of the demonstration, we show the costs of both operations when being carried out on the experimental computer.

We implement the RPE and the ring signature schemes on a computer with Core i7 2.81GHz processor and 8Gb RAM. The ring signature scheme is implemented on the curve ECSecp256k1 as mentioned above whereas the revocation predicate encryption scheme is implemented on the curve $y^2 = x^3 + x$ over a field of composite order. In this experiment, the length of each prime number composing the order is 512 bits. The curve allows performing pairing of type A1, the pairing over a group of composite order, which is the core building block of RPE. The performance of RPE is measured according to the dimension of encryption vectors whereas the performance of the ring signature scheme is evaluated based on the number of members in a ring. As illustrated in Figure 5.4, the ring signature scheme is practical as it takes less than 2.5s to create a signature on behalf of a large group of 100 members. Moreover, the signature verifying algorithm only requires 1s for such a signature, respectively. However, the revocable predicate encryption is slightly inefficient as encrypting a message with a 10-dimensions vector requires 50s and the decryption requires 7.5s, accordingly. The inefficiency of RPE is due to the expensive cost of the cryptographic operations performed over a pairing group of composite order, such as the pairing and the modular exponentiation. In [213], Freeman develops a method to convert cryptosystems built on composite-order groups, including the original predicate encryption scheme [206], to prime-order groups to improve performance. Applying this method in RPE helps to greatly reduce the computational overhead. Indeed, on our given computer, performing a pairing over a prime-order group is around 33 times faster than the same operation over a comparable composite-order group (15ms versus 500ms). Similarly, 35 times is the improvement factor with regard to the modular exponentiation.

5.9 Conclusions

In this chapter, we presented a data sharing platform to ensure four critical requirements necessary for constituting a robust platform, which are data confidentiality, data access control, user privacy, and data availability. To achieve these purposes, we leveraged the blockchain technology as well as various cryptographic primitives. In the proposed platform, we not only provided secure end-to-end encryption for data storage and sharing but the guaranteed user privacy and auditable access control mechanism also added more value onto the platform, making it more attractive compared to traditional systems especially in the contexts where user privacy matters. In addition, the improvement of data availability is also achieved thanks to the high performance content distribution system. To evaluate the practicability of the platform, we succeeded to prove the privacy and security requirements as defined in the section 5.3.2. We also implemented and measured the performance of the necessary smart contracts and the

integrated cryptographic primitives built in concordance with the proposed architecture of our privacy-preserving data sharing platform.

Chapter 6

Conclusion and Future Work

Contents

6.1 Foreword

In this chapter, we summarize the contributions of this dissertation and highlight the possible future extensions that could be promising to further develop our proposed solutions to make them usable in other use cases.

6.2 Conclusion

In this dissertation, we present our different approaches to enable secure data storage and data sharing which are fundamental in the nowadays digital transformation wave. To this end, we concentrated on two of necessary components constituting such solutions: authentication protocols and secure data sharing schemes. For each, we not only investigated the underlying problems of existing solutions to propose effective improvements but also presented new approaches, which include additional security and privacy features.

Password-based authentication protocols remain the most practical authentication method regardless of the inherent vulnerability to dictionary attacks. Many efforts have been made to make the protocols stronger by implementing them on the TLS protocol, which relies on CA-based PKIs, or incorporating more authentication factors such as smartcard, and biometry. Our first contribution in this dissertation was to leverage the signcryption technique to introduce a new paradigm of designing PAKE protocols, which allow two parties to mutually authenticate each other and negotiate a secure session key without relying on any additional centralized infrastructure. We proved the security of the proposed protocol, namely PSKE, in the Find-then-Guess Model which has been widely accepted and become a reference for PAKE protocol designs. Moreover, we also showed that PSKE outperforms the existing PAKE protocols in terms of both the communication and computational costs.

Besides proposing improvements on the traditional authentication approach, as the second contribution in this dissertation, we leveraged the blockchain technology to propose a privacy-enhancing decentralized public key infrastructure for digital identity management. The proposed infrastructure is then the base layer to design authentication protocols with desirable security and privacy guarantees. More concretely, the proposal puts the management of identities to the hands of users. Users are responsible for all operations over their identities, including registration, update, revocation, and recovery without relying on any centralized entities. All these operations are then publicly verifiable by the identity owners due to the transparency and consistency of the Ethereum blockchain, which is the smart contract platform chosen to illustrate this contribution. Generally, the transparency of the blockchain paves the way for harming user privacy if the infrastructure is not properly and securely designed. Our proposed key decoupling mechanism and anonymous key registration allows avoiding these potential risks. We also discussed some applications of the proposed DPKI in the context of data storage and sharing and others.

The more people move towards data outsourcing solutions, the more complex security risks concerning data access control that service providers need to deal with to ensure the safety of a huge volume of data. To reduce reliance on the service providers with respect to data confidentiality in the context of group-based data sharing, and as a third contribution of this dissertation, we presented a forward-secure data outsourcing solution. The aim of the proposal is to strengthen the security of existing systems by providing users with effective means of protecting data confidentiality which incorporates an additional security notion, called forward-secrecy. Due to the flexibility of our proposed solutions, data owners have the right to flexibly define access control policies on their outsourced data so that only authorized users can decrypt the data. In the system model, the service providers only play the intermediary role of transferring encrypted data between users and they are incapable of decrypting the data without the explicit permission from the data owners. In addition, the forward-secrecy notion also reinforces the security of the system. It ensures that revoked users can no longer decrypt the data previously shared with them.

Moving beyond the traditional approach, in the fourth contribution of this dissertation, we leveraged the blockchain technology to present a new way that users outsource and share data. The proposed data sharing platform was constructed to simultaneously ensure data confidentiality, fair data access control, data availability, and user privacy. Whereas many users are still hesitant to use centralized services due to potential risks concerning user privacy and data availability, the combination of these four mentioned factors makes it a valuable platform in practice, thus extending the user base. Indeed, the third contribution mainly targets the data confidentiality protection for any arbitrarily-sized group of users, there still exist issues related to user privacy, data censorship and even data availability in the centralized approach. More specifically, the providers can still identify who is sharing data with whom or the relation between parties involved in some outsourced data. The fourth contribution is therefore introduced to effectively solve all these issues.

6.3 Future Work

In the dissertation, we focused on building two-party authentication and key exchange protocols, including PSKE and protocols that can be built on top of the proposed DPKI. However, the applicability of these protocols are evidently not limited to this context as digital authentication

is mandatory in any applications in which user identity needs to be verified before being granted access to required services. We discuss the possible extensions of the protocols which could lead to more use cases.

With respect to PSKE, an extension to a three-party setting, in which the mutual authentication between two users are mediated by an *honest-but-curious* server, could be possible. More specifically, two users share with the server two distinct passwords and try to authenticate each other based on these passwords. The term *honest-but-curious* means the server follows the prescribed protocol but tries to extract as much information as possible based on the available information. The idea of the three-party setting has been investigated in the literature [214,215] and formally refined by Abdalla et al. [5]. In addition to a formal security for PAKE protocols in a three-party setting, Abdalla et al. also proposed a generic way to securely convert a two-party PAKE to a three-party PAKE protocol. Our future work of PSKE could follow this strategy and apply more performance optimizations to propose an efficient construction.

Regarding the privacy-enhancing decentralized public key infrastructure, we would like to extend the infrastructure by integrating an anonymous credential system [216]. Specifically, an anonymous credential system allows users to make assertions about their identity while maintaining privacy. For example, a user receives a credential which contains a set of attributes describing itself, i.e., name, age, country, and which is signed by the government, which is commonly called credential issuer. The signing procedure in this context is realized via blind signature schemes [113, 217]. This is to ensure that the credential issuer does not know what it signs but is still convinced that the signing data is constructed correctly and valid. This property of the blind signature schemes guarantees user privacy requirement even in the case that the credential issuer and verifier collude to correlate the user's activities. Moreover, an anonymous credential system also supports minimal information disclosure. That is, the user is allowed to show only necessary information which is required by the credential verifier while hiding the rest of information in the credential. Advanced techniques like zero-knowledge proofs and commitment schemes also allow proving that some attributes signed by the credential issuer satisfy some conditions without revealing what these attributes are. A typical example of this context is that a user can prove that its age is in a range, i.e., $[18, 30]$, without revealing the real age. As user privacy is a priority requirement in the proposed DPKI, the integration of an anonymous credential system would help broaden the use cases of our proposal.

Both data sharing solutions proposed in the dissertation are quite complete. The potential extensions to these works would lie in the implementation and deployment over a large scale system to obtain better evaluation of the system performance and users satisfaction. In particular, the decentralized data sharing platform was designed to run on the Ethereum blockchain, which is actually expensive in terms of storage and execution of the deployed smart contracts. Even though Ethereum has been proven to be the leading technology in the smart contract platform space due to many aspects, i.e., the global distribution of nodes participating in securing the blockchain, a huge community supporting the platform, and a rich set of toolkits facilitating the smart contract development, there have been many other promising platforms emerging to compete each other. Cardano [84] and Algorand [85] are two of them, which apply scientific approaches to provide provably secure consensus mechanisms. However, these are still at early days of development compared to Ethereum. Once the developments are finished, the migration of our platform to these blockchains with the aim of optimizing even more is also part of our perspective.

Author Publications

Journals:

- Hoang, V. H., Lehtihet, E., and Ghamri-Doudane, Y. Privacy-Enhancing Decentralized Public Key Infrastructure. [**In preparation for submission**]

Conferences

- Hoang, V. H., Lehtihet, E., and Ghamri-Doudane, Y. (2019, April). Password-Based Authenticated Key Exchange Based on Signcryption for the Internet of Things. In *2019 Wireless Days (WD)* (pp. 1-8). IEEE.
- Hoang, V. H., Lehtihet, E., and Ghamri-Doudane, Y. (2019, June). Forward-Secure Data Outsourcing Based on Revocable Attribute-Based Encryption. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC) (pp. 1839-1846).* IEEE.
- V. H. Hoang, E. Lehtihet, and Y. Ghamri-Doudane, Privacy-Preserving Blockchain-Based data sharing platform for decentralized storage systems, in *IFIP Networking 2020 Conference (IFIP Networking 2020), (Paris, France), June 2020.*

Bibliography

[1] S. Kent and R. Atkinson, "Ip encapsulating security payload (esp): Rfc 4303 (proposed standard)," 2005.

[2] A. Freier, P. Karlton, and P. Kocher, "The secure sockets layer (ssl) protocol version 3.0," *IETF*, vol. 3, pp. 1–67, 2011.

[3] T. Ylonen and C. Lonvick, "The secure shell (ssh) transport layer protocol," tech. rep., RFC 4253, January, 2006.

[4] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Advances in Cryptology — EUROCRYPT 2000*, pp. 139–155, 2000.

[5] M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *International Workshop on Public Key Cryptography*, pp. 65–84, Springer, 2005.

[6] P. Eronen and H. Tschofenig, "Pre-shared key ciphersuites for transport layer security (tls)," tech. rep., RFC 4279, December, 2005.

[7] F. Bersani and H. Tschofenig, "The eap-psk protocol: A pre-shared key extensible authentication protocol (eap) method," tech. rep., RFC 4764, January, 2007.

[8] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pp. 232–249, 1994.

[9] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[10] H. Krawczyk, "Sigma: The 'sign-and-mac'approach to authenticated diffie-hellman and its use in the ike protocols," in *Annual International Cryptology Conference*, pp. 400–425, Springer, 2003.

[11] E. Rescorla and T. Dierks, "The transport layer security (tls) protocol version 1.3," 2018.

[12] J. R. Prins and B. U. Cybercrime, "Diginotar certificate authority breach "operation black tulip"," *Fox-IT, November*, p. 18, 2011.

[13] S. Garfinkel, *PGP: pretty good privacy.* " O'Reilly Media, Inc.", 1995.

[14] A. Abdul-Rahman, "The pgp trust model," in *EDI-Forum: the Journal of Electronic Commerce*, vol. 10, pp. 27–31, 1997.

[15] D. Chen, X. Li, L. Wang, S. U. Khan, J. Wang, K. Zeng, and C. Cai, "Fast and scalable multi-way analysis of massive neural data," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 707–719, 2014.

[16] M. Ali, R. Dhamotharan, E. Khan, S. U. Khan, A. V. Vasilakos, K. Li, and A. Y. Zomaya, "Sedasc: secure data sharing in clouds," *IEEE Systems Journal*, vol. 11, no. 2, pp. 395–404, 2015.

[17] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology—CRYPTO 2001*, pp. 213–229, 2001.

[18] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," 1992.

[19] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proceedings of the 12th ACM conference on Computer and communications security*, pp. 364–372, 2005.

[20] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Annual International Cryptology Conference*, pp. 617–630, Springer, 2003.

[21] T. D. Wu, "A real-world analysis of kerberos password security.," in *Ndss*, 1999.

[22] S. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy.*

[23] S. Patel, "Number theoretic attacks on secure password schemes," in *Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No.97CB36097).*

[24] D. P. Jablon, "Strong password-only authenticated key exchange," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 5, pp. 5–26, 1996.

[25] M. Zhang, "Analysis of the SPEKE password-authenticated key exchange protocol," *IEEE Communications Letters*, vol. 8, no. 1, pp. 63–65, 2004.

[26] Q. Tang and C. J. Mitchell, "On the security of some password-based key agreement schemes," 2005.

[27] V. Boyko, P. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using diffie-hellman," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 156–171, Springer, 2000.

[28] M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," in *Proceedings of the 2005 International Conference on Topics in Cryptology*, CT-RSA'05, pp. 191–208, 2005.

[29] Y. Zhang, Y. Xiang, W. Wu, and A. Alelaiwi, "A variant of password authenticated key exchange protocol," *Future Generation Computer Systems*, vol. 78, pp. 699–711, 2018.

[30] D. Pointcheval and G. Wang, "Vtbpeke: Verifier-based two-basis password exponential key exchange," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pp. 301–312, 2017.

[31] J. Katz, R. Ostrovsky, and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 475–494, Springer, 2001.

[32] K. Kobara and H. Imai, "Pretty-simple password-authenticated key-exchange under standard assumptions.," *IACR Cryptol. ePrint Arch.*, vol. 2003, p. 38, 2003.

[33] S. Jiang and G. Gong, "Password based key exchange with mutual authentication," in *International Workshop on Selected Areas in Cryptography*, pp. 267–279, Springer, 2004.

[34] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pp. 307–328, 2019.

[35] F. Hao and P. Y. A. Ryan, "Password authenticated key exchange by juggling," in *Proceedings of the 16th International Conference on Security Protocols*, Security'08, pp. 159–171, 2011.

[36] M. Abdalla, F. Benhamouda, and P. MacKenzie, "Security of the j-pake password-authenticated key exchange protocol," in *2015 IEEE Symposium on Security and Privacy*, pp. 571–587, IEEE, 2015.

[37] S. M. Bellovin and M. Merritt, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pp. 244–250, 1993.

[38] M. Abdalla, O. Chevassut, and D. Pointcheval, "One-time verifier-based encrypted key exchange.," in *Public Key Cryptography*, vol. 3386, pp. 47–74, 2005.

[39] C. Gentry, P. MacKenzie, and Z. Ramzan, "A method for making password-based key exchange resilient to server compromise," in *Annual International Cryptology Conference*, pp. 142–159, Springer, 2006.

[40] P. MacKenzie, "The pak suite: Protocols for password-authenticated key exchange," in *IEEE P1363. 2*, Citeseer, 2002.

[41] P. Mackenzie, "More efficient password-authenticated key exchange," in *Cryptographers' Track at the RSA Conference*, pp. 361–377, Springer, 2001.

[42] P. MacKenzie, S. Patel, and R. Swaminathan, "Password-authenticated key exchange based on rsa," in *International conference on the theory and application of cryptology and information security*, pp. 599–613, Springer, 2000.

[43] T. D. Wu *et al.*, "The secure remote password protocol.," in *NDSS*, vol. 98, pp. 97–111, 1998.

[44] M. Bellare and P. Rogaway, "Provably secure session key distribution: The three party case," in *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pp. 57–66, 1995.

[45] S. Blake-Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," in *IMA international conference on cryptography and coding*, pp. 30–45, Springer, 1997.

[46] M. K. Boyarsky, "Public-key cryptography and password protocols: The multi-user case," in *Proceedings of the 6th ACM conference on Computer and communications security*, pp. 63–72, 1999.

[47] S. Halevi and H. Krawczyk, "Public-key cryptography and password protocols," *ACM Transactions on Information and System Security (TISSEC)*, vol. 2, no. 3, pp. 230–268, 1999.

[48] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 531–545, Springer, 2000.

[49] M. Bellare and P. Rogaway, "Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 317–330, Springer, 2000.

[50] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith, "Efficient and non-interactive non-malleable commitment," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 40–59, Springer, 2001.

[51] H. Krawczyk, "The order of encryption and authentication for protecting communications (or: How secure is ssl?)," in *Annual International Cryptology Conference*, pp. 310–331, Springer, 2001.

[52] J. H. An, Y. Dodis, and T. Rabin, "On the security of joint signature and encryption," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 83–107, Springer, 2002.

[53] Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption)," in *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pp. 165–179, 1997.

[54] C. Badertscher, F. Banfi, and U. Maurer, "A constructive perspective on signcryption security," in *International Conference on Security and Cryptography for Networks*, pp. 102–120, Springer, 2018.

[55] J. Baek, R. Steinfeld, and Y. Zheng, "Formal proofs for the security of signcryption," *Journal of cryptology*, vol. 20, no. 2, pp. 203–235, 2007.

[56] T. Okamoto and D. Pointcheval, "The gap-problems: A new class of problems for the security of cryptographic schemes," in *International workshop on public key cryptography*, pp. 104–118, Springer, 2001.

[57] Y. Zheng and H. Imai, "How to construct efficient signcryption schemes on elliptic curves," *Information processing letters*, vol. 68, no. 5, pp. 227–233, 1998.

[58] R. Steinfeld and Y. Zheng, "A signcryption scheme based on integer factorization," in *International Workshop on Information Security*, pp. 308–322, Springer, 2000.

[59] B. Libert and J.-J. Quisquater, "A new identity based signcryption scheme from pairings," in *Proceedings 2003 IEEE Information Theory Workshop (Cat. No. 03EX674)*, pp. 155–158, IEEE, 2003.

[60] B. Libert and J.-J. Quisquater, "Efficient signcryption with key privacy from gap diffie-hellman groups," in *International Workshop on Public Key Cryptography*, pp. 187–200, Springer, 2004.

[61] J. K. Liu, J. Baek, and J. Zhou, "Online/offline identity-based signcryption revisited," in *International conference on information security and cryptology*, pp. 36–51, Springer, 2010.

[62] S. S. D. Selvi, S. S. Vivek, and C. P. Rangan, "Identity based public verifiable signcryption scheme," in *International Conference on Provable Security*, pp. 244–260, Springer, 2010.

[63] S. S. D. Selvi, S. S. Vivek, D. Vinayagamurthy, and C. P. Rangan, "Id based signcryption scheme in standard model," in *International Conference on Provable Security*, pp. 35–52, Springer, 2012.

[64] M. Barbosa and P. Farshim, "Certificateless signcryption," in *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pp. 369–372, 2008.

[65] T. Pandit, S. K. Pandey, and R. Barua, "Attribute-based signcryption: Signer privacy, strong unforgeability and ind-cca2 security in adaptive-predicates attack," in *International Conference on Provable Security*, pp. 274–290, Springer, 2014.

[66] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional signcryption: Notion, construction, and applications," in *International Conference on Provable Security*, pp. 268–288, Springer, 2015.

[67] A. W. Dent, "Hybrid signcryption schemes with outsider security," in *International Conference on Information Security*, pp. 203–217, Springer, 2005.

[68] R.-J. Hwang, C.-H. Lai, and F.-F. Su, "An efficient signcryption scheme with forward secrecy based on elliptic curve," *Applied Mathematics and computation*, vol. 167, no. 2, pp. 870–881, 2005.

[69] F. Bao and R. H. Deng, "A signcryption scheme with signature directly verifiable by public key," in *International workshop on public key cryptography*, pp. 55–59, Springer, 1998.

[70] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 409–426, 2006.

[71] S.-M. Hong, S.-Y. Oh, and H. Yoon, "New modular multiplication algorithms for fast modular exponentiation," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 166–177, Springer, 1996.

[72] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[73] W. T. Polk, D. F. Dodson, W. E. Burr, H. Ferraiolo, and D. Cooper, "Cryptographic algorithms and key sizes for personal identity verification," 2005.

[74] M. Bellare and P. Rogaway, "The autha protocol for password-based authenticated key exchange," in *IEEE P1363*, pp. 136–3, 2000.

[75] B. Haase and B. Labrique, "Aucpace: Efficient verifier-based pake protocol tailored for the iiot," *IACR Cryptology ePrint Archive*, vol. 2018, p. 286, 2018.

[76] D. P. Jablon, "Extended password key exchange protocols immune to dictionary attacks," in *Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, WET-ICE '97, pp. 248–255, 1997.

[77] B. Laurie, "Certificate transparency," *Communications of the ACM*, vol. 57, no. 10, pp. 40–46, 2014.

[78] L. Chuat, P. Szalachowski, A. Perrig, B. Laurie, and E. Messeri, "Efficient gossip protocols for verifying the consistency of certificate logs," in *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 415–423, IEEE, 2015.

[79] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention.," *IACR Cryptology ePrint Archive*, vol. 2014, p. 803, 2014.

[80] A. Yakubov, W. Shbair, A. Wallbom, D. Sanda, *et al.*, "A blockchain-based pki management framework," in *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain 23-27 April 2018*, 2018.

[81] D. Madala, M. P. Jhanwar, and A. Chattopadhyay, "Certificate transparency using blockchain," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 71–80, IEEE, 2018.

[82] J. Chen, S. Yao, Q. Yuan, K. He, S. Ji, and R. Du, "Certchain: Public and efficient certificate audit based on blockchain for tls connections," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 2060–2068, IEEE, 2018.

[83] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15, 2018.

[84] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*, pp. 357–388, Springer, 2017.

[85] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, 2017.

[86] J. Abadi and M. Brunnermeier, "Blockchain economics," tech. rep., National Bureau of Economic Research, 2018.

[87] L. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1545–1550, IEEE, 2018.

[88] "https://namecoin.info/, accessed on 17/09/2015 at 10:31,"

[89] S. Matsumoto and R. M. Reischuk, "Ikp: Turning a pki around with decentralized automated incentives," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 410–426, IEEE, 2017.

[90] L. Dykcik, L. Chuat, P. Szalachowski, and A. Perrig, "Blockpki: An automated, resilient, and transparent public-key infrastructure," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 105–114, IEEE, 2018.

[91] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to bitcoin," *Designs, Codes and Cryptography*, vol. 87, no. 9, pp. 2139–2164, 2019.

[92] D. Boneh, M. Drijvers, and G. Neven, "Bls multi-signatures with public-key aggregation,"

[93] A. Yakubov, W. Shbair, and R. State, "Blockpgp: A blockchain-based framework for pgp key servers," in *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, pp. 316–322, IEEE, 2018.

[94] M. Al-Bassam, "Scpki: A smart contract-based pki and identity system," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pp. 35–40, 2017.

[95] B. Qin, J. Huang, Q. Wang, X. Luo, B. Liang, and W. Shi, "Cecoin: A decentralized pki mitigating mitm attacks," *Future Generation Computer Systems*, 2017.

[96] L. Axon and M. Goldsmith, "Pb-pki: A privacy-aware blockchain-based pki," 2016.

[97] W. Jiang, H. Li, G. Xu, M. Wen, G. Dong, and X. Lin, "A privacy-preserving thin-client scheme in blockchain-based pki," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[98] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pp. 41–50, IEEE, 1995.

[99] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic, "Distributed denial of service attacks," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 3, pp. 2275–2280, IEEE, 2000.

[100] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[101] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[102] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 552–565, Springer, 2001.

[103] A. Bender, J. Katz, and R. Morselli, "Ring signatures: Stronger definitions, and constructions without random oracles," in *TCC*, pp. 60–79, Springer, 2006.

[104] J. Herranz and G. Sáez, "Forking lemmas for ring signature schemes," in *INDOCRYPT*, pp. 266–279, Springer, 2003.

[105] H. Shacham and B. Waters, "Efficient ring signatures without random oracles," in *International Workshop on Public Key Cryptography*, pp. 166–180, Springer, 2007.

[106] N. Chandran, J. Groth, and A. Sahai, "Ring signatures of sub-linear size without random oracles," in *International Colloquium on Automata, Languages, and Programming*, pp. 423–434, Springer, 2007.

[107] O. Goldreich, "Zero-knowledge twenty years after its invention.," *IACR Cryptol. ePrint Arch.*, vol. 2002, p. 186, 2002.

[108] G. Brassard, D. Chaum, and C. Crépeau, "Minimum disclosure proofs of knowledge," *Journal of computer and system sciences*, vol. 37, no. 2, pp. 156–189, 1988.

[109] L. Babai, "Trading group theory for randomness," in *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pp. 421–429, 1985.

[110] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Annual International Cryptology Conference*, pp. 390–420, Springer, 1992.

[111] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on computing*, vol. 18, no. 1, pp. 186–208, 1989.

[112] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems," *Journal of the ACM (JACM)*, vol. 38, no. 3, pp. 690–728, 1991.

[113] S. A. Brands, "An efficient off-line electronic cash system based on the representation problem," 1993.

[114] G. Brassard, C. Crépeau, and M. Yung, "Everything in np can be argued in perfect zero-knowledge in a bounded number of rounds," in *International Colloquium on Automata, Languages, and Programming*, pp. 123–136, Springer, 1989.

[115] J. L. Camenisch, R. R. Enderlein, A. Lehmann, and G. Neven, "Privacy-preserving attribute-based credentials," Sept. 18 2018. US Patent 10,079,686.

[116] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung, "Multi-authority secret-ballot elections with linear work," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 72–83, Springer, 1996.

[117] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pp. 329–349, 2019.

[118] U. Feige, D. Lapidot, and A. Shamir, "Multiple non-interactive zero knowledge proofs based on a single random string," in *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pp. 308–317, IEEE, 1990.

[119] Y. Oren, "On the cunning power of cheating verifiers: Some observations about zero knowledge proofs," in *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pp. 462–471, IEEE, 1987.

[120] A. De Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge with preprocessing," in *Conference on the Theory and Application of Cryptography*, pp. 269–282, Springer, 1988.

[121] R. Cramer and I. Damgård, "Secret-key zero-knowlegde and non-interactive verifiable exponentiation," in *Theory of Cryptography Conference*, pp. 223–237, Springer, 2004.

[122] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Conference on the theory and application of cryptographic techniques*, pp. 186–194, Springer, 1986.

[123] M. Bellare, A. Boldyreva, and A. Palacio, "An uninstantiable random-oracle-model scheme for a hybrid-encryption problem," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 171–188, Springer, 2004.

[124] R. Canetti, O. Goldreich, and S. Halevi, "On the random-oracle methodology as applied to length-restricted signature schemes," in *Theory of Cryptography Conference*, pp. 40–57, Springer, 2004.

[125] J. Groth, R. Ostrovsky, and A. Sahai, "Perfect non-interactive zero knowledge for np," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 339–358, Springer, 2006.

[126] J. Groth, R. Ostrovsky, and A. Sahai, "Non-interactive zaps and new techniques for nizk," in *Annual International Cryptology Conference*, pp. 97–111, Springer, 2006.

[127] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 415–432, Springer, 2008.

[128] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Annual International Cryptology Conference*, pp. 410–424, Springer, 1997.

[129] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[130] F. Hao, "Schnorr non-interactive zero-knowledge proof," *RFC Editor, Tech. Rep.*, 2017.

[131] M. Blum, "Coin flipping by telephone a protocol for solving impossible problems," *ACM SIGACT News*, vol. 15, no. 1, pp. 23–27, 1983.

[132] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[133] I. Damgård, "Commitment schemes and zero-knowledge protocols," in *School organized by the European Educational Forum*, pp. 63–86, Springer, 1998.

[134] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Annual international cryptology conference*, pp. 129–140, Springer, 1991.

[135] A. Shamir, "How to share a secret," *Communications of the ACM*, pp. 612–613, 1979.

[136] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *International conference on the theory and application of cryptology and information security*, pp. 452–473, Springer, 2003.

[137] L. Chen, Z. Cheng, and N. P. Smart, "Identity-based key agreement protocols from pairings," *International Journal of Information Security*, vol. 6, no. 4, pp. 213–241, 2007.

[138] C. Boyd and K.-K. R. Choo, "Security of two-party identity-based key agreement," in *International conference on cryptology in Malaysia*, pp. 229–243, Springer, 2005.

[139] V.-H. Hoang, E. Lehtihet, and Y. Ghamri-Doudane, "Password-based authenticated key exchange based on signcryption for the internet of things," in *2019 Wireless Days (WD)*, pp. 1–8, IEEE, 2019.

[140] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably authenticated group diffie-hellman key exchange—the dynamic case," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 290–309, Springer, 2001.

[141] M. Abdalla, J.-M. Bohli, M. I. G. Vasco, and R. Steinwandt, "(password) authenticated key establishment: from 2-party to group," in *Theory of Cryptography Conference*, pp. 499–514, Springer, 2007.

[142] J. Alwen, S. Coretti, and Y. Dodis, "The double ratchet: security notions, proofs, and modularization for the signal protocol," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 129–158, Springer, 2019.

[143] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 275–286, Springer, 1994.

[144] M. Burmester and Y. Desmedt, "A secure and scalable group key exchange system," *Information Processing Letters*, vol. 94, no. 3, pp. 137–143, 2005.

[145] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pp. 321–334, 2007.

[146] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *International Conference on Pairing-Based Cryptography*, pp. 248–265, 2009.

[147] Q. Dong, D. Huang, J. Luo, and M. Kang, "Achieving fine-grained access control with discretionary user revocation over cloud data," in *2018 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9, 2018.

[148] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE transactions on knowledge and data engineering*, pp. 2271–2282, 2013.

[149] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Springer, 2005.

[150] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98, 2006.

[151] Garner, "https://www.avatier.com/products/identity-management/resources/gartner-iam-2020-predictions/."

[152] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization.," in *Public Key Cryptography*, pp. 53–70, 2011.

[153] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 256–266, Springer, 1997.

[154] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 255–271, Springer, 2003.

[155] R. Canetti, S. Halevi, and J. Katz, "Chosen-ciphertext security from identity-based encryption," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 207–222, Springer, 2004.

[156] N. Attrapadung, B. Libert, and E. De Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *International Workshop on Public Key Cryptography*, pp. 90–108, 2011.

[157] B. Waters, "Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions," in *Annual International Cryptology Conference*, pp. 619–636, Springer, 2009.

[158] A. Lewko and B. Waters, "New proof methods for attribute-based encryption: Achieving full security through selective techniques," in *Annual Cryptology Conference*, pp. 180–198, Springer, 2012.

[159] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 62–91, Springer, 2010.

[160] M. Chase, "Multi-authority attribute based encryption," in *Theory of Cryptography Conference*, pp. 515–534, Springer, 2007.

[161] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 121–130, 2009.

[162] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Annual international conference on the theory and applications of cryptographic techniques*, pp. 568–588, Springer, 2011.

[163] M. Green, S. Hohenberger, B. Waters, *et al.*, "Outsourcing the decryption of abe ciphertexts.," in *USENIX security symposium*, vol. 2011, 2011.

[164] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[165] H. Cui, R. H. Deng, X. Ding, and Y. Li, "Attribute-based encryption with granular revocation," in *International Conference on Security and Privacy in Communication Systems*, pp. 165–181, 2016.

[166] X. Liang, R. Lu, X. Lin, and X. S. Shen, "Ciphertext policy attribute based encryption with efficient revocation," 2010.

[167] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes," in *Ima International Conference on Cryptography and Coding*, pp. 278–300, 2009.

[168] J. K. Liu, T. H. Yuen, P. Zhang, and K. Liang, "Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list," in *International Conference on Applied Cryptography and Network Security*, pp. 516–534, 2018.

[169] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Annual International Cryptology Conference*, pp. 258–275, 2005.

[170] A. Lewko, A. Sahai, and B. Waters, "Revocation systems with very small private keys," in *Security and Privacy (SP), 2010 IEEE Symposium on*, pp. 273–285, 2010.

[171] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pp. 261–270, 2010.

[172] H. Cui, R. H. Deng, Y. Li, and B. Qin, "Server-aided revocable attribute-based encryption," in *European Symposium on Research in Computer Security*, pp. 570–587, 2016.

[173] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1214–1221, 2011.

[174] L. Zu, Z. Liu, and J. Li, "New ciphertext-policy attribute-based encryption with efficient revocation," in *Computer and Information Technology (CIT), 2014 IEEE International Conference on*, pp. 281–287, 2014.

[175] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information sciences*, pp. 355–370, 2014.

[176] K. Liang, L. Fang, W. Susilo, and D. S. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*, pp. 552–559, 2013.

[177] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," *Advances in Cryptology—EUROCRYPT'98*, pp. 127–144, 1998.

[178] G. Ateniese and S. Hohenberger, "Proxy re-signatures: new definitions, algorithms, and applications," in *Proceedings of the 12th ACM conference on Computer and communications security*, pp. 310–319, 2005.

[179] G. Taban, A. A. Cárdenas, and V. D. Gligor, "Towards a secure and interoperable drm architecture," in *Proceedings of the ACM workshop on Digital rights management*, pp. 69–78, 2006.

[180] S. Lee, H. Park, and J. Kim, "A secure and mutual-profitable drm interoperability scheme," in *The IEEE symposium on Computers and Communications*, pp. 75–80, IEEE, 2010.

[181] Y.-R. Chen, J. Tygar, and W.-G. Tzeng, "Secure group key management using uni-directional proxy re-encryption schemes," in *2011 Proceedings IEEE INFOCOM*, pp. 1952–1960, IEEE, 2011.

[182] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Applied Cryptography and Network Security*, pp. 288–306, 2007.

[183] C. Sur, C. D. Jung, Y. Park, and K. H. Rhee, "Chosen-ciphertext secure certificateless proxy re-encryption," in *IFIP International Conference on Communications and Multimedia Security*, pp. 214–232, Springer, 2010.

[184] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Annual international cryptology conference*, pp. 213–229, 2001.

[185] M. Ito, A. Saito, and T. Nishizeki, "Secret sharing scheme realizing general access structure," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 72, no. 9, pp. 56–64, 1989.

[186] A. Beimel *et al.*, *Secure schemes for secret sharing and key distribution*. Technion-Israel Institute of technology, Faculty of computer science, 1996.

[187] M. Karchmer and A. Wigderson, "On span programs," in *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pp. 102–111, IEEE, 1993.

[188] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *International Conference on Information Security Practice and Experience*, pp. 13–23, Springer, 2009.

[189] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Crypto*, pp. 537–554, 1999.

[190] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, pp. 850–855, 2011.

[191] B. Lynn, *On the implementation of pairing-based cryptosystems*. PhD thesis, 2007.

[192] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[193] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing privacy enhancing technologies*, pp. 46–66, Springer, 2001.

[194] R. Dingledine, M. J. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Designing Privacy Enhancing Technologies*, pp. 67–95, Springer, 2001.

[195] J. Shen, T. Zhou, X. Chen, J. Li, and W. Susilo, "Anonymous and traceable group data sharing in cloud computing," *IEEE Transactions on Information Forensics and Security*, pp. 912–925, 2017.

[196] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "Fairaccess: a new blockchain-based access control framework for the internet of things," *Security and Communication Networks*, pp. 5943–5964, 2016.

[197] G. Zyskind, O. Nathan, *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*, pp. 180–184, IEEE, 2015.

[198] J. Liu, X. Li, L. Ye, H. Zhang, X. Du, and M. Guizani, "Bpds: A blockchain based privacy-preserving data sharing for electronic medical records," in *GLOBECOM*, pp. 1–6, IEEE, 2018.

[199] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018.

[200] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*, pp. 53–65, Springer, 2002.

[201] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, pp. 68–72, 2003.

[202] D. D. F. Mazières, *Self-certifying file system*. PhD thesis, Massachusetts Institute of Technology, 2000.

[203] D. Mazieres and M. F. Kaashoek, "Escaping the evils of centralized control with self-certifying pathnames," in *Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications*, pp. 118–125, 1998.

[204] "Filecoin: A decentralized storage network,"

[205] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *TCC*, pp. 253–273, Springer, 2011.

[206] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *EUROCRYPT*, pp. 146–162, Springer, 2008.

[207] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International conference on the theory and applications of cryptographic techniques*, pp. 506–522, Springer, 2004.

[208] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," in *Annual international cryptology conference*, pp. 205–222, Springer, 2005.

[209] J. Lai, R. H. Deng, and Y. Li, "Fully secure cipertext-policy hiding cp-abe," in *ISPEC*, pp. 24–39, Springer, 2011.

[210] F. Laguillaumie and D. Vergnaud, "Designated verifier signatures: anonymity and efficient construction from any bilinear map," in *SecureComm*, pp. 105–119, Springer, 2004.

[211] M. Qu, "Sec 2: Recommended elliptic curve domain parameters," *Certicom Res., Mississauga, Canada, Tech. Rep. SEC2-Ver-0.6*, 1999.

[212] https://github.com/vanhoanHoang/Privacy-Preserving-Data-Sharing-Platform.

[213] D. M. Freeman, "Converting pairing-based cryptosystems from composite-order groups to prime-order groups," in *EUROCRYPT*, pp. 44–61, Springer, 2010.

[214] J. W. Byun, I. R. Jeong, D. H. Lee, and C.-S. Park, "Password-authenticated key exchange between clients with different passwords," in *International Conference on Information and Communications Security*, pp. 134–146, Springer, 2002.

[215] C.-L. Lin, H.-M. Sun, and T. Hwang, "Three-party encrypted key exchange: attacks and a solution," *ACM SIGOPS Operating Systems Review*, vol. 34, no. 4, pp. 12–20, 2000.

[216] C. Garman, M. Green, and I. Miers, "Decentralized anonymous credentials.," in *NDSS*, Citeseer, 2014.

[217] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *International Conference on Security in Communication Networks*, pp. 268–289, Springer, 2002.