DOCTORAL SCHOOL
MEDITERRANEA UNIVERSITY OF REGGIO CALABRIA

DEPARTMENT OF INFORMATION ENGINEERING, INFRASTRUCTURES
AND SUSTAINABLE ENERGY
(DIIES)

PHD IN
INFORMATION ENGINEERING

S.S.D. ING-INF/05
XXXIV CYCLE

BALANCING THE DEMANDS OF ANONYMITY, PRIVACY, AND ACCOUNTABILITY: CHALLENGES AND NEW SOLUTIONS IN VARIOUS APPLICATION CONTEXTS

CANDIDATE
ANTONIA RUSSO

ADVISOR
Prof. GIANLUCA LAX

COORDINATOR
Prof. TOMMASO ISERNIA

REGGIO CALABRIA, April 2022

Finito di stampare nel mese di **Aprile 2022**

Edizione csdA Centro Stampa d'Ateneo

Quaderno N. 54

Collana *Quaderni del Dottorato di Ricerca in Ingegneria dell'Informazione*
Curatore *Prof. Tommaso Isernia*

ISBN 978-8-89-935264-6

Università degli Studi *Mediterranea* di Reggio Calabria
Salita Melissari Feo di Vito. Reggio Calabria

ANTONIA RUSSO

BALANCING THE DEMANDS OF ANONYMITY, PRIVACY, AND ACCOUNTABILITY: CHALLENGES AND NEW SOLUTIONS IN VARIOUS APPLICATION CONTEXTS

The Teaching Staff of the PhD course in

INFORMATION ENGINEERING

consists of:

Tommaso ISERNIA (coordinator)
Pier Luigi ANTONUCCI
Giuseppe ARANITI
Francesco BUCCAFURRI
Salvatore COCO
Giuseppe COPPOLA
Mariantonia COTRONEI
Lorenzo CROCCO
Dominique DALLET
Claudio DE CAPUA
Francesco DELLA CORTE
Giuliana FAGGIO
Fabio FILIANOTI
Patrizia FRONTERA
Sofia GIUFFRÈ
Giorgio GRADITI
Voicu GROZA
Antonio IERA
Gianluca LAX
Aimè LAY EKUAKILLE
Giacomo MESSINA
Antonella MOLINARO
Andrea Francesco MORABITO
Giacomo MORABITO
Rosario MORELLO
Domenico ROSACI
Giuseppe RUGGERI
Mariateresa RUSSO

To my previous and future generations

Contents

Part I Background

Part II Anonymity and Digital Identity

Part IV Privacy and Social Networks

Part V Final Conclusions

List of Figures

Listings

1

Introduction

Over the years, digital transformation and innovation have changed our connection with work, sociability, progress, and life, making up new daily habits and processes. Moreover, the recent pandemic has accentuated and accelerated the need to transfer to the online world communication, employment and collaboration with co-workers and those close. Nevertheless, above all, these challenging times drove the need to be aware of our digital identity as a reflection of our physical identity as subjects within online communities and networks, as suggested by the eIDAS Regulation (Electronic Identification Authentication and Signature) [86].

The European Regulation aims to ensure full interoperability in the Member States for electronic signatures, identification, and authentication services. It gives European citizens the possibility to access online services of other EU countries (university services, banking, public administration services, other online services) using the same credential. The eIDAS principles are based on the security, trust, and interoperability of electronic services carried out by citizens all over EU countries [187].

As human beings, the concept of security in a broad sense, and sometimes also conceived as safety, is one of the priorities we aspire to in life through the undertaken decisions and actions. However, security can have different declinations in the online world: for instance, security can refer to ensuring users' data anonymity, or guaranteeing privacy of sensitive data, or allowing accountability when users access online services, communicate within social networks, carry out business processes, or exchange data [212]. In particular, *anonymity* is the condition in which any (direct or indirect) identifiers of individual subjects are unknown. Instead, *privacy* refers to an individual's control over activities in keeping them hidden and exclusive to the individual, even though everyone is aware of their identity. Finally, *accountability* is defined as the possibility to account the responsibility for the actions and behaviour of users who use a computer system.

This thesis analyses the needs for anonymity, privacy, and accountability in various applications and proposes new solutions to balance these properties among them from three different perspectives, which are:

1. anonymity and digital identity;
2. privacy and accountability;
3. privacy and social networks.

These perspectives aim to balance benefits deriving from being connected and the need to protect our digital identity and online actions and decisions. In particular, these three concepts seem antagonistic and contrast with each other. However, there is a need to find adequate balancing in each of the properties. These needs also vary within the different application contexts that will be presented in the following chapters.

The concepts of anonymity, privacy, and accountability are well-known and have been recently remarked also by the issuance of the General Data Protection Regulation (GDPR) [104], the new European Union privacy law that puts guidelines and regulations on how data have to be processed, used, stored, or exchanged to protect and ensure individuals data privacy [240]. Within this regulation, seven essential data protection principles have been established: lawfulness; fairness and transparency; purpose limitation; data minimization; accuracy; storage limitation; security; accountability.

This regulation was issued to respond to the various attacks that have targeted online users' sensitive information and also to the unclarity of social networks' privacy settings [68]. In fact, one of the most recent attacks affected 700 million LinkedIn users. Their data have been reportedly advertised on the dark web in June 2021, revealing information relating to real accounts, including users' full names, email addresses, phone numbers and physical addresses [255]. Alternatively, in 2018, the discovery that Facebook gave access to the personal data of more than 87 million users to Cambridge Analytica fueled interest in the risks of privacy violations [133]. These occurrences alarmed organizations to pay attention to the GDPR rules established to ensure data protection in every data life cycle of the company itself.

In the end, as usual, we will see that two main worlds must converge at once: innovation and security. On the one hand, the rapid progress increases the demand and performance standards of services in terms of availability, diversification and delivery. However, on the other hand, the security properties of such services must be guaranteed to create safe, shareable and healthful environments.

In the following, we will discuss each of the proposed perspectives with regard to various application contexts.

1.1 Anonymity and Digital Identity

The starting point of our study is related to one of blockchain technology's strengths: the anonymity of users [193]. The rapid development of this technology made possible its exploitation in many real-life scenarios [239]. However, there are situations where having a reference to certain identity is necessary to enable accountability, trust and transparency of actions performed over the blockchain.

This consideration introduces a new perspective of anonymity over the blockchain. In particular, we propose a solution that integrates an eIDAS-compliant digital identity with the blockchain via Identity-Based Encryption. The proposed approach aims to create a direct link between the pair of cryptographic keys used to sign and verify a blockchain transaction and the user's digital identity. This approach is a primitive that can be exploited to add further functionalities to several real-life applications.

A first use of the proposed primitive is to enable transactions and contracts among secure digital identities over Ethereum. Therein, we consider the case in which cryptocurrencies and tokens are transferred among verified users. After users link their identities to their blockchain addresses, a suitably-developed dedicated smart contract is in charge of verifying the validity of such transactions and authorizing resources' transfers.

A second use of the primitive is proposed for remote clinical services. In that regard, a fog middleware that provides end-users with the advantages of mobility, low latency and location awareness is exploited [105, 278]. Each user is provided with an IoT medical device used to monitor and analyze clinical parameters, and the closest fog server to the user elaborates such data. The proposed approach guarantees secure identification and authentication of patients while supporting anonymity and unlinkability.

The last use of the proposed primitive is related to the GDPR Regulation, which states some recommendations about the “right to be forgotten" that consists of obtaining the erasure of subjects' personal data from the controller. We propose a blockchain-based scheme that allows users to control the personal data revealed when accessing a service. Furthermore, the proposed solution provides mechanisms for revoking the authorization to access a service and for guessing the identity of a user only in cases of need.

1.2 Privacy and Accountability

The second aspect that we investigate links the needs derived from guaranteeing the privacy and still accountability of online actions. Indeed, accountability could seem in contrast to the growing demand for privacy. However, accountability is often necessary in the context of access control, where the main security research challenges consist in guaranteeing the anonymity of a user accessing an online service and yet disclosing the identity of a user who accessed an online service in case of need.

We propose an access control scheme that provides anonymity, access unlikability, and accountability. The system is based on a public blockchain and relies on identity and access control providers. Any user exploits different blockchain addresses to interact with the involved entities. The used blockchain addresses are linked to each other. Every entity can verify the transactions generated by the users by using the blockchain because all information needed to implement access control is publicly available on the blockchain.

A similar need occurs when dealing with sensitive medical data, where security and privacy issues arise, mainly related to unauthorized access to e-health records, especially when different healthcare organizations maintain records. A blockchain-based solution allows e-health record sharing by granting access only to authorized entities. This proposal relies on a public blockchain representing an entity that can offer a proper trust level of the entire system to patients and provides the needed automatism to the different phases. Furthermore, the exploitation of blockchain can avoid the linkage between a patient's identity and e-health records.

Also, in an energy trading scenario, many privacy and security concerns arise from energy producers and consumers [112]. We focus on an Ethereum-based solution for energy trading, assuring both the accountability of energy transactions and users' privacy. During energy-based transactions, users' privacy and data confidentiality should be considered. Through a smart contract, we implemented a protocol that achieves the authentication of users and allows the tracking of energy-driven transactions logged and stored in a public blockchain. The main benefit of smart contracts over Ethereum is that different parties with conflicting interests can exchange value without trusting each other. However, accountability is still required: in case of need, the customer's identity is linked to the service delivered and communicated to the appropriate parties.

The same concerns arise in the context of attribute-based service delivery, where we mix the power of attribute-based encryption schemas and blockchain technology to provide consumers' privacy and accountability. In particular, a blockchain-based solution integrates the features of smart contracts with an Attribute-Based Encryption scheme. As a result, the solution allows users to securely access services only based on attributes without disclosing their identity to the service suppliers.

1.3 Privacy and Social Networks

The third research problem we analyze combines privacy over social networks and, at the same time, the exploitation of social networks to build privacy-preserving solutions when accessing online services. Indeed, the availability of a massive amount of personal information has raised privacy concerns for online users [78, 218, 258]. Moreover, they may be not well informed about data they share on the Internet and, therefore, about their privacy choices.

We deal with an emerging issue related to the user's privacy settings in social networks. Until now, all privacy settings are managed only by social networks, which may change the settings without the user's conscious consent. For this reason, we propose a solution exploiting blockchain technology to store the privacy settings and to verify them at any moment and in a transparent way. A smart contract deployed on the blockchain determines whether the privacy settings assigned to the user by the social network are compliant with those declared in advance by the user. This solution is compliant with the GDPR Regulation achieving accountability and allowing a social network to prove the correct management of the user's privacy choices. These choices are not self-certified by the social network and instead stored in a decentralized way on the blockchain that guarantees the integrity and authenticity of data.

Another relevant consideration regarding privacy is the "data minimization principle". According to the GDPR, every online service should implement this principle. When accessing an online service, users must authenticate to prove their real identities. Responding to the logic of selective disclosure of attributes [257], we propose a solution based on using a social network in charge of providing users with the means of issuing and verifying claims and credentials. The solution allows users to control the information shared with the service and attribute providers. In particular, the user shares the credentials on the social network in a secure way. The service provider can verify the credentials by the social network as a secure and transparent repository of the selected and hidden information related to the users' credentials.

There is also a privacy problem in the personalization of advertisements in social networks. Indeed, users are willing to see advertisements that might interest them, instead of general and not personalized advertising. In these cases, the social network is the only owner of the user's interest. However, this information is an important asset that cannot be shared, so a third party can't ensure that a social network has sent advertising only to interested users. We face this problem by proposing a technique allowing a social network to prove to a third party that a user reached by the advertising is interested in that.

1.4 Outline of the thesis

The thesis is divided into four main parts. First, in the next part, we provide essential background information used in the entire thesis. Specifically, we present some cryptographic primitives and the fundamentals of blockchain technology, concepts exploited in several solutions. Then, we highlight the notion of digital identity focusing on two recognized regulations: GDPR and eIDAS.

Part II focuses on a new perspective of anonymity over the blockchain and contains new models to allow users complete control over their data by exploiting their digital identities. In Chapter 4, we propose a model that enables the binding of both sender and receiver of a blockchain transaction to a public digital identity. Chapter 5 presents a solution for performing Ethereum transactions among secure digital identities not yet registered to a blockchain-based system. Chapter 6 presents a system that allows users to prove the possession of some attributes without disclosing their whole identities while guaranteeing a certain degree of anonymity. Finally, in Chapter 7, we propose a solution that allows a company to exploit the advantages of fog computing, keeping compliance with the GDPR.

Part III is devoted to privacy-preserving solutions that simultaneously consider the accountability requirements to guarantee access to confidential information only to authorized entities. In Chapter 9, we face some relevant security challenges in the context of access control by proposing an access control scheme relying on blockchain technology. Chapter 10 defines a solution that integrates the features of Ethereum's smart contracts with an Attribute-Based Encryption scheme applied to a real-life scenario of service delivery. In Chapter 11, we propose a solution for energy trading in smart grids based on Ethereum blockchain and smart contracts that meets accountability and privacy requirements. Finally, Chapter 12 focuses on a solution that allows the sharing of health records yet guaranteeing access only to authorized entities and avoiding the linkage between patient's identity and e-health records.

Part IV shows different perspectives on privacy and social networks. In Chapter 13, we investigate the consequences of a social network's possible misbehaviour regarding users' privacy settings. Next, in Chapter 14, we propose a new solution for the management of personal data that is based on the use of a social network in charge of providing users with the means of issuing and verifying claims and credentials for accessing online services. Finally, Chapter 15 explores an approach to match user's preferences and advertising campaigns in such a way that the social network can prove this matching.

Finally, in Part V, we draw our conclusions.

Part I

Background

This section introduces some basic concepts frequently used in this thesis. First, we present some cryptographic primitives exploited in several solutions of this thesis. Then, we deepen fundamentals of the blockchain technology, explaining the main categories and features. Then, we focus on the concept of digital identity. Finally, this section's end provides some information on two recognized regulations: GDPR and eIDAS.

2

Introduction

The solutions proposed in this thesis make extensive use of cryptographic primitives, which are described in the following. Furthermore, many of the proposed solutions rely on the blockchain technology, digital identity, and GDPR and eIDAS regulations, which are the topics of the subsequent sections.

2.1 Cryptographic primitives

The first primitive we describe is the hash function, which is a one-way function that receives an input and returns a bit string with a fixed length, called `digest`. The security and reliability of a hash algorithm lie in the fact that the function is not invertible. Furthermore, it must never be possible to create two different messages with the same digest intentionally. Well-known cryptographic hash functions are SHA-1, SHA-256, RIPEMD-160. For example, SHA256 [196] is a cryptographic hash developed by National Security Agency (NSA) and returns a 256-bit digest, whereas RIPEMD160 [222] is a cryptographic hash designed in the open academic community and returns a 160-bit digest.

A blockchain address is generated by applying these functions. In particular, each blockchain user needs to have a private and a public blockchain key. The private key is a randomly generated 256-bit string. The public key is generated by the private one using a cryptographic function named *elliptic curve point multiplication*. In particular, the used algorithm is Curve Digital Signature Algorithm (ECDSA) and the elliptic curve is `secp256k1` [185]. The blockchain address of a user is computed from the public key by applying the SHA-256 respectively, and RIPEMD-160 [61].

Data encryption is a security method where information is encoded and can only be accessed by users possessing the correct encryption key. This method is often applied in two different forms, symmetric or asymmetric cryptography. Symmetric cryptography is a technique that exploits a single and secret cryptographic key to encrypt and decrypt data. All parties involved in the communication have to exchange the key used to encrypt the data before decrypting it. This fact sometimes could be a disadvantage of this technique. The most widely used symmetric algorithms are AES-128, AES-192, and AES-256 [97]. Asymmetric cryptography is based on the use of a public and private key for each user. Public keys are typically arranged by a Public Key Infrastructure, which binds public keys with the respective identities of entities (like people and organizations) through a process of registration and issuance of certificates by a certificate authority (CA). However, there are cases in which predistribution of keys is inconvenient or infeasible due to technical restraints: in these situations, Identity-based Encryption is a solution [57].

Identity-based Encryption (IBE) [2] allows any party to generate a public key from a known identity value (for example, an e-mail address). A trusted third party, called the Private Key Generator (PKG), generates the corresponding private key. To operate, the PKG first publishes a master public key and retains the corresponding master private key (referred to as master key). Given the master public key, any party can compute a public key corresponding to identity by suitably combining the master public key with the identity value. To obtain a corresponding private key, the party authorized to use the identity ID contacts the PKG, which uses the master private key to generate the private key for the identity ID. As a result, parties may encrypt messages (or verify signatures) with no prior distribution of keys between individual participants once their identity is known and well-defined. However, to decrypt or sign messages, the authorized user must obtain the appropriate private key from the PKG by proving the possession of the true identity. The most used IBE systems have been proposed by Boneh-Franklin [49] and by Sakai–Kasahara [230].

2.2 Blockchain technology

Blockchain is a technology that could have the capacity and potency for enhancing and changing various aspects of economy and society, and for this reason, it can be considered as a disruptive technology [247]. It was proposed by [193], and it is defined as a distributed ledger that stores, in a transparent and immutable way, transactions executed among users. Information is stored inside blocks, whose number and dimensions are continuously growing. Every block is linked to the chain by its header. The header contains the hash of the previous block and a timestamp. The transactions that take place in blockchain are stored inside blocks and contain information on the recipient's public address, the characteristics of the transaction, and the cryptographic signature, which guarantees the integrity and authenticity of the transaction. Every operation has to be confirmed and validated by blockchain participants, and this concept is summed up by distributed consensus. This way, users can trust the system of the public ledger, without trusting a central authority or a third-party intermediary. It has been proved that perceived privacy in using blockchain positively could affect and influence users' trust and attitudes towards blockchain technology [236, 238]. Over time, blockchain assumed different meanings and definitions: the first one is called Blockchain 1.0, and it is referred to as the Bitcoin paradigm. This kind of system represents a platform in which it is possible running and deploying all the operations carried out with cryptocurrency in digital payment systems.

In the following, a survey on the most commonly used blockchain technologies is presented, highlighting their advantages and drawback.

Blockchain networks can be defined as permissionless or permissioned [120]. The former, also known as public blockchains, are widely used in the domain of cryptocurrencies and financial markets, instead, the latter, also known as private blockchains, have entered the domain of businesses applications and institutional practices. The key characteristics of permissionless blockchains are anonymity and full transparency of transactions over open source protocols. In contrast, permissioned blockchains are developed by private entities and for this reason the network transparency and participants' privacy is controlled by the organization itself. Hybrid blockchains are used in organizations requiring a private, permission-based system alongside a public permissionless system. This setting allows the organization to control access to data stored in the blockchain. Consortium or federated blockchains are similar to hybrid blockchains, but controlled and shared among multiple organizations.

The first version of a network implementing the blockchain technology, Bitcoin, was defined in [193] and allows us to replace a single centralized party managing a service with a distributed ledger of replicated, shared, and synchronized digital data spread across different servers. Data are saved in a growing list of records, called blocks, and each block contains a cryptographic hash of the previous block, a timestamp, and transaction data [62]. Blockchain can record transactions between two parties efficiently and in a verifiable and permanent way [127]: it is managed by a peer-to-peer network of node running a common protocol for validating blocks. Once saved, the data in a block cannot be modified without alteration of all previous blocks, which requires a too high power computation. Despite this kind of blockchain is widely spread for exchanging cryptocurrency by bitcoin transactions, it doesn't support the development of smart contracts.

Ethereum [269] is a Blockchain 2.0 that enables the possibility to create and run smart contracts, programs executed over the Ethereum computing infrastructure. It is considered the second largest and global cryptocurrency platform and is a permissionless blockchain where developing decentralized applications through the use of smart contracts. This platform can be considered a system that is globally shared and implementing a cryptographically secure transaction-based state machine [269]. A smart contract is defined as a piece of code verifying and enforcing conditions that stipulate a digital contract between parties that does not require a third intermediary. Smart contracts are written in Solidity, an object-oriented and high-level Turing complete programming language. It exists a practical and conceptual issue about the external data used to verify and perform decision inside smart contracts, that is the "oracle" presence in Ethereum. An oracle has the purpose of connecting decentralized applications with third-party services in a trust and secure way, in order to get data from outside blockchain and execute any API call preventing data integrity and authenticity. Provable [214] is the most famous oracle service for smart contracts and blockchain applications. It consists of three main entities: data-source, query, and oracle. When a smart contract requires data from a data source outside blockchain, it sends a query to Provable and calls a function passing the result of the query as an input. Provable aims to demonstrate that the data taken from the original data-source is authentic, by linking returned data with an authenticity proof document.

Among permissioned blockchains, Hyperledger Fabric [28] is an implementation of an open-source private blockchain running smart contracts and is intended to form a foundation for developing applications with a modular architecture. Being private, access to the network is restricted to selected participants. Hyperledger Fabric allows components, such as consensus and membership services, to be plug-and-play.

IOTA [211] networks were designed for IoT applications and are permissionless blockchain networks that are built on Tangle, a new data structure based on a directed acyclic graph, which does not need blocks, miners, or any chain. For this reason, IOTA transactions are free. Although this platform can yield better performance than Ethereum and Hyperledger blockchains, its primary limitation is that devices may not be not capable of performing the Proof of Work, resulting in a bottleneck when transactions occur [98].

EOSIO [90] is the first blockchain platform that uses the Delegated Proof of Stake consensus algorithm. Converse to traditional proof-of-work-based systems, EOSIO is public, permissionless, and suffers from serious attacks derived from exploiting vulnerabilities in DApps and leading to millions of dollars lost for EOSIO users, as discussed in references [124, 215]. MultiChain [110] is a platform enabling the creation and deployment of private blockchains to be developed and used inside organizations. The system administrator sets a series of user permissions to introduce controls over transactions and block size.

Chain Core [89] is another platform for private blockchains and is typically used to initiate and transfer financial assets based on permission from the blockchain infrastructure. Corda [52] is a distributed ledger platform for recording and enforcing business agreements among institutions. Chain Core and Corda also rely on smart contracts and allow participants to manage permissions. Open Chain [201] is an open-source distributed ledger technology based on the Partitioned Consensus: every Open Chain instance has one authority validating transactions. This platform aims to manage the digital assets of organizations in a scalable and secure way.

2.3 Digital identity

A digital identity is the core information about an individual, organization, application, or device that exists online. This term also denotes aspects of civil and personal identity. Furthermore, the entire collection of the information generated by a person's online activity is linked to her/his digital identity. Another similar definition given by ISO/IEC 24760-1 reports digital identity as a set of attributes related to an entity [3]. In this section, we briefly survey the main technologies related to digital identity. Open Authorization (OAuth) [198] is an open access delegation protocol used by users to provide a third party (typically a site or an application) with the ability to access their personal information registered on a site without providing them with credentials to access this site. This protocol is widely used, especially in social networks, by many big companies (examples are Facebook, Twitter, Google) to allow their users to share profile information with third parties. OAuth is designed to use the HTTPs protocol for communication and exploits the release to the third party of tokens by an authorization server, once the user approves the proxy. These tokens are used as credentials to access shared information. OpenID is another decentralized authentication protocol promoted by the OpenID non-profit foundation. By this protocol, a site administrator is supported in managing the users' authentication procedure, because no credential for user's login has to be stored. By OpenID, user access different sites with the same digital identity and password. In this protocol, the third party that handles authentication is the OpenID identity provider, while a site compatible with OpenID is called a relying party. The protocol is distributed among the identity providers and there is no central entity that manages authentication or decides who can act as a provider or identity provider. The first version of OpenID was published in 2005 by Brad Fitzpatrick, creator of the Live-Journal community and with the name Yadis (yet another distributed identity system). In 2007, Symantec included OpenID as a supported standard. In 2008, the OpenID 2.0 release was published and carried out by several major providers (Yahoo, Google, IBM, Microsoft, VeriSign, MySpace). The third and latest version, called OpenID Connect, was released in 2014.

Windows CardSpace [5] is a Microsoft software for digital identity management released in 2007. Born with the purpose of providing an environment robust against phishing attacks, CardSpace stores digital identities and provides a graphical interface for their management. When an application or a site needs to obtain information about the user, it generates a request for that information. The request is intercepted by CardSpace, which starts a graphical interface that shows the information stored and associated with that application or site. At this point, CardSpace contacts the digital identity provider to obtain the information to be shared, which is returned as a signed XML file, to guarantee its authenticity and integrity. In 2011, Microsoft registered a development of CardSpace, due to the technological changes and feedback received from partners and users. At the same time, Microsoft has shifted interest towards the U-Prove project. U-Prove is an advanced cryptographic technology, combined with identity solutions on existing standards, aimed to find a compromise to the eternal dilemma between identity and privacy guarantee with two important privacy-preserving features: (1) unlinkability and (2) selective disclosure of attributes.

2.4 GDPR and eIDAS

The General Data Protection Regulation (GDPR) [104] is a European Union privacy law that covers the protection of EU residents' personal data. As stated in [106], personal data is information that, directly or indirectly, can identify an individual. The GDPR remarks the importance of seven basic principles of data protection: 1) lawfulness, fairness, and transparency, 2) purpose limitation, 3) data minimization, 4) accuracy, 5) storage limitation, 6) security, 7) accountability.

Although how to apply these principles is not stated, they represent the spirit of the regulatory framework. Thus, compliance with these principles is fundamental to build any data-processing framework in practice. Indeed, this regulation emphasizes the importance of applying these principles to any company, and it is not possible to be GDPR-compliant without implementing these rules in the data life cycle of the company.

In some solutions proposed in this thesis, we focus on one third principle that regards data minimization (Article 5.1.c) and requires entities to process only *adequate, relevant, and limited personal data that is necessary*. This regulation does not define what the terms adequate, relevant, and limited means but states that data processing should only use as much data as is required to successfully accomplish a given task, and data collected for one purpose cannot be used for a different purpose without obtaining a new consent. This means that companies must limit personal data collection to data that are absolutely necessary for carrying out the purpose for which data are processed [156].

The Regulation (EU) N 910/2014 [86] on electronic identification and trust services for electronic transactions in the internal market provides a normative basis to enable secure electronic interactions between businesses, citizens, and public authorities.

eIDAS establishes a legal framework for electronic signatures, electronic seals, electronic time validations, electronic documents, certified electronic delivery services and services relating to authentication certificates for websites. Compared to electronic identification systems, the regulation requires that each member state has to notify the electronic identification systems provided to citizens and companies for the purpose of mutual recognition. Thanks to the principle of mutual recognition and reciprocal acceptance of interoperable electronic identification schemes, eIDAS wants to simplify the use of electronic authentication against public administrations, both by companies and by citizens. The regulation aims to create a level playing field for trust service providers who are currently operating in a context where differences in national laws in the various Member States are a source of legal uncertainty and additional burdens.

All Member States have to notify their eID schemes (national electronic identification schemes) to the European Commission, which are published in the Official Journal of the European Union. Both people and companies can access public services provided by an EU Member State using the eID of another EU Member State: this concept aims at promoting cooperation between states. Interoperability between different eID-schemes is reached by defining the interfaces between eIDAS-Nodes. However, the eID ecosystem consists of various actors that should be available in all EU countries: the most important is the node operator, which controls that an eID node behaves correctly and implements the function of the connection point between the attribute provider, the identity provider and the service provider.

The public digital identity is recognized by law in a Country or at international level making the basis for non-repudiable accountable applications. There is a concrete instantiation of this notion in the European Union. Indeed, it is based on the eIDAS Regulation.

The European Regulation establishes electronic recognition procedures common to all EU countries: so digital services become European. Moreover, it defines common rules guaranteeing full interoperability at Community level not only for certified electronic signature tools but also for citizens' web identification and for third-party services (e.g. electronic seals, time validation, electronic delivery service). Each Member State maintains it own electronic identification systems, which have to be accepted by all other member states. For example, Italy has notified to the EU Commission the institution of SPID, the Italian public system for the management of the digital identity of citizens and businesses [87].

Estonia, for example, has already notified its eID scheme to the European Commission. Estonia has long-term experience in using electronic authentication, and in the technical document about the Estonian eID scheme [66], the digital certificate of identity concept is highlighted and treated deeply.

On 22nd August 2017, the Federal Republic of Germany has notified the German eID scheme in accordance with the eIDAS Regulation to the European Commission. The German eID [99] is designed to provide security and trust during the identification process with two general purposes: the German identity cards and German resident permits. Mutual authentication between the chip of the eID card and the relying party is guaranteed, adding a secure, protected channel for direct communications.

The National Identification and Authentication System in Croatia (NIAS) [42] is the central identification and authentication system for e-services through all the country. The basic function of NIAS is guaranteeing electronic identification and secure authentication of users for e-services. NIAS distinguishes three entities: issuers of electronic credentials, providers of e-services, and users of e-services.

Portugal [91] has pre-notified three eID schemes: the national eID card, the mobile eID solution, and the Professional Attributes Certification System. The Identity Card is a smartcard-based eID combining four identification numbers (i.e., fiscal, social, health, and civil ID), replacing paper-based ID cards.

In Italy, the Public System used for the management of digital identity, named SPID [87], has been designed in compliance with eIDAS Regulation, and it allows the access to online services of the public sector with a single credential set. A user can use SPID credentials for education, for public administration services, for the health system, and many other services. There is a high number of services enabled by SPID, and nowadays, they are growing in different online areas. In general, an eIDAS-compliant eID offers various advantages related to the secure cross-border authentication through different current eID schemes in Europe. The eIDAS key benefits are interoperability, also on the legal side, and security and trust, because of the validity of transactions made across borders.

Part II

Anonymity and Digital Identity

The concept of anonymity appears very often in the IT world as a possibility to keep one's actions separate from one's identity. Over the years, anonymity has become a cornerstone of blockchain technology, allowing millions of blockchain users to execute transactions without any reference to their real identities. More precisely, the blockchain's anonymity turns into the concept of pseudo-anonymous, as users exploit an alias (i.e., their blockchain address) to create transactions. This alias could be valuable and allow the actions' linkability: an adversary may not know who a user is are but can still attribute activities to them. In opposition, new and revolutionary applications can exploit the anonymity. Indeed, the potential of blockchain technology opened several challenges in many real-life scenarios. For example, referencing a specific identity in identity-aware applications is necessary to enable accountability, trust, and transparency of actions performed over the blockchain. Often, when accessing an online service, users must perform authentication to prove their real identities. However, in some cases, the grant of services could be based on the disclosure of simple subject's characteristics [122] (e.g., being of age). In fact, in these applications, new paradigms are emerging. The aim is to reveal only the useful information of one's identity in line with the GDPR principle of data minimization. In this part of the thesis, we will look at a new perspective of anonymity over the blockchain and propose new models to allow users' complete control over their data and promptly manage the attributes necessary to perform online actions.

For this purpose, in Chapter 4, we propose a model that enables the binding of both sender and receiver of a blockchain transaction to a public digital identity. The solution exploits eIDAS-compliant identification schemes for handling public digital identities and Identity-based Encryption (IBE) for associating a digital identity with a public key. The research is published in [61], and to the best of our knowledge, it was the first attempt to create a non-anonymous blockchain, which can be used in all cases in which the author of a transaction has to be identified with certainty and legal effect.

An instantiation of this model is presented in Chapter 5 to enable Ethereum transactions among secure digital identities not yet registered to a blockchain-based system. We consider the transfer of cryptocurrencies and tokens implemented by a suitably-developed dedicated smart contract, in charge of verifying the validity of such transactions and authorizing resources' transfers. The solution's feasibility is assured by the concurrent role of the IBE's Private Key Generator acting as a service provider of the public digital identity system, as stated in [57].

Chapter 6 focuses on the data minimization of users' personal information when accessing an online service. Indeed, in compliance with the GDPR [104], we propose a system that allows users to prove the possession of some attributes without disclosing their whole identities, while guaranteeing a certain degree of anonymity. Using the most useful features of blockchain, the proposed system provides suitable mechanisms for revocation and accountability of such attributes. The proposed scheme allows a company to comply with the data minimization principle stated by GDPR, yet ensuring that access-control policies are respected. These aspects highlight the practical importance of this research, which has been published in [228].

The data minimization principle discussed above becomes crucial in the healthcare scenario: it limits data processing to only data that are necessary in relation to the purposes for which they are processed. Indeed, in many applications, knowing the identity of users or linking different accesses of the same user do not respect the data minimization principle. In Chapter 7, we address two common privacy issues of a solution based on fog computing: a fog server should not know the identity of the user, and it should be guaranteed the unlinkability of user's accesses to the same fog server in different moments. We propose a solution that allows a company to exploit the advantages of fog computing, keeping the compliance with the GDPR. This research has been published in [60].

3

Related Work

In this section, we survey the state-of-the-art proposals related to the main challenges faced in this part of the thesis. First, we focus on the existing identity-aware applications exploiting blockchain technology. Then, we discuss some proposals addressing the need for data secure authorization and access in several contexts. In [17], the authors review applications relying on blockchain. They highlight the potential benefit of such technology in manufacturing supply chain and a vision for the future blockchain ready manufacturing supply chain is proposed.

Indeed, blockchain has started to become the technical core of cryptocurrency, access control systems, asset management, banking, e-voting, etc. [207, 45, 76], thanks to the assurance of authenticity and uniqueness of transactions.

The paper [149] states that digital supply chain integration is becoming increasingly dynamic. Access to customer demand needs to be shared effectively, and product and service deliveries must be tracked to provide visibility in the supply chain. Business process integration is based on standards and reference architectures, which should offer end-to-end integration of product data. The authors of this study investigate the requirements and functionalities of supply chain integration, concluding that cloud integration can be expected to offer a cost-effective business model for interoperable digital supply chains. Moreover, they explain how supply chain integration through the blockchain technology can achieve disruptive transformation in digital supply chains and networks.

In [134], the authors highlight that the need for blockchain-based identity management is particularly noticeable in the Internet age, as we have faced identity management challenges since the dawn of the Internet. They observe that blockchain technology may offer a way to circumvent this problem by delivering a secure solution without the need for a trusted, central authority. It can be used for creating an identity on the blockchain, making it easier to manage for individuals, giving them greater control over who has their personal information and how they access it. The proposed solution stores users' encrypted identity, allowing them to share their data with companies and manage it on their own terms.

In [56], the authors focus on Public Digital Identity System (SPID), the Italian government framework compliant with the eIDAS regulatory environment. They observe that a drawback limiting the real diffusion of this framework is that, despite the fact that identity and service providers might be competitor private companies, SPID authentication results in the information leakage about the customers of identity providers. To overcome this potential limitation, they propose a modification of SPID to allow user authentication by preserving the anonymity of the identity provider that grants the authentication credentials. This way, information leakage about the customers of identity providers is fully prevented.

The paper [252] focuses on pseudonymisation, a concept that was only recently formally introduced in the EU regulatory landscape. In particular, it attempts to derive the effects of the introduction of pseudonyms (or pseudonymous credentials) as part of the eIDAS Regulation on electronic identification and trust services and, ultimately, to compare them with the effects of pseudonymisation within the meaning of the General Data Protection Regulation (the GDPR). The paper examines how eIDAS conceives pseudonymisation and explains how this interpretation would translate in practical uses in the context of a pan-European interoperability framework.

In [55], an advanced electronic signature protocol that relies on a public system for the management of the digital identity is proposed. This proposal aims at implementing an effective synergy to provide the citizen with a unique, uniform, portable, and effective tool applicable to both authentication and document signature.

In [46], the authors propose a security framework that integrates the blockchain technology with smart devices to provide a secure communication platform in a smart city. The authors observe that, despite a number of potential benefits, digital disruption poses many challenges related to information security and privacy.

In [79], the authors explore an environment in which in-store customers supplement company drivers can take on the task of delivering online orders on their way home. The results of their computational study provide insights into the benefits for same-day delivery of this form of crowdshipping, and demonstrate the value of incorporating and exploiting probabilistic information about the future.

The study carried out in [183] highlights that passengers and freight mobility in urban areas represents an increasingly relevant component of modern city life. On one side, it fosters economic growth, but, on the other, it also generates high social costs. Congestion and pollution are two problems policy-makers want to curb adopting appropriate measures. In this context, [183] analyses the feasibility and behavioral levers that might facilitate the diffusion of crowdshipping in urban areas. Two are the main objectives of the paper. The first is to investigate under which conditions passengers would be willing to act as crowdshippers. The second is to find out under which conditions people would be willing to receive their goods via a crowdshipping service. Crowdshipping can generate positive impacts, such as the reduction of total and ad-hoc trips, by optimizing, through sharing, the use of resources and infrastructures. This study focused on University students, show that 87% of students would, in principle, be willing to act as crowdshippers (i.e. supply) with an adequate compensation, while 93% of them are willing to receive their goods through a crowdshipping system (i.e. demand) under certain conditions, especially characterized by delivery timing and punctuality.

The authors of [206] provide an overview of the blockchain technology and its potential to disrupt the world of banking through facilitating global money remittance, smart contracts, automated banking ledgers and digital assets. In this regard, they provide a brief overview of the core aspects of this technology, as well as the second-generation contract-based developments. From there, their work enforces key issues that must be considered in developing such ledger based technologies in a banking context.

The paper [37] provides a high-level understanding of how blockchain technology will be a fundamental tool to improve supply chain operations. It illustrates theoretical and conceptual models for use of open blockchain in different supply chain applications with real-life practical use cases as is being developed and deployed in various industries and business functions.

The authors of [263] propose an overview of what smart contracts are and what are their main challenges for the future. In particular, they state that smart contracts have three main characteristics: *(i)* autonomy, *(ii)* self-sufficiency and *(iii)* decentralization. Autonomy means that the contracts and the initiating agents do not need to be in further contact. Self-sufficient means that smart contracts are able to raise funds by providing services and spending them when needed. Furthermore, smart contracts are decentralized as they do not are valid on a single centralized server, but they are distributed and self-executed across network nodes. As they can be seen as a distributed application, they have to face almost the well-known challenges of them, such as the reentrancy vulnerability, the privacy issues, how to guarantee the reliability of external information, and so on.

Another topic related to our proposal regards digital identity, in which we can find a rich literature. Bitnation [8] is the world's first Decentralized Borderless Voluntary Nation (DBVN). Bitnation started in July 2014 and hosted the first blockchain for refugee emergency ID, marriage, birth certificate, World Citizenship and more.

The website proof-of-concept, including the blockchain ID and Public Notary, is used by tens of thousands of Bitnation Citizens and Embassies around the world.

In [22], the authors proposed SCPKI, an alternative PKI system based on a decentralized and transparent design using a web-of-trust model and a smart contract on the Ethereum blockchain, to make it easily possible for rogue certificates to be detected when they are published. The web-of-trust model is designed such that an entity or authority in the system can verify fine-grained attributes of another entity's identity, as an alternative to the centralized certificate authority identity verification model.

The paper [73] argues that existing laws, specifically the federal Electronic Signatures in Global and National Commerce Act ("ESIGN") and state laws modeled on the Uniform Electronic Transaction Act ("UETA"), render blockchain-based smart contracts enforceable and therefore immediately usable.

The study [180] deals with a new approach to access control based on blockchain technology. The policies that express the right to access a resource are published inside blockchain. That way, every user can check if policies and resources match. Considering a blockchain, its capabilities of transparency and immutability allow a distributed consensus and auditability preventing a party from denying the rights granted by the policy.

In [108], a new cryptographic system for fine-grained access control of shared encrypted data, called Key-Policy Attribute-Based Encryption (KPABE), is developed. In this system, ciphertexts are matched with sets of attributes, and private keys are associated with access structures.

The concept of Ciphertext-Policy Attribute-based Encryption is formalized in [44]. In this solution, the policy is associated with the ciphertext and the attributes with the key. A user can decrypt a ciphertext if the user's attributes pass through the ciphertext's access structure.

The authors of [34] propose a solution combining CP-ABE with KP-ABE and consider a scheme in which both policy and attributes are associated with the ciphertext and key. The attributes are related to the ciphertext, and the policy designs the users who can decrypt. The policy states the kind of ciphertext the user can decrypt.

Attribute-Based Access Control (ABAC) is defined as logical access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes. The study [121] defines ABAC to understand the real applications of this mechanism. Attribute-Based Access Control is analyzed in real use cases to improve scalability, feasibility, and performances of applications in which the information sharing within and between organizations is expected.

The authors of [229] propose the concept of Attribute-Based Encryption. Authors provide an original type of Identity-based Encryption in which the identity consists of an attribute set. Users, with an identity and their attributes, can decrypt a ciphertext encrypted with the same attributes.

The Attribute-based proxy re-encryption scheme (ABPRE) [169] extends the ABE scheme empowering users with delegating capability in the access control environment. A proxy can be chosen by users to re-encrypt a ciphertext related to a specific access policy.

Concerning the revocation of attributes, the study [126] contains a solution exploiting ciphertext-policy attribute-based encryption for an access control mechanism. The solution is related to an efficient implementation provided with an attribute and user revocation capability.

Attribute-based encryption has always been considered as a technology for solving the problem of data privacy and fine-grained access control in traditional cloud storage systems based on a centralized storage architecture. The development of blockchain technology allows the building of a decentralized storage mode that could overcome the problem of a single point of failure in traditional cloud storage. The authors of [261] propose a framework that combines the Ethereum blockchain and ABE technology to implement data storage and sharing scheme for decentralized storage systems.

Blockchain technology can provide patients with immutable records regarding their medical data, said Electronic Health Records (EHRs). In [114], an attribute-based signature scheme with various authorities is presented to enforce the validity of EHRs stored in a blockchain. This system allows patients to possess the control of generating, managing, and sharing EHRs with other authorized data consumers in a secure environment.

In a cloud computing environment, service providers can be allowed to take care of confidential data, and this permission may raise potential security and privacy issues [259]. The cloud service provided, by adopting an encryption system, has to support fine-grained access control and also provide high performance and scalability. The authors propose a scheme to help companies use cloud servers to share confidential data efficiently.

A new architecture for access control in the IoT is provided in [210]. This framework, based on blockchain technology, overcomes the FairAccess [203] and other issues derived from the architecture, evaluating a new decentralized and authorization process for authorization in IoT environments. FairAccess is an authorization management framework that is fully decentralized and privacy preserving and uses blockchain as a decentralized access control manager. With the help of blockchain, this solution introduces various and new types of transactions to delegate, revoke, and grant access.

4

Integrating Digital Identity and Blockchain

Blockchain is a recent technology whose importance is rapidly growing. One of its native features is pseudo-anonymity, since users are referred by (blockchain) addresses, which are hashed public keys with no link to real identities. However, when moving from the use of blockchain as simple platform for cryptocurrencies to applications in which we want to automatize trust and transparency, in general, there is not the need of anonymity. Indeed, there are situations in which secure accountability, trust and transparency should coexist (e.g., in supply-chain management) to accomplish the goal of the application to design. Blockchain may appear little suitable for these cases, due to its pseudo-anonymity feature, so that an important research problem is to understand how to overcome this drawback. In this solution, we address this problem by proposing a scheme that mixes the mechanism of public digital identity with blockchain via Identity-Based-Encryption. We define the solution and show its application to a real-life case study.

4.1 Introduction

Blockchain [193] is a recent technology used in many application contexts, such as financial services, industry 4.0, smart city, share trading. It was defined in [193] and allows us to replace a single centralized party managing a service with a distributed ledger of replicated, shared, and synchronized digital data spread across different servers. Data are saved in a growing list of records, called blocks, and each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. Blockchain can record transactions between two parties efficiently and in a verifiable and permanent way [127]: it is managed by a peer-to-peer network of nodes running a common protocol for validating blocks. Once saved, the data in a block cannot be modified without alteration of all previous blocks, which requires a too high power computation.

Blockchain has several features: it is completely decentralized, since there is no central authority regulating data; it guarantees irreversible transactions, because once a transaction is generated, there is no way to delete or modify it; it is a trustless system, since it allows the transfer of sensitive information on a non-trust network by trusting the system on the whole not the system participant; it shows a pseudo-anonymous nature, since anybody can create a blockchain address to be used for transactions and it is no way to trace back it to his/her identity if appropriate precautions are taken [189]. It is worth noting that anonymity, in the original notion of blockchain, is a fundamental feature, as blockchain is born with the cryptocurrencies in mind and, for many years, cryptocurrencies were the sole applications for blockchain.

However, in the last years, also thanks to the advent of new blockchains and smart contracts, we are witnessing the shift from the use of blockchain as simple platform for cryptocurrencies to complex applications in which we want to automatize trust and transparency, and to take advantage from the other features of blockchain. In these cases, in general, we do not need anonymity anymore. Indeed, there are situations in which accountability, trust and transparency should strictly coexist, and accountability should be implemented by allowing a secure association with real-life identities. This requirement may derive from many different needs: it might be just an opportune measure to prevent unresolvable disputes, or it could derive from compliance with the law. For these cases, blockchain appears little suitable, especially when the domain of the involved actors is open and not confined inside a single organization, which is a prerequisite for the suitability of blockchain itself. Consider, for instance, the management of the flow of goods and services (supply chain) [71]: it involves the movement and storage of raw materials, of work-in-process inventory, and of finished goods from a point of origin to a point of consumption. Typically, a supply chain is managed by a platform, a sets of technologies and processes promoting information sharing and coordination. There exist platforms for same day e-commerce home delivery in which consumers use a smart phone to browse and shop a broad range of products aggregated from nearby retail stores. Then, customer orders are handled by nearby independent couriers for pick-up and delivery to the customer. However, the platform acts as a trusted third party, thus it has to be always online and trusted by all participants. If at least one of the two conditions does not hold, using a blockchain makes sense. In this case, it should be necessary that anybody generating a transaction can be identified, but the current version of blockchain allowing pseudo-anonymous transactions does not help us.

For all the above reasons, an important research problem is to understand how to overcome the native pseudo-anonymity of blockchain in order to support identity-aware applications.

In this solution, we address this problem by proposing a solution that mixes the mechanism of public digital identity with blockchain via Identity-Based-Encryption. We found this way the most suitable and not explored (so far) approach, because it accomplishes all the aimed requirements. Identity-Based-Encryption (IBE) gives a direct role to the notion of identity, so allowing a direct link between the pair of cryptographic keys used to sign and verify a transaction and the identity of the transaction signer. On the other hand, public digital identity allows us to give a concrete definition of the identity to be used in IBEs by solving one of the problems of the concrete solutions based on IBEs, which is the proof of identity to the party issuing private keys (i.e., the Private Keys Generator).

As public digital identity, we use the notion compliant with eIDAS [86], a recent European Union regulation on electronic identification fully effective from 2016. It establishes the principle of mutual recognition and reciprocal acceptance of interoperable electronic identification schemes among Member States, and we chose it because (1) it is expected that, in the next years, eIDAS will be used by the most of EU citizens, (2) it is based on robust cryptographic primitives so that it can be considered secure, and (3) it has full legal effect.

We observe that an attempt of direct integration of public digital identity with a blockchain-based application would not provide a good result in terms of trust. Indeed, we should require that some entity of the application (even a smart contract if we adopt a blockchain like Ethereum) should play as a Service Provider of the public digital identity system (like in [29]). This implicitly requires the trust in this node, concerning the assessment of identity. In contrast, the use of IBEs requires that only Identity Providers (and this is an assumption accepted also in eIDAS) and the Private Keys Generator of IBEs are trusted parties, that are parties external to the application. Clearly, Identity Provider and Private Keys Generator might also coincide.

4.2 Ideal solution

We recall that the basic goal of this solution is to integrate blockchain and public digital identity. In this section, we sketch what we identify as the ideal solution of the problem above, in the sense that it implements the above integration in the most direct and strong way.

Suppose we have an IBE system with Private Key Generator PKG and a public identity digital system with identity provider IP (assumed unique, w.l.o.g.). For simplicity, we assume we are not considering blockchains allowing smart contracts (i.e., Blockchain 2.0), even though the generalization to every kind of blockchain is straightforward. Therefore, we focus our attention just on the elements related to our problem, which are the blockchain addresses and, consequently, the form of transactions. Obviously, the organization of blocks, the consensus protocol, the mining process, and the other aspects of the blockchain are outside the scope of our problem.

Specifically, the elements of the blockchain we are considering in this section are:

1. the blockchain address, denoted by A_u, of a user u and obtained as $A_u = h_1(h_2(P_u))$, where h_1 and h_2 are two proper cryptographic hash functions (as typically done in blockchains), and P_u is a public key of u in the cryptosystem used in the blockchain;
2. the transaction, which we schematically denote as a tuple $\langle P_{u_s}, i, A_{u_r}, c\rangle$, where P_{u_s} is the public key associated with the user *sender*, i denotes the input transactions, A_{u_r} denotes the blockchain address of the user *recipient* (assumed unique for simplicity) and c is the payload of the transaction (e.g., in Bitcoin, it represents the amount of money transferred by this transaction). The transaction is signed by using the secret key S_{u_s}.

Our idea is the following. We assume that u is equipped with a public digital identity granted by IP and let UID be the universal identity number of the user in the public digital identity system (recall that such an identification number exists in real-life public digital identity systems and it is independent of the identity provider, in case of multiple identity providers). Let denoted by IBE^P_{UID} and IBE^S_{UID} the IBE public key and secret key derived by the identity UID, respectively. Recall that, on the basis of the master key, IBE^P_{UID} can be obtained by any party with no need of further information. On the contrary, IBE^S_{UID} is released by PKG through a secure channel to any party able to demonstrate to be the owner of the identity UID. What we require is that PKG becomes a service provider in the public digital identity system, which means that it recognizes in a secure way the identity of people by leveraging the federated authentication protocol involving IP and a (strong) authentication session of the user at IP. Therefore, in order to release secret keys, PKG will require a secure authentication session done according to the protocol of the public digital identity system.

This allows us to design a blockchain in which the address of the user u, recognized in the public digital identity system by the identifier UID, is obtained as: $A_u = h_1(h_2(IBE^P_{UID}))$ (we recall that h_1 and h_2 are two cryptographic hash functions). Therefore, the sources and the recipients of a transaction are derived directly from UIDs, thus from public digital identities, and impersonation is not possible provided that it is not possible in the public digital identity system. Specifically, a transaction

$\langle P_{u_s}, i, A_{u_r}, c\rangle$ done by the user u_s with identity UID_s and having as recipient the user u_r with identity UID_r, is signed by the IBE secret key $IBE^S_{UID_s}$ and verified by the IBE public key $IBE^P_{UID_s}$, which everyone can compute on the basis of the IBE public master key, once the identity UID_s is known. This allows us also to represent the transaction as: $\langle UID_s, i, UID_r, c\rangle$. This representation reflects a nice feature of our solution, in which blockchain addresses are intensionally always existing in the blockchain domain, even though they are not materialized, provided that the corresponding identities exist in the public digital identity system. As a consequence, a given transaction moving a token (or money) to a user u may exist in the blockchain without requiring any action from u on the blockchain (the creation of a key-pair), as identities are implicitly blockchain addresses.

One could argue that a similar solution makes us lose the full decentralization of the blockchain paradigm. This is necessarily true if we want to rely on the current notion of public digital identity system, which is inherently centralized. However, a different notion of digital identity could be applied, also fully decentralized and based on blockchain itself like [128] or [162].

It is worth noting that the ideal solution here presented implicitly requires that blockchain (public and private) keys are compliant with the adopted IBE scheme (for example, RSA [223]). Unfortunately, this is not the case of existing blockchains: for an instance, Bitcoin blockchain adopts the elliptic curve secp256k1 [185], which is not compliant with any IBE scheme and a definition of an IBE scheme on this cryptographic scheme is not feasible.

For this reason, to give a more practical value to this solution, we implement in the next section a workaround that allows us to basically obtain the same result by leveraging any existing blockchain. Specifically, we chosen Bitcoin blockchain because it is one of the most used, but any other blockchain could be considered, also by extending the approach toward smart-contract-supporting blockchains like Ethereum. Consider that, in this case, any solution (like [29]) that implements the integration between the public digital identity system and the blockchain by directly giving the role of service provider to smart contracts, does not reach the goal in a satisfactory way from the security point of view, because it requires that the service providers (internal to the application domain) are trusted third parties (TTPs). Conversely, in our solution, TTPs are only TTPs of the external systems (i.e., the identity provider of the public digital identity system and the Private Key Generator of the IBE system).

4.3 A practical solution

Starting from the considerations done in the previous section, in this section we provide a practical solution that does not relax any security feature w.r.t. the ideal one. It is practical in the sense that it does not require changes of blockchain formats and protocol, thus operating on the exiting ones. For the sake of presentation, we describe the solution on the Bitcoin blockchain, which is widely used.

The actors in our scenario are:

- *Users*, physical or legal people using a public digital identity for authentication.
- *Identity Providers*, which create and manage public digital identities.
- *IBE Services*, public or private organizations providing the mapping between a public digital identity and a pair of asymmetric encryption keys (called IBE keys).
- a *Blockchain*, a Distributed Ledger.

In our proposal, we can identify the following operations.

1. *Digital Identity Issuing*. First, a user creates his/her public digital identity. To do this, he/she must be registered to one of the *Identity Providers*, which is responsible for the verification of the user identity before issuing the public digital identity and the security credentials.
   A public digital identity is identified by the pair $\langle username, IP \rangle$, where *IP* is the identifier of the identity provider that issued the public digital identity and *username* is a string. Moreover, there exists a string *UID* (Universal ID), which identifies a public digital identity. For example, the user X registered by the Identity Provider Y is identified by the UID X@Y. It is worth noting that UIDs are supported by the Public Digital Identity Systems.
2. *IBE private key gathering*. To obtain the IBE private key, a user contacts the Private Key Generator (PKG) of the IBE service to receive the master public key, if it is not already known. Then, the Private Key Generator, by acting as a service provider of the public digital identity system, authenticates the user by an eIDAS-compliant scheme, as illustrated in Fig. 6.2.
   First, the user using a browser (`User Agent`) sends to PKG a request for gathering the IBE private key (Step 1). Then, PKG replies with an authentication request to be forwarded to *Identity Provider* (Step 2). If the received request is valid, *Identity Provider* performs a challenge-response authentication with the user (Steps 3 and 4). In case of successful user authentication, *Identity Provider* prepares the statement of user authentication, which is forwarded to PKG (Step 6). Finally, PKG provides the user with the IBE private key (Step 7).

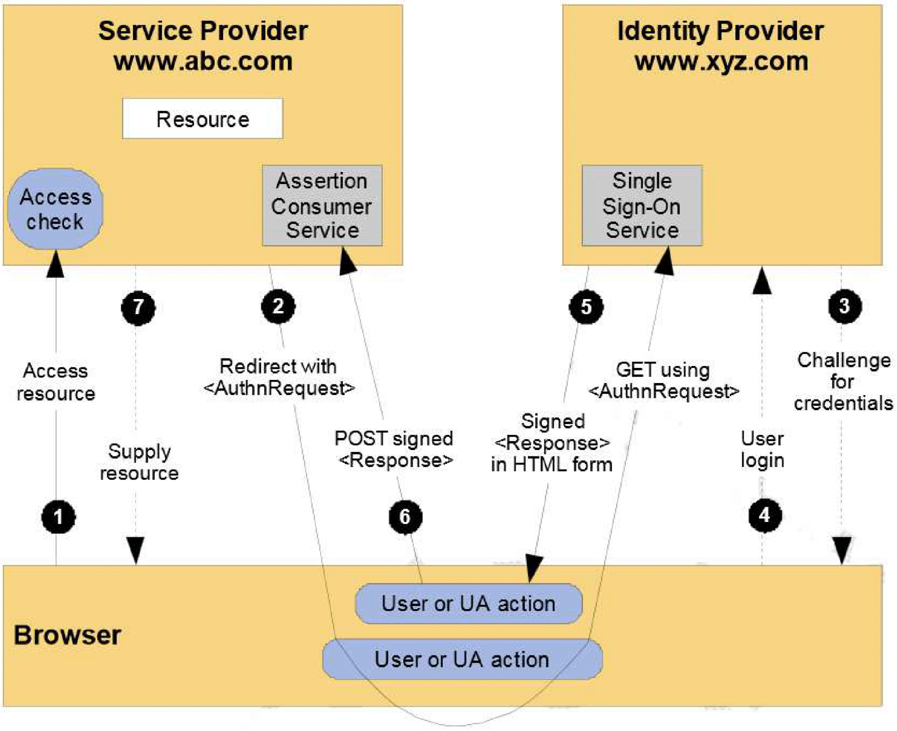


Fig. 4.1: Data flow in an authentication process.

3. *Blockchain Registration*. First, the user generates a pair of private and public blockchain keys, and, starting from the public one, the blockchain address A is computed. Then, the user generates on the blockchain a transaction from A to A, having as payload $\langle UID, E(A) \rangle$, where UID is the universal ID of the public digital identity of the user, and $E(A)$ is the encryption of the user's blockchain address by the user's IBE secret key. By this transaction, the user links hes/his public digital identity to the blockchain address A: indeed, by computing $E(A)$, the user proves the knowledge of the IBE secret key associated with this UID.
4. *Transaction*. When a user S (sender) wants to carry out a transaction with a user R (receiver), the following operations are done:
   a) S obtains the universal ID of R, say IUD_r.
   b) S searches for the transaction having IUD_r in the payload: this is the transaction done by R in the blockchain Registration step.
   c) S extracts from this transaction the blockchain address of R, say A_r.
   d) S generates a blockchain transaction from her/his blockchain address A_s to A_r (the value of the payload depends on the application).

Now, it should be easy to understand how to know the public digital identity of a user involved in a blockchain transaction. Consider a blockchain transaction from the (blockchain) address A_s to the (blockchain) address A_r, and assume we are interested in knowing the identity of the user associated with A_r[1].

The first operation to do is to search for the transaction having A_r as sender and receiver (i.e., the transaction done in the Blockchain Registration step). If it is not found, this means that A_s did not execute the protocol correctly, because she/he generated a transaction to an unregistered user (clearly, it is not possible that the registration transaction of A_r has been deleted because blockchain transactions are immutable). Thus, we assume that this registration transaction, say T, is found.

Now, after verifying the authenticity and integrity of T (i.e., that it has been signed by the blockchain public key associated with the address A_r), the payload $\langle p_1, p_2 \rangle$ is extracted.

Next, the IBE public key $IBE^K_{p_1}$ derived from the string p_1 is computed and used as public key to decipher p_2. If the decryption of p_2 corresponds to A_r, then we are sure that the receiver (i.e., A_r) of the transaction T is associated with the public digital identity p_1.

Clearly, by repeating the same procedure starting from A_s instead of A_r, we can identify also the user associated with A_s, who generated the transaction.

4.4 Case study and implementation details

In this section, we instantiate the general approach presented in the previous section to a specific scenario and we show the generated data both to better explain how our proposal works and to demonstrate its compliance with the Bitcoin blockchain.

Among the numerous applications that can benefit from our solution, we selected crowdshipping, which is very timely (as remarked in Section 14.6).

Crowdshipping refers to the phenomenon of recruiting citizens to serve as couriers: a person already traveling from point A to point B takes a package with him and, making a stop along the way, delivers the package to another person in exchange for a reward. The objective is reducing pollution and road traffic using, as a delivery carrier, a person who is already on the move.

Zipments [6], active in New York since 2014, and PiggyBee [4], online since 2012, are probably the most known crowdshipping platforms. Being a centralized approach, the platform has to be a trusted party because it is in charge of receiving and storing log activity: clearly, an attack on the system or a malicious behavior

[1] For the sake of presentation and to avoid to introduce new notations, in the following, with a little abuse of notation, we use the address A_u also to refer to the user u, thus meaning "the user associated with the address A_u"

of the platform provider could compromise accountability. To address this problem, the use of blockchain is a solution: all the information needed to guarantee accountability, especially the delivery of a package between two users, is stored in the blockchain. In particular, we considered the basic step of a crowdshipping system, which occurs when a user, say Alice, delivers a package to another user, say Bob. Alice needs both: (1) to be sure that the person receiving the package is Bob and (2) to have a proof of delivery. Our solution guarantees both the goals without using a centralized crowdshipping platform.

We implemented a Java prototype to test our solution in a crowdshipping scenario: it is composed of a module implementing the IBE system and a module implementing the access to the blockchain. We did not need to implement the identification scheme compliant with eIDAS, because it is a service used by our prototype. We show all the operations carried out by the two users and the generated data.

1. *Digital Identity Issuing*. Both Alice and Bob have a public digital identity: thus, they have been identified by an identity provider, say `example.com`, which gave each of them a public digital identity and a credential for authentication (typically, a password). Now, assume that the username of Alice is `alice` and the username of Bob is `bob`. Thus, the UIDs of Alice and Bob are `alice@example.com` and `bob@example.com`, respectively. Observe that, for the sake of presentation, we used the same identity provider (i.e., `example.com`) for both the users: however, no problem arises in case the public digital identities are issued by different identity providers, because the solution does not depend on the particular UID of the user.
2. *IBE private key gathering*. To obtain the IBE private key, a user connects to the site of the IBE system by the browser (i.e., the user agent) and sends a request for accessing the service. Observe that the IBE system acts as a service provider in this step, because it needs to authenticate the user before issuing the private key. Then, the IBE system replies to the user agent with an authentication request to be forwarded to the identity provider. The identity provider is selected according to the user's UID.

   If the received request is valid, the identity provider performs a challenge-response authentication with the user. In case of successful user authentication, the identity provider prepares the *assertion* containing the statement of the user authentication for the IBE service provider. The assertion contains the reference to the request message, the authenticated user, the identity provider, the personal information about the authenticated user, the temporal range of validity, and the description of the authentication's context. The assertion is signed by the identity provider to guarantee integrity and authenticity.

Now, the assertion returned to the user agent is forwarded via `http POST` Binding to the IBE service provider. The IBE system verifies the assertion and provides the user with her/his IBE private key. We denote by IBE_U^S the IBE private key of the user U.

Concerning the user's IBE public key, they are computed starting from the master public key and the user's UID. We denote by IBE_U^P the IBE public key of the user U.

In Table 4.1, the IBE public and private keys of Alice and Bob are reported: they are represented by Base58Check encoding [1], which is used for blockchain addresses (see later).

3. *Blockchain Registration*. Each user needs to have a private and a public blockchain key. The private key is a randomly generated 256-bit string. The public key is generated by the private one by means of a cryptographic function named *elliptic curve point multiplication*. In particular, the used algorithm is Curve Digital Signature Algorithm (ECDSA) and the elliptic curve is `secp256k1` [185]. The use of these functions is necessary to guarantee the compatibility of our solution with blockchain.

   We denote by BKC_U^S and BKC_U^P the blockchain private key and public key of the user U. In Table 4.1, the blockchain public and private keys of Alice and Bob are reported.

   The blockchain address A of a user is computed from the public key K as A =RIPEMD160(SHA256(K)), where SHA256 [196] is a cryptographic hash developed by National Security Agency (NSA) and returns a 256-bit digest, whereas RIPEMD160 [222] is a cryptographic hash designed in the open academic community and returns a 160-bit digest.

   We denoted by A_U the blockchain address of the user U. In Table 4.1, the blockchain addresses of Alice and Bob are reported. Observe that blockchain addresses are usually represented by Base58Check, an encoding similar to Base64 but modified to remove non-alphanumeric characters and letters which might look ambiguous when printed. It is therefore designed for human users who manually enter the data by copying from some visual source.

   Finally, each user generates on the blockchain a transaction with her/his address as both sender and receiver, having as payload $\langle UID, E(A)\rangle$, where UID is the universal ID of the public digital identity of the user, and $E(A)$ is the encryption of the user's blockchain address done by the user's IBE private key.

4. *Transaction*. Now, both Alice and Bob have their public digital identity associated with a blockchain address. Suppose that Alice has to deliver a package with $ID = AB123$ to Bob and, consequently, she needs a proof of delivery from Bob. In

a real-life situation, we can image that carriers run a mobile app on their smartphones to manage transaction generations. We can suppose also that the package ID is a QRcode printed on the box, so it can be easily read by the mobile app running on carrier's smartphone. Moreover, the same mobile app can show another QRcode reporting the UID of the owner, in such a way that when Alice has to deliver the package to Bob, Bob can show his UID by his mobile smartphone and vice versa.

Once the package ID and the UID of Alice have been collected, Bob's mobile app generates a transaction to A_{Alice} (i.e., the Alice's blockchain address) including in the payload the type of operation carried out (i.e., package receiving) and the id of the product. This transaction is signed by Bob with the blockchain private key and stored on the blockchain.

Alice can read on the blockchain this transaction and checks its correctness: clearly, this is done by the app mobile. This transaction represents the proof of delivery of the package from Alice to Bob.

Observe that in some context it could be necessary also an additional proof: in this case, Alice can generate a transaction to Bob having, in the payload, "package sending" as the type of operation and the id of the product, in such a way that Bob can proof the reception of the package from the correct user (i.e., Alice).

4.5 Conclusion

In this chapter, we discussed about the benefits deriving from the possibility of binding the sender or the receiver of a blockchain transaction to a public digital identity. We proposed an architecture eIDAS-compliant identification schemes for handling public digital identities and Identity-based Encryption for associating a digital identity with a public key. This architecture has been implemented by a Java prototype and used to validate the proposal in a crowdshipping scenario.

To the best of our knowledge, this is the first attempt to create a non-anonymous blockchain, which can be used in all cases in which the author of a transaction has to be identified with certainty and legal effect.

Symbol	**Value**
IBE^{S}_{Alice}	2UBUArXzNjLYArGyk46pn6yJVrik5x5sFRne1H2ACznMeBeAfvJyMdbdY 5aDofJ2NxjTQHwCUdpRiPMfKo4Y7CNhrwVq4KDFTiA3KavkyX7b7dE U1CVB1SwEZkMQL2KoD5erHSVsCwcKiqm6yPESsZMPWhEUkWio47D SETVP672AYrnw4E8zMH18gvrnPaERiLd9KL8Z9ZhYui7NL7NWA4oJ8kq GXSsJX85gTheFyswfSXya4HrwfXtQYotrpq7uuS7rKgGGsAhazE7Ceg6mY cMUSdbPg9drR21EUx3LrD3z3sp8QhFvDBpLkdhEZMGsngjDgdu24rZZh M5beQUCC56WVWefRE
IBE^{P}_{Alice}	C9vPE,185xNAn8quf9s4vrechMHXj6PDZakHJ532JYvGbQt7obcLqyyeLub 7VTXcY8qPg3FJj3TvPgEV3CAEx4K9U8diTkGj1xe2dZFicQLWk68KnGy eDZimALtNHm2hEvjFJSDinMtBEQAZrZUXGBrqN7QFT5imVLV821kEJ b1sMh3HMfnhEmucXsn1MDKgCymhkrXHKsFfFRyta9wzQvQKRmdNhT 5FruGXTV8VbA6XG1VaszpUco7EiUzLfayZdA8YWw3umeZDr5bi3AenxM WHXw6ptZFPg7rkf2YgmbHKqFnBCEyDT2gsF8XF9S1rkWjiqsKxef1ZC63 czWiRvfAZ3A5xfo7R9QAsk
IBE^{S}_{Bob}	19HktTsakupP856Q2whGm4ExkLambnHpKNDEFUngnwdLi6RQJrL5T2R zYKTbd1FowwHgLJnzLbNV9vNyJC3TamygAqAbzgxtt6oHmaR6Uj49uD3 6zAWb6Zkn7TDN6uqfaMVWwjVq6rh2GV4Rai66a1EPtRBRbdxoYS9pVE vmB74juFNxwmsbkxxCa3uU1xYi65HGWBo96S6z2NYkxvwEfUrbhikKhxw jbRuCcFpubgtsHPs2k5SyrY1shXXJMX4ZCqgb2BwseJEEiLo11j4BMgmo53 yoWrj3sA1Ysh7NNQDvREAptZnaVhJg7RRZRvdMe7WGFxW6yunrynV9 VXR2jwZk2KnYn96
IBE^{P}_{Bob}	Meip8,185xNAn8quf9s4vrechMHXj6PDZakHJ532JYvGbQt7obcLqyyeLub7 VTXcY8qPg3FJj3TvPgEV3CAEx4K9U8diTkGj1xe2dZFicQLWk68KnGye DZimALtNHm2hEvjFJSDinMtBEQAZrZUXGBrqN7QFT5imVLV821kEJb 1sMh3HMfnhEmucXsn1MDKgCymhkrXHKsFfFRyta9wzQvQKRmdNhT5 FruGXTV8VbA6XG1VaszpUco7EiUzLfayZdA8YWw3umeZDr5bi3AenxM WHXw6ptZFPg7rkf2YgmbHKqFnBCEyDT2gsF8XF9S1rkWjiqsKxef1ZC6 3czWiRvfAZ3A5xfo7R9QAsk
BKC^{S}_{Alice}	z4KNrFydhCHUt15g9N3MDX4Di2WfuE1JzMMZHRFtVCWkpx6DTH1H VTqBCtdwCL1ERwVfeto3A5pU8G8Fgkv8V2G
BKC^{P}_{Alice}	2f6eQs8MtzDWd1dj95LncxAfEseGNvn7LhbUYrg8kPSUNp5gYKN9zwkvwm ZtPFoFpjPrqdEgeYj3jzbvzKvComRQ7iF9JSE3JGY6UfNBYshkZzXG8qkJ WM93MDDSz6rJzYfAZ8rVU6n9xLLH2CeSSfhv5QZW9MqjcT3v7Mpsahh HtYHUK7
BKC^{S}_{Bob}	22PMoQ4VMGhGwep4KxxqHLB4JtPZFJ5AvLWar5ndP3fvGHaRbsEW2H yw55qAZR9TMyG9Z49P3tApibixFZo2SwXV
BKC^{P}_{Bob}	2f6e6iHZ4yg6h5oTaNj5tvTpgbhbjGJ63DUrA7Zi9n8aut1wtWQ9ELNt3iMeZ 7taavtwv55bZUY2MQdNFmAhoozstxUt6km8j791bWwDtaZGmkxv7vb5bzy sLtnDre8fgQ8GfLuv3F5yEmzweGv69S
A_{Alice}	1QCw797xQbc7UjbpeMeCcdxyj9SpbVnvN4
A_{Bob}	1JCn8rVTLTKf8H1hJQc9Jx2s9FTEwPNUDk

Table 4.1: Value of the data generated in our running example.

5

Ethereum Transactions and Smart Contracts among Secure Identities

One of the limitations of the current blockchains is that recipients of transactions (originated from both users and smart contracts) must preliminarily sign up the system. In contrast, the nature of blockchain would allow the implementation of services with a high degree of flexibility and interoperability, once the subjects can be securely identified someway. In this chapter, we overcome this limitation by integrating Public Digital Identity with Ethereum via Identity-Based-Encryption (IBE). An important feature of the solution is that it does not require additional trust w.r.t. that necessary for IBE and Public Digital Identity systems.

5.1 Introduction

The interest towards blockchain [193] is constantly increasing during the last years, due to its power to enable new business scenarios. Blockchain technologies attract the attention of both industries and researches, in various fields, besides computer science, mainly also economics and law. As a consequence, any aspect regarding those technological features that impact how applications can be designed and used is very relevant.

One of the current limitations of blockchains (even smart-contract oriented) is that actors of transactions and smart contracts are required to voluntarily subscribe to the services of the application platform implemented over blockchain. This aspect makes blockchain platforms little appropriate to those situations in which services may involve dynamically unregistered subjects. The proposal of this solution regards the following situation. Consider a set S of subjects who are identifiable in a certain (secure) way. For the moment, it does not matter how. Suppose in this set there are users who may be involved in a service based on blockchain (for example, Ethereum), in different moments, depending on dynamic conditions. Thus, for example, `Alice`, who is already registered to the service, has to transfer money (or a given token) to `Bob`. But `Bob`, despite being in S, is not in the service yet. In other words, we would like to enable some *suspended* actions, to avoid to compromise the liveness of the system. Indeed, according to the features currently supported by Ethereum (and the other blockchains), Alice's action should be denied by the system until Bob signs up the system. We obtain a similar use case when the sender is a contract instead of a human user.

Our proposal, contextualized in the Ethereum environment, is aimed to overcome the above limitation, by enabling over Ethereum transactions and contracts among (secure) digital identities, whose existence is independent of the specific application platform. This allows the design of flexible, dynamic and interoperable services, with considerable benefits in many cases, especially in crowd-based or multi-organization domains.

To do this, we faced a number of problems. The first one is which notion of digital identity we may adopt to have a realistic result. One could think of an identity built as a combination of (verified) social network profiles owned by the subject being identified. This could be an option, but we think that whether a Public Digital Identity System exists, like those that are compliant in EU with the eIDAS regulation [86], this is the best way to follow. Thus, in our schema we refer to SPID [87], which is the Italian System of Public Digital Identity introduced in accordance with the eIDAS initiative.

The second point is how to link in a secure way digital identities and Ethereum addresses. Our solution leverages Identity-Based-Encryption (IBE) [2], which gives a direct role to the notion of identity and then a direct link between Ethereum keys and identity of the user, once she/he is able to provide the PKG (i.e., the party issuing the IBE private key) with the proof of her/his SPID identity. From this point of view, this solution is an evolution of the work presented in [62], in which the idea of integrating IBE and blockchain is presented for the first time in the simpler context of Bitcoin blockchain, thus without the possibility of involving unregistered users. We highlight that the role of IBE is crucial in our proposal, because a direct integration of SPID with blockchain (like in [29]) would require that a blockchain-side entity (an application or a smart contract) should play as a Service Provider of the public digital identity system. This would require the full trust in this entity, concerning the assessment of identity.

5.2 Our proposal

The goal of this solution is to allow the association of a digital identity with a blockchain transaction. Among the possible mechanisms to handle digital identity, such as OAuth [198], OpenID [9] Windows CardSpace [5], we refer to the notion of public digital identity, which has been defined by the Regulation (EU) N. 910/2014 [86]. Our choice is motivated by the fact that we expected that, in the next years, public digital identity will involve the most of EU people: for example, on February 2017, Germany notified its national identity which has more than 40 million registered citizens [10].

In our solution, we have the following types of entity:

- a user, a physical or legal person using a digital identity for authentication. Each user can be associated with one or more public digital identities.
- a public identity digital system with identity provider IP, which creates and manages public digital identities. Without loss of generality, we assume it is unique.
- an IBE system with Private Key Generator PKG. It is managed by a public or private organization and provides the mapping between a digital identity and a pair of asymmetric encryption keys (called IBE keys).
- a Distributed Ledger allowing smart contracts (i.e., Blockchain 2.0).

In this scenario, we identify the following types of operation that users carry out.

1. *Digital Identity Registration*. To obtain a digital identity, a user must be registered to the public identity digital system. In this phase, the real identity of the user is verified before issuing the public digital identity and the security credentials.
   A public digital identity is identified by the pair $\langle username, IP\rangle$, where *IP* is the identifier of the identity provider that issued the public digital identity and *username* is a string. For example, the user X registered by the Identity Provider Y is identified by the X@Y. Moreover, any Public Digital Identity System compliant with eIDAS defines also a string *UID* (Universal ID), which is a single numeric identifier independent of the identity provider, in case of multiple identity providers.
2. *IBE private key gathering*. To obtain the IBE private key, a user contacts the Private Key Generator (PKG) of the IBE service to receive the master public key, if it is not already known. Then, the Private Key Generator, by acting as a service provider of the public digital identity system, authenticates the user by an eIDAS-compliant scheme.
   First, the user using a browser (`User Agent`) sends to PKG a request for gathering the IBE private key (Step 1). Then, PKG replies with an authentication request to be forwarded to *Identity Provider* (Step 2). If the received request is valid, *Identity Provider* performs a challenge-response authentication with the user (Steps 3 and 4). In case of successful user authentication, *Identity Provider* prepares the

statement of user authentication, which is forwarded to PKG (Step 6). Finally, PKG provides the user with the IBE private key (Step 7).

3. *Blockchain Binding*. By this operation, a user associates his IBE public key IBE_p^K with his blockchain address A. First, the user generates a pair of private and public blockchain keys, and, then, the blockchain address A of the user is computed as the cryptographic hash of the public key. Then, the user generates a transaction from A to A on the blockchain, having in *data* field $\langle UID, E(A) \rangle$, where UID is the universal ID of the public digital identity of the user, and $E(A)$ is the encryption of the user's blockchain address by the user's IBE secret key. This transaction is called *binding transaction*. By this transaction, the user links her/his public digital identity to the blockchain address A: indeed, by computing $E(A)$, the user proves the knowledge of the IBE secret key associated with this UID.
4. *Transaction*. Suppose a user S (sender) wants to send to a user R (receiver) a transaction and let v be the value of the transaction (i.e., the amount of virtual money to transfer). In this case, the following operations are done. First, S obtains the universal ID of R, say UID_r and searches for the most recent binding transaction having UID_r in the payload: this search can be successful or not. If a transaction $T = \langle UID_r, E(A_r) \rangle$ of this type is found, then:
   a) S deciphers $E(A_r)$ by using the IBE public key calculated from UID_r, to verify that the authenticity of the signature (observe that $E(A_r)$ works as a signature to prove that the right user has generated the binding transaction). If this check fails, T is ignored and another search is carried out.
   b) After deciphering $E(A_r)$, the blockchain address of UID_r is obtained.
   c) S generates a blockchain transaction from his blockchain address A_s to A_r, with value v.

   Consider now the case in which no transaction of this type is found, which is the most interesting case. This means that the user R exists but has not yet joined the blockchain. In this case, S generates a blockchain transaction from his blockchain address A_s to the blockchain address of a specific smart contract, say A_{sc}, specifying both UID_r and v. This smart contract stores the information that there is a *sleeping* transaction to UID_r, from the sender A_s and value v.
5. *Cashing*. Suppose that a user R, after registering to the blockchain, wants to receive the *sleeping* transactions sent to him before his registration (i.e., those transactions sent to the smart contract *sm* and intended for him). Then, he generates a blockchain transaction, named *cashing transaction*, from his blockchain address A_r to the blockchain address A_{sc} (i.e., the same smart contract referred above), having his UID (i.e., UID$_r$) in the payload. Now, the smart contract searches for the most recent binding transaction T sent from A_{sc} and computes the IBE pub-

lic key IBE_r calculated from UID_r. Then, it extracts $E(A)$ from the payload of T and deciphers $E(A)$, verifying that UID_r is obtained. Finally, it extract from the stored sleeping transactions those sent to UID_r (if any): for each transaction found, a new transaction to A_r is generated, with the same value as the found transaction.

In the next section, we show how to implement this solution in Ethereum.

5.3 Implementation

In this section, we instantiate the general approach presented in the previous section to the specific environment of Ethereum: in particular, we show all the operations carried out by two Ethereum users, say Alice and Bob.

1. *Digital Identity Registration*. Both Alice and Bob have a public digital identity: thus, they have been identified by an identity provider, say `example.com`, which gave each of them a public digital identity and a credential for authentication (typically, a password). Now, assume that the username of Alice is `alice` and the username of Bob is `bob`. Thus, the UIDs of Alice and Bob are `alice@example.com` and `bob@example.com`, respectively. Observe that, for the sake of presentation, we used the same identity provider (i.e., `example.com`) for both the users: however, no problem arises in case the public digital identities are issued by different identity providers, because the solution does not depend on the particular UID of the user.
2. *IBE private key gathering*. To obtain the IBE private key, a user connects to the site of the IBE system by the browser (i.e., the user agent) and sends a request for accessing the service. Observe that the IBE system acts as a service provider in this step, because it needs to authenticate the user before issuing the private key. Then, the IBE system replies to the user agent with an authentication request to be forwarded to the identity provider. The identity provider is selected according to the user's UID.

   If the received request is valid, the identity provider performs a challenge-response authentication with the user. In case of successful user authentication, the identity provider prepares the *assertion* containing the statement of the user authentication for the IBE service provider. The assertion contains the reference to the request message, the authenticated user, the identity provider, the personal information about the authenticated user, the temporal range of validity, and the description of the authentication's context. The assertion is signed by the identity provider to guarantee integrity and authenticity.

Now, the assertion returned to the user agent is forwarded via `http POST` Binding to the IBE service provider. The IBE system verifies the assertion and provides the user with her/his IBE private key. We denote by IBE_U^S the IBE private key of the user U.

Concerning the user's IBE public key, they are computed starting from the master public key and the user's UID. We denote by IBE_U^P the IBE public key of the user U.

3. *Blockchain Binding*. Each user needs to have a private and a public blockchain key. The private key is a randomly generated 256-bit string. The public key is generated by the private one by means of a cryptographic function named *elliptic curve point multiplication*. In particular, the used algorithm is Curve Digital Signature Algorithm (ECDSA) and the elliptic curve is `secp256k1` [185].

   The Ethereum address A of a user is computed from the public key K by apply Keccak-256 [43], and finally taking the last 20 bytes of that hash. We denoted by A_U the blockchain address of the user U. Finally, each user generates the binding transaction having as payload $\langle UID, E(A) \rangle$, where UID is the universal ID of the public digital identity of the user, and $E(A)$ is the encryption of the user's Ethereum address done by the user's IBE private key.

4. *Transaction*. Now, both Alice and Bob have their public digital identity associated with a blockchain address. Suppose that Alice has to send some Ether money to Bob, but Bob has not an Ethereum wallet (i.e., he has not an Ethereum address). Clearly, we can image that users run an application (on a PC or a smartphones) to manage transaction generations.

   First, Alice has to know the UID of Bob: the UID of Bob as well as the amount of money to transfer are inserted into the application, which generates a transaction to the smart contract. In Listing 5.1, we give an implementation of this contract written in Solidity, which is a JavaScript-like language. For the sake of presentation, we do not explain every line of the code: we assume the reader is familiar with Solidity and Oraclize, which is the leading oracle service for smart contracts and blockchain [7] In particular, it is called the function `pay`, using the UID of Bob as parameter (Lines 13-15): this function stores the amount that will be given to Bob when he will register (by `payUID`).

5. *Cashing*. After Bob creates his wallet, he can ask for receiving the amount from the *sleeping* transactions sent to him before his registration. To do this, he generates a cashing transaction to the smart contract illustrated in Listing 5.1, by calling the function cash. The smart contract first checks if there is some amount for Bob. If any, an oraclize function is used (Line 21), which returns the Ethereum

address of Bob by the callback function. Finally, a money transfer to Bob is carried out by the smart contract and the amount to pay to bob is reset (Lines 33-34).

By this protocol, we enable on Ethereum the possibility to send money to users without the need to know their blockchain address. The suitable use of the secure digital identity guarantees that only the correct user receives money.

```
 1  pragma solidity ^0.4.25;
 2
 3  import "github.com/oraclize/ethereum-api/oraclizeAPI_0.4.25.sol";
 4  import "github.com/Arachnid/solidity-stringutils/strings.sol";
 5
 6  contract SleepingEther is usingOraclize {
 7    mapping(bytes32=>string) uidMapping; //mapping between queryID and bool
 8    mapping(string=>uint) payUid; //mapping between UID and eth value to send
 9    address public addr;
10    using strings for *;
11    string pi;
12
13    function pay(string uid) public payable {
14        payUid[uid] += msg.value; // add the ether addressed to uid
15     }
16
17     function cash (string uid) public payable{
18        if(payUid[uid]>0)
19           if (oraclize.getPrice("URL") <= address(this).balance) {
20              pi = "URL".toSlice().concat(uid.toSlice());
21              bytes32 queryId = oraclize_query("URL", pi);
22              uidMapping[queryId]=uid;
23           }
24     }
25
26    function __callback (bytes32 myid, string result, string uid) public {
27        if (msg.sender != oraclize_cbAddress())
28          revert ();
29        bytes memory tempEmptyStringTest = bytes(result);
30        if(tempEmptyStringTest.length != 0){
31           addr = parseAddr(result);
32           uint tot= payUid[uidMapping[myid]];
33           addr.transfer(tot);
34           payUid[uid]=0;
35        }
36     }
37  }
```

Listing 5.1: Code of the smart contract.

5.4 Conclusion

In this chapter, we presented a solution to integrate Public Digital Identity with Ethereum to enable transactions (both external and internal) directed to subjects not yet registered to the system. The solution leverages the concept of Identity-Based-Encryption (IBE). The goal is achieved by giving to the Private Key Generator (PKG) of the IBE the role of Service Provider of the Public Digital Identity system. It is worth noting that, we only treat the case in which a given amount of cryptocurrency is transferred, but the transfer of tokens with identifier can be easily implemented by using the interface ERC721.

6

Accessing Online Services with Minimal Personal Information Disclosure

The General Data Protection Regulation highlights the principle of data minimization, which means that only data required to successfully accomplish a given task should be processed. In this solution, we propose a Blockchain-based scheme that allows users to have control over the personal data revealed when accessing a service. The proposed solution does not rely on sophisticated cryptographic primitives, provides mechanisms for revoking the authorization to access a service and for guessing the identity of a user only in cases of need, and is compliant with the recent eIDAS Regulation. We prove that the proposed scheme is secure and reaches the expected goal, and we present an Ethereum-based implementation to show the effectiveness of the proposed solution.

6.1 Introduction

In the digital era, information is a valuable asset: for instance, think about social networks, which collect information of hundreds of millions of people such as personal data, friends, visited places, listened to music, watched TV, preferences, and interests. Such data are monetized in several ways, such as to produce context-aware information that influences users' preferences to recommend specific items (custom advertising). For this reason, often accesses to services require that a user authenticates.

However, in many situations, there is not a real need to be aware of personal information, which is done only to collect rich data. Consider the case of a merchant selling products for adults (e.g., liquor or cigarettes), which only needs to check that acquirers are of a certain age: the request of the identity card to show the birth date has the side effect of disclosing personal information, such as name, surname, nationality. Although this situation is not very worrying in real life, the problem is relevant in the digital world because disclosed data can be stored and processed automatically. As collected data may contain private information that could be transferred to unauthorized parties, privacy-preserving proposals in this context are gaining attention [237, 145]. Indeed, recently many service providers require the disclosure of less sensitive information.

This privacy problem is well-known and has been recently remarked also by the issuance of the General Data Protection Regulation [104], the new European Union privacy law that puts guidelines and regulations on how data have to be processed, used, stored, or exchanged to protect and ensure individuals data privacy [159, 240]. In this context, the problem we address is how to build a system able to reach four research goals:

RG1. when accessing a service, a user should be allowed to provide the minimal amount of personal information needed to access the service;

RG2. only authorized users should access a service;

RG3. there should be the possibility to revoke the permission given to a user to access a service;

RG4. the user's identity should be revealed in case of need.

For this purpose, we exploit the power of blockchain [193], a recent technology used in many fields, such as finance, smart cities, society progress driving [239]. Blockchain is a fully distributed repository that stores transactions. Several nodes distributed in a peer-to-peer fashion have the control over stored information and run programmable rules in the form of *smart contracts* [141]. By replacing a single centralized party with a distributed ledger of replicated, shared, and synchronized data, blockchain guarantees transparency, traceability, and immutability of registered information.

In this solution, we propose a blockchain-based system that allows users to prove the possession of some attributes without disclosing their identity. Moreover, our proposal provides suitable mechanisms to allow revocation and accountability. Revocation concerns the possibility to make invalid a credential when a user loses possession of an attribute (for example, a driver's license) or the credential is stolen or expired. Accountability is a feature that allows a party in cooperation with other trusted parties to guess the identity of a user in cases of need. Differently from the state of the art, our proposal does not rely on sophisticated cryptographic primitives, which reduce the efficiency of a solution. An important aspect is related to the Regulation (EU) N 910/2014 [86], which regards electronic identification and trust services for electronic transactions in the EU internal market. This regulation provides a normative basis to enable secure electronic interactions between businesses, citizens, and public authorities. Among others, eIDAS introduces the role of the Attribute Provider, an entity responsible for providing information about electronic identities. The issuance of eIDAS opens the possibility to design new solutions for attribute certification that can be very effective because it is expected that in the next years eIDAS will involve most of EU people. We exploited this opportunity so that the proposed solution is compliant with the eIDAS infrastructure, which increases its effectiveness. We instantiated the general solution to a real-life scenario and described the detailed data workflow to show how our approach can be implemented by Ethereum, the most used blockchain enabling smart contracts.

6.2 Scenario and problem formulation

In this section, we introduce a scenario to present the addressed problem.

The scenario is composed of the following actors:

- *Users*, physical or legal people with an eIDAS-compliant digital identity.
- *Identity Providers*, eIDAS-compliant entities that create and manage digital identities.
- *Attribute Providers*, eIDAS-compliant entities that are in charge of verifying and validating the possession of attributes.
- *Service Providers*, which supply users with (online) services.
- a *Blockchain*, a Distributed Ledger in which smart contracts can be deployed.

We consider the case of a user who needs to access an online service supplied by a service provider, but this access is granted, provided that the user has the permissions and attributes to access it. The considered service is a car rental booking, in which users need to demonstrate the possession of two attributes: a driver's license and the age (because the rental price depends on the user's age). Thus, a service provider that offers a rental car service needs to be sure about the possession of the driver's license and to know the age of any user who wants to rent a car. Generally, in such cases, a service provider asks users to provide all personal data, which characterize them as legal and natural people. In some cases, the identification can be performed with the support of a third party (such as a social network authentication procedure).

The problem is that users have to reveal many personal and (possible) sensitive data that are not useful to gain the service requested, and this could damage their privacy and expose them to various attacks or future vulnerabilities. The disclosure of specific attributes can help to reduce this effect: users are responsible for the information they want to share with third parties, that is, they can decide to show data in a granular way.

In summary, the goals to reach are:

1. the service provider should be aware of the minimal amount of the user's information needed to access the service;
2. only authorized users should access a service;
3. there should be the possibility to revoke the permission given to a user to access a service;
4. the user's identity should be revealed in case of need.

Concerning the last item, it is worth noting that the recent definition of the eIDAS environment can help to face this problem with new tools that did not exist until a few years ago [212]. Indeed, the identity providers defined by eIDAS keep a series of information related to the digital identity of users, called unqualified or elementary attributes. Furthermore, the attribute providers defined by eIDAS are authorized to certify a qualification and can add other attributes to the digital identity of the user.

As a consequence, we propose a solution based on the disclosure of the attributes selected by users who want to access the service. Besides eIDAS, our proposal also relies on the recent technology of blockchain, which allows us to design a decentralized approach to maintain a hidden link between users and their attributes. blockchain is in charge of guaranteeing transparency and immutability of actions (or transactions) and allowing stakeholders to perform and verify a secure access control relying on attributes through smart contracts.

6.3 Conceptual model

Starting from the scenario described in the previous section, here we describe the solution designed to solve the considered privacy issue. The entities in our scenario cooperate by performing the tasks described in the following.

1. *Digital Identity issuing*. Any user running our solution needs to have an eIDAS digital identity. To do this, the user needs to register to one of the available identity providers, which is responsible for the verification of the user identity before issuing the credential of the digital identity.
   The identity provider of a user knows a list of elementary attributes (e.g., date of birth) of the user. On the other side, there are also attribute providers that manage other and not elementary attributes related to this identity (for example, a Motor Vehicle Office plays the role of attribute provider for a driver's license).
2. *Blockchain registration*. In this proposal, users will be referred by their Ethereum address, so that any user has to create an *external owned account*, characterized by a public address and controlled by a private key. The public key derives from the private one and is computed by a cryptographic function of type *elliptic curve*

point multiplication. Typically, the blockchain address is derived from the public key by a cryptographic hash function. For example, in Bitcoin, a user with public key K has an address computed as RIPEMD160(SHA256(K)), where [196] is a cryptographic hash developed by National Security Agency (NSA) and returns a 256-bit digest, whereas [222] is a cryptographic hash designed in the open academic community and returns a 160-bit digest (i.e., the address).

3. *Credential issuing*. This operation is carried out by the user when a credential for one or more attributes is needed and involves an attribute provider (say AP[1]). According to the eIDAS protocol, AP acts as a service provider and needs to identify the user by an eIDAS-compliant scheme, as illustrated in Fig. 6.1.

   First of all, the user sends a request for a credential to AP (Step 1). Then, AP replies with an authentication request to be forwarded to the *Identity Provider* (Step 2). The *Identity Provider* performs a challenge-response authentication with the user (Steps 3 and 4). In case of successful user authentication, the *Identity Provider* prepares the assertion of user authentication, which is forwarded to AP (Step 6). This way, AP is aware of the user's digital identity and can verify if the user owns the required attributes. In this case, AP performs a second task to know the user's blockchain address (Step 7): AP sends a random string to the user and waits for receiving a blockchain transaction with this string from the address of the user, say A.

   If such a transaction is received, then AP replies to the user with the requested credential: it consists of an assertion reporting the blockchain address A of the user, the verified attributes, and a (suitable) URL belonging to the Web domain of the attribute provider (e.g., `http://www.attributeprovider.com/8f7b19f38f4c4b10b52de9727e9f0538`). Finally, the attribute provider publishes at this URL the digest of the credential as a proof of authenticity and integrity of the credential (this replaces a cryptographic signature).

4. *Credential using*. When a user needs to access a service granted only to people with some attributes (e.g., a driver's license or being of age), the user can exploit a credential for such attributes obtained with the procedure described in the previous step. The user sends the credential assessing the possession of the requested attributes to the service provider (say SP) supplying this service. The verification of the credential validity is carried out from SP by the call to a function of the smart contract SC, which receives this credential and executes the following steps: (1) extracts from the credential the value of the URL field, (2)

[1] For the sake of simplicity, let us assume only one attribute provider handles the needed attributes: in case more attribute providers are involved, this operation is repeated for each of them.

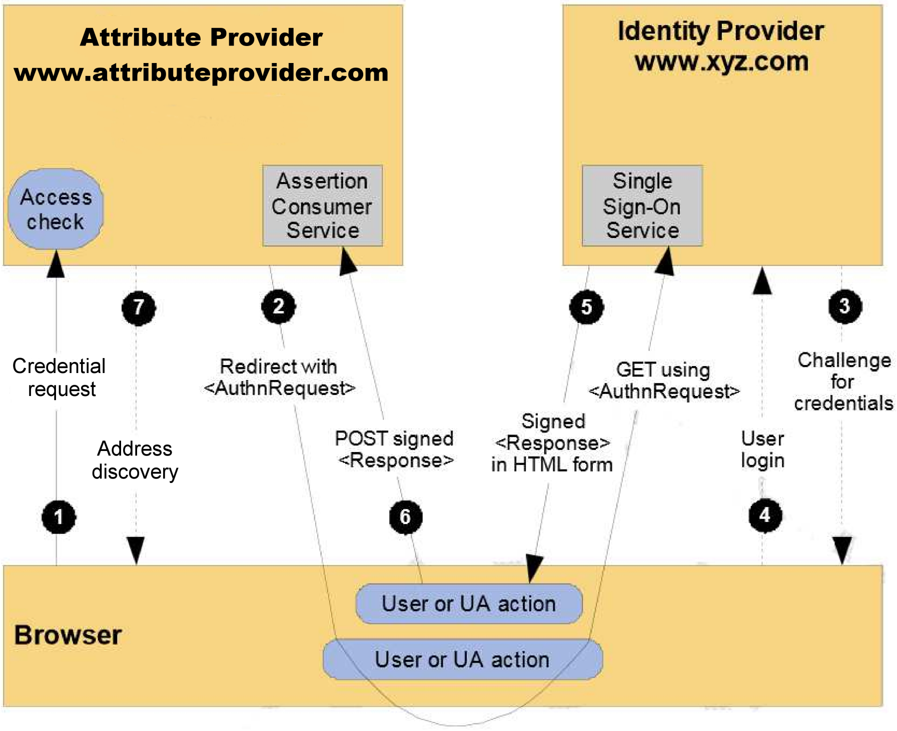


Fig. 6.1: Data flow in assertion issuing process.

verifies that this URL belongs to the domain of the attribute provider, (3) downloads the file at this URL, (4) calculates the digest of this file, say D, and (5) calculates the digest of the received credential, say D'. If and only if $D = D'$, then the function returns that the credential is valid. Only in this case, the user will be able to access the service; otherwise, the access is denied.

5. *Credential Revocation*. Revocation concerns the possibility to make invalid a credential when a user loses possession of an attribute (for example, a driver's license) or the credential is stolen or expired. Revocation is carried out by the attribute provider: for each credential to be revoked, the attribute provider extracts the specified URL where the credential is published and removes the document on this URL, making the file unreachable. This way, the user cannot provide the proof of the requested attribute to any service provider, which cannot find the credential.

The proposal here presented is defined at a conceptual level and does not consider several (orthogonal) aspects, such as the smart contract and the exchanged data. These aspects are the subject of the next section.

6.4 Implementation and proof of concept

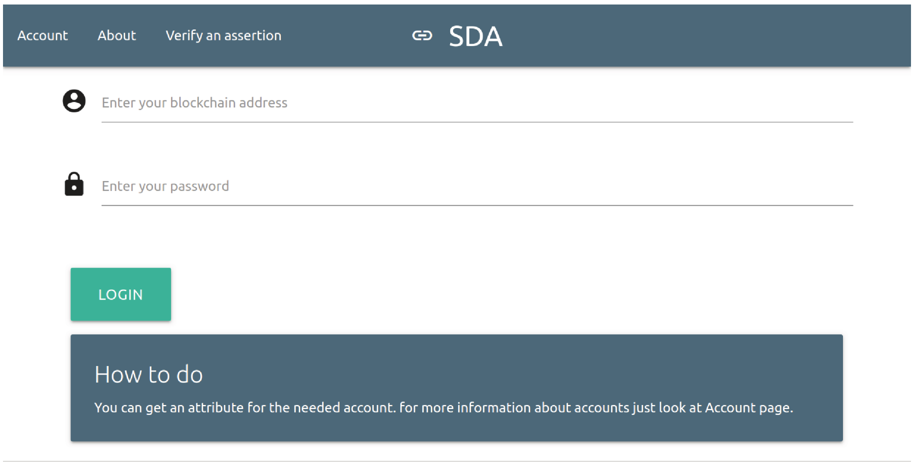
In this section, we instantiate the general approach presented above to a real-life scenario to help the reader to understand better how our solution can work. We describe the detailed data workflow to show how our approach can be implemented in a real blockchain: we used Ethereum, which is the most used one enabling smart contracts.

For the sake of presentation, we refer to a simplified scenario in which the service to access requires the user to be of age (for example, in the case of age-restricted videos, which are not visible to users who are under 18 years of age). Our implementation is based on the Ethereum blockchain, and the environment in which we tested our solution is the Ropsten testnet blockchain, a free blockchain based on Ethereum using proof of work. The smart contract is build using [15] and exploits [214] to import data from external sources. We implemented a JAVA decentralized web application (DAPP) by the [264] library and used Infura [93] as blockchain infrastructure to access the Ethereum blockchain.

Now, we describe how the operations defined in our proposal are implemented in this scenario.

1. *Digital Identity issuing*. The public digital identity of a user U can be issued in order to perform a secure authentication based on attributes. The Identity Provider IP stores a list of the elementary attributes of the user. The Attribute Provider AP manages not elementary attributes and checks the identity of the user.
2. *Blockchain registration*. We created an Ethereum address for any entity involved (user, identity provider, and attribute provider). First, we generated a couple of asymmetric keys for each of them. Then, the Ethereum address is computed from the public key by applying Keccak-256 [43] and taking the last 20 bytes of that hash. In practice, we installed the *MetaMask* extension in Google Chrome, created the new accounts, and saved the seed words for restoring the MetaMask accounts.
3. *Assertion issuing*. This operation is carried out by the user when a proof of attribute is needed.

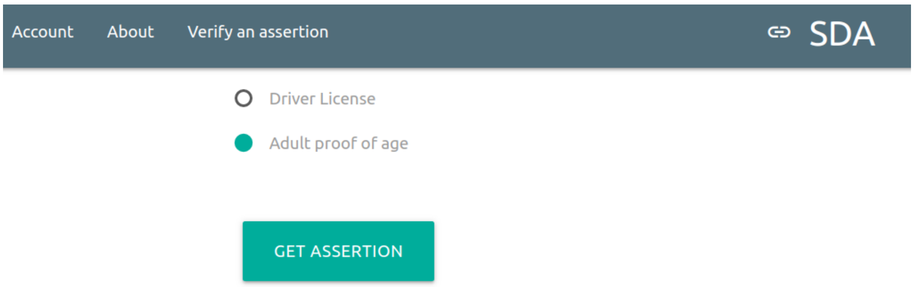
   To obtain the assertion, the user connects to the AP's site by a browser and sends the request for an assertion. Now, the user is authenticated by the chosen eID (this part is skipped in our implementation because it is a standard procedure). Then, AP replies with a challenge-response authentication with the user (see Fig. 6.2). In case of successful user authentication, the attribute provider shows a page with all the user's attributes so that the user can select the ones to be certified, as shown in Fig. 6.3.



How to do

You can get an attribute for the needed account. For more information about accounts just look at Account page.

Fig. 6.2: User authentication.



○ Driver License

● Adult proof of age

Fig. 6.3: Attribute selection.

Now, *AP* prepares a response that includes the assertion of the attributes selected by the user: this message is based on the SAML format and according to [85] the most relevant fields are:

- an attribute `Version` with the version of SAML used in the message;
- an element `StatusCode` with the outcome of the request;
- an attribute `ID` containing the assertion identifier;
- an attribute `IssueInstant`, which specifies the instant at which the assertion is issued;
- an element `Issuer`, which refers to the issuer of the message;
- an element `AuthnInstant`, which specifies the instant at which the authentication has been performed.
- an element `AuthnContextClassRef`, which specifies the used authentication method based on a particular class reference.

An example of a response message is shown in Fig. 6.1. The status code contains the URL used by *AP* to publish the assertion. The assertion contains the version, the ID, the issue instant, the reference to the issuer, the blockchain address of the subject, the end of validity, and the attribute name. Moreover, the assertion has temporal data regarding the start of validity instant, the timestamp of the authentication, and the type of certified attributes (proof of age).

```
<saml2p:Response
  xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" Version="2.0">
  <saml2p:Status>
    <saml2p:StatusCode>
          https://attributeProvider.com/zkik1ONcxdxaUIoqCZfX.xml
    </saml2p:StatusCode>
  </saml2p:Status>
  <saml2:Assertion
    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
        ID="47fd4a3a-16ea-415a-9b16-f9f034785388"
        IssueInstant="2020-01-22T09:19:29.170Z" Version="2.0">
    <saml2:Issuer>attributeProvider.com</saml2:Issuer>
      <saml2:Subject>
            <saml2:NameID
            Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
            0x8fa7173202d86C746bd884C9f116E356600c6b0E
      </saml2:NameID>
            <saml2:SubjectConfirmation
            Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml2:SubjectConfirmationData
                NotOnOrAfter="2020-07-21T09:19:29.163Z" />
      </saml2:SubjectConfirmation>
    </saml2:Subject>
    <saml2:AuthnStatement
            AuthnInstant="2020-01-22T09:19:27.138Z">
      <saml2:AuthnContext>
        <saml2:AuthnContextClassRef>
            urn:oasis:names:tc:SAML:2.0:ac:classes: proof of age
        </saml2:AuthnContextClassRef>
      </saml2:AuthnContext>
    </saml2:AuthnStatement>
  </saml2:Assertion>
</saml2p:Response>
```

Listing 6.1: Creation of a new auction

Moreover, *AP* publishes at the URL reported in the assertion a file containing the assertion and calculates the digest of this file. Finally, *AP* calls the function *indexDigest* of the smart contract (see Fig. 12.1, Lines 54-59), which generates a token. This token is the digest of a nonce generated from the block timestamp and incremented by one every time a new token is generated. This ensures its uniqueness. Observe that it is not a secret value (all data in the smart contract are not secret). This token is used as an index to find such an assertion digest in a mapping (Line 57).

Finally, the attribute provider sends to the user a JSON file containing the URL where the assertion is available, and the token *t*. An example of this file is reported in Fig. 6.2.

```
{
  "location":"https://attributeProvider.com/zkik1ONcxdxaUIoqCZfX.xml",
  "token":"4ce9b145974bad0dbeca705f89d3f44161fc8226348497cd67854 b0edcf7e465"
}
```

Listing 6.2: Example of JSON credential.

4. *Assertion using*. Suppose the user needs a service supplied by the service provider SP and U is required to possess an attribute, for example, a proof of age. To prove to be of age, U sends the JSON file received by the attribute provider in the previous step to SP.
   To verify this credential, SP calls the function *verifyAssertion* of the smart contract SC (Fig. 12.1, Lines 38-49), giving as input the `token` and the `location` extracted from the JSON file. This function is payable (i.e., it can receive Ether) and exploits the Provable oracle to download the assertion located at the given URL (Line 44). Then, it retrieves the digest of the assertion previously stored in *digestSet* (Line 57) and adds it to *verifiedDigest*, a mapping between digests and query identification (i.e., *content* in Line 45). Moreover, the query is included in a set of pending query (line 46).
   Once Provable returns the query result, the *callback* function is called automatically. This function has the oracle query identification Id and the query result as parameters. First, the digest of the assertion downloaded by Provable is calculated (Line 29) and compared to the digest previously stored in *verifiedDigest* (Line 30). The result of this comparison is stored in *resultSet*.
   Then, the Service provider SP calls the function *checkResult*, which takes the query identification Id as a parameter. This function returns the content of the *resultSet* at the index Id, which represents the result of the previous function (Lines 50-53). Specifically, the value 1 denotes a valid assertion, the value 2 denotes an incorrect assertion, whereas the value 0 denotes that the query result is not available (for example, in case of wrong id of the query).

Concerning the smart contract, we showed a more extensive implementation in which also events are used (see Lines 11-17 and the calls to function *emit*). Logged events can provide support in case of need (for example, in the event of a complaint). The implementation of this solution by a Java prototype is available on Github at the address `https://github.com/DroBaptiste/SelectiveDisclosureOfAttribute`. Concerning the smart contract, it has been deployed on Ropsten and is reported at `https://ropsten.etherscan.io/address/0x2cF05A44F23A92581088c17e7C8c7D88B2F8d0f2#code`, where the `provable.sol` library has been included.

6.5 Security analysis

In this section, we discuss how our proposal reaches the expected goals, which are:

1. the service provider should be aware of the minimal amount of the user's information needed to access the service;

2. only authorized users should access a service;
3. there should be the possibility to revoke the permission given to a user to access a service;
4. the user's identity should be revealed in case of need.

Let us start with the first property. The service provider is aware of (i) the JSON file and (ii) the assertion. The former does not contain any information about the user; the latter contains only data that the user selected. Provided that (1) there is no collusion among identity provider, attribute provider, and service provider, and (2) the user does not select more information than the needed one to access the service, the first goal is reached.

Concerning the second item, it is clear that authorized users can access the service by following the protocol. Thus, we have to prove that unauthorized users cannot create a valid assertion. In our analysis, we assume that the protocol is correctly run by identity and attribute providers because they are trusted entities. Consequently, an attacker cannot tamper with his/her identity or attributes, which are guaranteed by identity and attribute providers. When the attacker is able to create an assertion that passes all the security checks, we have to consider the following two possibilities. The attacker has created a false assertion and is able to create the signature done by the attribute provider. This means to break the cryptographic primitives or to guess the private key. The probability of any of these cases is negligible. Another possibility is that the attacker has violated the smart contract. If we assume no error exists in the smart contract code, then the adversary can only try a 51% attack {sayeed2019assessing. Again, the probability of this attack is negligible.

Concerning the attribute revocation, it can occur for several reasons: 1) most of the attributes have an expiring date from which their validity ends, 2) a user can choose to change the status of a single attribute (in our scenario attributes could be separate and disjointed) or 3) the most significant cause is when the attribute provider ought to revoke the expired attribute. How the revocation occurs is evident in the protocol. Thus, we focus on a possible misconduct of attribute providers, that is the unfair revocation of an attribute. Consider the case of an attribute provider AP that has issued an adult proof of age credential to a user U. Now assume that AP revokes this credential unfairly so that the credential validation fails. However, if U has stored this credential, then by computing the digest d of the credential and by invoking the function *indexDigest* of the smart contract (see Fig. 12.1, Lines 54-59), U can prove that a token is associated with d, and thus, that AP had issued this credential. This possible malicious behavior of AP is detected and, thus, contrasted. Clearly, in some cases, other information could be necessary to close the complaint.

As for the last goal, we reach this result by exploiting an eIDAS-compliant identification scheme. The robustness of an identification scheme depends on the degree to which it adheres to the technical specifications and best practices. Even if the standards used for identification systems can vary by country, the compliance with eIDAS ensures acceptable robustness. Our scheme allows a party in cooperation with other trusted parties to guess the identity of a user in cases of need. This possibility is given because each credential has an identifier, and each attribute provider stores the mapping between each generated credential and the user identity. Recall that a service provider is not aware of the identity of the user accessing the service, and the only information known is about the user's attributes certified by the credential. However, in case of valid reasons (for instance, a terrorist who rented a car), the cooperation between the service provider (which knows the credential identifier) and the attribute provider that issued such a credential (which knows the user's identity associated with this credential) allows us to uncover the user's identity.

Concerning this aspect, we observe that there is the possibility of running offline the scheme by relaxing some of the security requirements. Consider the case in which the URL is not available: the protocol should deny the request of access request because the credential cannot be retrieved. However, as it is done for micropayments when the economic value of the service is limited, the service provider could accept to receive the credential from the user instead of downloading it from the (unavailable) URL. Observe that in this case, the credential verification is done again by the function *indexDigest*, thus proving that a token is associated with this credential. Clearly, in this way, the check of credential revocation has not been carried out: however, if the cost of making unavailable the web site of the attribute provider (for example, by a Denial-of-Service attack) is higher than the price of the service, it is not advantageous for a malicious user to run this attack for obtaining the service. The choice about providing the service at this risk is left to the service provider.

```

 pragma solidity >= 0.5.0 < 0.6.0;
 import "./provable.sol";

 contract verificationContract is usingProvable {
 mapping (bytes32 => bytes32) private digestSet;
 mapping (bytes32 => bytes32) private verifiedDigest;
 mapping (bytes32 => bool) private pendingQueries;
 mapping (bytes32 => uint) private resultSet;
 uint nonce;

 event LogConstructorInitiated(string nextStep);
 event LogVerification(string saml);
 event LogNewProvableQuery(string description);
 event AssertionResult(uint answer);
 event TokenIndexed(bytes32 token);
 event queryInitiated(bytes32 id);
 event TransactionMade();

 constructor() public payable {
     nonce = block.timestamp;
     emit LogConstructorInitiated("Constructor was initiated");
 }
 function pay() payable public {
     emit TransactionMade();
 }
 function __callback(bytes32 myid, string memory result) public{
     if (msg.sender != provable_cbAddress())
         revert();
     require (pendingQueries[myid] == true);
     bytes32 digest = sha256(abi.encodePacked(result));
     if (verifiedDigest[myid] == digest) {
         resultSet[myid] = 1;
     } else {
         resultSet[myid] = 2;
     }
     emit LogVerification(result);
     delete pendingQueries[myid];
 }
 function verifyAssertion(bytes32 _token, string memory _url) public payable{
     if (provable_getPrice("URL") > address(this).balance) {
         emit LogNewProvableQuery("Oraclize query was NOT sent, please add some ETH to cover for the query fee");
     } else {
         string memory str = strConcat("binary(",_url,").slice(0,10000)");
         emit LogNewProvableQuery("Oraclize query was sent, standing by for the answer..");
         bytes32 content = provable_query("URL", str ,300000);
         verifiedDigest[content] = digestSet[_token];
         pendingQueries[content] = true;
           emit queryInitiated(content);
     }
 }
 function checkResult(bytes32 id) public returns(uint) {
     emit AssertionResult(resultSet[id]);
     return resultSet[id];
 }
 function indexDigest(bytes32 _digest) public{
     bytes32 token = sha256(abi.encodePacked(nonce));
     nonce++;
     digestSet[token] = _digest;
     emit TokenIndexed(token);
 }
}
```

Listing 6.3: Code of the smart contract.

6.6 Conclusion

In this chapter, we proposed a system that allows users to prove the possession of some attributes without disclosing their whole identity. Our solution relies on blockchain, which is a very recent technology but considered mature and already widely adopted in many application contexts. The use of a blockchain platform such as Ethereum allowed us to make transparent cryptographic operations to user and system, thus making the solution more effective, robust, and secure. Using certain features of blockchain, the proposed system provides suitable mechanisms for revocation and accountability. The proposed solution has been implemented by a JAVA decentralized Web application that exploits Ethereum as blockchain. As a real scenario for validation, we considered an infrastructure compliant with the eIDAS Regulation, in which the attribute providers defined by eIDAS are authorized to certify a qualification or some attributes of a digital identity.

In this chapter, we described an example in which a company needs to know if a user is of age but collects the user's date of birth. This data processing violates the minimization principle and, according to Article 83 of [104], this infringement is subject to administrative fines up to 20M EUR or, in the case of an undertaking, up to 4% of the total worldwide annual turnover of the preceding financial year.

This aspect highlights the practical importance of our solution: the scheme we proposed allows a company to comply with the data minimization principle stated by GDPR, yet ensuring that access-control policies are respected. Although the use of our solution does not allow a company to know the identity of users accessing a service, in case of valid reasons (for instance, a terrorist who rented a car), it is possible to uncover user's identity with the support of a trusted party, which is the attribute provider that certified user's age. This is an important added value of our proposal, especially in this period in which the balance between privacy right and security right is difficult to determine.

7

Allowing Privacy-Preserving Fog Computing with Digital Identity Assurance

Nowadays, there is an increasing demand for cloud-based remote clinical services, both for diagnosis and monitoring. The COVID-19 pandemic has dramatically amplified this need. E-government programs should quickly go towards the expansion of this type of services, also to avoid that people (especially elderly) renounce treatment or adequate health care. However, to be effective, latency between IoT medical devices and the cloud should be reduced as much as possible. For this reason, fog computing appears the best approach, as part of the elaboration is moved closer to the user. However, some privacy threats arise. Indeed, these services can be delivered only based on secure digital identity and authentication systems, but the intermediate fog layer should learn nothing about the identity of users and the link among different service requests. In this chapter, we propose a concrete solution to the above issue by leveraging eIDAS-compliant digital identity and by including a cryptographic protocol to provide anonymity and unlinkability of user's access to fog servers.

7.1 Introduction

The use of information technology and communication to improve the delivery of information and services to citizens is one of the main goals of e-government. *Remote clinical services*, which consist in the digital transmission of medical information for remote medical diagnosis and monitoring, are assuming a very important role, also due to the increasing need determined by the COVID-19 pandemic [84, 150]. This claim is widely recognized, even just by referring to *remote patient monitoring*, which allows continuous, real-time, non-invasive monitoring through wearable devices that wirelessly transmit patient information to a healthcare entity.

Therefore, e-government programs should devote a lot of resources to the development of these services, which give considerable benefits to public health, also beyond the emergency we are living.

Large-scale services, especially those requiring a certain server-side computational load, must be thought of as services provided under the cloud paradigm. For massive and ubiquitous remote clinical services, it would be anachronistic not to use this approach, because a traditional client-server solution does not scale [19].

Moreover, cloud-based solutions give many other advantages: the reduction of the size of data centers, the dynamic adaptation of power computation for peak times and low-use times, the improvement of worker collaboration by allowing dispersed groups of people to meet virtually and share information in real-time, the availability of data and applications independently of the part of the world where a worker is.

However, in the case of remote clinical services, there is a specific issue to consider. Indeed, the latency between IoT medical devices and providers is critical. Therefore, standard cloud-based architectures are not the best solutions.

To overcome this issue, fog computing has been proposed [50]: it extends the cloud close to the device that produces or generates the data, exploiting its network connection, storage, and computing features. In this case, the device is known as a *fog server*, and examples include switches, routers, cameras. The literature has shown that fog computing is the most promising solution to reduce latency without renouncing to the cloud paradigm [105, 278, 191, 181]: indeed, this approach allows a user to communicate only with the closest fog server, which queries the cloud only when this is necessary. Moreover, by designing a suitable scheme for moving user's requests from a fog server to another fog server when a user moves, we can enhance the user's mobility [178]. In fact, mobility is another important feature to take into account in our application setting.

Besides the numerous advantages such as efficiency, low-latency, resource-load optimization among others, some security issues should be considered in the context of e-health, especially remote clinical services [144]. For this reason, the provision of services should be done by adopting strict security measures, among which the user's identification with a high level of assurance. Therefore, strong mechanisms to manage digital identities and authentication should be used by the cloud provider. However, in a solution adhering to the fog computing paradigm, there is an intermediate layer to consider, also from the security and privacy point of view. Indeed, the layer between the cloud and user introduced by fog computing belongs to third parties that should be not aware of the real-life identity of users as well as the content of their interactions with the provider, even though users have to be authenticated by fog server to allow service delivery [51, 129, 64]. Also, the possibility for a fog server to link different (even anonymous) interactions would be an intolerable privacy leakage.

We provide a solution to the above trade-off by proposing a fog-computing-based approach for remote clinical services, which guarantees the security level and the technological features introduced by the eIDAS Regulation [187] for the identification and authentication of users. Indeed, this approach can solve the security issues related to the access to a service relying on fog computing [232]. The privacy threats introduced from the fog middleware are contrasted by using a cryptographic protocol that supports anonymity and unlinkability while ensuring strong authentication.

This solution takes origin from the proposal given in [64], which basically focuses on the security of the device authentication by allowing a device to be authenticated by a fog server without sharing any secret and using the same credential for any fog server. However, the solution given in [64] is not secure in the adversarial model of honest-but-curious fog servers considered in this scheme because the authentication with the fog server is based on the user identity. In fact, this solution overcomes the above drawbacks.

7.2 Fog Computing

Cloud is migrating to the edge of the network, and the components of the network are aligning towards a virtualization infrastructure, called *fog computing*. Fog computing extends the cloud computing paradigm to the edge of the network and facilitates innovative applications and services for IoT devices [50]. This emerging model provides the end-users with some advantages such as mobility, low latency, and location awareness related to a widespread geographical distribution of nodes. These advantages are suitable for a wide number of applications in the fields of Smart Cities, Grid or Wireless Sensors, and Actuators Networks.

As mentioned above, fog computing cooperates with cloud computing. In Figure 7.1, it is represented a scheme of the three-layers infrastructure made of the cloud and fog computing, and the end-user.

Fog computing faces new security and privacy challenges besides those derived from cloud computing. The authors of [190] observe that the existing security and privacy measurements for cloud computing cannot be directly applied to fog computing due to its features. Indeed, the surveys [144, 277] individuate the security challenges and give the corresponding solutions about trust and authentication, network security, secure data storage in the fog computing technology. These papers face up different security areas and highlight the recommendation to take among main applications such as healthcare systems, vehicular networks and road safety, video stream processing.

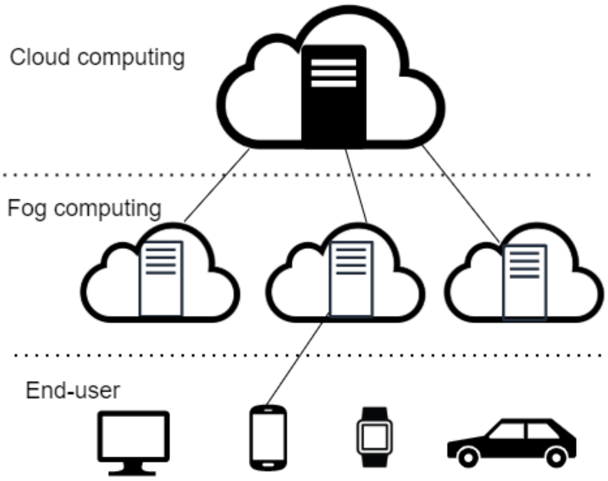


Fig. 7.1: Example of Fog Computing architecture.

In a fog computing application, the users' privacy has to be preserved: particularly, location privacy and usage pattern privacy are two very relevant issues. Although these two problems have been studied and addressed for various applications, the study [119] provides a solution to maintain the best possible delay and energy consumption performance, still considering the privacy protection of users.

The authors of [281] identify the access controls problems related to fog computing technology: users must be authorized by the cloud or the fog servers to access a resource or a service, and at the same time, fog servers and cloud need to be authenticated reciprocally. Access control models are presented to highlight their application and their aim of protecting user's privacy and ensure system security in an environment of fog computing.

The study [276] underlines that the access of fog computing services and resources needs to be authenticated and authorized. The solution they provide is a prototype fog computing platform, evaluating users' privacy awareness in order to preserve them from some known attacks. Account hijacking is an attack based on social engineering, in which, after having stolen credentials, an attacker simulates the victim's behavior in the network. On the other hand, considering the access to a fog server, the insider threat is a malicious threat [245] that involves an internal actor of an organization who could gain the network access with no fair intentions.

Fog computing applications are growing, and the requirements of fog platforms have to be diversified for the specific needs. The study [20] presents a representative collection of actual or proposed solutions based on the fog computing paradigm. By dividing the proposals into categories, the authors highlight the specific features that fog platform designers can follow during the development of the application.

Nowadays, the number of connected IoT devices is rising challenges into various sectors, such as healthcare, energy, smart cities, education. The presence of these different types of devices and technologies connected simultaneously raises some problems.

A relevant aspect to be addressed is the security related to safe and reliable operations of IoT-connected devices. As suggested in [217], identity-based cryptography (IBC) plays a promising role in IoT: a feasibility study of the applicability of IBC in IoT is proposed.

The study [278] proposes some solutions to improve current security systems and protocols and aim at addressing security and privacy challenges in fog computing.

The authors of [232] propose a smart hub to provide real-time information in a public environment, such as an airport, and based on fog computing. They analyze the security and privacy issues to provide users with good service. The eIDAS Regulation is taken into account as a possible option to carry out the registration at the system.

A lightweight privacy-preserving data aggregation scheme is proposed in [177]. The authors demonstrate that this scheme is privacy-preserving and resistant to false data injection from external attacks. An approach against stolen-device attacks is proposed in [51]. A Physically Unclonable Function enables secure authentication and message exchange among the IoT devices, and the proposed scheme provides identity-based authentication and repudiation and achieves an efficient key agreement between two IoT devices connected to the same authentication server.

Fog computing presents many advantages [244] for different scenarios, such as smart traffic lights and connected vehicles or software-defined networks. The authors focus on a man-in-the-middle attack, in which gateways used as fog servers may be compromised or replaced by malicious ones. Indeed, the attacker takes control of gateways and even of victims' communications, and, then, exploits cascading vulnerabilities related to users and fog infrastructure.

Inside a platform that provides a carpooling service, users' sensitive information could be disclosed at the expense of their privacy. In order to address this issue, a solution based on fog computing is proposed in [164]. Specifically, the authors highlight the privacy and security aspect of the solution: a private blockchain is used to store carpooling records. Platform users, such as passengers and drivers, perform anonymous authentication and encrypt data before transmission.

It is evident that fog computing and its security issues are relevant topics in the literature. To the best of our knowledge, the solution we propose is the first one (1) exploiting eIDAS-compliant digital identity for the identification and authentication, and (2) preserving privacy and unlikability of users among different fog servers.

7.3 Scenario and problem formulation

In this section, we present a general scenario and define the security problems our proposal solves.

Generally, fog servers can acquire and process data sent from authorized users: for example, if a user is near a fog server, she/he can decide to communicate with this server instead of the cloud. When a user moves from a fog server to another fog server, the service used by the user can be provided by the latter fog server after verifying the authorization of the user.

In the considered scenario, we can identify the following actors:

- Users, who are the owners of processed data.
- Users' devices, which are health devices, typically wearable, generating user data.
- Cloud servers, a group of computers connected over the Internet, providing storage and computing power available on-demand by users.
- Fog servers, a middle layer between the cloud and the users, enabling efficient data process. Fog servers do not know each other and could be untrusted.
- The Identity Provider, an entity that creates, maintains, and manages the user's identity information and provides an authentication service. Note that users can have different digital identities, for example, issued by different Identity Providers: this is encouraged to increase system resilience. Clearly, in the case of multiple identities, a user chooses the Identity Provider to use for authentication.

The problem we face is to strengthen classical solutions of fog computing by achieving the following objectives:

1. the solution should be resistant against stolen-device attacks, which occur when an adversary has the physical possession of the device of the victim so that device-based authentication can be performed successfully. For example, in the literature, there exist several solutions that exploit *Physical Unclonable Functions* (PUFs), which are low-cost primitive exploiting the unique random patterns in a device and are applied for secure key generation and key agreement [51].
2. for privacy reasons, a fog server should not know the identity of the user exploiting the service;

3. again for privacy reasons, the unlinkability of the accesses of the user to the same fog server in different moments should be guaranteed.

7.4 Existing solution and improvements

In this section, we describe the solution presented in [64] on which our proposal is based.

We consider a initial situation in which a user is close to Fog Server A and is exploiting this server for storage and computing health data. For example, the user could be at home. Then, the user moves to another place near Fog Server B, which will be in charge of continuing the job done by Fog Server A.

In this scenario, we aim at strengthening classical solutions of Fog computing against several attacks, and in particular against stolen-device attacks. A stolen-device attack occurs when an adversary has the physical possession of the device of the victim, so that it can be performed successful device-based authentication (for example, when only PUF is adopted [51]). We contrast this attack because in our solution, the authentication is carried out by requiring also information known to the user.

The steps that we can identify in our solutions are described in the following.

Digital Identity Registration. In this step, users obtain their digital identity by the identity provider. The identity provider, after verifying user's personal data, issues the digital identity and credentials. The digital identity is a set of (personal) data containing at least the following attributes:

- a string PersonIdentifier, which is an identifier of the digital identity;
- a string FamilyName, the surname of the user;
- a string FirstName, the name(s) of the user;
- a date DateOfBirth, the date and year the user was born.

The user's credential is a pair ⟨username, password⟩ that the user will exploit to authenticate. In general, since there can be more identity providers, a user can be associated with one or more digital identities (for example, for redundancy reasons, in case one identity provider is not available).

Authentication. This step starts when the user U is close to the Fog Server B, so that there is the need to transfer data from Fog Server A (previously used by the user) to Fog Server B, typically because U moved and is closer to B than A. First, by a web request, U sends to the Fog server B a request for data transfer (Step 1 in Figure 7.2).

Then, the user exploits her/his digital identity for authentication with the Fog server B. In particular, the Fog server B authenticates the user by an eIDAS-compliant scheme.

The Fog server B sends the user a request for authentication to be sent to the Identity Provider; the request is forwarded by the user's browser to the Identity Provider (Step 2). This authentication request contains the reference to the Fog Server B, as starting the request.

The Identity Provider verifies the correctness of the request received and, if it is valid, carries out a challenge authentication with the user. The user authenticates by the credentials issued in the phase Digital Identity Registration, described above (Steps 3 and 4).

If the user successfully completed authentication, the Identity Provider prepares the assertion containing the user's authentication statement intended for the Fog Server B. This assertion is returned to the user by the Identity Provider and forwarded to the Fog server B which verifies whether the authentication has been successful (Step 5).

In the positive case, user's data are moved from the Fog server A to the Fog server B. In general, Fog Server B could not know where user's data are (i.e., they are stored by Fog Server A) or could not know how to contact Fog Server A. For these reasons, the transfer of user's data from Fog Server A to Fog Server B cannot be done directly from these servers, but has to pass through the Cloud.

This operation is carried out as follow: initially, Fog server B makes a data request to the Cloud (Step 6), which contains the assertion of user's authentication. The Cloud verifies the authenticity of the request and the assertion and looks for where user's data are, that is Fog Server A. Then, the Cloud forwards this request to Fog server A (Step 7), which transfers user's data to the Fog server B (Step 8).

With regards to the three objectives defined in Section 7.3, we note that the proposal presented in [64] reaches only the first objective (i.e., it is resistant against stolen-device attacks), but fails with respect to objectives 2 and 3. In this solution, we provide an improvement of [64] by defining a technique that is able to guarantee users privacy by preventing fog server from knowing user's identity (objective 2) and from linking different accesses of the same user to the fog server (objective 3).

7.5 Our solution

In this section, we describe our proposal to improve the security of fog computing in a scenario of mobility and in which the users' privacy is considered a relevant issue. Although the chosen application setting is that of remote clinical services, in

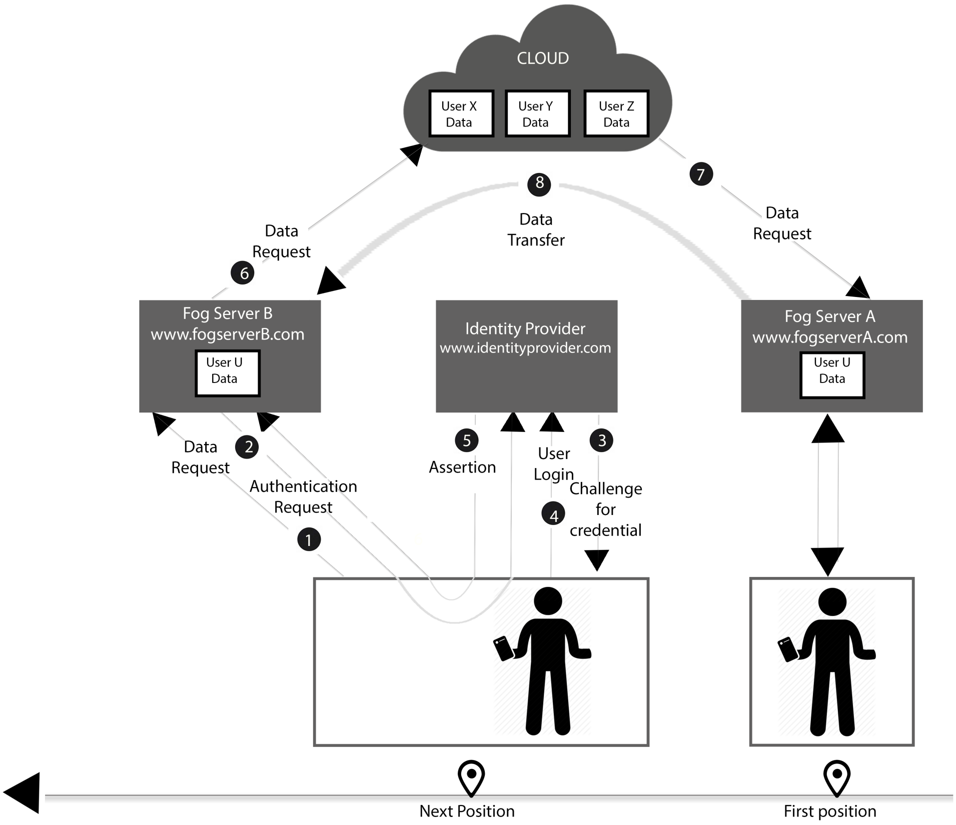


Fig. 7.2: Solution presented in [64].

principle the solution can be considered from a more general point of view. As a matter of fact, no different real-life application setting mixes all the features that motivate our research.

The solution we present exploits *anonymous credentials*, which allow an entity to prove statements about itself and its relationships with other entities anonymously.

We suppose that users are provided with their public digital identity to perform the authentication by the cloud and fog servers in an eIDAS compliant scheme. Indeed, the user is provided with a pair of ⟨username, password⟩, and these credentials are used to access a service or a resource granted by service providers (cloud or fog server).

We introduce the notation used in the following. We define the set $F = \{f_0, \dots, f_n\}$, where f_0 is the cloud and $f_1, \dots, f_n$ are fog servers. Moreover, $\mathrm{ID}(f)$ denotes the identifier of the fog/cloud f.

We define the element *(anonymous) credential* as $C = \langle ID, \text{exp\_time}, \mathrm{ID}(f_o), \mathrm{ID}(f_d), C(f_o), K, s \rangle$, where:

- ID is the identifier of the credential, which is usually derived from the timestamp of the issuing.

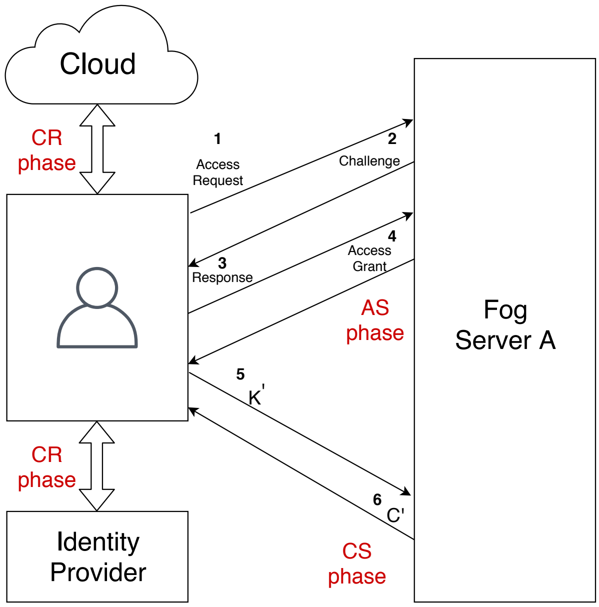


Fig. 7.3: Data flow in our solution.

- exp_time (expiration time) is a determined date or time after which the credential should no longer be used. The validity of a credential is set on the basis of the specific application (so that it is not a core aspect in this solution).
- f_o and f_d are two distinct elements of the set F, which are said *(credential) origin fog* and *destination fog*, respectively.
- $C(f_o)$ is the certificate of the origin fog f_o. This field is optional and can be set to *null*.
- K is a public key.
- s is the signature of the credential.

Now, we are ready to present our proposal, whose phases are schematized in Figure 9.1 and described in the following:

Setup. This phase is carried out at the beginning. Here, the cloud generates a certificate, based on the standard X.509 [74], for each fog server. A certificate is signed by the cloud and contains information about the fog server (such as its identifier) and the fog server's public key. Each fog server secretly stores the corresponding private key. Observe that the cloud has a certificate too, which is self-signed and is known by all fog servers.

Moreover, users generate a digital identity by an Identity Provider. After verifying the user's data, the Identity Provider issues the digital identity and access

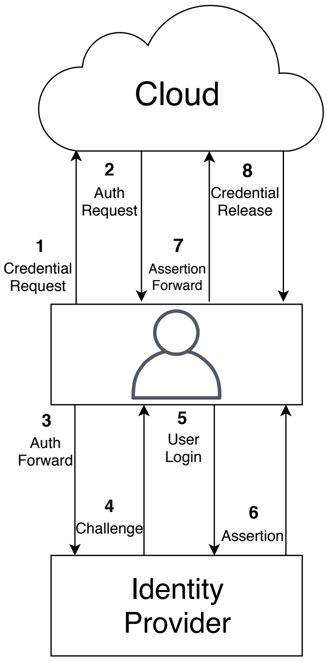


Fig. 7.4: Credential release phase.

information. The digital identity is a set of (personal) data containing at least the following attributes (according to the eIDAS scheme) that we recall are:

- a string PersonIdentifier, which is an identifier of the digital identity;
- a string FamilyName, the surname of the user;
- a string FirstName, the name(s) of the user;
- a date DateOfBirth, the date and year the user was born.

The user's access information is a pair ⟨username, password⟩ that the user will exploit to authenticate. In general, since there can be more Identity Providers, a user can be associated with one or more digital identities (for example, for redundancy reasons, in case one Identity Provider is not available).

CR. In this phase, said *Credential Release*, the user is authenticated by the cloud and receives an *(anonymous) credential* that will be used next. Specifically, the user exploits her/his digital identity for authentication with the cloud by an eIDAS-compliant identification scheme.

The operations carried out in this phase are detailed in Figure 7.4. First, the user contacts the cloud to request the credential (Step 1 of Figure 7.4); then, the cloud sends the user a request for authentication (Step 2), which is forwarded to the Identity Provider by the user's browser (Step 3). The Identity Provider verifies that the received request is valid (i.e., it is in the expected format and is signed by the sender), and starts a challenge authentication with the user. The user authenticates by the access information issued in phase Setup (Steps 4 and 5). If the user completed authentication, the Identity Provider prepares a response including an assertion, which is returned to the user by the Identity Provider (Step 6) and forwarded to the cloud (Step 7). In case of valid assertion, the user authentication successes (Step 8).

Now we explain how this credential is used. Suppose that the user needs to connect to the fog server f (because it is the closest one). A pair of asymmetric cryptographic keys (K_p, K_s) is generated: the private key K_s is known by the user, the public key K_p is also known by the cloud. Moreover, the cloud generates and releases to the user an anonymous credential in which $f_o = f_0$ (i.e., the origin fog f_o is the cloud f_0), $f_d = f$, $C(f_o)$ is null, $K = K_p$, and s is the signature of this credential done by the cloud.

AS. *Access to the Service*. Once the user has acquired a credential C, she/he can use it for authentication with a fog server, said f (e.g., the fog server A in Figure 9.1). First, the user sends a request to f that includes the credential C described above (Step 1 of Figure 9.1). Observe that the fog server cannot know the identity of the user from C, because no identifying information is included in C. The fog server f verifies the validity of C carrying out the next checks:

1. f extracts the public key of the credential signer f_o from the certificate $C(f_o)$. Observe that if $f_o = f_0$ (i.e., the certificate has been released by the cloud), the public key needed for the verification is extracted from the cloud certificate, which is publicly available. Then, f extracts the signature s and this signature is verified to guarantee the integrity and authenticity of the credential. In the positive case, the fog server keeps on the other checks.
2. The credential has a validity time, which, if expired, enforces the deny of the user request.
3. The fog server checks that the value of ID(f_d) in the credential is correct.
4. Each fog server maintains a list of already received credentials so that f checks that the value of the filed ID of C is not included in this list. Moreover, this ID is now added to this list.

After the validity of the credential is checked, f randomly generates a value x as a challenge, encrypts x by the public key K included in C, and transmits this information to the user (Step 2), who must return the initial value x, thus proving that she/he was able to decrypt the challenge (Step 3).

If all the above checks succeed, then the fog server accepts the user's request and grants the service (Step 4).

CS. Credential Switch. In a scenario of mobility, it may occur that while a fog server f_a is elaborating the user's data, the user moves from a point close to another fog server f_b. The user could exploit the current credential to request access to f_b. However, in this case, the third issue described in Section 7.3 arises (i.e., we cannot guarantee the unlinkability of the accesses of the user). The phase credential switch is carried out by the user to solve this problem. Specifically, a new pair of asymmetric cryptographic keys (K'_p, K'_s) is generated, in which the private one K'_s is known only by the user (Step 5 in Figure 9.1). Then, f_a generates a new anonymous credential C' in which $f_o = f_a$, $f_d = f_b$, and $K = K'_p$ (Step 6). Now, the user has a new (different) credential and can exploit this credential to access f_b. Clearly, this credential is verified by the procedure described in phase AS. This way, the fog server f_a can authorize the user to access a service provided by another fog server f_b, without relying on the cloud.

By comparing our solution with the one described in Section 7.4, we observe that the number of eIDAS-based authentications is reduced since anonymous credentials are used instead of that authentication. Moreover, the use of anonymous credentials is more efficient (and less invasive) than eIDAS authentication, because no interaction with the user is needed.

7.6 Security analysis

In this section, we discuss the security of our solution. We start from our threat model: we assume that any fog server can be an honest-but-curious adversary (i.e., a legitimate participant in the system that not deviates from the defined protocol but attempts to learn all possible information from legitimately received messages [173, 275]). We assume no collusion attack occurs [111]: thus, we do not consider the possibility that two or more fog servers collude each other to break the security properties.

We observe that all messages and credentials exchanged by the parties are signed by the sender, which ensures their integrity and authenticity.

Concerning the Requirement 1 listed in Section 7.3 (i.e., robustness against stolen-device attack), we observe that the use of digital identity allows us to contrast stolen-device attacks because we implemented a two-factor authentication [182]: indeed, users authenticate by something they know (eIDAS password) and something they have (the device).

The second security property requires that any fog server does not know the identity of the user using the service: this is guaranteed because the (anonymous) credential defined in Section 10.2 does not contain any personal information about

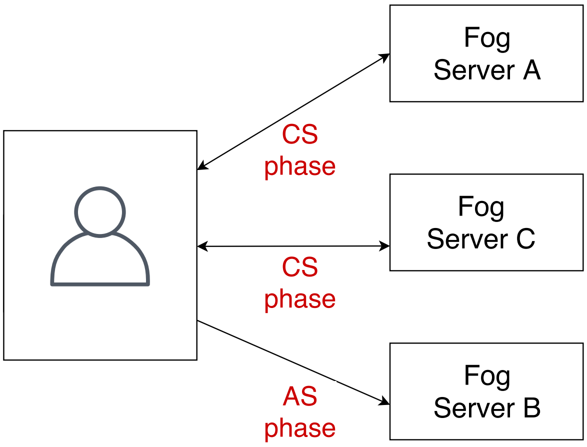


Fig. 7.5: Credential switch phase.

the user. Only the cloud knows this information, and without the collusion with the cloud (as assumed in our threat model), no fog server can guess the user identity.

The third security property requires the unlinkability of the accesses of the user to the same fog server in different moments. Concerning this aspect, consider that the credential-switch phase is carried out to generate a sort of *authentication token*, which is used to access another fog server without the need to contact the cloud or to provide any identifying information. Moreover, since this token changes each time, users accesses are unlinkable, and no tracking of users is possible.

Observe that, after a phase Credential Switch having f_a as origin fog and f_b as destination fog, when the user accesses f_b, the latter might guess that the user comes from f_a. To avoid this, we allow other switches involving further fog servers. This way, the user introduces *obfuscation* in such a way that the above information (i.e., the fog server previously used by the user) cannot be guessed with certainty. An example is provided in Figure 7.5, where the user contacts the fog server f_c before accessing f_b in such a way to simulate to come from f_c instead of the actual fog server f_a.

Note that each fog server stores the credential received by a user to access the service. This is done to avoid that someone re-uses a credential before the expiration. Moreover, replay attacks are avoided because an adversary who eavesdrops or intercepts a credential cannot fraudulently use it to impersonate the user. Indeed, the adversary cannot respond to the challenge because she/he does not know the private key to be used to decrypt the challenge.

7.7 Use case

This section aims to help the reader to understand how our solution works through the description of a use case that is relevant from the application point of view.

Consider an e-health ecosystem: therein, patients are increasingly becoming central in healthcare, and the IoT technology can enable this process towards patient-centric healthcare. Fog computing is a strategic technology able to fulfill the requirements of computing, real-time interactions, data storage, and network connectivity for the IoT devices connected to the cloud. Furthermore, fog servers are closer than the cloud to the medical devices that produce data, thus reducing the latency and traffic towards the cloud [152, 249, 151].

In this use case, fog servers process and filter personal e-health data. A user is provided with an IoT medical device used to monitor and analyze her/his heart rate. The device is wearable and has limited computation power and resources so that collected data should be sent directly to the cloud, which analyzes them and provides the user with the required results of the analysis. By adding the middle layer of fog computing, the elaboration of such data is carried out closer to the user. However, the adopted solution should offer the following features:

- for privacy reasons, a fog server should not know the identity of the user exploiting the service;
- again for privacy reasons, the unlinkability of the accesses of the user to the same fog server in different moments should be guaranteed.

The proposed solution guarantees these two characteristics. Indeed, users gain access to the fog server through anonymous credentials so that their identities are hidden to the fog server. The second feature to guarantee concerns unlinkability. We observe that in the protocol defined in our proposal, users who need to access a service contact various fog servers during the credential switch phase (see Figure 7.5). At the end of this phase, an anonymous credential is returned chosen among the credentials obtained by the fog servers. Moreover, users can collect as many anonymous credentials as they want; however, they use each credential only one time to avoid that reusing the same credential makes it possible to link their accesses and elaborated data.

7.8 Conclusion

Fog computing is an emerging topic and aims at extending the benefits of the cloud by improving its effectiveness and efficiency in providing mobile users with data and applications by exploiting the awareness of their location. However, moving data and applications from one fog server to another one raises several security and privacy problems.

In this chapter, we focused on two privacy issues: a fog server should not know the identity of the user, and it should be guaranteed the unlinkability of the accesses of the user to the same fog server in different moments.

Indeed, we proposed a solution which exploits the authentication mechanism offered by the EU Regulation eIDAS, thus directly exploitable by all EU citizens.

Thanks to the adoption of our proposal, a company can exploit the advantages of fog computing keeping the compliance with the GDPR. Indeed, the main implication of our study is related to the possibility of offering a solution for using fog computing in a way that is compliant with the GDPR principles, and, in particular, with the principle of data minimization, which limits data processing to only data that are necessary in relation to the purposes for which they are processed. Indeed, in many applications, knowing the identity of users or linking different accesses of the same user do not respect the data minimization principles.

Part III

Privacy and Accountability

In the online and offline world, we experiment with the constant need to keep personal information confidential. This need is crucial in situations where data is highly sensitive and concerns our private life. As a matter of fact, it is worth noting that the leakage of such information to unauthorized subjects can harm the health and safety of each of us. Therefore, the technology and bodies in charge of control have to ensure the appropriate use of data. However, while it is necessary to guarantee this right, on the other hand, there are situations in which it is required to keep track of actions and responsibility for actions. Therefore, there is this continuous need to balance the privacy of online users on the one hand and but to develop mechanisms that make it possible to attribute responsibility for actions on the other hand. This part of the thesis will present privacy-preserving solutions that simultaneously consider the accountability requirements, which are strongly necessary for some specific scenarios. Furthermore, we will deepen how such solutions can guarantee access to confidential information only to authorized persons when needed.

In Chapter 9, we face some relevant security challenges in the context of access control by proposing an access control scheme relying on blockchain technology. This scheme guarantees either the anonymity of users and the unlinkability of their different requests by exploiting different blockchain addresses at each interaction with the other entities. At the same time, we address the need of guaranteeing the accountability of such requests by linking the users' blockchain addresses in a verifiable chain stored by several parties in a distributed way. This research has been published in [155].

Another need occurs in the context of attribute-based service delivery where the power of attribute-based encryption schemas and the blockchain technology meet the consumers' privacy and accountability requirements. For this reason, in Chapter 10, we elaborate a practical solution that integrates the features of Ethereum's smart contracts with a (Ciphertext-Policy) Attribute-Based Encryption scheme. To show the effectiveness of our proposal, we instantiate the general model to the real-life scenario of service delivery [53].

Likewise, during energy-based transactions, users' privacy and data confidentiality should be considered. Indeed, Chapter 11 highlights how accountability and privacy still need to coexist in this scenario. We propose a solution for energy trading in smart grids based on Ethereum blockchain and smart contracts. The protocol achieves the authentication of users and allows the tracking of energy-driven transactions logged and stored in a public blockchain.

Assuring only authorized access to sensitive e-health records while preserving patient's privacy is still a challenge, especially when different healthcare organizations maintain these records. In Chapter 12, we propose a solution that allows the sharing of e-health records guaranteeing access only to authorized entities and avoiding the linkage between patient's identity and e-health records. The proposal relies on a public blockchain representing an entity offering a proper trust level of the entire system to patients and offering the needed automatism to the different phases.

8

Related Work

This section surveys the most relevant proposals related to the balance of privacy and accountability in different applications. First, we focus on the most recent access control proposals exploiting also blockchain technology. Then, we discuss some privacy-preserving solutions to protect users' privacy while securely granting access to several services.

In [233], the authors discuss various access control policies already proposed in the literature. Access control is considered as a relevant requirement of any information management system to protect users and resources from unauthorized accesses. Various access control models have been studied to preserve the information protection. The authors of [200] propose a model named T–RBAC and based on the role-based access control. The name of this model denotes the importance of the task in an enterprise environment where T–RBAC is supposed to be used.

The Usage Control is a promising approach to handle the access control process in an information system [204]. This model puts together access control, trust management, and digital rights management for controlling the usage of digital information objects. The proposed solution enables finer-grained control with privacy issues in enterprise and non-enterprise environments. The components involved in the systems are the subjects, objects, and the policies.

In [235], the authors provide a literature review and a taxonomy of the current ABAC models. They highlight the open or unexplored problems, such as the scalability, the delegation, and the suitability of proposed solutions. The paper [227] deals with the implementation of an anonymous authentication in a decentralized access control scheme in the cloud for secure data storage. In the proposed system, the cloud is in charge of verifying the users' authorization without knowing their identity. Moreover, the access policy for each stored record is managed by the cloud.

The authors of [160] highlight the importance of an anonymity-based authentication and implement a blockchain-based RBAC model that provides role-based access control. The model is simulated on an Ethereum-based through the use of smart contracts, and the authors claim that their technique is more efficient in gas use than the existing RBAC model.

Nowadays, the security and privacy issues in the Internet of Things are enormous. The paper [202] presents a distributed access control framework, named FairAccess, which is based on blockchain. The authors exploit blockchain to enforce access policies in distributed environments through the use of smart contracts. In this case, smart contracts need gas to be executed, and this is the price to pay to make authorization decisions.

In [109], the first KP-ABE scheme was proposed, where a policy is associated with the decryption key and the attributes are associated with the ciphertext. In order to decrypt the ciphertext, the attributes have to match the policy. Similarly, in [44], the notion of CP-ABE was formalized, where the policy is associated with the ciphertext and the attributes with the key.

On the other hand, regarding the ciphertext, the attributes are related to the latter and the policy states which type of user can decrypt the ciphertext. An evolution of ABE is ABPRE [169]. In [168], a CP-ABPRE scheme is presented that uses a semi-trusted proxy to transform a ciphertext encrypted under a certain policy to another ciphertext under a different policy.

Platforms which rise with the aim to share services among consumers have the need of a Trusted Third Party (TTP) [48], a Decentralised App (DAPP) based on Ethereum smart contracts can resolve the requirement of having a trusted intermediary. The authors propose a DAPP for the sharing of objects and all the processes of renting are ruled by the smart contract.

In service marketplaces scenarios, a blockchain that runs smart contracts can enable the concept of trustless intermediation. The need of considering a trusted figure, who plays the role of trusted intermediary, lays on the service markets nature: the authors of [147] present a distributed approach to the problem and propose a new concept of decentralized and trustless service marketplace.

In [116], the authors consider the delivery of physical assets, the main requirement is establishing, through a smart contract, a series of agreements between the involved parties; this way, accountability of actions is preserved. In Lelantos [26], a blockchain based anonymity-preserving physical delivery system is proposed. This system can offer to consumers the fair exchange of services and the unlinkability of operations. Obviously the actors can operate in a anonymous way using a pseudonym revealed to the others parties, and all the processes are ruled also by a smart contract.

In [250], the author provides an overview of Ethereum and the principle of operation; moreover, open problems are listed and treated, differentiating the level of abstraction. The paper [24] collects different proposals on smart contracts, related to security, privacy issues, codifying and performance. Several platforms for smart contracts are compared and applications are examined. There are many technical and social implications in using smart contracts: [197] investigates the advantages of machine-readable smart contracts. The author highlights existing gaps in industry solutions using smart contracts and proposes their solution.

In [103], a SWOT analysis of blockchain is drafted to outline the advantages and disadvantages in using this technology in different areas. In particular, authors focus on the insurance field and highlight the scenarios in which it can be worthwhile to improve blockchain and smart-contract applications.

The survey [27] provides an overview of solutions exploiting the blockchain technology in energy sector. The authors classify the proposals into different categories based on the field of activity (e.g., e-mobility, grid management, decentralised energy trading), the platform used, and the relative consensus algorithm. The authors of [27] introduce security and identity management as a possible outcome of the blockchain technology in energy applications. They conclude that smart contracts simplify and make faster the cooperation and competition among energy suppliers.

The authors of [21] focus on the security and privacy challenges of energy trading in smart grids. The proposed system, PriWatt, relies on Bitcoin and Bitmessage: the former technology guarantees security and privacy without the need of a third party, and the latter assures anonymity through encrypted messages in messaging streams. A system limitation regards the message redundancy in the communication necessary to guarantee high levels of privacy and security.

The system provided in [274] is based on an Ethereum private blockchain that allows the participation of only authorized users. No identity management mechanisms are implemented but the access control and authentication are guaranteed through the blockchain's smart contract feature of restrict modifiers.

The authors of [82] present a Secure Private blockchain-based platform assuring the privacy of producers and consumers. While the producer can exploit different energy accounts, the consumers' privacy is preserved by changeable public keys of their smart meters. Nevertheless, to reduce the computation, the negotiation between the producer and consumer is conducted off-the-chain. This choice limits the security properties of energy bids that are not evaluated by the smart contract as our solution contemplates.

In [102], the authors propose a solution to implement traceable energy governance in Smart Grid Networks. The schema provides a transparent and traceable tracking of energy usage and consumption via the blockchain transactions. This proposal uses permissioned blockchain and super-nodes in charge of validating users' identities and activities. The authors of [40] deal with Energy Storage Units (ESU) in smart grids. In their proposal, they use certified pseudonyms and smart contracts with no centralized authority.

The authors of [101] solve the problem of privacy in an energy trading scenario with a consortium blockchain-oriented approach. During the energy trading phases, the authors introduce a privacy-preserving module named Black Box Module (BBM), whose main principle is to create a mapping accounts for energy sellers.

In [243], the authors face the problem of privacy in the blockchain-based solutions for energy trading in smart grids. Their proposal is based on the function-hiding inner product encryption to match every bid with its bidder. However, this solution requires a central trusted entity, the Distributed System Operator, that acts as a mediator between the user and the network.

A smart and scalable distributed ledger system for smart grids is proposed in [38]. The authors analyzed the properties of this new protocol and instantiated it in an electrical vehicles scenario. Ecash is the energy cryptocurrency of the system, used as a digital asset for energy transactions. These transactions are added in form of a directed acyclic graph. The validation of transaction is done by checking the balance amount of Ecash spent or used in the transaction and through the Proof of Time instead of the Proof of Work of Bitcoin. If the transaction is validated by more than half of the total SmartChain then the transaction is considered valid. Two chains are proposed: the seller and the buyer chains where the respective transactions are stored. This proposal is opposed to the current solution relying on the already existing blockchain technologies, as our schema does. Indeed, the authors of [38] design a new system inspired by the blockchain paradigm and aimed at meeting the limited computational resources of electric vehicles.

9

Blockchain-based Access Control supporting Anonymity and Accountability

In information security, access control is the selective restriction of access to an online resource or service. One of the most used access control models is Attribute-based Access Control, in which access rights are granted to users by evaluating suitable attributes (user attributes, resource attributes, and environment conditions). An important aspect of access control is to guarantee that the identity of the user accessing a service is preserved. In this solution, we deal with this problem and propose a new scheme based on a blockchain to ensure that only authorized users can access a service, yet preserving anonymity and unlinkability of their accesses. Moreover, the cooperation among several trusted parties allows the identification of the user accessing a service in case of need.

9.1 Introduction

Blockchain has been recently proposed as a solution to several application problems [58, 54]. This emerging technology is a secure storage relying on a distributed consensus protocol able to validate the data added to it [180]. Indeed, blockchain is a distributed and transparent public repository of transactions executed by users and shared among a large number of nodes [247]. Transactions are stored inside a chain only if they are validated by blockchain nodes. Validation is done by a distributed consensus algorithm [272], on which the performance of the blockchain network depends.

Blockchain users create a wallet and are provided with a couple of private and public keys. The private one is used to sign the transactions and aims to guarantee security and authenticity. The public address of a wallet is generated starting from the public key. The users can perform blockchain transactions once they create their wallets [75]. Moreover, users can generate countless blockchain addresses in order to preserve their pseudonymity, which is another important added value of blockchain technology. Generally, a transaction is a transfer of value among blockchain users. Inside a transaction, there is the reference to the recipient's public address and other suitable data, named transaction *payload*. The blockchain technology presents many advantages [282], such as the transparency and immutability of records and the pseudonymity of transactions. These properties could be exploited in access control systems.

Access control systems regulate the accesses to protected resources or operations inside a computer system. The process of access control involves the authentication and the authorization of subjects through a series of security policies in such a way that only the legitimate accesses can take place. The security policies can rely on several security models proposed in the literature [200] and are shared among the different entities of the access control mechanism.

In the context of access control, some significant security research challenges are related to:

- how to guarantee the anonymity of a user accessing an online service supplied by a service provider;
- how to ensure that two different requests of a user to access an online service are not linkable;
- how to disclose the identity of a user who accessed an online service in case of need.

In this solution, we provide an access control scheme that provides the three features above. Our scheme is based on a public blockchain and relies on identity and access control providers. The users who request access to a protected resource supplied by an online service provider are provided with blockchain accounts. Also identity providers, access control providers, and service providers have blockchain accounts, in such a way that all information needed to implement the access control is on the blockchain and publicly available.

Any user exploits different blockchain addresses to interact with identity, access control, or service providers. The used blockchain addresses are linked to each other, and every entity can verify the transactions generated by the users by using the blockchain. Moreover, our solution stores the list of the blockchain addresses used by a user, and each provider involved in the access stores a part of this list (i.e., the link between two addresses adjacent in the list).

Concerning the anonymity, the user reveals his/her identity only to the identity provider, which maintains a mapping between the identity and the blockchain address generated by the user to be identified.

The unlinkability of different requests of the same user is reached by exploiting different blockchain addresses at each interaction with the other entities.

At the same time, we need the guarantee the accountability of the requests. We reached it by linking the users' blockchain addresses in a verifiable chain locally and partially stored by several parties. In case of need, a party in cooperation with other trusted parties can restore the chain and guess the identity of a user.

9.2 Access control

In this section, we introduce some important concepts related to access control as well as the access control models proposed in the literature, which are used in the rest of the solution.

Access control regards the processes carried out to protect users and resources from unauthorized accesses inside any information management systems. A subject is an entity able to access a protected object containing information. An authorized subject is provided with privilege, that is an authorization to carry out some actions on the objects.

In order to develop an access control system, three important abstraction layers have to be taken into account: the security policy, the security model, and the security mechanism [233]. The security policy defines high-level requirements related to the authorization rules that are formally stated in the security model. The security mechanism is the lower layer defining the functions that implement the control policies described in the security model. Many security models have been proposed in the literature to describe security properties in an access control system [200].

The discretionary access control (DAC) is a flexible policy based on the identity of resources' owners. That is, a resource's owner can define the access rules and authorized operations of that resource and modify them anytime. The access control list (ACL) is an example of DAC. An access control list defines the authorized operations and the authorized users for every resource. This type of access control is not suitable in our case because we want to guarantee the user's anonymity.

Contrarily to the DAC, the non-discretionary access control techniques (NDAC) rely on established and non-modifiable rules. An example of NDAC is the mandatory access control (MAC). In the mandatory access control, the control policies are released by a central authority, such as the system's administrator, not by the single user able to access a resource.

In the role-based access control (RBAC), privileges are associated with the roles carried out by subjects. In an organization, a role is made of permissions or responsibilities referred to a subject or group of subjects. Therefore, the definition of roles is the central point of this model.

The attribute-based access control (ABAC) is defined as an access control methodology where authorization is determined by the possession of attributes associated with the subject, object, and policy or rules. The attributes generally describe these entities and are easily modifiable and verifiable by the authority in charge of releasing them.

It is evident that in some access control models, the identity of the subject is not necessary to gain the authorization for a resource. In the role-based access control, subjects have to demonstrate to perform a specific role, whereas the owned attributes are enough in the attribute-based access control. In our solution, we exploit an attribute-based access control scheme.

In the ABAC model, the entire process of access control can be summarized as follow. When a subject requests access to a protected object, the access control mechanism has to verify that the subject is authorized. That is, the subject possesses the attribute necessary to access the resource. Furthermore, also the object attribute and the environmental conditions (i.e., not related to the subject or object but linked to the environment, such as time and zone) have to be validated. If the conditions are fulfilled, the subjects gain access. Otherwise, the subjects are not authorized for that resource.

9.3 Our proposal

In this section, we present the proposed solution: we start by describing the scenario considered in this solution, which is composed of the following actors:

- *Users (U)*, who are physical people whose anonymity in accessing a service should be guaranteed.
- *Identity Providers (IP)*, which create and manage digital identities.
- *Access Control Providers (ACP)*, which are in charge of verifying an access control policy.
- *Service Providers (SP)*, which offer online services only to authorized users.

Now, we describe the protocol allowing us to solve the faced problem. It is schematized in Figure 9.1 and is composed of the following phases:

Setup. This phase is used to initialize the environment and to perform some preliminary operations. First, the blockchain to be used is chosen. As we will see, we exploit the basic features of a blockchain (i.e., the distributed repository) so that any blockchain could be used. Although our solution is orthogonal to the used blockchain, for the sake of presentation, we will refer to the Bitcoin Blockchain when need. According to the chosen blockchain, the following two functions are

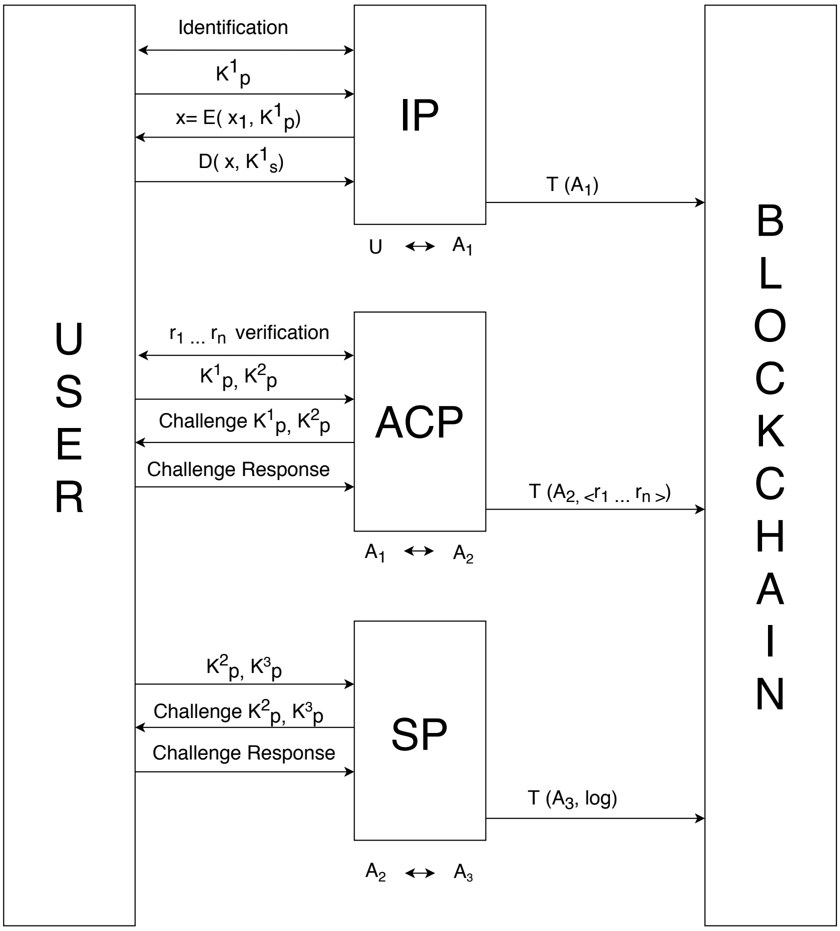


Fig. 9.1: Scheme of our solution.

defined: `sign(M,k)` and `verify(S,M,k)`, where M is a message, k is a cryptographic key, and S is a signature. Specifically, the former returns the signature of the message M by the key k, the latter verifies the validity of the signature S of the message M by the key K and returns true if and only if S is a valid signature. For example, in Bitcoin, these two functions are based on elliptic curves.

All the actors know the chosen blockchain and the defined functions.

BAG. Blockchain Account Generation. Our solution is based on blockchain so that any actor needs an *account* to use the blockchain. This phase, which is carried out at least one time[1] by any actor of the scenario, aims to generate a pair of cryptographic (public and private) keys and a blockchain address. In most of the blockchains, the private key is a randomly generated string with a suitable large number of bits (typically, 256 bits), whereas the public key is obtained by applying a cryptographic function to the private key. Then, the associated blockchain address is obtained by applying a suitable function to the public

[1] As it will be clear later, this phase is run each time a user needs a new account.

key as $A = f(K_p)$. Typically, this function is implemented by applying a cryptographic hash function and by keeping the last n bits (e.g., $n = 160$ in Bitcoin). Each actor joining the system generates a pair of cryptographic keys and obtains a blockchain address. Given an account K, K_s denoted the associated secret key, K_p the public key, and $A = f(K_p)$ the associated address.

UI. *User Identification*. This phase is carried out by each user to register his/her digital identity from an identity provider ID. First, the identity provider verifies the user identity by a recognition in person, via webcam, or online. Then, it collects the surname, name(s), date of birth of the user, and all the other personal data useful for identification.

Now, let K_s^1 and K_p^1 be the user's private and public key, respectively. The user U needs to prove to be the owner of the blockchain address A^1: for this purpose, U sends K_p^1 to the identity provider. The identity provider generates a random x^1 and calculates x=$\mathtt{Encrypt}(x_1, K_p^1)$, which is sent to the user.

The user calculates x'=$\mathtt{Decrypt}(x, K_s^1)$ and sends back this value to the identity provider. In turn, the identity provider verifies that $x' = x^1$, which means that U is the owner of the public key K_p^1 and the address $A^1 = f(K_p^1)$.

In this case, the identity provider generates the *registration* transaction, which is a blockchain transaction from the address of the identity provider to the address A^1, with no further data.

The purpose of this transaction is to store on the blockchain that A^1 is associated with an identified user. Moreover, the identity provider stores the mapping between A^1 and the personal data of U, collected in the identification step. This mapping is useful for accountability.

AC. *Access Control*. In this step, the user contacts a suitable access control provider ACP to receive a *proof* that he/she satisfies some requirements, say $r_1, \ldots, r_n$.

First, ACP verifies that the user satisfies the requirements $r_1, \ldots, r_n$, and, as remarked in Section 9.2, we use an attribute-based access control scheme.

Then, the user U creates a new account K^2 by using the procedure described in the BAG step. U sends K_p^1 and K_p^2 to ACP. The access control provider checks on the blockchain if there exists a registration transaction for the account K^1 (i.e., a transaction from any identity provider to the address A^1). If this transaction is not found, the procedure alts. Otherwise, ACP picks a random x^2 and calculates a=$\mathtt{Encrypt}(r_2, K_p^2)$ and x=$\mathtt{Encrypt}(a, K_p^1)$. Then, ACP sent a to the user as a challenge.

In response to the challenge, the user calculates a'=$\mathtt{Decrypt}(x, K_s^1)$ and x'=$\mathtt{Decrypt}(a', K_s^1)$. The latter value is returned to the access control provider, which checks that $x' = x^2$. In the positive case, ACP has the proof that U is the owner of both the address A^1 and A^2.

In this case, the ACP generates the *authorization* transaction, which is a blockchain transaction from the address of the access control provider to the address A^2, having $r_1,\ldots,r_n$ as payload (usually, the data field in a transaction).

This transaction saves on the blockchain the information that A^2 satisfies the requirements is $r_1,\ldots,r_n$. Moreover, the access control provider stores the mapping between A^1 and A^2, which is used for accountability.

SA. Service Access. In this step, the user contacts the service provider to require a service. First, the user creates a new account K^3 as described in the BAG step. To prove to be authorized to this service, the user sends the public keys K_p^2 and K_p^3 to the service provider SP. SP verifies that the account K^2 satisfies the requirements $r_1,\ldots,r_n$ by searching on the blockchain for an authorization transaction sent to the address $f(K_p^2)$, having in the payload at least $r_1,\ldots,r_n$. If this transaction is not found, the procedure alts. Otherwise, SP generates a random x^3 and calculates a=`Encrypt(`r_3`,`K_p^3`)` and x=`Encrypt(`a`,`K_p^2`)`. Then, SP sent a to the user as a *challenge*.

In response to the challenge, the user calculates a'=`Decrypt(`x`,`K_s^2`)` and x'=`Decrypt(`a'`,`K_s^3`)`. The value x' is sent back to the service provider, which can verify that $x' = x^3$. In this case, the service is granted to the user, and a *log* transaction is generated. This transaction is sent from the address of the service provider to the address $f(K_p^3)$ and has in the payload the log information (typically, it contains the id of the service and the timestamp).

In the next section, we discuss how our solution reaches the expected goals.

9.4 Validation

Our solution aims at providing an access control mechanism that guarantees anonymity, unlinkability, and accountability in accessing online services. In our security analysis, we assume the following properties hold:

1. the random generated numbers x_1, x_2, and x_3 are never re-generated used by any actor. This can be guaranteed provided that the number domain is suitably large (for example, 128-bit random numbers currently satisfy this requirement);
2. the cryptographic primitives `Encrypt` and `Decrypt` are robust and cannot be broken. The elliptic curves used in several blockchains (e.g., Bitcoin adopts `secp256k1` [185]) currently satisfy this requirement;
3. private keys are kept secret and cannot be guessed;

4. the information about the mapping of the addresses stored by each entity is not shared or made publicly available;
5. the user discloses personal information only during the Identification Step;
6. Identity Provider, Access Control Provider, and Service Provider do not collude.

Under these assumptions, we show how the expected security properties are guaranteed. We start from anonymity, which requires that the name of the user accessing the service is not given or known from the service provider, the access control provider, or any third party (except the identity provider, which is the entity that knows the user identity by the scheme). Observe that the identity provider publishes the blockchain address of the user so that the user is identified by a public-key only. In blockchain this is called pseudonymity and differs from anonymity because an attacker wishing to de-anonymize a user tries to construct the one-to-many mapping between users and public-keys [220]. We prevent this attack by storing the association between user and address on the identity provider only. As a consequence, there is no possibility to break pseudonymity and anonymity.

The second property is unlinkability, which means that a user may make multiple uses of services without other parties being able to link these uses together. This is achieved by forcing the user to generate a new account after each iteration. This way, at each iteration, the providers see a new blockchain address that appears randomly generated.

The last requirement is to guarantee accountability, that allows a party in cooperation with other trusted parties to guess the identity of a user in cases of need. The identity of a user who accessed a given service can be guessed as follows. First, from the log transaction of this service, the address A^3 is extracted. Then, the service provider returns the address (say A^2) associated with A^3, by using the locally stored mapping. Now, an authorization transaction to A^2 is searched on the blockchain and let ACP the access control provider that generated this transaction. Again, by using the ACP's local mapping, the associated address A^1 is found. A new search for a registration transaction sent to the address A^1 returns the identity provider that identified the user and that can provide the requested information. Observe that all these transactions are found if the protocol has been correctly run by the identity provider, access control provider, and service provider.

9.5 Conclusion

In this chapter, we faced some relevant security challenges in the context of access control, by proposing an access control scheme relying on the blockchain technology. For this reason, users who request access to a protected resource supplied by an online service provider are provided with blockchain accounts. Also the other entities involved in the scenario, which are identity providers, access control providers, and service providers, have their blockchain accounts. The aims of our solution is to guarantee the anonymity of a user and the unlinkability of different requests of the same user. This is reached by exploiting different blockchain addresses at each interaction with the other entities. At the same time, we need the guarantee the accountability of the requests. We reached it by linking the users' blockchain addresses in a verifiable chain locally stored in a distributed way by several parties.

10

Privacy-Preserving Service Delivery with Accountability Requirements

The main benefit of smart contracts over Ethereum is that different parties with conflicting interests can exchange value without trusting each other. As a matter of fact, solutions in which service delivery is regulated by smart contracts are proliferating. Sometimes, services can be negotiated and delivered only on the basis of some attributes, without disclosing the identity of the customer to the service supplier. However, accountability is still required, so that, in case of need, the identity of the customer should be linked to the service delivered and communicated to the appropriate parties. In this chapter, we propose a practical solution to the above problem that integrates the features of Ethereum with a (Ciphertext-Policy) Attribute-Based Encryption scheme. To show the effectiveness of our proposal, we instantiate the general model to a significant use case.

10.1 Introduction

Ethereum [94] is one of the blockchain platforms attracting the interest of both research and industry, mainly due to the power of *smart contracts*. Indeed, when different parties with conflicting interests have to exchange value, a problem is how to prevent that one of the parties, in a certain moment, misbehaves to obtain an advantage, so that the agreement is not concluded fairly for everyone. Smart contracts solve this problem. The consensus mechanism implemented by Ethereum guarantees that all the contract steps are automatically executed in a transparent way, according to the agreed rules, without the need that the parties have to trust each other.

There are many situations in which a service can be delivered to a customer only on the basis of some requirements the customer has to satisfy, such as the age, the professional title, the possession of a licence, etc. At the same time, it could be desirable not to disclose to the (potentially untrustworthy) service supplier other identifying information to prevent data misuse even in the less severe hostile case of an honest-but-curious provider.

To reach this goal, one could think of standard techniques based on anonymous credential [65], but, to be realistic, a solution to the problem of anonymous payment should be provided, together with an appropriate level of guarantee that anonymity does not compromise obligations, non-repudiability and accountability of the agreement.

To the best of our knowledge, no solution has been proposed for this general problem so far. The idea of this solution is to leverage the power of smart contracts to obtain all the above requirements. Pseudonymity of Ethereum can ensure a good level of privacy, but the problem of implementing attribute-based contracts is not solved, at the moment, by native features of Ethereum. The focus of this solution is just this, and the direction we follow is the integration of a public attribute certification process (possibly based on the ecosystem designed by the eIDAS EU Regulation [86]) into the smart-contract features at a cryptographic level, thus by exploiting a Ciphertext-Policy Attribute-Based Encryption Scheme (CP-ABE) [44].

We observe that an attempt of bypassing the cryptographic link between attribute possession and Ethereum transactions would not provide an adequate result in terms of trustworthiness. Indeed, we should require that an entity of the application (maybe a smart contract) should obtain by a Third Trusted Party (the Attribute Provider, in the eIDAS system) the proof of the possession of certain attributes for a given pseudonymous individual. This implicitly requires full trust in this node, concerning the assessment of attributes. In contrast, our solution requires that only the party certifying the attributes (assumption fully accepted in eIDAS) and the Private Keys Generator (PKG) of the CP-ABE are trusted parties, which are parties external to the application. Moreover, often attributes can be certified by Government bodies, which could also play the role of Private Keys Generator.

It is worth noting that, in our solution, the link between attributes and real-life identity is known only by the party certifying the attributes. Moreover, all the actions are immutably recorded over the blockchain and, in case of need, the link between the pseudonymous used in a certain transaction and the real-life identity of the customer can be disclosed by collecting information from the PKG and the parties certifying the attributes. This guarantees the accountability requirement of our solution.

10.2 Proposed solution

In this section, we present the architecture and the solution we propose to allow service delivery only to users having certain requirements, by preserving their identity. In Figure 14.1, the entities involved in the scenario are depicted: we have a user *U* who needs a service s provided by one of the several available service suppliers (SS). Moreover, we have several Attribute Providers (AP), which are in charge of checking if a user fulfills or not some specific attributes. Finally, we have the Public Key Generator PKG, which is the Trusted Party issuing ABE private keys.

We introduce some preliminary background notions.

1. let SS be the service supplier providing the service s;
2. let $\mathcal{A}$ be the access structure [41] representing the policy associated with the service s;
3. let $P = \{a_1, \ldots, a_n\}$ be the attributes of U that are compliant with the policy $\mathcal{A}$; [1];
4. we denote by $OW(a_i)$ the Attribute Provider that is in charge of checking if U fulfills or not the attribute a_i;
5. we denote by Eth_x an Ethereum address owned by x, where x can be U, SS, or a smart contract;
6. we model an Ethereum transaction T as a tuple $\langle id_T, Eth_{src},$ $Eth_{dest}, data\rangle$, where id is the identifier (usually, it is the digest of the transaction), src and $dest$ are the sender and receiver of the transaction, resp., and $data$ is the payload. Observe that, transactions include also an additional field `value`, which is not relevant for our scope, so that it is not considered here.
7. $KCK(M)$ denotes the Keccak digest of the message M.
8. Setup(k): This algorithm receives a security parameter k and returns a public parameter PK and a master secret key MSK.
9. KeyGen(MSK;P): This algorithm takes as input a set of attributes P and the master secret key MSK. It outputs a private key SK associated with S.
10. $Encrypt(PK, M, \mathcal{A})$ denotes the encryption of the message M under the policy $\mathcal{A}$ (see item (2) above).
11. $Decrypt(CT, SK)$ denotes the decryption of the ciphered message CT with the ABE private key SK.

Now, we describe the steps carried out by the different actors of our scenario. These steps are also summarized in Figure 14.1.

Step 1: service request. First, the user U asks for the service s supplied by SS. For this purpose, U generates an Ethereum transaction $T_1 = \langle id_{T_1}, Eth_U, Eth_{SS}, data_1\rangle$, where id_{T_1} and $data_1$ are the identifiers of the transaction and the service s, respectively.

Step 2: challenge start. When SS receives the service request T_1, the service supplier first checks whether Eth_U is not *revoked* (the detail about how this check is imple-

[1] For the sake of presentation, with an abuse of notation, we use $\mathcal{A}$ meaning the policy represented by $\mathcal{A}$.

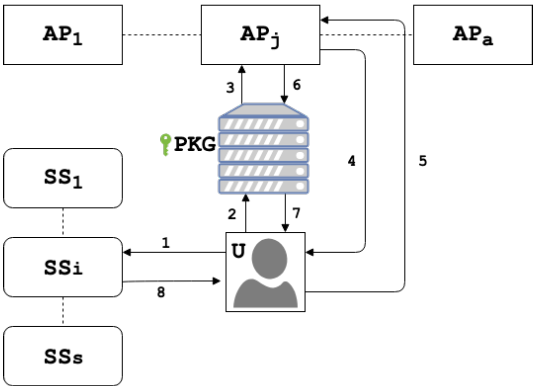


Fig. 10.1: Scenario and steps of our solution.

mented is given in Step 5). If Eth_U is not revoked, SS acknowledges the request and generates a *challenge* needed to verify that the user is able to prove the possession of all the required attributes to satisfy $\mathcal{A}$. In particular, SS creates the new policy $\mathcal{A}'$ requiring, in addition to the conditions expected by $\mathcal{A}$, also the possession of the attribute a_{n+1}, where a_{n+1} represents the possession of the Ethereum address Eth_U. In words, the new policy enforces that the user has to satisfy the requirements of the policy $\mathcal{A}$ and, moreover, she/he owns the Ethereum address Eth_U. This is done in such a way to guarantee that only the user with address Eth_U can overcome the challenge.

The challenge starts with the generation of the transaction $T_2 = \langle id_{T_2}, Eth_{SS}, Eth_{SC}, data_2\rangle$, where SC is a smart contract and $data_2$
$= \langle X_0 = id_{T_1}, X_1 = P', X_2 = Encrypt(PK, R, \mathcal{A}'), X_3 = KCK(R), X_4$
$= Eth_U\rangle$, where $P' = P \cup \{a_{n+1}\}$ and R is a secret value (the challenge solution). In words, $data$ contains the reference to the first transaction done by U, the set of attributes, the challenge consisting of the encryption of a value R that can be deciphered only by users satisfying the policy $\mathcal{A}'$, and the digest of R computed by the Keccak function (the solution verification). The smart contract receives the transaction and *waits* for the user reply.

Step 3: challenge reply. The user U looks for the transactions made by SS with $X_0 = id_{T_1}$ (i.e., in the first input of data field) – thus, the transaction T_2. By using id_{T_1}, U can find the challenge and extract $X_1 = P'$.

In order to prove the possession of all the attributes in P', U needs the ABE private key associated with the attributes P'. The task carried out to obtain this key is the following.

1. U contacts PKG and asks for the ABE private key associated with the attributes P' (message 2 in Figure 14.1);

2. PKG computes the set $AP = \bigcup_{x=1}^{n+1} OW(a_x)$, which is composed of the attribute providers of the attributes involved in P';
3. now, each $AP_i \in AP$ performs a challenge-response-based authentication with the user U (messages 4 and 5);
4. In case of successful authentication, if the user owns the attributes requested, AP_i generates an *assertion* of attribute certification for PKG (message 6), which is signed by AP_i to guarantee integrity and authenticity. Moreover, AP_i stores the mapping between the user identity and the assertion identifier, which can be used in case of revocation or accountability;
5. After collecting all the attributes certifications, PKG invokes KeyGen($MSK;P'$), which generates the ABE private key $sk_{P'}$ for U. PKG sends this key to U (message 7) and stores the mapping between the received assertion and the Ethereum address, again for accountability or revocation reasons.

Observe that this procedure is carried out only the first time U needs the ABE private key: indeed, this key will be used also for the next accesses to services with the same policy $\mathcal{A}'$.

Now, U extracts X_2 from T_2, calls Decrypt(X_2,$sk_{P'}$), and obtains R'. In order to demonstrate the knowledge of the ABE private key and, consequently, to prove the possession of the attributes P', U generates another transaction $T_3 = \langle id_{T_3}, ETH_U, ETH_{SC}, R' \rangle$.

Step 4: agreement. The call to the smart contract starts the automatic check of the challenge reply. The smart contract acts as follows:

1. finds the *pending* challenge for the Ethereum address Eth_U, and retrieves $X_3 = KCK(R)$;
2. extracts R' from T_3;
3. computes $R^* = KCK(R')$;
4. if $R^* = X_3$, then U overcomes the challenge and the grant for providing the user U with the requested service is given.

Step 5: revocation. This step is performed whenever a user U loses an attribute a_i. In this case, the attribute provider $OW(a_i)$ searches for all the assertions previously sent to PKG mapped to U (these are stored into a map - see Step 3): this list of assertions is sent to PKG as *revoked* assertions.

PKG receives this list and searches for the Ethereum address mapped to each assertion. Then, an Ethereum transaction of revocation to each of these addresses is generated, reporting the revocation of the ABE key associated with this Ethereum address. This way, the list of revoked Ethereum addresses is stored on Ethereum and can be used in Step 2 to check if the Ethereum address of the user requiring the service has been revoked.

10.3 Implementation

In this section, we describe the implementation of our solution. In order to test our proposal, we implemented the adopted CP-ABE scheme in JAVA language, leveraging the libraries [80]. This was necessary to include, in our system, some lightweight implementation of the scheme, tailored with our specific scenario, in which policies are single domain-dependent attribute (this results in concrete two-attribute policies since each policy must include also the Ethereum address as second attribute).

Furthermore, we designed also the smart contract used to ask for a service and decide whether to grant it. The smart contract is written in Solidity [15, 77], a high-level Turing-complete and object-oriented language.

The smart contract, whose code is reported in Listing 15.1, stores by `Owner` (Line 3) the Ethereum address of the service supplier, which is initialized with the address of the party that deployed the smart contract. The `struct data` (Lines 4-10) represents the skeleton of the challenge that must be won by the user to obtain the requested service. Mappings in Lines 11-12, are used, respectively, to save pending transactions (on which there is an open challenge) and to retrieve the challenge from a given transaction.

In Solidity, the modifier can be seen as an extension of a function and it is used to check a condition prior to executing the function. So, we implement the `modifier OnlyOwner` (Lines 17-20) in the `function cStart` (challenge start) (Lines 22-26) in such a way the condition that has to be checked is related to the sender of the transaction. In particular, `OnlyOwner` requires that the sender must be the owner of the smart contract, otherwise the function cannot be executed.

The function `cStart` can be called only by the service supplier and is used to reply to a *service request* of a user. In particular, the input of the function is the challenge, which is saved in an instance of `struct data` (Line 23); moreover, we set the mapping `txs_pending` for the given transaction `tx` to *true* so that, from now on, there is an open challenge for that specific request . Finally, we map the given transaction `tx` to this challenge.

The function `cResponse` (challenge response) (Lines 27-45) is called by the applicant of the service to reply and, possibly, win the challenge. This function receives the address of the challenge transaction and the challenge solution `r1`. First, if there is an open challenge for the given transaction (Line 28), we verify that the applicant who is calling the function is actually the target of that challenge (Line 30), and compare keccak256(r1) with kck_R of the challenge (Line 32). In case of success, the challenge is no more pending (Line 33) and the service can be grant (this part is application dependent so we omitted its implementation). In case any of the previous checks are not passed, the function ends (revert()).

We implemented this smart contract by *Remix - Solidity IDE* [14] and used *Ropsten* as testnet, with the support of *Metamask* [13], which consists of a browser extension that allows us to run dApps (decentralized applications) directly on the browser without running a full Ethereum node. The deploy of the contract on the Ropsten Test Network costs 844 Micro(ETH) (in April 2019, this is about 0,13 $), the function cStart costs 644 Micro(ETH) (about 0,11 $) and the function cResponse costs 24 Micro(ETH) (about 0,0037 $). From this analysis, we can say that the implementation of our solution is feasible and cheap.

```
 1
 2  pragma experimental ABIEncoderV2;
 3  contract Granting {
 4    address owner; //the service supplier
 5    struct data{ //the challenge
 6      bytes32 tx;
 7      string[] attributes;
 8      bytes32 encrypted;
 9      bytes32 kck_R;
10      address user;
11    }
12    mapping(bytes32 => bool) public txs_pending;
13    mapping(bytes32 => data) public fromTx_toData;
14
15    constructor () public {
16      owner = msg.sender;
17    }
18    modifier onlyOwner(){
19      require (owner == msg.sender);
20       _;
21    }
22
23    function cStart (bytes32 tx, string[] memory attributes, bytes32 encrypted, bytes32 kck_R, address user) public
            onlyOwner {
24      data memory d1 = data(tx, attributes, encrypted, kck_R, user);
25      txs_pending[tx]= true;
26      fromTx_toData[tx] = d1;
27    }
28    function cResponse(bytes32 tx, uint256 r1) public {
29      if(txs_pending[tx]==true){
30        data memory d2= fromTx_toData[tx];
31        if (msg.sender==d2.user){
32          bytes32 rs= keccak256(abi.encodePacked(r1));
33          if(rs==d2.kck_R) {
34            txs_pending[tx]= false;
35            //Grant the service
36          }
37          else { revert();}
38        }
39        else { revert();}
40      }
41    }
42  }
```

Listing 10.1: Code of the smart contract.

10.4 Case study

In this section, we show how our approach is instantiated to obtain a practical solution in a real-life scenario, which is a car sharing.

Suppose John wants to rent a car (i.e., the *service*, in our scenario) s at the company Car4U (which plays the role of *service supplier* SS). As Attribute-Based Encryption, we adopt a solution derived by the scheme [44]. Thus, we consider given a cyclic group $\mathbb{G}$ with order p, a generator g of $\mathbb{G}$, an hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$.

The Public Key PK and the Master Secret Key MSK are calculated as $PK = (\mathbb{G}, g, h = g^{\beta}, e(g,g)^{\alpha})$ and $MSK = (\beta, g^{\alpha})$, where α and β are two random elements $\in \mathbb{Z}_p$.

For the sake of presentation, we consider the case in which the service supplier policy consists of only one requirement. In particular, the access structure $\mathcal{A}$ of the policy for renting a Car4U car consists of the attribute a_1 *having the driving license*. Furthermore, the company *Car4U* has deployed its own smart contract described in Section 13.5.

Now, we detail the use case proposed above.

Step 1: service request. Once John has decided the car to rent, he generates an Ethereum transaction T_1 to Car4U. We assume that the Ethereum address of Car4U is public and extracted by the site, or a QR code, or in a similar way. The payload of the transaction (i.e., the `data` field) contains the license plate of the car to rent as service identifier.

Step 2: challenge start. Let suppose that John's address has not been revoked: *Car4U* acknowledges the request and sends a *challenge* needed to verify that John satisfies the policy $\mathcal{A}'$, which means a_1 *having the driving license* and a_2 *having the Ethereum address* Eth_{John}.

Now, Car4U picks up a random $R \in \mathbb{G}_{\mathbb{T}}$ and encrypts R under $\mathcal{A}'$. The encryption algorithm picks up two random $r, s \in \mathbb{Z}_p$. The ciphertext is: $X_2 = (\tilde{C} = Me(g,g)^{\alpha s}, C = h^s, C_1 = g^{r+s}, C_2 = g^{2r+s}, C^{'}_1 = H(a_1)^{r+s}, C^{'}_2 = H(a_2)^{2r+s})$.

Finally, Car4U generates a transaction to call the function `cStart` of the smart contract having in the payload $\langle id_{T_1}, \{a_1, a_2\}, X_2, KCK(R), Eth_{John} \rangle$.

Step 3: challenge reply. Now, John requests to the Public Key Generator PKG the ABE private key sk_{a_1,a_2} built from the attributes a_1, a_2. Observe that $OW(a_1)$ is the Motor Vehicle Office, which is in charge of checking whether a user has a driver license. In order to show to be the owner of Eth_{John}, John is required to sign a challenge by the Ethereum private key associated with Eth_{John}: thus, the role of $OW(a_2)$ is played by PKG. Now, John proves his identity to the Motor Vehicle Office: this is done by an eIDAS-compliant identification scheme. Once the identity of the person is verified, the Motor Vehicle Office can check whether John has a driving license. If this is the case, the Motor Vehicle Office sends to PKG a SAML assertion [176] and stores the mapping between the assertion identifier and the identity of the user.

If John proves also to be the owner of Eth_{John}, PKG calculates the ABE private key for John: PKG picks up the randoms $t, t_1, t_2 \in \mathbb{Z}_p$. The decryption key is: $sk_{a_1,a_2} = (D = g^{\frac{\alpha+t}{\beta}}, D_1 = g^t H(a_1)^{t_1}, D_2 = g^t H(a_2)^{t_2}, D_1^{'} = g^{t_1}, D_2^{'} = g^{t_2})$. This key is sent to John and PKG stores the mapping between the assertion identifier and Eth_{John}.

Now, in order to decipher X_2 (extracted from T_2), John computes: $F_1 = \frac{e(D_1,C_1)}{e(D_1^{'},C_1^{'})}$, $F_2 = \frac{e(D_2,C_2)}{e(D_2^{'},C_2^{'})}$, and $A = \frac{F_1^2}{F_2}$.

Then, the deciphered text is obtained as: $R' = \frac{\tilde{C}}{e(C,D)/A}$.

It is easy to check the decryption procedure. $F_1 = \frac{e(D_1,C_1)}{e(D_1^{'},C_1^{'})} = \frac{e(g^t H(a_1)^{t_1}, g^{r+s})}{e(g^{t_1}, H(a_1)^{r+s})}$. Since g is a generator, $H(a_1) = g^k$ for some k. Thus, $F_1 = \frac{e(g^t g^{kt_1}, g^{r+s})}{e(g^{t_1}, g^{k(r+s)})} = e(g,g)^{t(r+s)}$. Similarly, $F_2 = e(g,g)^{t(2r+s)}$. Therefore, $A = \frac{F_1^2}{F_2} = \frac{e(g,g)^{2t(r+s)}}{e(g,g)^{t(2r+s)}} = e(g,g)^{ts}$ and $\frac{\tilde{C}}{e(C,D)A} = \frac{Me(g,g)^{\alpha s}}{e(h^s, g^{\frac{\alpha+t}{\beta}})/e(g,g)^{ts}} = M$. Finally, John calls cResponse of the same smart contract and sends the deciphered value R'.

Step 4: agreement. According to the called function, the service is grant if $KCK(R')$ is equal to $KCK(R)$.

Step 5: revocation. Let suppose now that John's driving licence is revoked. Since John has asked for and obtained the ABE private key before the revocation, he is able to pass the challenge for rent a new car, and, thus, he could rent a car without a driver license.

To solve this problem, when the driver licence of John is revoked, the Motor Vehicle Office sends to PKG the identifiers of all the assertions sent to PKG in the past related to John. PKG, which knows the Ethereum addresses related to these assertions, generates an Ethereum transaction to John's Ethereum address having in the payload a field *type* sets to *revocation* and a field *address* sets to this Ethereum address. In such a way, if John tries to request a service to any service supplier, this request will be discarded because his Ethereum address is contained in a transaction of revocation stored on Ethereum (recall that this check is done by the service supplier in Step 2).

10.5 Security analysis

In this section, we briefly discuss the security properties of our solution. We remark that this discussion is high-level and informal, accordingly to the nature of this solution, which actually aims to shortly describe the scientific core of a proposal contextualized in an industrial research project. Even though an accurate security analysis is still in progress, besides the following general discussion, the reader may identify throughout the solution, a number of detailed aspects related to security (for example, the revocation) which are mentioned within the description of the solution itself to argue its robustness.

Let start now by defining the *adversary model*. In our analysis, we assume that both PKG and Attribute Providers are trust parties and that the attacker can be a user, a service provider or external to the system. In addition to the standard security assumptions (unbreakability of cryptographic primitives and blockchain security properties), we assume that private keys and secret information are not disclosed by the owners and that users and service suppliers do not collude each other. The goal of the attacker is to break at least one of the following security properties: access-control, privacy, accountability, unlinkability, availability, non-repudiation.

Access control is reached. Indeed, to win the challenge, the user has to decipher a challenge by an ABE private key that PKG issues only to users who prove the possession of the attributes required by the policy.

The privacy requirement is that a service supplier is not aware of information identifying the user accessing the service. This goal is reached because the architecture of the solution allows the service supplier to know only the Ethereum address of the user, which is generated by the user and appears a random string. The level of protection of such information is that of pseudonymity given by blockchain, so it is not absolute, as de-anonymization is in general possible. However, for our specific context, unlike cryptocurrency transfers, if the user wants to protect her/his privacy by using always *one-shot* blockchain addresses, the above attacks on pseudonymity are not applicable.

Accountability is obtained by merging information coming from different parties. If we want to know the identity of the person who used the service s, we start from the transaction T_1 used to require the service and extract the Ethereum address used by this person. This can be done only by leveraging public information. Then, since PKG stores the mapping between Ethereum address and corresponding assertion, and any Attribute Provider stores the mapping between assertion identifier and digital identity, it is possible to disclose the identity of the person who used s.

Unlinkability is guaranteed provided that a user exploits a new Ethereum address for each service request (as discussed earlier, this measure makes ineffective also de-anonymization attacks).

Attacks on Availability are contrasted. The only attack that can be performed is a DoS, in which an attacker floods a service supplier with superfluous requests thus trying to overload it and prevent legitimate requests from being fulfiled. However, since any service request transaction has an even small price in Ether, an attack of this type would be very expensive.

Non-repudiation is obtained. Indeed, each action (service request, service grant, etc.) is logged into the Ethereum blockchain and it can be verified. Moreover, transactions are signed by an Ethereum private key, which is kept only by the owner of the address. Again, Ethereum transactions cannot be modified after they are validated. Moreover, the integrity of transactions is guaranteed by the properties of public blockchains. Thus, no party can repudiate an action.

10.6 Conclusion

In this chapter, we basically presented the idea to mix the power of smart contracts and blockchain with the power of ABE schemes, leveraging also the concrete ecosystem defined by the European Regulation eIDAS to be more effective. We identified, as a scenario in which such a combination can be fruitful, the context of attribute-based service delivery, in which it is not required that the user fully trusts the service supplier and that user's privacy must be protected unless a-posteriori higher-order reasons do not require the disclosure of her/his identity. In this case, full accountability is guaranteed. We defined the formal scheme of our solution, implemented it over Ethereum, and showed in a use case how the solution performs.

11

Privacy-Preserving Energy Trading in Smart Grids

The need for a flexible, dynamic, and decentralized energy market has rapidly grown in recent years. As a matter of fact, Industry 4.0 and Smart Grids are pursuing the path of automation of operations so that all the steps among consumers and producers are getting closer. This leads towards solutions that exploit the paradigm of public blockchain, which represents the best platform to design flat and liquid markets for which providing trust and accountability to mutual interactions becomes crucial. On the other hand, one of the risks to face in this situation is that personal information is exposed to the network, with intolerable threats to privacy. In this chapter, we propose a solution for energy trading, based on the blockchain Ethereum and smart contracts.The solution aims to be a concrete proposal to accomplish the needs of energy trading in smart grids, including the important feature that no information about the identity of the peers of the network is disclosed in advance.

11.1 Introduction

Due to the continued growth in energy demand, the issues of increasing its production, on the one hand, and to limit environmental pollution, on the other hand, are becoming global challenges.

Of course, it is necessary yet not sufficient to extend the usage of renewable energy. Only in 2018, renewable energy raised by 4%, accounting for almost one-quarter of global energy demand growth [100].

Moreover, there is a need to improve and update the classic electric grid infrastructure, to make it more efficient, flexible, and dynamic. In recent years the new concept of *Smart Grids* (SGs) is emerging. A smart grid can be considered as the evolution of classic grids having the main target to be eco-friendly, faster than the classic one, and more innovative. Smart grids are born also to improve the overall reliability of the whole energy cycle and to guarantee a better ratio demand/response so that the financial field is interested as well by applying a new energy market pattern [241]. Moreover, by increasing the energy demand and the number of entities involved in the energy market, smart grids have to face the problem of guaranteeing a certain level of data and message availability in transmissions among peers of the network [63].

Energy systems in smart grids are taking the direction of decentralized architectures in which a device, known as *smart meter*, can manage requests and responses through the whole network. Since it would be not appropriate to implement centralized protocols over smart grids, it is fundamental to accommodate this decentralized and distributed direction by using technologies that are decentralized and distributed as well. This way, blockchain technology appears to be the best solution, because of its proven properties, such as immutability, transparency and decentralization [205]. Indeed, thanks to the evolution of the blockchain paradigm originally born with Bitcoin blockchain [193] (mainly devoted to the cryptocurrency Bitcoin), blockchains supporting *smart contracts*, like Ethereum [92], can be viewed as platforms for secure, interoperable, and decentralized applications, in which conflicting parties may establish agreements and exchange value without trusting each other. Energy trading in smart grids perfectly fits with these features. Therefore, an interesting research direction is to investigate how to fully exploit the power of blockchain and smart contracts to envisage innovative applications and to increase the effectiveness of the notion of smart grid. Observe that the use of blockchain may introduce flexibility among operations carried out by stakeholders inside the energy trading market. In particular, if we overlap these features with the energy industry we can deduce that the sector that can benefit most from them is energy trading among applicants and bidders. Indeed, a blockchain-based solution for energy trading is able to improve accountability, reliability, fairness and to reduce time and costs.

However, there are still open challenges and limitations in the implementation of blockchain-based applications to energy market trading, such as the scalability, security and efficiency [112].

This solution is just placed in this research track, by proposing an Ethereum-based solution for energy trading aimed also to enhance scalability with respect to other approaches presented in literature. The blockchain enables parties to transfer assets without the participation of a trusted third-party and all the transactions are stored and validated by the network, with no centralized unit control. To the best of our knowledge, our approach presents an innovative aspect w.r.t. existing related proposals. Indeed, the power of smart contracts is also exploited to manage the offers in a blind fashion, so we can actually talk about an auction, in which identities are disclosed only when the agreement is established. Interestingly, the entire auction is managed with no intervention of any external referee.

11.2 Scenario and motivations

From the beginning, the electricity grid was conceived as a centralized system in which energy is produced in huge power plants. It is clear that this kind of system has limits in reliability, availability, and, as a consequence, in business terms. Moreover, the growing world population generates a rising demand for energy and, due to the increasing level of pollution, the request for sustainable and renewable energy is necessary. Indeed, investing in renewable energy is becoming central in most of the world governments for environmental protection. Energy is a raw material and for this reason it can be exchanged; energy trading term means buying, selling, and moving energy from where it is produced to where it is needed. The concept of Energy Internet [283] stands for the open, collaborative, and interactive process of energy production and consumption. Through the years, the energy systems have been developed into four different stages: decentralized or centralized energy systems, either distributed or smart and connected energy systems.

A decentralized approach can evaluate the energy exchange [280], considering the decreasing price of distributed energy resources in the ten past years, known energy consumers can become prosumers, that is they can both consume and generate energy. The consumers, instead, only purchase energy. There are several business initiatives whose aims are to improve the energy use and consumption all over a (smart) grid. Many of these initiatives are characterized by similar actors and operations. Indeed, the actors in an energy trading scenario could be represented by:

- *Consumer*, a physical person who needs to buy electricity.
- *Prosumer*, an entity that acts as an energy supplier (such as farmers with wind turbines or an individual who produces additional energy) and at the same time uses and buys electricity. In detail, we can consider a prosumer as a consumer with the ability to produce energy as well. So, every prosumer is a consumer while the contrary is not always true.
- *Retailer*, which buys electricity from prosumers and sells it to customers (both prosumers and consumers). The retailer is also responsible for getting customers connected to the network and for customers' billing and service.

The operations carried out by the actors could be divided into three phases, implemented through different approaches, as the study [260] suggests. The first step consists of making aware the network about the own energy supply and demand.

This step requires the adequate controls to ensure the privacy and security of the actors. The second step regards the matching among consumers and prosumers. Specifically, the consumer chooses the most suitable prosumer able to fulfill the request. Many times, this phase is implemented through an auction process. The transaction settlement is the last phase, it consists of establishing the rules, among the parties, to guarantee the transfer of energy.

During the various processes in which the prosumers are currently involved performing an energy trading protocol, their identities are forced to be disclosed, leading to some privacy problems.

The aim of this solution is to provide a protocol that takes into account the security and privacy requirements in an energy trading scenario.

When a consumer demands for energy, an auction starts, the winner prosumer stipulates a contract with the consumer. During these phases, being aware of the actors' identity could cause a possible impairment. Furthermore, dynamicity is required because prosumers are not known first. For this reason, an important issue (addressed in this solution) is to implement a privacy-preserving approach in the auction phase.

Exploiting the blockchain technology in an energy trading scenario can include the well-known advantages of distributed ledger, such as the elimination of a central governing institution, a distributed consensus, and the immutability and accountability of transactions. Observe that more auctions can be executed at the same time by suitably designing smart contracts. The execution of more auctions does not bring more possibilities of malfunction in the consensus agreement thanks to the properties of the Ethereum blockchain that prevent from latency and propagation problems by implementing a modified *GHOST* protocol (Greedy Heaviest Observed Subtree). Furthermore, the blockchain protocol prevents from double-spending attempts by design.

At the same time, designing a smart grid fully automated can be advantageous and helpful in the cost reduction of transactions and electricity. Although the other proposed solution consisting in the integration of blockchain for energy trading seems to solve the problem, there are still open issues such as the spreading of energy trading in a public blockchain, or the responsibility in the transactions derived from the anonymity ensured by blockchain. For these reasons, in this solution, we propose an approach that includes the management of the actors' identity to make transactions accountable.

This way, the final agreement will be achieved among not anonymous entities.

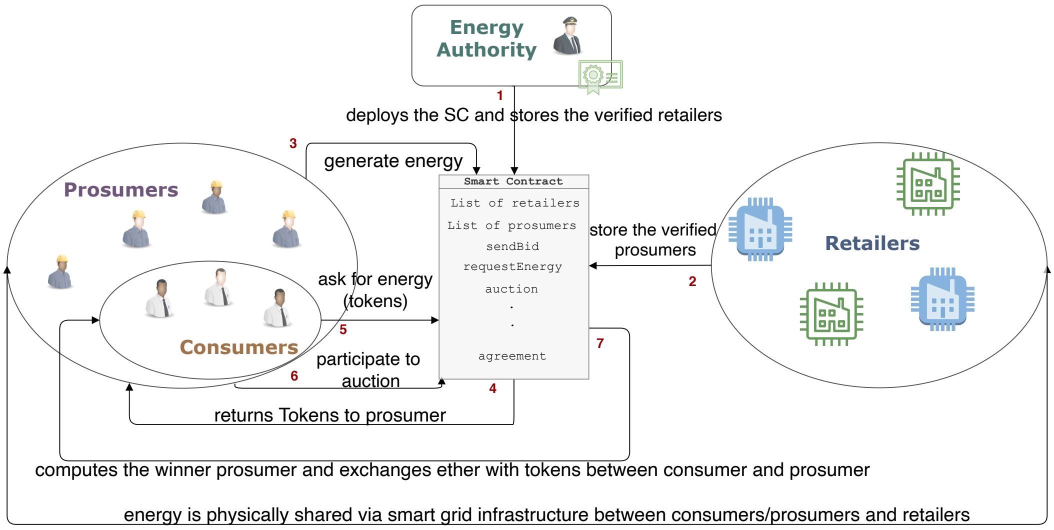

Energy Authority
1
deploys the SC and stores the verified retailers
3
generate energy
Smart Contract
List of retailers
List of prosumers
sendBid
requestEnergy
auction
agreement
Prosumers
Consumers
ask for energy (tokens)
5
participate to auction
6
4
returns Tokens to prosumer
7
store the verified prosumers
2
Retailers
computes the winner prosumer and exchanges ether with tokens between consumer and prosumer
energy is physically shared via smart grid infrastructure between consumers/prosumers and retailers

Fig. 11.1: Architecture of our solution.

11.3 Our solution

In this section, we describe our proposal. First, we present the involved entities and, then, we present the main steps of the entire process.

Figure 13.4 illustrates the overall architecture of our solution.

As described in Section 11.2, the actors we consider are *consumers*, *prosumers*, and *retailers*. We exploit the Ethereum blockchain to store the information in a distributed and immutable way and also to guarantee the security properties. Consequently, in our proposal, we include a new actor, the *Energy Authority*, which is the entity that deploys the smart contract needed to drive our solution.

In our solution, the following steps can be identified:

Setup.

In the initialization phase, a suitable smart contract SC is deployed on Ethereum by the Energy Authority. It implements the functions that are described and used in the following. Moreover, both prosumers and retailers register an Ethereum address.

System Registration.

In this phase, the entities join the system. First, the owner of the smart contract identifies each retailer and verifies its Ethereum address by a challenge-response scheme. In particular, the retailer must sign a challenge sent by the owner by using the private key of its Ethereum account. For each verified retailer, the smart contract owner invokes a function of SC and gives the retailer's address A_R as an input parameter. This function adds A_R to the list of the *verified* retailers L_R managed by the smart contract. A verified retailer R can register one or more prosumers P in the system. This operation is done by calling another function of SC and giving the prosumer's address A_P as an input. Again, the retailer verifies the prosumer's address by a challenge-response procedure. The result of the function call is the inclusion of this address to the list of the *verified* prosumers L_P, which is also managed by the smart contract. These procedures are repeated every time the smart contract owner wants to add a new retailer or a retailer wants to add a new prosumer to the system. At the end of this phase, it is possible to verify whether an Ethereum address, that we call Main Ethereum Address MEA, is associated with a verified retailer or prosumer.

Energy Production.

The actor involved in this step is the prosumer, which generates energy and trades it with the retailers. In particular, given the prosumer P_i, she/he can transfer a given amount of energy, say E, to the retailer R_j thanks to the smart grid infrastructure. Indeed, in the smart grid environment, there exists an IoT device, the smart meter, that is fundamental to link the consumer to the whole energy infrastructure. We propose an easy extension of such a smart meter that will include also the possibility of connecting to the Ethereum blockchain. This can be reached by adding a new feature on this device that will have associated an Ethereum address and, through the Internet, it will be able to interact with the blockchain network. In particular, this device acknowledges an input and output energy transfer in terms of tokens via SC.

The smart contract sends a certain amount Tk of tokens to the prosumer. The value Tk is computed as $Tk = E \cdot c_{i,j}$, where $c_{i,j}$ is the exchange rate between the prosumer P_i and the retailer R_j. Moreover, SC generates, at this point, an event *Transfer* to log the operation carried out.

Energy Request.

In this step, a consumer (or a prosumer acting as a consumer) asks for energy (i.e, tokens) by building a request containing the amount of energy needed. In particular, this task is carried out by calling the function `newAuction()` and giving as parameters the amount of requested energy, and two timestamps d_1 and d_2 used as deadlines of the auction. Observe that the consumer does not call this function by the Ethereum account generated during the System Registration step, but generates a new address for the function call, which we call Temporary Ethereum Address (TEA). This address is generated by the same consumer or prosumer and is a disposable address since it is never reused for another auction in order to avoid to link different auctions done by the same actor. Indeed, recall that a public blockchain allows everyone to rebuild the graph of interactions and transactions among users. Thus, if an actor uses her/his *Main Ethereum Addresses* (MEA) in every step of the solution, a competitor retrieves her/his bids so obtaining commercial and economic advantages for the future. For this reason, our model requires that only the winner prosumer and the energy applicant disclose their MEAs only in the *Agreement* phase. At this point, the smart contract starts a blind auction with a fixed deadline d_1.

Auction.

Any prosumer can participate in the auction by bidding a price p for this supply. In particular, the prosumer has to call the function `SendBlindBid()` of the smart contract by giving it the blind offer of the price $H(p||r)$, where r is a random value and H stands for a cryptographic hash function. This way, the real bid is hidden to the other competitors.

We remind that to prevent identity disclosure the prosumer uses a new TEA to participate in this auction.

Awarding.

At the auction deadline d_1, each prosumer that participated in the auction calls the function `sendBid()` and passes as parameters the values p and r in plain-text to disclose its offer.

After all participants reveal their offers or after the deadline d_2 established previously by the energy applicant, the auction is awarded to the best bidder. In fact, the energy applicant retrieves the best offer related to its auction by calling the function `endAuction()`, which computes the best offer and returns the winner bidder.

Before establishing the winner, this function calculates $H(p||r)$ and verifies that the result is equal to the value submitted in the previous step, thus validating the offer.

Agreement.

Now, both the consumer and the awarded prosumer must disclose their identities. For this purpose, the prosumer has to link its Temporary Ethereum Address TEA used in the previous phase to its Main Ethereum Address MEA (which is publicly available) by generating a transaction from MEA to TEA and another from TEA to MEA.

This way, the prosumer proves to be the owner of both the Ethereum addresses. It is now necessary to check that the MEA associated with the awarded prosumer has, at least, p token available in its wallet. This means that the prosumer can fulfill the consumer request. If this check fails, the smart contract discards the awarded prosumer and, by shifting the list of prosumers participants in the auction, it repeats the operations with the newly awarded prosumer. This cycle is repeated until all the requirements are fully satisfied.

At this point, the consumer has to prove to be the owner of the address A used during the auction. To do this, the prosumer generates a random value r (challenge), which is sent to the consumer. The consumer generates a new transaction from A to MEA of the prosumer having as payload r, thus proving to be able to win the challenge. Moreover, the consumer uses an identity-based authentication scheme to disclose her/his identity: for example, schemes such as OpenId-Connect and SAMLv2 [192] can be used (this aspect is out of the scope of the solution). If these operations succeed, the consumer and prosumer complete the auction by exchanging tokens and ethers as resulting from the energy request and auction.

Redeem Tokens for Energy.

This step can be carried out by everyone with tokens in their Ethereum wallets, so both prosumers and consumers, which want to redeem tokens for energy. In this step, the energy applicant has to send tokens towards the retailer by using a given function of the smart contract. This function will check that the sender has the amount of token in the wallet and that the recipient of this amount is a registered retailer. If these controls succeed, then tokens are transferred from the applicant wallet to the retailer one. At this point, the retailer sends to the applicant electricity via the smart grid's infrastructure and generates, at the same time, an Ethereum transaction with the information about the amount of energy sent.

However, since the retailer is not fully trusted (as it happens in real-life architectures as well), it is necessary to adopt some countermeasures to contrast a hypothetical malicious behavior of the retailer. At this point, the applicant's smart meter plays a fundamental role.

There are, potentially, four options: (*i*) the energy received is compliant with the agreement, (*ii*) the energy received is less than the agreement, (*iii*) the energy received is more than the agreed amount, (*iiii*) the energy is not received. Based on these situations, the smart meter will generate, as an answer, an Ethereum transaction by calling a function of *success* or *failure*. In this last case, a dispute arises between the energy applicant and the retailer. The Energy Authority is involved as a *super party* to effectively mitigate and solve the problem.

11.4 Implementation

In this section, we present a possible implementation of our proposal and describe the Ethereum smart contract that provides the needed functionalities. First, it is necessary to set-up the environment useful for the development of such a smart contract. In particular, we use *Remix* as Integrated Development Environment (IDE) and *MetaMask* that is a browser extension that allows us to run Decentralised Applications (dApps) directly on the browser without running a full Ethereum node. The programming language is Solidity [15] and the smart contract has been deployed on the Ropsten TestNet. For the sake of presentation, we only focus the attention on the most relevant aspects of the implementation.

In Table 11.1, we show costs associated with our implementation. In particular, we focus on the most common and used functions of the smart contract and also the entire (and unique) deployment of the smart contract, reporting both the values in milliether and in US dollars (in July 2020).

First, we had to declare the token *ERC*20 interface in such a way our smart contract can inherit it by implementing its functions. We gave the token the name of *SET*, which stands for both *Smart Energy Token* and *Smart Energy Transfer*. Because of the aim of such a token, the *ICO* period is not necessary so that the initial total supply was given totally to the developer of the smart contract by means of the constructor function. Indeed, in Solidity, the constructor method is called and executed only when the smart contract is deployed. We use also this properties for storing the information about the actual developer of the smart contract in the `owner` variable. We remind that, in our case, the developer of the smart contract is the Energy Authority.

Another fundamental Solidity properties we exploited is the modifier, which is used to limit the access to functions. In particular, in Listing 11.1, we implemented a modifier that, if declared in a given function, limits the access only to the developer of the contract. For example, we used this modifier in function `_add_retailer()`, which can be called only by the Energy Authority to add the addresses of verified retailers to this particular list. An analog pattern has been used also in the case of the insertion of verified prosumers into the list by declaring the function`_add_prosumer()` with the corresponding modifier `onlyRetailer` and. Generally speaking, this pattern is used every time a function needs this kind of restriction.

```
 modifier onlyOwner() {
        if (msg.sender != owner) {
            revert();
        }
        _;
    }
function _add_retailer(address _new_retailer)onlyOwner public{
    retailers_list[_new_retailer]=true;
    }
```

Listing 11.1: Application of the Solidity `modifier` in our smart contract

In the whole demand-response cycle, the first operation that is carried out in the Ethereum environment is the *Energy Request*. In particular, in Listing 11.2, we implemented the function `newAuction()` that generates a new auction on the system. The applicant has to declare how many kWhs are needed and the periods of time she/he wants to wait for the completion of the whole process. In detail, the applicant has to give two timeouts to the function. The first timeout denotes the period of time in which the auction is active while the second one denotes the period of time until the prosumer can send the plaintext bid.

When the new auction is created the smart contract adds it into the mapping all_auctions and the function emits also an event to log this operation.

```
function newAuction(uint kWh, uint timeout1, uint timeout2)public {
   uint id_auction = getID();
   all_auctions[id_auction].consumer = msg.sender;
   all_auctions[id_auction].active = true;
   all_auctions[id_auction].end_of_auction = now+timeout1;
   all_auctions[id_auction].end_of_disclosurement = now+timeout1+timeout2;
   emit newAuctionGenerated(msg.sender, id_auction, kWh, now+timeout1, now+timeout1+timeout2, now+timeout1+timeout2);}
```

Listing 11.2: Creation of a new auction

At this point, prosumers can send their blind bids to answer the token request by calling the function sendBlindBid and, before the second timeout expires, they call the function sendBid(), in which they reveal the real offer made 11.3.

```
function sendBlindBid(uint idAuction, bytes32 blind, bytes32 hashRandom) public returns (bool) {
   require(all_auctions[idAuction].active==true && now<all_auctions[idAuction].end_of_auction, "The auction is now closed");
   blindBid[msg.sender].idAuction=idAuction;
   blindBid[msg.sender].blind=blind;
   blindBid[msg.sender].hashRandom=hashRandom;
   blindBids[idAuction].push(blindBid[msg.sender]);
   return true;}

function sendBid( uint idAuction, uint cost , uint _random ) public returns (bool){
   require(all_auctions[idAuction].active==true && now>all_auctions[idAuction].end_of_disclosurement , "It's too late");
   if(blindBid[msg.sender].blind == keccak256(abi.encodePacked(toBytes(cost),toBytes(_random)))){
      bid[msg.sender].cost=cost;
      bid[msg.sender].idAuction=idAuction;
      bid[msg.sender].bidderAddress=msg.sender;
      bid[msg.sender].random=_random;
      bids[idAuction].push(bid[msg.sender]);}
      return true;}
```

Listing 11.3: Functions sendBlindBid() and sendBlind()

The next step is to compute the winner prosumer after the end of the auction. So, the tokens applicant calls the function endAuction() that first checks whether the auction is closed and, if this operation successes, it computes the winner prosumer. The code of these steps is shown in Listing 11.4.

Now, the winner prosumer and the energy applicant have to send, respectively, tokens and ethers to the smart contract, which will collect and exchange them with each other. Since the prosumer participated in the auction with a temporary Ethereum address *TEA*, now it has to use the main Ethereum address *MEA* to send tokens and receive ether. In particular, the function putToken() is called by the main Ethereum address of the prosumer, to demonstrate it is the real owner also of the address that won the auction. To achieve this goal, the prosumer has to carry on the following steps. First, it has to sign the hashed *MEA* with the private key corre-

Function	*Milliether*	*US Dollars*
`newAuction()`	0,149	0,035
`sendBlindBid()`	0,023	0,005
`sendBid()`	0,027	0,006
`endAuction()`	3	0,71
`Whole Smart Contract`	4,684	1,12

Table 11.1: Costs of the deployment of the Smart Contract

sponding to the temporary Ethereum address that has been used to participate in the auction. At this point, the prosumer uses its MEA to send this signed hashed information together with tokens in such a way to demonstrate it is the actual possessor of both the MEA and the TEA.

Finally, the energy applicant, which can be both a prosumer or a consumer, has to exchange its tokens with the retailer to obtain physically the energy needed. This operation is carried out by calling another function that is used to receive tokens and triggering the dispatch of the electricity thanks to the smart grid infrastructure.

```

function getBestValue(uint idAuction)public returns(offer memory){
   require(all_auctions[idAuction].consumer==msg.sender && now > all_auctions[idAuction].end_of_disclosurement);
   offer memory _o = bids[idAuction][0];
   uint best_cost= bids[idAuction][0].cost;
   uint n=bids[idAuction].length;
   uint pos = 0;
   for(uint j=1;j<(n);j++) {
      if (bids[idAuction][j].cost<best_cost && bids[idAuction][j].unvalid == false ){
         best_cost = bids[idAuction][j].cost;
         _o = bids[idAuction][j];
         pos = j;}}
      bids[idAuction][pos].unvalid=true;
      return _o;
      }

function endAuction(uint idAuction) public returns ( address, uint) {
   require(all_auctions[idAuction].consumer==msg.sender && now>all_auctions[idAuction].end_of_auction, "The auction
         is still active");
   offer memory best_offer=getBestValue(idAuction);
   address winnerAddress= best_offer.bidderAddress;
   uint winnerBid = best_offer.cost;
   all_auctions[idAuction].winner=winnerAddress;
   emit eventEndAuction(winnerAddress, winnerBid);
   return (winnerAddress, winnerBid);
   }
```

Listing 11.4: Ending of the auction and computation of the winner prosumer

11.5 Security analysis

In this section, we discuss the security properties and the adversary model of the solution described above. We show that the following security properties are guaranteed:

Data confidentiality refers to protecting information from unauthorized users. In our case, the real values of the bids should be hidden and protected from the other auction competitors until the auction deadline.

Data integrity refers to the completeness, consistency, and accuracy of data. Data used for energy trading should not be tampered with: in particular, once declared, the price of bids during the auction phase should not be modified by anyone.

Privacy requires that no identifying or sensitive information is disclosed if not necessary. In our case, both prosumers' and consumers' identity information should be preserved during an auction to assure fairness.

Authentication guarantees the verification of the identity of the entities accessing a protected system or a resource. We require that, after the auction, the involved actors are aware of their reciprocal identity.

Accountability assures that the operations carried out in a collaborative system occur in an open and accountable way. In our solution, we refer to the accountability of every transaction among actors.

Reliability is the probability that a system can perform a predetermined function under given conditions for a given time. In our scenario, reliability means that the actors can exploit system functionalities, such as the request for energy, the auction, or the agreement between prosumers and consumers ensuring the continuity of correct services.

After describing the security properties to guarantee, we define the adversary model. In our analysis, the energy authority is a trusted party and behaves responsibly and correctly in the system. In contrast, a retailer, a prosumer, or a consumer can be malicious and act as an adversary internal to the system. Clearly, the adversary can also be an external entity of the system. In our attack model, the adversary cannot compromise the behavior of the energy authority and cannot guess randomly generated values, secret information, blockchain private keys, passwords of the other entities. Furthermore, the adversary cannot execute transactions from the Ethereum accounts of the other entities. The adversary cannot break the cryptography primitives (e.g., it cannot revert cryptographic hash values or decrypt ciphered messages) and cannot perform physical attacks on the infrastructure (e.g., tampering with smart meters). The goal of the adversary is to violate at least one of the security properties listed above.

Let start by describing how these properties are guaranteed.

Data confidentiality is reached during the auction. Indeed, the prosumer does not send to the smart contract the price p of the supply in plain text but sends the value $H(p\|r)$, where r is a random value. To violate the confidentiality of the price p, the adversary should either (1) break the one-wayness property of H or (2) guess the random r and use a brute-force approach. Both of these possibilities are unfeasible.

Concerning data integrity, the price p of the supply offered in the auction as $h = H(p\|r)$ cannot be modified. Suppose that, in the awarding phase, the adversary sends the values p_1 and r_1, with $p_1 \neq p$, thus trying to change the offered price. As the smart contract calculates $h_1 = H(p_1\|r_1)$, if $h_1 \neq h$, this attack is detected. Having that $h_1 = h$ with $p_1 \neq p$ is impossible because this would violate the second pre-image resistance property of cryptographic hash function [224]. Moreover, the integrity of the values sent to the smart contract cannot be tampered with, thanks to the immutability of blockchain transactions: when transactions are mined by the network, data contained into the transactions are stored and not modifiable any more.

The privacy of the users is obtained because the identity of the auction winner and the consumer is disclosed only after the end of the auction, in the last phases of the energy request/supply. Indeed, the auction participants do not use their main Ethereum address MEA, which is linked to their identity, but a Temporary Ethereum Address TEA that is randomly generated and used only for this auction. In effect, the reuse of blockchain addresses is strongly discouraged since the initial adoption of the blockchain technology [39]: Ethereum addresses are pseudo-anonymous and their reuse can favor the break of pseudo-anonymity of the owners. It is worth noting that not reusing the main address at each auction also contrasts an attack based on behavior. Indeed, an attacker could track and link the activities of prosumers and consumers to gather useful information for predictive analysis based on energy consumption or the price offered for supply.

The authentication is achieved by using a challenge-response protocol, a protocol widely used for authentication [186], which is robust provided that the random number used as a challenge is generated from a sufficiently large domain and is never reused. The awarded prosumer has to link its TEA to its MEA. To do this, the prosumer signs by the TEA private key the value MEA, thus declaring its MEA. This association is guaranteed by the secretness of the TEA private key. Consumers have also to disclose their identity when a request of energy is supplied by the winner prosumer. The robustness of this authentication depends on the corresponding robustness of the digital identity chosen. Indeed, our solution is orthogonal to the identification scheme. We suggest the use of a digital identity compliant with the eIDAS Regulation [83], which is recognized to be robust and provides a normative basis for secure electronic interactions among citizens and companies all over Europe.

Accountability is reached because all the operations of energy production, energy request, energy provision, and payment are logged and stored in a public blockchain.

By looking at the Blockchain transactions, it is possible to verify the behavior of any entity. The accountability of the operations carried out in the entire environment avoids the arising of disputes among the actors: no one can claim something different from what has been reported on the blockchain.

The reliability of the solution is based on the features of blockchain. The robust Ethereum network counts a large number of nodes that work for keeping alive the network, ensuring the reliability of the blockchain-based solutions. Observe that, each actor is advantaged by well-behaving: indeed, participating in the auction requires a fee to be paid by every participant. This fee is not refunded in case of protocol violations. For example, the prosumer winner is discouraged from not providing the offered token because, in this case, the participation fee is not refunded by the smart contract (thus, protecting against attacks aiming at the denial of service).

11.6 Conclusion

In this chapter, we proposed a solution for energy trading in smart grids based on Ethereum blockchain and smart contracts. Smart grids are a domain in which the power of blockchain can be profitably exploited to achieve the aimed goals. This schema witnesses the above claim, by showing that smart contracts can enable a robust solution allowing energy trading as an auction with no referee and without requiring that different parties trust each other. An important aspect we remark here is that the implementation issues regarding smart contracts, including efficiency, scalability and costs, have been fully addressed, to provide a concrete proposal. Also the security analysis does not identify drawbacks of the solution that, in conclusion, appears promising.

12

Enabling Secure Health Information sharing among Healthcare Organizations by Public Blockchain

This work offers a solution to deal with the several security and privacy issues in sharing and exchanging health records of patients among different healthcare organizations, yet avoiding unauthorized access. This work relies on the use of a public blockchain, which is an improvement with respect to existing proposals, which exploit private or consortium blockchains. After the analysis of the advantages coming from sharing and exchanging health records of patients to have a vision of patients' medical histories, this work analyzes the security issues of this scenario and proposes the use of the blockchain technology (1) to avoid the linkage between patient's identity and e-health records and (2) to grant access to e-health records only to entities authorized by patients. The main results are the use of an eIDAS-based digital identity to control access to these records and a concrete implementation by adopting the Ethereum blockchain. The resulting solution is deeply described, and effectiveness and affordability of the proposal have been shown. The originality of this work is given by the use of 1) a public blockchain instead of a private or consortium blockchain and 2) an eIDAS-based digital identity for access control. In this way, the setup of the system is significantly simplified and it does not require the acquisition of additional resources and the related maintenance.

12.1 Introduction

The word *e-health* refers to the provision of health services using digital technology [279]. In e-health, each patient is associated with electronic health records (EHRs) that can be used for diagnosis and monitoring. Doctors may access a patient's e-health records (typically named *personal health records*) generated during the previous visits to have a clear and complete vision of the medical history without the need to ask the patient. In some cases, accessing patients' medical history is possible only if e-health records have been generated by the same healthcare organization or if a suitable sharing service between two organizations is available. On the other hand, during their life patients go to different healthcare service providers, resulting in a widespread of their e-health records among many and independent repositories, each one maintained by a different healthcare organization. Consequently, technology able to improve e-health record sharing and exchanging among healthcare organizations represents a need for the healthcare domain but also a challenge in the research community because several security and privacy problems arise in this new setting: health data are sensitive and their access should be grant only to authorized entities [143, 142].

Although the protection of EHRs is a primary goal in the healthcare industry, the number of security breaches increases every year [18]. The issue of laws and regulations to protect health information is not sufficient. For example, in 1996 the Health Insurance Portability and Accountability Act was issued in the United States. This act highlighted that the confidential section of electronic medical records needs to be protected and established standards to protect patients' privacy during electronic medical record exchange and sharing [199]. Despite this act, security issues continue to occur in many health organizations [47], and insider abuse is the prevalent cause of privacy breaches [146].

In this scenario, e-health clouds are gaining increasing popularity to facilitate data storage and sharing in healthcare [16, 167]. For example, the proposal described in [137] introduces a three-factor authentication combining password, smart card, and biometrics. This proposal resists various existing attacks, such as impersonation attack in the registration phase, offline password guessing attack in the login, and password change phase; furthermore, this proposal offers revocation. However, the adoption of cloud-based solutions leads to a series of challenges, especially how to ensure security and privacy of highly sensitive health data for the cloud. Even if the cloud is expected to be a trusted party to manage data, the cloud could misbehave because it is under attack or inadequately protected.

Recently, solutions based on *blockchain* as a distributed public repository storing users' transactions are very relevant. Generally, a transaction is a transfer of value among blockchain users who create a wallet and have two keys: the private key is used to guarantee security and authenticity of transactions, whereas the public key is used to generate the wallet identifier (address). Blockchain nodes accept, verify, and validated transactions received from other nodes by running a distributed consensus algorithms [273]. Blockchain presents many advantages that could be exploited in public health and social services and the paradigm of Blockchain 2.0 enables the creation and the development of smart contracts, pieces of codes executed within the entire network. Smart contracts allow the exchange of value and data among users, automatically verifying conditions and making decisions without third parties.

In the context of blockchain-based solutions for healthcare, important research challenges concern how to guarantee that:

1. patient's identity is known without error;
2. this identity is not linkable to an e-health record;
3. only authorized entities can access e-health records.

In this solution, we address these concerns. With regard to the first one, we rely on the concept of identity introduced by the eIDAS Regulation (EU) N. 910/2014 [86], which applies to businesses, citizens, and public authorities all over EU countries. There are several advantages of using eIDAS-compliant digital identity schemes: one of them is to have certain and legal validity all around the EU countries.

The second concern is about the link between patient and record that could be discovered by analyzing information stored in the blockchain. To hide this link, we designed a suitable cryptographic scheme that is proven to be secure.

With regard to the last concern, we observe that solutions proposed in the literature are based on private blockchains, in which only authorized entities can read or write transactions [271], or consortium blockchain, which are managed by a limited number of entities and do not implement the distributed nature of ideal blockchain [279]. In contrast, we rely on a public blockchain and designed a scheme to allow only entities authorized by patients to access their e-health records.

The impact of our solution in terms of economic, social or human development is relevant: the choice of using a public blockchain reaches the result of allowing any party anywhere in the world to access health records in a secure way. Moreover, the use of a widespread digital identity and a public blockchain combined with a new mechanism for controlled access to e-health records allows us to design a solution that can be exploited by a large number of patients to share their e-health records securely.

12.2 Proposed solution

In this section, we present the solution we propose to share electronic health records in a secure way. First, we present an overview of the approach to allow the reader to understand the idea underlying the proposed solution. Then, we give all the implementation and technical details.

12.2.1 Overview

An electronic health record is a collection of records storing patient health information in a digital format. As these records can be generated by different data sources

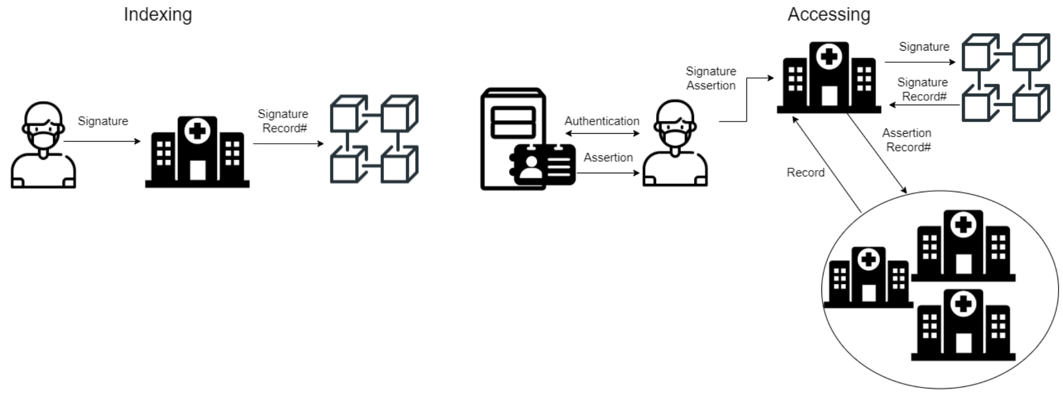


Fig. 12.1: Overview of the proposed solution.

and accessed by different healthcare clinics, these records should be shared among suitable entities. In our solution, we use a public blockchain for indexing e-health records and an eIDAS identity scheme to control access to these records. Figure 12.1 summarizes the idea underlying our proposal.

After an e-health record of a patient is generated, its indexing is done as follows. The patient produces a suitable string, say signature, and this signature can be generated only by the patient. Then, the healthcare clinic publishes on the blockchain the association between this signature and the e-health record. Access to e-health records is done in the following way. First, the patient authenticates by an identity provider, which returns a suitable assertion. The doctor who needs to access patients' e-health records receives from the patient such assertion and the signature generated previously. The doctor uses the blockchain to obtain the references to the patient's e-health records from the signature. Then, the assertion is used as proof of authorization to access such records. This scheme is simple to understand but does not explain how signatures, authentications, assertions, and references are generated to guarantee the three expected goals. These aspects are addressed in the next section.

12.2.2 Domain model

To present the proposed solution, we formalize the domain model as described by the UML class diagram depicted in Figure 12.2. The main concepts that can be identified in our solution are represented as UML classes in this model, where relationships are modeled as UML associations.

The class `Patient` represents the primary concept of our domain: it represents people who receive or need medical care. Each patient is characterized by an ID and personal information (e.g., first name, family name, date of birth, etc.). Each patient has a set of `Devices` and has associated a set of `EHRs`. As said before, each EHR contains a set of `Records`, each of them has been generated by a `DataGenerator`.

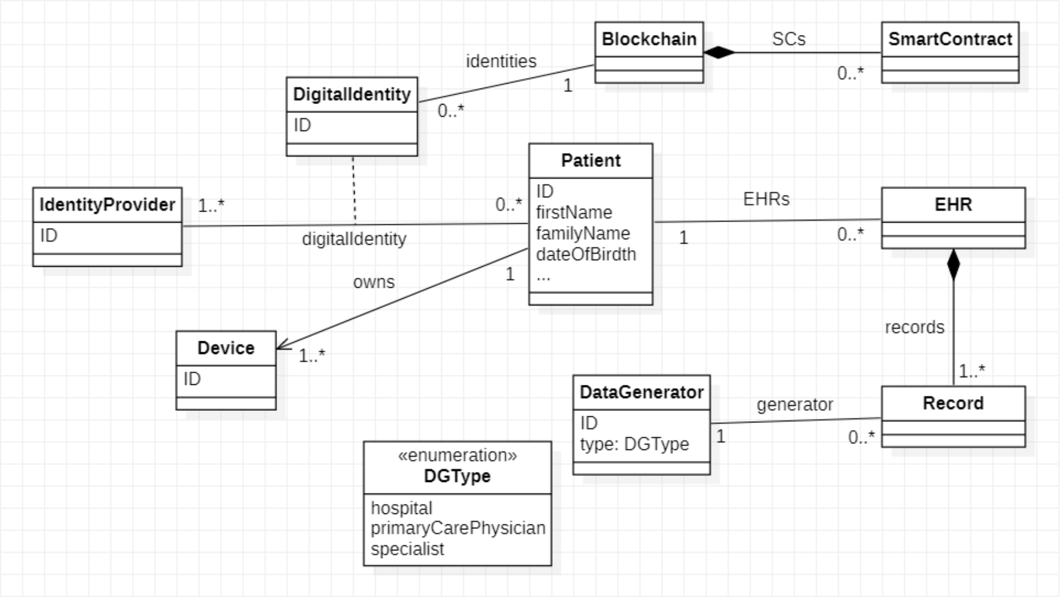


Fig. 12.2: Domain model.

Different types of data generators exist, such as hospitals, primary care physicians, specialists, and each record has a specific reference to the related data generator.

Moreover, each patient is associated with `IdentityProvider`, which represents an entity that creates, maintains, and manages the user's identity information and provides an authentication service. Patients can be associated with different Identity Providers, and this redundancy is useful in case an Identity Provider is unavailable: in this case, a user can choose which digital identity to use. Each Identity Provider releases to the patient her/his digital identity, which represents the unique identifier representing the patient as a user for that specific digital identity.

At last, it is available a public `Blockchain`, a Distributed Ledger in which `smart contracts` can be deployed. Now, we are ready to describe the interactions among the entities that allow us to provide an effective solution to the faced problem.

12.2.3 Implementation

The scheme we propose is based on five phase: system setup, identity registration, EHR indexing, EHR request, and EHR release. In the following sections, we describe each phase detailing the exchanged messages.

System Setup

In this phase, the actors perform the operations to initialize the environment. First, all data generators are associated with an identifier (e.g., an incremental counter). In a more general way, we could think of an aggregation of data generators, for example, at the level of cities. In this case, this identifier could be of the form $ID_{dg} = \langle ID_s, ID_c, ID \rangle$, where ID_s and ID_c are references to the state and city in which the data generator is, and ID is an incremental integer. For example, ⟨UK, London, 1⟩ could be the reference to the data generator in London associated with the number 1 (e.g., it could be the largest hospital).

Moreover, an asymmetric cryptographic scheme is chosen (e.g., RSA) that will be used when needed. A one-way hash function $H(x)$ that receives an input x and returns a bit string y with fixed length is also chosen. The requirement is that, given y, it should be difficult to find any message x such that $H(x) = y$ (*one-wayness*). Several hash functions can fulfill this requirement, for example, SHA-1, SHA-256, RIPEMD-160.

Finally, a suitable smart contract SC is deployed on the public blockchain, whose code is described in the following.

Identity registration

This phase is performed by a patient to obtain a digital identity from an Identity Provider. After verifying the user's data, Identity Provider issues the digital identity and access information. The digital identity is a set of personal data containing at least the following attributes (according to the eIDAS scheme [86]):

- a string PersonIdentifier, which is an identifier of the digital identity;
- a string FamilyName, the surname of the user;
- a string FirstName, the name(s) of the user;
- a date DateOfBirth, the date and year the user was born.

The user's access information is a pair ⟨username, password⟩, which will be used to authenticate. Moreover, the user generates a pair of asymmetric cryptographic keys (K_p, K_s) of the cryptographic scheme chosen in phase Setup: the private one K_s is known only by the user.

EHR Indexing

Consider the case in which a patient U goes to a hospital (i.e., a data generator) for a visit. After the visit, an e-health record of U is generated (for example, the result of an electrocardiogram), and the phase EHR indexing is carried out.

First, U is requested to authenticate by an eIDAS compliant eID scheme. The user connects to the hospital website and receives a request for authentication, which is forwarded to Identity Provider. Identity Provider starts a challenge authentication with the user. If the user completed authentication by using the access information (i.e., username and password) received in phase Identity Registration, Identity

Provider prepares an assertion, which is returned to the user by Identity Provider and forwarded to the hospital. In case of valid assertion, the user authentication successes.

Now, U calculates $F = H(S)$, where H is the cryptographic hash function chosen in phase Setup and S is the signature of the identifier of the digital identity of U (i.e., the encryption of the string *PersonIdentifier*) computed by the private key of U.

Then, the hospital calls the function `index` of the smart contract SC, which receives $\langle F, ID_{dg}, ID_{ehr} \rangle$, where ID_{dg} is the identifier of the data generator as defined in phase Setup and ID_{ehr} is the identifier of the generated e-health record of U. This function stores the mapping between F and the pair $\langle ID_{dg}, ID_{ehr} \rangle$ (thus, enabling the possibility to receive this pair starting from F).

EHR Request

Consider now the case in which U goes to another hospital and needs to access the previous e-health records. For this purpose, U is requested to authenticate by an eIDAS compliant eID scheme. According to the eIDAS Technical Specification [86], the Authentication Request contains the following fields (among others):

1. a unique attribute `IDauthentication_request` generally obtained by a combination of origin and a timestamp;
2. an element `Issuer` that identifies the Service Provider from which the request had origin;
3. an attribute `Destination` that is the address of the contacted Identity Provider for the authentication.

Differently from the standard protocol described above, here the field `IDauthentication_request` is set to the hash value $H(t, ID_{dg}, ID_U)$, where t is the timestamp, ID_{dg} is the identifier of the data generator, and ID_U is the identifier of the digital identity of U.

After U authenticates, the assertion A is generated by Identity Provider and returned to the hospital. Moreover, U signs ID_U by her/his private key and calculates the digest F of this signature (as done in the previous phase).

Now, the hospital calls the function `access` of the smart contract SC, which receives F (i.e., the generated digest) and returns a set of pairs $\langle ID_{dg_i}, ID_{ehr^i_j} \rangle$, in which each element refers to a record of U stored by the data generator ID_{dg_i} (clearly, this function exploits the mapping generated by the function `index`).

After this list is received, a request for accessing the e-health records of the user is sent to each data generator ID_{dg_i}. This request is a tuple $R = \langle A, t, ID_{dg}, ID_U, ID_{ehr^i_1}, \ldots, ID_{ehr^i_p} \rangle$, where we recall A is the assertion previously generated, t, ID_{dg}, ID_U are the timestamp, the identifier of the data generator, and the identifier of the digital identity of U used to generate A, respectively, and $ID_{ehr_1^i},\ldots,ID_{ehr_p^i}$ are the p records of U stored in the data generator ID_{dg_i} (the smart contract call has returned these references). It is worth noting that the request contains A, which is a proof of the permission given by the patient to access her/his health data (as explained in the next phase).

EHR Release

When a data generator ID_{dg_i} receives an EHR request R, the following checks are performed to verify the validity of the request:

1. it is verified that A is signed and the signature is not expired as requested by the eIDAS specifications;
2. t, ID_{dg}, ID_U is extracted from R and it is verified that $H(t, ID_{dg}, ID_U)$ is equal to the value of the field IDauthentication_request;
3. it is verified that the field PersonIdentifier of the assertion is equal to ID_U (i.e., the identifier of U's digital identity).

If all of the above checks succeed, the requested records are returned (to ID_{dg} in our case).

This concludes the access to the e-health records of the patient, which is granted to a healthcare service provider authorized by the patient. All these phases and interactions are schematized by the UML sequence diagram reported in Figure 13.5. In the next section, we discuss how our solution reaches the expected goals.

12.3 Validation

The validation of our proposal is here conducted against a set of requirements we want to guarantee. The first requirement of our proposal is to guarantee that the identity of the patient is known without error. This is reached by the use of an eIDAS-compliant eID scheme, which is universally considered secure provided that the minimum security requirements are respected (e.g., the user does not disclose her/his secret access information).

The second requirement is ensuring that the patient's identity is not linkable to an e-health record. We observe that the only link stored on the blockchain is the tuple $\langle F, ID_{dg}, ID_{ehr}\rangle$ generated in phase EHR Indexing, where $F = H(S)$ is a (ciphered) reference to the patient (i.e., F is the digest of a signature done by the patient) and ID_{dg} and ID_{ehr} refer to the e-health record, respectively. Thanks to the one-wayness of the hash function it is hard to find S starting from F. Moreover, since S can be

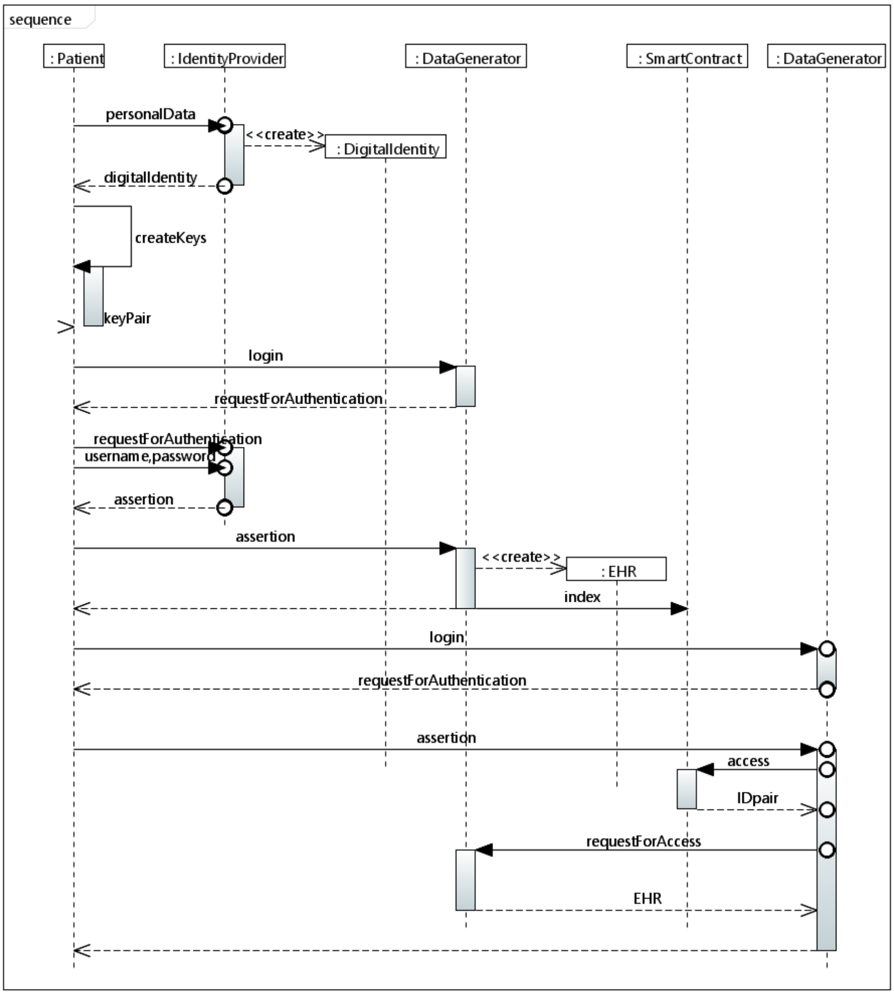


Fig. 12.3: Sequence diagram describing our solution.

generated only by the patient (because her/his private key is needed for the generation), also F can be generated only by the patient. In summary, the link between user identity and e-health record exists, but the reference to the user is ciphered and can be decrypted only with the support of the patient.

The last requirement is to guarantee that only authorized entities can access e-health records. Observe that Identity Provider issues the assertion after identifying the patient and requiring her/his authorization. Moreover, the field `IDauthentication_request` is set to $H(t, ID_{dg}, ID_U)$, where t is the timestamp and ID_U is the identifier of the digital identity of U: before releasing the record, the data generator verifies that 1) the request is not expired and 2) field PersonIdentifier of the assertion is equal to ID_U to ensure that the requested record is relative to the patient to whom the assertion is issued.

An aspect we highlight is the simpleness and cheapness of the implementation of our solution. For this purpose, we sketch in Listing 12.1 a basic code of the smart contract implementing our solution. We used Solidity [15], the language for the development of the smart contracts in Ethereum, which is an object-oriented and high-level language.

```
1  pragma solidity ^0.7.4;
2  pragma experimental ABIEncoderV2;
3
4  contract ehealth {
5     struct record {
6        uint id_dg;
7        uint id_ehr;
8     }
9     mapping(uint => record []) records;
10
11    function index(uint id, uint id_dg, uint id_ehr) public {
12       record memory _record = record(id_dg,id_ehr);
13       records[id].push(_record);
14    }
15
16    function access(uint _id) public view returns (record [] memory){
17       return records[_id];
18    }
19 }
```

Listing 12.1: Sketch of the smart contract.

For the sake of presentation, we do not show the code implementing accessory functionalities (e.g., CRUD pattern design, modifier). The struct `record` models an e-health record (Lines 4-7), and the property `records` stores the mapping between patient and e-health records (Line 8). From this code, the reader can understand the implementation of the functions `index` and `access`, which implement the functionalities described in Section 12.2.

Finally, we consider the cheapness of the solution: we computed the cost to deploy the smart contract, which is 547 Micro(ETH) (in June 2021, this is approximately 1.44$), and the cost of the call to the function index is 184 Micro(ETH), which is approximately 48 cents. This result allows us to conclude that the implementation of this smart contract is very cheap.

12.4 Conclusion

The novel solution for allowing the sharing of health records proposed in this chapter guarantees the granted access only to authorized entities and avoids the linkage between patient's identity and e-health records. The proposal relies on a public blockchain that represents an entity that can offer a proper trust level of the entire system to patients and offers the needed automatism to the different phases. The innovation of this solution is related to its simplicity that enables also a cheap implementation in existing blockchain technologies.

Part IV

Privacy and Social Networks

Social networks allow us to be always connected and share with the rest of the world the pieces of our identity that we want. While these online communities have been created to build genuine and significant connections, known risks exist in sharing information within these networks. The first risk involves basic social network users not being fully aware of their rights in terms of privacy. For example, they may not completely know what personal information is still online and, therefore, they are not aware of their past and present privacy choices. Therefore, the primary need is to guarantee social networks' honesty that means avoiding their misbehaviour. On the other hand, preserving the integrity of users' privacy choices. In 2021, 4.48 billion users of social networks [81] exist and very often, they exploit social networks to have access to various online services. Again, the problem that exists is the leakage of information or the revealing of unnecessary data. In this historical moment, the user must be central and fully aware of the information they share with the social network and the service provider. For these reasons, in this part of the thesis, we explore two different perspectives: How to guarantee the integrity of users' privacy on social networks, also relating to various additional services, such as personalized advertising, and how social networks can implement useful mechanisms responding to the logic of selective disclosure of attributes in accessing a service online.

In particular, in Chapter 13, we investigate the consequences of a social network's possible misbehavior regarding users' privacy settings. These settings are stored by the social network, which acts as a privileged party and could modify the user's choices to spread their data at any time without a user being able to prove this violation. To protect users' privacy, we propose an approach that combines the use of blockchain technology with a highly adaptable model to define users' privacy settings in social networks. A smart contract is able to determine whether the change of the privacy settings is compliant with what the user has chosen and declared. The approach aims to provide online social networks with a tool to demonstrate their honesty while allowing users to have control over their privacy. This research has been published in [157].

In Chapter 14, we give another perspective on social networks responding to the logic of selective disclosure of attributes. In particular, we propose a new solution for managing personal data based on the use of a social network in charge of providing users with the means of issuing and verifying claims and credentials for accessing online services. A service provider can verify the attributes of a user by the support of the social network as a secure and transparent repository of the selected and hidden information. The effectiveness of our proposal has been shown in a real-life scenario using Facebook as a social network [156].

Contextually, social networks can reach users with hundreds of advertisements on products or services every day. As a result, users are more and more willing to declare their interests in receiving personalized advertising [270]. In Chapter 15, we propose an approach to match users' preferences and the advertising campaigns in such a way that the social network can prove this matching. Indeed, leveraging blockchain, a social network can manage user preferences and demonstrate that a user chose a particular interest in case of need (e.g., complaint from a user or a company). Our solution has been developed on Ethereum and has been shown to be cheap and effective in [154].

13

Matching Desired and Real Privacy Settings of Social Network Users

Social networks store a considerable amount of personal data, which are also a source of information for business. To comply with users' privacy rights, all social networks allow users to select the level of privacy they desire. However, what occurs if the privacy choices of a user are modified unilaterally by the social network? The privacy settings chosen by the user are stored by the social network, which acts as a privileged party, *which could tamper with the user's choices at any time. This solution addresses this problem and proposes a decentralized approach to manage the privacy settings of a user. Any change in the privacy settings of a social network user is validated by a smart contract to ensure that it is compliant with users' expectations. The proposed solution has been implemented as an Ethereum-based decentralized application to validate the effectiveness of the proposed approach.*

13.1 Introduction

The amount of personal data available on the Web has grown exponentially, and a concentration of such data is placed in the most prominent social networks, such as Facebook, Instagram, Twitter, YouTube, and LinkedIn. The availability of a massive amount of personal information has raised privacy concerns about how social networks manage these data [78, 218, 258]. Conversely, data are an essential asset for social networks: they can sell personalized advertising with characteristics that match those of users because they know users' demographic information, activities, purchasing patterns, interests, and much more information. The contrast between the need to preserve users' data privacy and the need for social networks to monetize users' data is evident: for this reason, social networks allow users to choose to what extent they want their data to be disclosed.

However, allowing a user to choose the desired privacy settings may not be sufficient; for example, in 2018, the discovery that Facebook gave access to the personal data of more than 87 million users to Cambridge Analytica fueled interest in the risks of privacy violations [133]. In 2019, a study performed by the European Commission [248] concluded that many people do not know how to change privacy settings in social networks.

Concerning the problem of privacy settings in social networks, several studies have been performed, primarily focusing on enabling users to specify correct privacy requirements [226, 68, 225, 221] and detecting incorrect sharing settings [188, 35, 174]. In this solution, we manage a new issue related to the user's privacy settings in social networks. Although a user can set whether personal data can be transferred to third parties and which data can be displayed to whom via the Internet, the user's choices are saved by the social network and used to manage his/her data accordingly. However, what occurs if the choices of the user are modified unilaterally by the social network? Consider the following hypothetical example. *John* accuses the social network *Fantasy* of having permitted a third party to access his data and asks for compensation. *Fantasy* replies by showing the permission given by *John* to share his data. At this point, such a dispute cannot be resolved: *Fantasy* could have tampered with the stored privacy settings of *John* to hide a data breach; on the other hand, *John* could have simulated this privacy violation to obtain compensation. This issue occurs because the privacy settings chosen by a user are stored by the social network, which acts as a *privileged party*, which could modify the user's choices at any time. To the best of our knowledge, no solution to this problem has been proposed in the literature. Existing approaches based on data encryption or decentralized storage [69, 123, 48] cannot be used for the most popular social networks. Indeed, social networks show their service for free and generate most of their revenue by offering custom advertisements to specific types of consumers [36, 246]. Because the advertisement-based business model of social networks strongly depends on user data, any approach aimed at hiding such data from social networks, such as schemes based on data encryption or decentralized storage, is not effective in practice.

Our paper proposes a solution to this problem that is based on the decentralized storage of users' privacy settings in such a way that the social network cannot modify such settings without this change being tracked. The proposed approach is based on *blockchain*, a recent disruptive technology that has already been applied in many fields, such as finance and smart cities. The blockchain is a fully distributed cryptographic system that guarantees transparency, traceability, and immutability of registered information. The control is distributed among several nodes in a peer-to-peer fashion and ensures that transactions comply with programmable rules in the form of *smart contracts* [141]. The blockchain is in charge of storing both the privacy preferences of a user and the privacy settings assigned to him/her by a social network. Moreover, using suitable smart contract functions, the blockchain determines whether the privacy settings assigned to the user by the social network are compliant with those declared by the user.

A further benefit derived from the use of the proposed approach is that it favors the achievement of one of the goals of the recent General Data Protection Regulation (GDPR) [104], the new European Union privacy law. The GDPR is intended to apply guidelines and regulations to how data are analyzed, managed, stored or exchanged and applies to organizations that are registered in the EU, organizations that have an establishment or subsidiary in the EU, and to any organization that sells goods or shows services and must process or track the personal data of EU residents. The GDPR places strict obligations in terms of accountability on organizations to show their compliance with the regulation and also implies that organizations will have to maintain written records of the processing activities they perform. The use of the proposed solution achieves accountability because it allows a social network to show the correct management of the user's privacy choices. Indeed, such choices are not self-certified by the social network, as currently occurs; however, they are stored in a decentralized way on the blockchain that guarantees the integrity and authenticity of data.

The proposed approach does not aim to prevent a social network from violating the user's privacy but only to detect whether a violation has occurred in case of dispute. For example, in the case of *John* and *Fantasy* introduced above, the use of the proposed scheme allows anyone to determine whether the social network misbehaved based on information publicly available on the blockchain. The proposed approach is designed in such a way that the information stored in the blockchain does not show any sensitive information about the user.

The primary contributions of this solution are:

1. we identify an issue in the management of the user's privacy settings that allows a social network to perform and hide a privacy breach;
2. we propose a blockchain-based scheme for detecting the malicious behavior of a social network that exploits this issue;
3. we analyze the privacy settings of five prominent social networks (Facebook, Instagram, Twitter, YouTube, and LinkedIn) and define a graph-based model to represent the concepts and relations of privacy settings concisely;
4. we implement the proposed approach for Facebook, Instagram, Twitter, YouTube, and LinkedIn to demonstrate its effectiveness.

13.2 Privacy settings in social networks

In this section, we describe the privacy choices that a user can make in the most used social networks.

We start with the user's privacy settings on Facebook [96]. The "Privacy Setting and Tools" section allows users to choose who can see their activity (future posts). A user can choose among several levels of visibility of posts: public, available to friends on Facebook, available to friends on Facebook except for certain selected users, available to a specific group of friends, available to a custom group of people, or private. Also, users can choose who can find their profile. To be more specific, a user can choose if anyone can send a friend request or if the sender must be friends with at least one of the user's friends. Other privacy settings in this section include the possibility of choosing whether other users can see account information such as the user's friends list, phone number, and email address. The friends' list can be public, private, or available to friends of friends; the user's phone number and email address can be set to be available to everyone, friends, or friends of friends. In the "Timeline and Tagging Settings" section, users are shown more privacy settings that manage their profile privacy. Users can filter comments and decide whether to allow their friends to post on their timeline. They can also choose who can see what others post on their timeline and whether to allow other users to share their posts to stories or not. Also, this section displays tagging options: users can choose who can see posts they are tagged in, and they can also decide if they want to review tags before they are submitted. Facebook also allows its users to block specific users, apps, or pages from interacting with their profile or from performing specific actions. In addition to sharing content, Facebook users can also chat: the privacy options associated with this feature allow users to choose if they want to show their activity status or not. Facebook users can even change their story privacy settings, and Facebook stories can be public, available to friends and connections, available to friends, or available to a custom set of users. Last, location services, sometimes called location access, are available in Facebook's mobile app and help Facebook show its users location-based features, including allowing them to post content that is tagged with their location, obtain more relevant ads, find places and Wi-Fi nearby, and find nearby friends. When location services is on, a user can choose to turn Background Location on or off, which allows Facebook to access the device's precise location when the user is not using the app or not.

On Instagram [130], users can manage their privacy settings on different levels. First, users can manage account privacy, where anyone can view a user's profile and posts on Instagram by default. Users can decide to make their profile private so that only approved followers will be able to see its content. If a user's posts are set to private, only approved followers will see them on hashtag or location pages. Second, users can decide how photos and videos depicting them are added to their profile by the "Photos of you" section: they can choose to add them automatically to their "photos and videos of you" or not. If a user's profile is public, people can reshare the user's posts on their stories. There is an option to turn this feature off through the "Resharing stories" section. Similar to Facebook, Instagram allows users to block specific accounts from viewing their profile. Blocked accounts cannot view any sort of content posted by the user who blocked them. Instagram users can also decide if they want their activity status to be shown or not, and they can decide who can see their stories. Users can choose with whom to share their stories. Everyone on Instagram can send direct messages to any user, whether they follow them or not. However, messages from people other than one's followers are kept under a different section (Requests) in direct messages. While Instagram does not let users stop direct messages for regular messages, it can restrict direct message replies for stories. Instagram shows three privacy options for message replies to stories: Everyone, People you follow, and Off. Last, in the "comment controls" section, users can filter comments and choose who can post comments on their posts. Depending on what the user chooses, comments can be posted by Everyone, followed accounts, or followers.

In Twitter [254], privacy settings are categorized into three sections. The first section is about Tweets, which can be set to be public or protected. Public tweets, which is the default setting, are visible to anyone, including people who do not have a Twitter account, and protected tweets are visible only to followers. The second option in this section addresses the location of Tweets, which enables the addition of precise location information to a Tweet. This feature is off by default. When it is enabled, it allows Twitter to collect, store, and use Tweets' precise locations obtained by GPS. The last option in this section addresses allowing people to tag users in photos: this can be set to anyone, only following, or nobody. The second section is Direct Messages: by default, users can receive a private conversation request only by who follows them. However, a user can choose to receive requests from anyone on Twitter. The second option of this section allows users to turn on or off the notification that they have seen a message: by turning off this setting, a user would not be able to see read receipts from others. The third section, called Discoverability, allows others to find the user by email address or phone number.

The privacy settings on YouTube are simple: these settings allow users to choose who can see their liked videos, saved playlists, and subscriptions. By default, such resources are public, but a user can maintain one or more categories of resources private.

LinkedIn shows privacy settings in three sections: profile and network information, LinkedIn activity, and job-seeking preferences. Users can choose their profile visibility for viewers not logged in LinkedIn, and the profile's public visibility can be turned on or off. From this level, privacy settings on personal information, posts, or activities are generated; by default, LinkedIn sets primary information as public, but users can decide what category of data include in the public profile. Public profiles can be found through search engines. Public visibility can be switched on or off: if it is off, the profile will not be visible for not logged-in members; if on, users will show basic profile information (name, number of connections, industry, and region). Profile photos can be shared with connections, the network (LinkedIn members connected up to three degrees of separation), and anyone (all LinkedIn members). Users can choose to show their headline or not, posts and activities, current experience, past experiences, and education in the profile. Also, users can choose to show only the first letter of the last name. Personal information, such as email, can be shared with no one, 1st-degree connections, 1st- and 2nd-degree connections, and anyone on LinkedIn. With regard to connections, users can choose to hide or not hide them from other connections. LinkedIn also implements the concept of "views": if a user sees another user's profile, the user with the viewed profile will be notified. LinkedIn protects the user's privacy via three different levels of profile viewing options: name and headline, private profile characteristics, and private mode. Concurrently, a user can manage active status by three options: no one, connections, and all LinkedIn members. Users can choose whether or not they want to share changes in jobs or education with the network and to notify the network if they have been mentioned in a blog or article post. Users can allow others to be mentioned or tagged in content, such as posts, comments, and tags in the photo. Users can decide whether LinkedIn can save the information entered when applying to jobs (internal or external application) directly on LinkedIn. When users apply for a job, they can choose to share their full profile with the job poster. Users can be open to opportunities or not: in the first case, recruiters will be able to find users by career interests. Users can also decide to create a job alert for companies: they will be notified of new jobs matching their skills. Last, users can choose to share or not share their interests with recruiters.

The description of the choices that social networks show about privacy is used in the next section to define a model to represent the privacy settings of a user.

13.3 Modeling privacy settings

The purpose of this section is to define a model to represent the privacy settings of a user in a social network. To do this, we introduce certain preliminary definitions.

Algorithm 1 Generation of the Privacy Feature Graph

```
Input  $S$: social network
Variable  $PF^S = \langle N, E \rangle$: privacy feature graph of $S$
  for each type i of user in S do
    add node $g_i$ to N of $PF^S$
    for each node $g_j \neq g_i$ in N do
      if $g_i \subseteq g_j$ then
        add edge $(g_i, g_j)$ to E of $PF^S$
      end if
    end for
  end for
  for each type i of information in S do
    add node $r_i$ to N of $PF^S$
  end for
  return $PF^S$
```

Definition 1. *Given a social network S, the* privacy feature *graph of S is a direct graph $PF^S = \langle N, E \rangle$, in which nodes are divided into two disjoint and independent sets G (groups) and R (resources); thus, $N = G \cup R$. A node in G represents a group of users (i.e., profiles) of the social network S, while a node in R represents a type of information in a user's profile. Given two nodes g_i, $g_j \in G$ with $i \neq j$, an edge from g_i to g_j denotes that all the users in g_j are also in g_i.*

The generation of the *privacy feature* graph is formalized in Algorithm 1.

Now, we show an example to help the reader understand this definition better.

Example 1. *In Figure 13.1, the privacy feature graph of Instagram is shown. In this graph, we have four group nodes $g_1, \ldots, g_4$ (i.e., $|G| = 4$) and five resource nodes $r_1 \ldots r_5$ (i.e., $|R| = 5$). As described in Section 13.2, an Instagram user can choose to show profile data to four categories of users: g_1 denotes all Instagram users; g_2 and g_3 denote the follower users and followers who are also followed by the account owner respectively; and g_4 denotes all the non-follower users who are followed by the account owner. Again, we have seen that on Instagram, privacy settings apply to five categories of resources: $r_1, \ldots, r_5$ denote photos and videos, stories, story message replies, comments, and active status, respectively. Finally, the edges from g_1 to g_2, g_3, and g_4 denote that g_1 is a superset of all the other group nodes (e.g., followers are included in everyone). Again, for the same reason, note that g_2 is a superset of g_3.*

Starting from the analysis shown in Section 13.2, we show how it is possible to build a privacy feature graph for each of the social networks considered in the previous section.

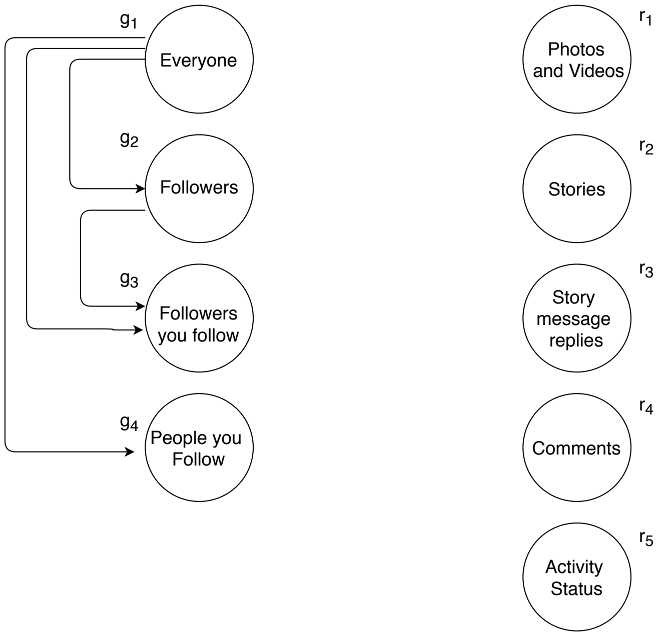


Fig. 13.1: Privacy feature graph of Instagram.

Facebook has five group nodes: everyone, friends, friends of friends, restricted users, and custom (the user can decide to include or exclude certain users). Resource nodes are made of posts, tagged posts, stories, friends' lists, profile information, and app information.

In Twitter, we identify three group nodes (everyone, everyone in Twitter, and followers) and the following resource nodes: tweets, location, tag, direct messages, notification, email, and phone number.

YouTube is simplest and only has the group node everyone and three resources: videos, saved playlists, and subscriptions.

In LinkedIn, there are three group nodes (everyone, connections, and members), and there are five resource nodes (posts, tagged posts, personal information, job applications, and job interests).

The model defined in this study can be extended to many other social networks. Now, we define a model for the privacy settings that a user can choose in a social network.

Definition 2. *Given a user U and a social network S with privacy feature graph $PF^S = \langle N, E\rangle$, the* privacy setting *graph of U in S is a direct graph $PS_U^S = \langle N, E \cup P\rangle$, where $p \in P$ is an edge $(g \in G, r \in R)$.*

This graph has the same nodes as PF^S, and a superset of edges: the additional edges w.r.t. PF^S are edges from a group node to a resource node. Specifically, an

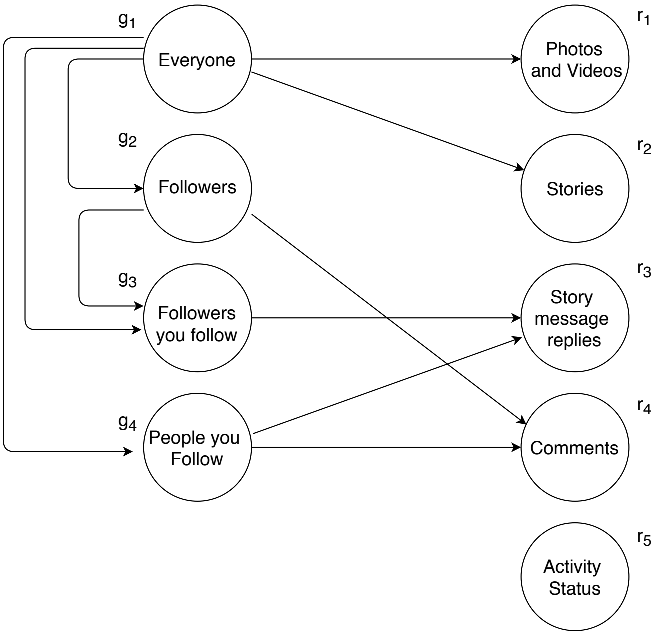


Fig. 13.2: Privacy setting graph of a customized profile.

edge ($g \in G, r \in R$) models that the users denoted by g can access the information shown by the resource r.

Example 2. *Considering Instagram again, we examine the privacy settings of three different users mapped by three privacy setting graphs $PS^S_{U_1}$, $PS^S_{U_2}$, and $PS^S_{U_3}$.*

The first user U_1 has a public profile. Then, $PS^S_{U_1}$ is similar to PF^S and has four edges $(g_1, r_1), \dots, (g_1, r_4)$, which indicates that everyone (g_1) can access all resources $r_1 \dots r_4$.

In the second case, user U_2 chooses to set her/his profile as private. In a private profile, only U_2's followers can access resources; therefore, $PS^S_{U_2}$ has as edges $(g_2, r_1), \dots, (g_2, r_4)$, where g_2 are the followers of U_2.

The most interesting and probably common case is when a user, say U_3, has customized privacy settings modeled by the privacy setting graph $PS^S_{U_3}$, such as the one shown in Figure 13.2. The edge (g_1, r_1) represents that every Instagram user can access photos and videos posted by U_3; (g_1, r_2) represents that every Instagram user can access stories posted by U_3. Story message replies can be sent by those users who are followed by U_3; this is shown through the edges (g_3, r_3) and (g_4, r_3). Permission to comment is given to followers and followed accounts (i.e., "People you follow and your followers"); this is shown by the edges (g_2, r_4). and (g_4, r_4). Finally, U_3 has chosen not to show the activity status: this is modeled by the lack of edges to the node activity status.

Now, we are ready to introduce the two definitions that will be widely used in the following.

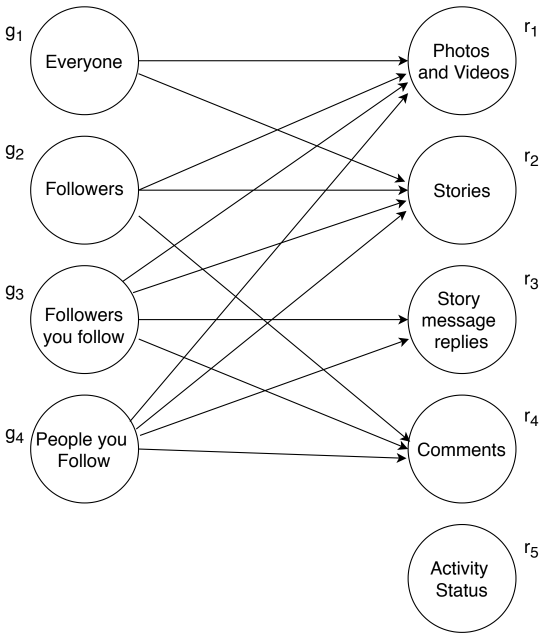


Fig. 13.3: Extended privacy setting graph.

Definition 3. *Given a privacy setting graph* $PS_U^S = \langle G \cup R, E \cup P \rangle$, *we define the* extended privacy setting *graph* $EPS_U^S = \langle G \cup R, P \cup P' \rangle$ *as the bipartite graph such that every edge connects a vertex in G to one in R, and P′ is the set of edges* (g_i, r_j) *such that there exist in* PS_U^S *the edges* (g_y, g_i) *and* (g_y, r_j) *with* $i \neq y$.

The following example shows an intuitive idea of how to build an extended privacy setting graph.

Example 3. *Consider the privacy setting graph shown in Figure 13.2. To obtain the bipartite graph, we must remove the edges between group nodes because they violate the bipartite requirement. For any edge* $e \in E$ *from* g_y *to* g_j*, if* g_y *has no edge to any resource node, then e can be removed. Otherwise, for any edge* $(g_y, r_j) \in P$*, we add to P′ a new edge* (g_j, r_j) *before removing e. Figure 13.3 shows the extended privacy setting graph derived from the privacy setting graph shown in Figure 13.2. For example, note that everyone can access* r_1 *and* r_2*, and everyone is a superset of* g_2*, leading to the addition of the edges* (g_2, r_1) *and* (g_2, r_2)*, when the edge* (g_1, g_2) *is removed.*

We conclude this section by defining a binary encoding of the model, which will be useful in the implementation of the proposal.

Definition 4. *Given an extended privacy setting graph* $EPS_U^S = \langle G \cup R, E \cup P \rangle$*, we define the* serialized privacy setting SPS_U^S *as the* $|G| \cdot |R|$*-bit string such that the x-th bit with* $1 \leq x \leq |G| \cdot |R|$ *is* 1 *if and only if there exists the edge* (n_i, r_j) *with* $i = (x-1)/|G| + 1$ *and*

	g_1	g_2	g_3	g_4
r_1	1	1	1	1
r_2	1	1	1	1
r_3	0	0	1	1
r_4	0	1	1	1
r_5	0	0	0	0

Table 13.1: Adjacent matrix of the considered SPS_U^S.

$j = (x-1)\%|G| + 1$, where / and % denote the quotient and the remainder of Euclidean division, respectively.

This definition allows us to represent an extended privacy setting graph using a bit string. In the following example, we show how this string is obtained.

Example 4. *Consider the extended privacy setting graph shown in Figure 13.3. This graph can be shown by the adjacency matrix reported in Table 13.1, whose 32-bit string encoding is $0x000FF370$, based on the definition given above.*

13.4 Proposed solution

In this section, we present the proposed approach to ensure that the privacy settings requested by a user cannot be modified by a social network without being detected.

We introduce a scenario in which we have four actors:

- an *online social network*, a decentralized and distributed computer network that shows services through the Internet.
- a *user*, a person using one social network to communicate with other people and share information and resources.
- a *Blockchain*, a Distributed Ledger enabling smart contracts, such as Ethereum.
- a *smart contract*, which is deployed on the blockchain.

Figure 13.4 shows the architecture of the proposed solution and the interactions among the actors performed based on the following operations.

1. *Social network registration*. Each social network needs an *External Owned Account* (EOA) to operate on the blockchain, and a blockchain address is generated as follows: a pair of private and public blockchain keys are generated, and then, the

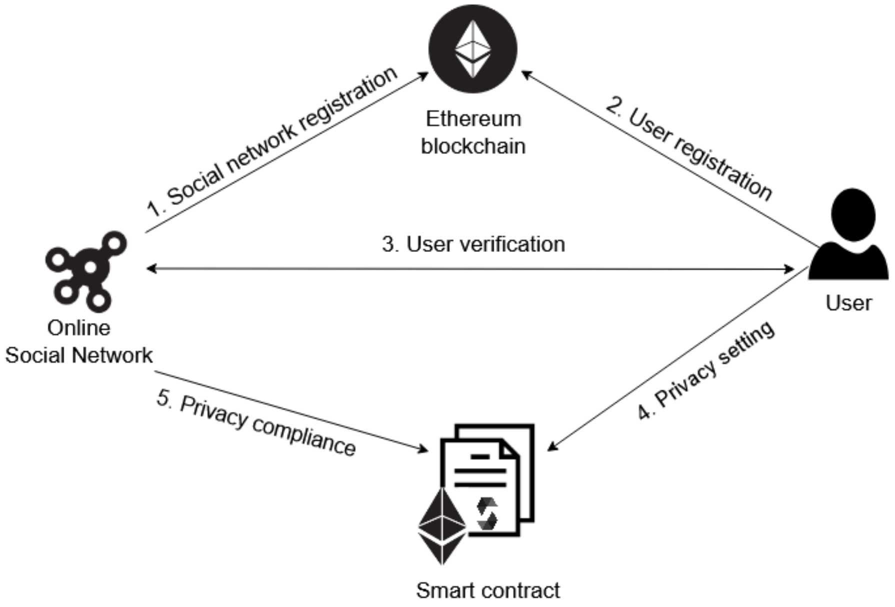


Fig. 13.4: Interactions between the actors.

corresponding blockchain address is computed as the cryptographic hash of the public key. Each social network makes its blockchain address publicly available.

2. *User registration*. A user also needs an EOA: following the same procedure described above, each user U can generate her/his blockchain address, say A. Also, each social network SN includes SN includes two new fields in the user's profile: the first is for the user's blockchain address, the second is filled in by the user's secret (e.g., a password), which works as a *salt*. Finally, U generates a transaction on the blockchain from A to the social network blockchain address, which is publicly available, with a payload $H(snid, A, salt)$, where *snid* is the ID of the user in the social network, and H is a suitable cryptographic hash function. Via this transaction, the user links her/his social network identifier to blockchain address A. This mapping can be verified only by knowing *salt*, and this is performed by the social network.
3. *User verification*. This operation is performed by a social network to verify that the blockchain address declared by one of its users is correct. Given a user U, this check is performed as follows: first, the social network extracts from U's profile the values *snid*, A, and *salt*. Then, it searches for a received transaction coming from address A having in payload $H(snid, A, salt)$. If no transaction is found, then this check fails; otherwise, the mapping is correct.
4. *Privacy setting*. This operation is performed by a user to declare her/his desired privacy settings. Specifically, this is performed by calling the function `setPrivacy` of the smart contract (say SM) and by passing the extended privacy setting bit string as a parameter (see Definition 4) derived by the privacy feature graph

that represents the desired privacy settings of the user (see Definition 2). The smart contract stores the settings associated with the user's blockchain address.

5. *Privacy compliance*. When a social network wants to assign or modify the privacy settings of user U, the function `checkSettingGraph` of SM is called to ensure that the new settings are compliant with the user's preferences stored on the blockchain,

   This function has A and B_U, where A is the blockchain address of the user, and B_U is the extended privacy setting bit string derived from the privacy feature graph representing the privacy settings to be assigned to the user. This function (1) extracts the bit string P representing the preferences of the user saved locally, if any, and (2) compares bitwise P and B_U to verify that a zero bit in P is associated with a zero bit in B_U. Only if this check does not fail, the function returns that the settings are compliant with the user's preferences.

In summary, the idea underlying the proposed approach is to exploit a blockchain smart contract to store both the desired privacy preferences of the user and the privacy settings assigned to the user by the social network. Also, the smart contract verifies whether the privacy settings assigned to the user by the social network are compliant with those declared by the user. Because all operations are stored in the blockchain, and transactions cannot be modified, this solution implements an easy method to show accountability of all the privacy assignments performed by a social network. A social network can use this solution as proof of having acted correctly, based on the accountability requirement of the recent General Data Protection Regulation (GDPR) [104].

13.5 Design and implementation

In this section, we describe the development of the proposed solution, which is based on a decentralized application (DApp), called *Your Privacy Manager DApp*, which runs on a blockchain. We start by discussing the choices concerning the architecture of the system.

13.5.1 System architecture

We start by surveying the most commonly used blockchain technologies and highlighting their advantages and drawbacks based on the interesting analysis reported in [231].

Ethereum [269] is considered the second largest and global cryptocurrency platform and is a permissionless blockchain enabling the creation of decentralized applications through the use of smart contracts. This platform can be considered a system that is globally shared and implementing a cryptographically secure transaction-based state machine [269]. A smart contract is defined as a piece of code verifying and enforcing conditions that stipulate a digital contract between parties that does not require a third intermediary. Smart contracts are written in *Solidity*, an object-oriented and high-level Turing complete programming language.

IOTA [211] networks were designed for IoT applications and are permissionless blockchain networks that are built on Tangle, a new data structure based on a directed acyclic graph, which does not need blocks, miners, or any chain. For this reason, IOTA transactions are free. Although this platform can yield better performance than Ethereum and Hyperledger blockchains, its primary limitation is that devices may not be not capable of performing the Proof of Work, resulting in a bottleneck when transactions occur [98].

EOSIO [90] is the first blockchain platform that uses the Delegated Proof of Stake consensus algorithm. Converse to traditional proof-of-work-based systems, EOSIO is public, permissionless, and suffers from serious attacks derived from exploiting vulnerabilities in DApps and leading to millions of dollars lost for EOSIO users, as discussed in references [124, 215].

Among permissioned blockchains, Hyperledger Fabric [28] is an implementation of an open-source private blockchain running smart contracts and is intended to form a foundation for developing applications with a modular architecture. Hyperledger Fabric allows components, such as consensus and membership services, to be plug-and-play.

MultiChain [110] is a platform enabling the creation and deployment of private blockchains to be developed and used inside organizations. The system administrator sets a series of user permissions to introduce controls over transactions and block size.

Chain Core [89] is another platform for private blockchains and is typically used to initiate and transfer financial assets based on permission from the blockchain infrastructure. Corda [52] is a distributed ledger platform for recording and enforcing business agreements among institutions. Chain Core and Corda also rely on smart contracts and allow participants to manage permissions.

Open Chain [201] is an open-source distributed ledger technology based on the Partitioned Consensus: every Open Chain instance has one authority validating transactions. This platform aims to manage the digital assets of organizations in a scalable and secure way.

This analysis provides a way to verify the suitability of each technology with respect to the needs of the proposed solution. The Hyperledger Fabric, MultiChain, Chain Core, and Open Chain platforms suffer from limitations due to their permissioned nature and goals [28, 89, 52, 201]: users belong to disparate environments and organizations, which makes it difficult to individuate who can arrange permissions [110].

Therefore, from the perspective of users' accessibility, Ethereum, IOTA, and EOSIO are valid options for the implementation of the proposed solution. Among these platforms, we choose Ethereum for developing the proposed solution because it is the most used and widespread permissionless Blockchain enabling smart contracts.

13.5.2 Implementation

In this section, we describe the implementation of the DApp called *Your Privacy Manager DApp*. This DApp provides a front-end to interact with the Ethereum blockchain via a smart contract, and the application's back-end code is executed on a peer-to-peer decentralized network.

The development environment relies on Truffle, a testing framework for blockchain that uses the Ethereum Virtual Machine (EVM)), which simplifies the DApp implementation process for developers.

For the deployment of the smart contract, Ganache is used. Ganache is part of the Truffle suite and is a personal blockchain used to develop distributed applications for Ethereum and Corda. Additionally, Ganache can be used to run tests and deploy contracts. The local test network produced by Ganache can be used for development purposes only. To deploy the smart contract on the Ethereum blockchain, a connection to the primary network is necessary.

A user can interact with the Ethereum decentralized application through the browser without the need to run a full Ethereum node but does need MetaMask. Unlike other wallets, MetaMask is a web browser plug-in that supports Brave, Google Chrome, and Firefox and provides a user interface to sign blockchain transactions and to manage identities.

For the implementation of the proposed application, we used Node.js, an open-source, cross-platform JavaScript run-time environment. The use of Node.js enables the use of the same programming language to develop both server- and client-side scripts.

The user interface consists of HTML and JQuery functions that display different DOM elements depending on what options the user selects. The core of the client-side application is a JavaScript file that contains all the functions necessary to load the smart contract, connect with the wallet, capture the user's desired privacy settings and call the smart contract functions.

Figure 13.5 schematizes the sequence diagram of the privacy setup process. A similar diagram can be traced to represent the verification of compliance. A user

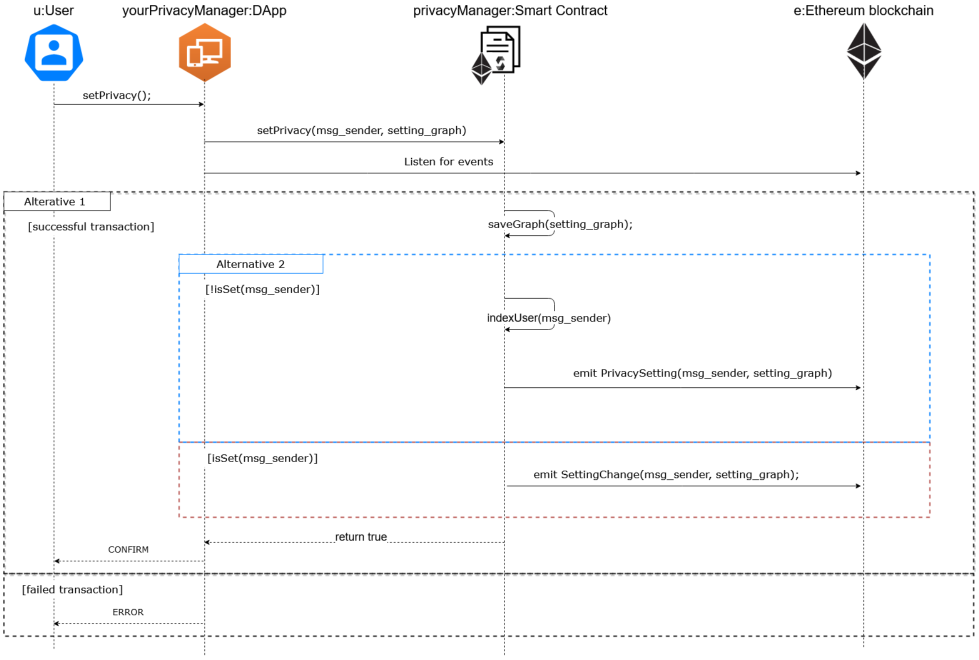


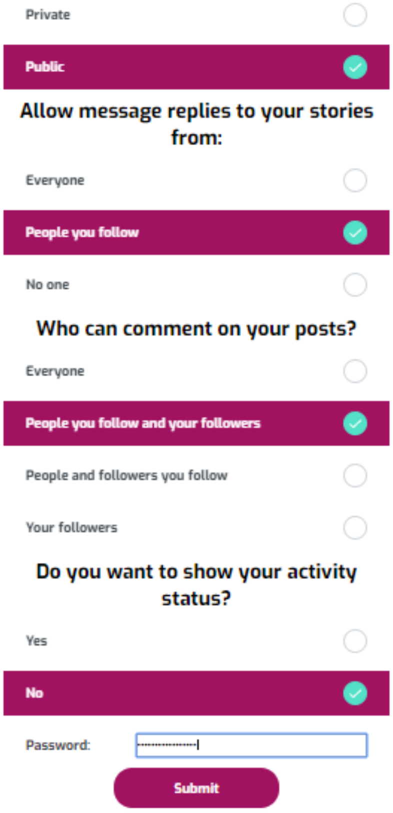
Fig. 13.5: Sequence diagram showing functional calls and events.

accesses the DApp using a web browser with Metamask and submits the desired privacy settings through a form. Then, the JavaScript asynchronous function `setPrivacy()` calls the setPrivacy function of the smart contract. The smart contract then saves the setting graph of the user (sender): Alternative 2 in Figure 13.5 records if this is the first time the user sets his/her privacy, and how smart contract associates this graph with the new user. The result of this operation is logged on the blockchain using the proper event. Alternative 1 represents the result of the operation requested by the user: if no error occurred in the process, a confirmation message is sent to the user. An error message is shown to the user in the case the transactions failed for any reason (e.g., out of gas, incorrect signature, exceeding block gas limit).

Now, we show the implementation of the privacy settings in the case of an Instagram user. We present the data flow among several actors: the user, its Dapp, the social network Instagram, and the Ethereum blockchain with the smart contract.

A preliminary operation that occurs only once regards the link between the user's Instagram account and blockchain address. For this purpose, the DApp includes three fields to be filled in by the user:

- *Username*, which is the username of the user on Instagram;
- *Blockchain address*, which is set to the user's blockchain account address, say *A*;
- *Password*, which is set to the password of the user on Instagram.



Private ○

Public ✓

Allow message replies to your stories from:

Everyone ○

People you follow ✓

No one ○

Who can comment on your posts?

Everyone ○

People you follow and your followers ✓

People and followers you follow ○

Your followers ○

Do you want to show your activity status?

Yes ○

No ✓

Password: ················

Fig. 13.6: Selection of the desired privacy settings on the DApp.

Once the user fills in these fields, the application generates a transaction to the social network blockchain address with a payload $\langle H(username, A, password^1)\rangle$.

The social network receives this transaction, extracts the blockchain address of the sender (i.e., A), searches for the username who has declared this address, and retrieves the password of this user. Now, the social network has all the data required to calculate the same digest and to verify that the digest received by the transaction has been correctly generated. This process assures that the link between the username and blockchain address has been sent by the user. To generate such a transaction, the password of the user and the private key associated with the blockchain address must be known because the transaction is signed.

Then, the user can select their desired privacy setting on the DApp. The part of the user interface relative to this operation is shown in Figure 13.6, which shows privacy settings that the user can choose. After having selected the desired options and input their password by clicking the button Submit, the DApp generates an Ethereum transaction for the smart contract. This transaction is built based on the

[1] For the sake of presentation, we assume that the OSN knows the user's password. Actually, only the password digest is known: thus, the password digest should be used in place of the password.

operation described in Section 13.3 and is handled by the function called PrivacyManager; the code of this function and the entire smart contract is reported in Listing 13.1. The smart contract is implemented in Solidity, a statically typed programming language designed for developing smart contracts, compiled to bytecode, and executed on a suitable virtual machine (EVM). The smart contract has been written using the CRUD pattern. Considering how the EVM currently works, this process is recommended. In the first portion of the code, a structure, UserSettingStruct, is defined using the struct keyword. This structure contains two types of information:

- *setting_graph*, of type bytes, is a dynamically sized array used to represent the setting graph.
- *index*, of type uint, represents the position of a user's blockchain address in the userSettingIndex array, based on the CRUD pattern.

Line 8 defines a mapping between the address of a user and an element of the previously defined structure. Finally, the array userSettingIndex is used to verify the correctness of this mapping.

The following events are also part of the smart contract:

- *PrivacySetting*: This event is emitted every time a user declares a desired privacy setting for the first time;
- *SettingChange*: This event is emitted every time a user updates the desired privacy settings;
- *CheckSetting:* This event is emitted whenever a privacy setup is configured through the OSN front-end and shows whether the privacy setting graph received as an input is compliant with the one previously set by the user.

The function isSet (Lines 14-17) checks if a user is associated with a setting graph. To do this, the function first checks if the userSettingIndex array is empty (Line 15). If the array is empty, no association exists, and the function returns false. If the array is not empty, the function retrieves the index associated with the input from the userSettingStruct struct through the mapping. Then, the function retrieves the address contained in the userSettingIndex array at the retrieved index position and checks if this address matches the input. A Boolean value is returned based on a comparison result.

The function setPrivacy (Lines 18-30) is used by users to set or update their privacy preferences. The input _setting_graph is a dynamically sized array of bytes used to represent the privacy setting graph associated with the user's preferences (see Definition 4). First, the isSet function is called (Line 19). If isSet returns false, the function indicates that the sender has never submitted a graph; thus, the new setting_graph value is stored as the value of setting_graph in the userSettingStruct struct associated with the address of the sender through the mapping (Line 21). Then, the address of the sender is pushed in the array userSettingIndex, and the new index is stored as a value for the index attribute in the structure. Finally, the PrivacySetting event is emitted.

Everything works similarly if the user had already submitted a graph in the past (Lines 25-29). In this case, only an update is required. Thus, only the graph contained in the structure must change, and the index remains the same. Once the update has occurred, the SettingChange event is emitted.

The function checkSettingGraph (Lines 31-50) is called by an OSN interested in associating a set of privacy options with a user. First, the function checks if the user address has ever been associated with a privacy preference graph. To do this, the isSet function is called. If the result of this call is false, the CheckSetting event is emitted, and the value of its third parameter is false because the user has never defined her/his privacy preferences. Otherwise, if isSet returns true, the checkSettingGraph function checks if the input privacy settings and those already associated with the user are compatible in size. If their sizes do not match, then the CheckSetting event is emitted, and the value of its third parameter is set to false.

```
1  pragma solidity ^0.5.10;
2
3  contract PrivacyManager {
4      struct UserSettingStruct {
5          bytes setting_graph;
6          uint index;
7      }
8      mapping(address => UserSettingStruct) private userSettingStructs;
9      address[] private userSettingIndex;
10     event PrivacySetting(address user, bytes setting);
11     event SettingChange(address user, bytes setting);
12     event CheckSetting(address OSN, address user, bool result);
13
14     function isSet(address _address) public view returns(bool exists) {
15         if (userSettingIndex.length == 0)
16             return false;
17         return (userSettingIndex[userSettingStructs[_address].index] == _address);
18     }
19     function setPrivacy(bytes memory _setting_graph) public returns(bool success) {
20         userSettingStructs[msg.sender].setting_graph = _setting_graph;
21         if (!isSet(msg.sender)) {
22             userSettingStructs[msg.sender].index = userSettingIndex.push(msg.sender) - 1;
23             emit PrivacySetting(msg.sender, _setting_graph);
24             return true;
25         }
26         else {
27             emit SettingChange(msg.sender, _setting_graph);
28             return true;
29         }
30     }
31     function checkSettingGraph(address _user, bytes memory _request) public {
32         if (!isSet(_user)) {
33             emit CheckSetting(msg.sender, _user, false);
34             return;
35         }
36         bytes memory setting_graph = userSettingStructs[_user].setting_graph;
37         if (setting_graph.length != _request.length) {
38             emit CheckSetting(msg.sender, _user, false);
39             return;
```

```
        }
        else {
            for (uint i = 0; i < setting_graph.length; i++)
                if ((setting_graph[i] | _request[i]) > setting_graph[i]) {
                    emit CheckSetting(msg.sender, _user, false);
                    return;
                }
            emit CheckSetting(msg.sender, _user, true);
        }
    }
}
```

Listing 13.1: Code of the smart contract.

Otherwise, the function compares the input setting graph `request` coming from the OSN with the user's request. This comparison is performed for each corresponding byte of the byte arrays by a bitwise OR: if the output of this OR operation is greater than the byte in the stored `setting_graph` in the `userSettingStructs`, then the settings are not compliant with the user's preferences. Eventually, a `CheckSetting` event that displays the OSN address (`msg.sender`), the user's address, and the result of the call is emitted: in particular, the result will be true if the request from the OSN is compliant with the user's, false otherwise.

Finally, we consider the case in which a user wants to update the privacy setting. This case is similar to the setting of the privacy seen above, with the only difference that instead of the `PrivacySetting` event, the `SettingChange` event is emitted. The use of events is necessary because the real value returned by a function is always the hash of the transaction that is created: transactions do not return a value to the front end because they are not immediately mined and included in the blockchain. As a solution, to obtain the return value from the function, the front end must maintain watching for that event. This process also implies that a listener must be active in detecting the events. Specifically, the OSN listens to the events, and when the `SettingChange` event occurs, the social network checks if the new privacy settings of the users are compliant with the existing settings associated with the user. If they are not compliant, then the social network has to update the user settings accordingly.

To validate the proposed smart contract, we used Etherscan, a Block Explorer and Analytics Platform for Ethereum, where it is possible to find all Ethereum transactions. When a function of the smart contract is called, the transaction begins, and after a few seconds, it is visible on Etherscan. Also, in the *Event Logs* section, the events emitted during the execution of the function are also shown. The contract has been implemented by *Remix - Solidity IDE* connected with Metamask, an extension for accessing Ethereum enabled distributed applications from the browser: the Ethereum testnet used is *Ropsten*.

The implementation of the proposed solution can be found on GitHub at `https://github.com/Lara-F/Your-Privacy-Manager`. The smart contract has been deployed on Ropsten and is reported at `https://ropsten.etherscan.io/address/0x3657b2322f2dde1fea4af963ae2f5b7837db4fe1`.

13.6 Related Work

In this section, we survey the most important proposals related to the proposed approach. We start by considering the literature related to social networks.

Facebook, Twitter, and Instagram are social media platforms that are familiar to many people, and many online users use them every day [246]. Studies [33, 30] explore the growing importance of social networks in contemporary development, investigating the relationship between social media and crime or homicide. Considering Facebook, the authors suggest that the association between Facebook penetration and crime or homicide is overall negative because social media are used essentially for productive ends. However, reference [33] highlights the prominence of a positive relationship between social media in terms of Facebook penetration and terrorism. The direct link between social media and governance dynamics is studied in references [32, 135], while that between social media and corruption is discussed in reference [136]. These studies show the growing importance of information and communication technology and the spread of social media in the daily life of citizens. These relationships also describe on the tourism sector [31].

With regard to user privacy, [268] highlights the fact OSNs' privacy settings and features are frequently concealed, too difficult to understand, or change so quickly that it is nearly impossible for users to maintain them. This complexity raises the question of whether OSN users' privacy settings match what they intend to share. In reference [179], the authors present the results of an empirical evaluation that estimates privacy objectives and behaviors and compares these with the privacy settings on Facebook. This study emphasizes that, although Facebook provides the opportunity to configure privacy preferences at a detailed level on most user data (e.g. each album, video, photo or status update), users have to manage so much data that, even if they used privacy-preserving features regularly, they would not be able to monitor everything. Considering that every participant in this study stated that they had identified at least a sharing violation, the results of this paper show that managing privacy settings is not an easy task both for OSNs and users. A similar study [174] shows results obtained by deploying a survey implemented as a Facebook application. The primary purpose of this study is to quantify the extent of the problem of handling privacy and measuring the discrepancy between the desired and real privacy features. The analysis shows that nearly half of the content users uploading to Facebook are exposed to all Facebook users as a result of being shared with the default privacy settings.

Another interesting perspective on this topic is given in reference [251], which discusses the privacy risks that come from various recent personalization tendencies. The authors note that, depending on the functionalities offered, each social network handles its users' privacy differently. In reference [284] the authors consider the fact that third parties have control over an enormous amount of personal data and, considering the recent rise in reported privacy violation incidents, they acknowledge that a solution to such problems is possible thanks to verifiable computing achieved using a decentralized network of peers accompanied by a public ledger.

A relevant topic related to the proposed approach is blockchain. In reference [123] the limitations and the advantages of using private blockchain in businesses are explored. The authors propose an attribute-based encryption security system that relies on a private-over-public (PoP) blockchain approach to overcome the drawbacks of private blockchain and simultaneously to benefit fully from the positive features of the public blockchain.

Blockchain and smart contracts are applied to many domains including AI, 5G, IoT, and proof of delivery systems. The study shown in reference [231] suggests how blockchain technology could transform AI, solving many shortcomings and challenges related to AI such as data security, collective decision making, and decentralized intelligence. Investigating on the features of blockchain architecture and platforms, the authors show a wide range of open research challenges for combining AI and blockchain technologies.

The authors of reference [67] discuss the key opportunities offered by blockchain technology in 5G networks, highlighting the challenges of scalability and interoperability among different blockchain platforms and 5G stakeholders. Concurrently, IoT applications are continuously growing and, these devices are deployed at a massive scale. The authors of reference [25] propose a blockchain-based authentication system and implementation relying on Ethereum smart contracts for IoT devices.

With regard to the proof of delivery of assets, references [116, 117] explore the use of the Ethereum blockchain to create decentralized PoD systems ensuring accountability and integrity. Their solution includes the contemporary presence of multiple transporters, eliminating the need for a trusted third party. In reference [118], another blockchain-based solution to show the delivery of digital assets is shown. Their proposal is developed by smart contracts exploiting the IPFS decentralized file system to ensure that the integrity of the agreement form between the parties is well maintained.

Every day sensitive and behavioral data about users are collected by social networks [69] and present a means for companies or attackers that use them for marketing or other purposes. The authors propose a blockchain-based model to guarantee the privacy of users and to protect sensitive personal information in a distributed environment. This approach focuses on data storage applications, and the DEPLEST algorithm solves the problem of data synchronization integrated with a new consensus protocol for blockchain ledgers.

The investigation of transactions recorded inside blockchains is also important. Reference [113] studies transaction features and the relationships between them. The authors introduce statistical laws of the data related to a framework of network science and they represent the relations between different user accounts as a graph. They believe that this statistical approach can be replicated to other cryptocurrency platforms.

The most recent literature related to the privacy aspects discussed in this paper is described in the following. The authors of reference [226] highlight the importance of self-determining privacy settings by digital service users: specifically, the selected privacy requirements must be correct and describe the real privacy demands of users. The study shows a categorization of the common types of specification privacy interfaces and different user types. The experiments identify how to increase the effectiveness, efficiency, and satisfaction of privacy policy specification interfaces.

In some cases, the privacy settings of many mobile apps are difficult to understand and locate by users [68]. These difficulties expose user privacy to various risks, without proper consent. The authors of reference [68] report a systematic study of this problem and analyze the user perception of privacy settings. They discovered that 82.16% of hidden privacy settings are set to leak user privacy by default. Privacy settings suffer from the "set it and forget it" issue [188]. The decisions made about users' personal preferences could change over time and users tend to forget this type of setting. The survey analyzes the behavior of 78 Facebook users to understand the potential risks of the incorrect shifting of privacy preferences and to identify error-prone and mismatched privacy settings.

Reference [225] describes the basic interest of users in transparency and privacy control measures, individuating two important topics: transparency and self-determination. Privacy settings are generally perceived as too complex, and taking actions and making decisions about privacy are difficult for many users. Even if users are interested in protecting their privacy, they are frequently hindered from making decisions.

This review of the literature shows the potential of blockchain technology in applications that support privacy and authorization management. The importance of privacy aspects in OSNs, such as users' awareness of validating their privacy choices, is evident. To the best of our knowledge, the unfair behavior of a social network that changes the privacy settings of the user is a new problem and no solution has been proposed in the literature. The proposed approach is the first technique that detects the alteration of the privacy settings of a user performed by a social network, and this result is obtained by exploiting the power of blockchain technology.

13.7 Discussion

In this section, we discuss certain aspects of the proposed solution.

We start by managing how the goals of the proposed solution are reached. The blockchain stores all events related to the privacy settings of a user, which are when a user states the desired privacy settings and when a social network assigns the privacy settings to a user. In the event of a complaint, a misbehaving party can be detected by looking at the events on the blockchain

The events on the blockchain allow a social network to demonstrate that they have acted honestly, thus providing accountability, because the smart contract verifies that the privacy settings assigned to a user are compliant with her/his expectations.

The blockchain stores the privacy settings of a user. However, no party except the user and the social network can link these settings to the user, because the settings are associated with just a blockchain address, which is pseudo-anonymous [284]. The link is generated in the step *user verification* and is published on the Blockchain as a digest and can be known only by guessing the password of the user, which can be assumed to be a secret only known by the user and the social network.

We include certain considerations related to the cost of implementing this solution by reporting the price of the operations related to the creation of and the calls to the smart contract. The deployment of the smart contract, which is performed only once, costs 525 Micro(ETH) (in December 2020, this is approximately 0.12 $); the call to the function `setPrivacy` costs 91 Micro(ETH) (approximately 0.02 $) and the function `checkSettingGraph` costs 29 Micro(ETH) (approximately 0.0065 $). Thus, we can conclude that by paying about 2 cents, users can set their privacy settings, and the OSN, with a cost of 0.65 cents per user, can check whether the privacy settings of the users are compliant with their privacy features. We believe that these costs can be borne by the social network, even though there can be applications in which this service is paid by users. This result allows us to state that the implementation of the proposed solution is cheap and effective.

Despite these important results, the proposed solution have certain limitations. As discussed above, the drawback of using Ethereum is related to the transaction fee. We noted in Section 13.5.1 that other implementations of blockchain could be used, such as IOTA [211] and EOSIO [90], to overcome this problem.

Another limitation of the proposed solution is related to the creation of the privacy feature graph of a social network (see Definition 1). The identification of group and resource nodes requires a study of the social network and cannot be automated. Also, any change in the privacy options of a social network requires updating the graph. Fortunately, these changes are not frequent in social networks but do also require a change in the user interface of the website and app.

A final aspect to consider regards the security of the smart contract code: it is critical to ensure that the smart contract code is bug-free and is not vulnerable to almost any security threats [118, 25]. One of the most effective approaches to create secure applications is to make the implementation open source (as we did) in such a way that many programmers may access and test the code. However, we must remember that it is impossible to make a system completely secure and still usable.

13.8 Conclusion

The problem of guaranteeing the privacy of users in the context of social networks is receiving increasing attention from the research community. The case of Cambridge Analytica and the issue of the GDPR Regulation have highlighted the need to increase research to discover possible privacy issues and suitable solutions.

This chapter provides knowledge to this field of study by highlighting possible misbehavior of a social network that can result in a privacy violation. The privacy settings of a user are stored by the social network, which acts as a privileged party and could modify the user's choices to spread his/her data at any time without a user being able to prove this violation. These aspects have become critical challenges with the issuance of the GDPR Regulation, and the proposed approach aims to provide OSNs with a tool to demonstrate their honesty. The proposed approach combines the use of blockchain technology with a highly adaptable model to define the privacy settings of users in social networks. The proposed solution has been implemented to show its effectiveness and cheapness: the Ethereum smart contract implementing the required functionality, and the decentralized web application that serves as a user interface to interact with the smart contract.

The change required for a social network is minimal. They should show users the possibility to declare the blockchain address used to manage privacy, which is associated with the decentralized application. Despite this negligible change, social networks would markedly benefit from using the proposed solution.

14

Exploiting Social Networks for Data Minimization according to the GDPR

In many application domains, there is a need to ensure that users satisfy some requirements to use a service: for example, there is a minimum age to buy alcoholic beverages or to watch some videos on YouTube. In these situations, organizations typically collect more personal information than the necessary to provide a better service. The consequence is a personal data leakage that violates the data minimization principle stated by the General Data Protection Regulation 2016/679. The proposed solution is a new approach for allowing individuals to maintain control over the disclosure of their data, deciding which information to disclose and for how long. Our approach is based on the use of social networks, and an implementation on Facebook is presented to show that the proposed solution is effective, cheap, friendly, and simple to adopt.

14.1 Introduction

Nowadays, the number of accessible online services is growing considerably and users have become the major players in the process of information exchange among parties. Often, when accessing an online service, users must perform authentication to prove their real identities. However, in some cases, the grant of a service could be based on the disclosure of only certain subject's characteristics [122]. Consider an online user who wants to access a media content reserved for subjects in possession of specific attributes, such as to be of age. Commonly, the user must fill in fields that contain also unnecessary information, such as name, surname, and nationality, and this results in a leakage of personal data. Moreover, users often are not sufficiently aware of the treatment of their data and ignore their privacy rights [158, 171].

With reference to the above example, the problem we address is finding a solution that guarantees two properties: 1) only adult people should access this content (access control) and 2) the service provider should know only that the accessing user is adult and not any further information (minimization). To remark the importance of this problem, we observe that one of the ten principles of *self-sovereign identity* [72] stands for the minimization of disclosed information, that is, the disclosure should involve the minimum and necessary amount of data. Recently, this principle has been claimed with the issuance of the General Data Protection Regulation (GDPR) [104], which makes data minimization a relevant goal in accessing online services. Most of the proposed solutions to this problem are based on blockchain technology, whose main advantage is that saved information is distributed and cannot get lost. However, if private information is lost, data on blockchain cannot be removed, modified, or hidden [262]. As observed in [257], blockchain technology is a good foundation but it is not a necessity: this result suggests us to think of a different approach.

We propose a new solution based on the use of a social network in charge of providing users with the means of issuing and verifying claims and credentials. Responding to the logic of selective disclosure of attributes, a user knows and controls the information shared with the social network and with other actors, such as attribute providers or service providers. The primitive operations on which our solution is based are the exclusive-or function, a hash function, and a pseudo-random number generator: using these simple functions and with the support of an attribute provider, a user can generate a credential. A service provider can verify a credential with the help of the social network, which works as a secure and transparent repository of hidden information. Moreover, users are identified and authenticated through an eIDAS authentication [86]. We instantiated the general solution to a real-life scenario using Facebook as a social network and described the detailed data workflow to show the effectiveness of our proposal.

Our solution offers several advantages with respect to the state of the art. The first is related to user-friendliness [138], as people are favorably disposed to work in the environment of social networks, which they well know. The second advantage concerns costs, as social networks can be used typically for free. The third advantage regards scalability and availability: social networks can manage a very high number of users, and redundancy is implemented to ensure service availability. Consequently, the probability of service interruption is very low. A further advantage is about compliance with GDPR. Indeed, art. 83 of GDPR states that the supervisory authority shall impose administrative fines in respect of infringements of privacy rights and fines should be effective, proportionate, and dissuasive. The use of a technique that reduce saved data in relation to the purposes for which they are processed can provide a company with a tool to avoid administrative fines. A final observation is that the presented technique applies to various contexts for protecting user's privacy in accessing online services, like e-health applications [163, 267] or wireless sensor networks [265, 266].

U	User
SP	Service Provider
A	Attribute
AP	Attribute Provider
IP	Identity Provider
SN	Social Network
C^X	Credential issued by X
P	Pseudo-random number generator
H	Cryptographic hash function
r	Random

Table 14.1: Notation.

14.2 Proposal description

In this section, we present the approach proposed to allow users to keep control over the personal data used for accessing services on the Web. We start by introducing the scenario and the notation.

14.2.1 Preliminaries

The scenario we consider is composed of the following typologies of entities:

- let U be the user, an individual who should own and control the personal data used to access online services;
- let SP be the service provider, a party creating and offering end-user services;
- let A be an attribute regarding a quality, a characteristic, or a competence ascribed to U;
- let AP be the Attribute Provider, an organization responsible for establishing and maintaining attributes of individuals, and issuing attribute credentials;
- let IP be the identity provider, an entity in charge of creating and managing digital identities;
- let SN be a social network.

We introduce the notation used in this solution (Table 14.1):

- given a social network SN, we denote the following functions of SN:
  - $Send(M, U)$, which denotes the sending of the private message M to the user U;
  - $Post(T, h)$, which denotes the posting of the text T indexed by the hashtag h;
  - $Search(h)$, which returns the texts posted with hashtag h.

- we denote by C^X a credential generated by an entity X;
- let P be a pseudo-random number generator;
- let H be a cryptographic hash function;
- r denotes a random bit string.

14.2.2 Proposal definition

The basic idea underlying our solution is that after the identity provider authenticates the user, the user's credential needed to access a service is composed of two parts: the attribute provider publishes the former, the user shares the latter via the social network. The two credential parts are built in such a way that each single component does not disclose any useful information (it appears like a random bit string), but it is possible to reconstruct the credential only by knowing both the parts. The attribute provider and the user can stop sharing the handled part, and this results in the revocation of the credential. Figure 14.1 depicts a conceptual overview of our approach.

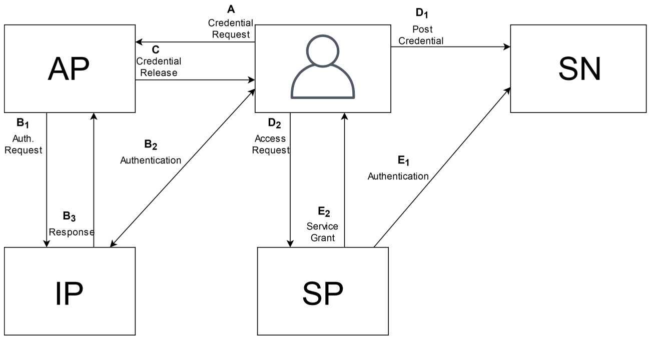


Fig. 14.1: Conceptual overview of our solution.

Our proposal is built on a social network offering the three functions described above (i.e., $Send$, $Post$, and $Search(h)$). Once the social network SN is chosen, its reference is communicated to all users and attribute providers. We expect that both users and attribute providers have a profile in SN. Thus, any new user or attribute provider has to register a profile in SN. As this profile is created specifically to access services by revealing minimal personal data, the user does not register (true) personal data and does not give any link to his/her real identity (this is done by setting the highest privacy degree). The connection to the profile of attribute providers in SN is publicly available.

There is a preliminary step used to initialize the environment, where two functions are defined:

- $H(x)$ is a one-way hash function that receives an input x and returns a bit string with a fixed length. In words, we require that given a value y, it should be difficult to find any message x such that $h(x) = y$. A cryptographic hash function is an excellent candidate to implement H (for example, SHA-1, SHA-256, RIPEMD-160).
- $P(x)$ is a function that receives an input x and, depending on x, generates a sequence of bits having the same properties as a series of random numbers. A Pseudo-Random-Number generator can implement this function.

These two functions are publicly available and known by all actors.

Now, we describe the procedures carried out by the different actors to implement our approach.

A. *Credential request.* This step is performed every time a user needs to certify the possession of some attribute. Let A be such an attribute and AP be an attribute provider in charge of certifying A. For the sake of presentation, here we assume that the credential contains only one attribute. However, in the case of more attributes, this procedure is repeated for each attribute.

   The credential request is sent by a private message from the social profile of the user to that of the attribute provider (Step A of Figure 14.1). Specifically, the user generates a random r_1^u, calculates $r_2^u = H(r_1^u)$ where H is the cryptographic hash function, and creates the request containing:

   a) the attribute of the user to be certified (i.e., A);
   b) the value r_2^u.

   This request is sent to AP by calling the social network function $Send(\langle A, r_2^u \rangle, AP)$.

B. *User identification.* After receiving the request, AP needs to identify the user: for this purpose, an eIDAS authentication is performed by using the eID scheme requested by the user (Step B_1). Observe that this authentication can be done by any eID scheme compliant with eIDAS: for the sake of completeness, we provide detail of this authentication process, which is schematized in Figure 14.2.

   First, the user sends the credential request to AP by using a browser named User Agent (UA) in Figure 14.2 (Step 1). Then, AP replies with an authentication request to be forwarded to the eIDAS identity provider declared by the user (Step 2). Now, the identity provider performs the authentication of the user by a challenge-response procedure (Steps 3 and 4), in which (typically) the user is requested to authenticate by login and password or similar mechanisms (for example, by sending an SMS to the phone number declared by the registering user and by asking to send back the text inside the SMS). In case of successful user authentication, the identity provider generates the response with the result of user

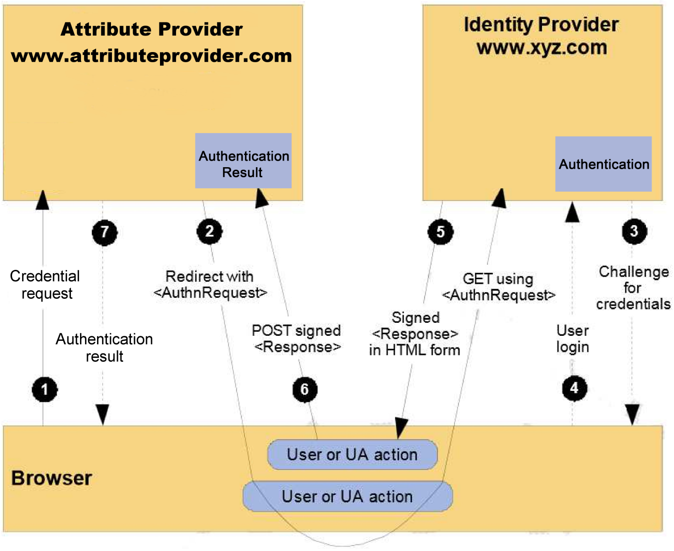


Fig. 14.2: Authentication process.

authentication (Step 5), which is forwarded to AP (Step 6). Finally, AP notifies the result of the authentication procedure (Step 7), and only in case of success, the next phase is carried on. The whole authentication is summed up by the steps B_1, B_2, and B_3 of Figure 14.1.

C. *Credential generation (*AP *side).* The attribute credential is generated by the cooperation between AP and the user.

After identifying the user, AP verifies that the requested attribute A is own by the user. If this is the case, AP prepares a credential C containing the certified characteristics, the temporal validity of the credential, and the reference to the user profile in SN (Step C).

Then, AP generates a random r^{AP} and calculates $C^{AP} = C \oplus P(r_2^u) \oplus P(r^{AP})$, where $\oplus$ is the exclusive-or (XOR) function and P is the function defined in the preliminary step. In other words, C^{AP} can be seen as C encrypted by two keys: r_2^u is chosen by the user, r^{AP} is selected by the attribute provider.

Finally, AP posts C^{AP} on its profile in SN, using $H(r_2^u)$ as an index of this post, thus calling the social network function $Post(C^{AP}, H(r_2^u))$.

D. *Credential use.* Suppose now that the user needs to access a service requiring the possession of the attribute certified by C and that this service is supplied by the service provider SP.

First, the user generates $C^U = P(r^{AP}) \oplus P(r_1^u)$ and posts C^U on his/her profile in SN (Step D_1 of Figure 14.1), using $H(r_1^u)$ as an index of this post (i.e., by exploiting the function $Post(C^U, H(r_1^u))$). In words, C^U can be seen as the key of the attribute provider used to encrypt C (i.e., $P(r^{AP})$) encrypted by $P(r_1^u)$.

Then, the user opens the web page of the site of SP and logins to the service by the social network account to prove to be the owner of this profile (Step D_2). Then, the user discloses r_1^u and r^{AP} to SP, which is enough to show the possession of the requested attribute, as discussed below.

E. *Credential verification.* SP needs to verify the correctness and validity of the proof presented by the user (Step E_1). Thus, it calculates $r_2^u = H(r_1^u)$ and searches for $\overline{C}^U$ and $\overline{C}^{AP}$ (they are the posts indexed by $H(r_1^u)$ and $H(r_2^u)$, respectively). This is done by calling the social network functions $\overline{C}^U = Search(H(r_1^u))$ and $\overline{C}^{AP} = Search(H(r_2^u))$.

   If $\overline{C}^U$ is not found in the profile of the user or $\overline{C}^{AP}$ is not found in the profile of an attribute provider, then the credential verification fails, and the service is not granted. Otherwise, SP computes $\overline{C}^U \oplus P(r_1^u) \oplus P(r^{AP})$ and verifies that this is equal to zero. In this case, SP calculates $\overline{C} = \overline{C}^U \oplus \overline{C}^{AP} \oplus P(H(r_1^u)) \oplus P(r_1^u)$ and tests that the result is a valid credential. Specifically, SP extracts from the credential the profile of the user and checks that it is equal to the profile of the user who authenticates. Then, SP extracts from the credential the certified attribute and checks that it is sufficient to access the service (Step E_2). In the positive case, the service is granted to the user. If any of the previous checks fails, the procedure is stopped, and the service access is denied.

F. *Revocation (AP side).* An important aspect concerns the credential revocation. AP should revoke a credential when a user loses the possession of an attribute, to make unusable a credential (for example, in case of withdrawal of the driving license of a user). Another reason for credential revocation is when the secrecy of the user's password to access the social network profile is compromised.

   In this case, AP can make unusable a user's credential C by merely removing the post C^{AP} from its social network profile. Clearly, if the user tries to use C, the service provider cannot find the deleted post in the profile of the attribute provider so that the verification procedure fails.

G. *Revocation (user side).* Also the user can wish to remove personal information after it has been used to access some service. Consider a patient with a disease credential, who wants to hide this information after the use. The patient can reach this goal by merely removing C^U from the profile in such a way that the credential C cannot be recovered. Clearly, excluding the case in which C^U has been copied while it is published in the profile (in this case, no solution to this problem exists), C cannot be recovered without the knowledge of C^U.

We will discuss the advantages and improvements of our solution with respect to the state of the art in Section 14.6.

14.2.3 Social network choice

As seen in the previous section, the proposed approach exploits some features provided by social networks, and in this sense, our solution is orthogonal to the underlying social network. Consequently, the aspect regarding how to implement our approach can be adequately addressed once the social network SN is given.

We analyzed the most known social networks concerning the requested features, which are sending a private message M to a user U (i.e., $Send(M,U)$); posting of the text T indexed by the hashtag h (i.e., $Post(T,h)$) and searching an indexed text with hashtag h (i.e., $Search(h)$).

The results of this analysis are reported in Table 14.2, in which the social networks are classified into three categories. The first category lists some social networks that are fully compliant with our proposal because they provide the requested features. We observed that most of the social networks offer such features. For instance, Instagram users (1) can send direct messages to other users, (2) can post textual information, named *post*, and they must be authenticated to post anything; (3) can use the hashtag symbol # to categorize their post by keywords, (4) can search for posts indexed by a given hashtag typically to find a conversation about a particular topic. Observe that on Instagram, posts have to be composed of at least an image or a video.

The second category (with Twitter, Weibo, Snapchat) is composed of social networks that have some limitations in the post function. Specifically, they have a limit in the length of the text to post (e.g., this limitation is 280 characters in Twitter), and, thus, in the credential size. As a consequence, they could be used only with a limited number of certified attributes or by applying compression in the represented text.

Finally, the last category shows social networks that do not support our solution.

14.3 Running example

In this section, we show the application of our proposal to a real-life case in which the students of a university U play the role of users, an e-learning site that offers for free some lectures only to such students is the service provider, whereas the attribute provider is done by the university U. Our solution aims to guarantee the following two requirements: 1) the service provider should know whether a given user is a student of U (access-control requirement) and 2) the service provider does not have to know the user identity (privacy requirement).

For the implementation of our solution, among all the social networks that could be used according to the analysis reported in Section 14.2.3, we selected Facebook because (1) it is one of the most famous one offering the required features and (2)

	$Send(M, U)$	$Post(T, h)$	$Search(h)$
Instagram	Yes	Yes	Yes
Facebook	Yes	Yes	Yes
Linkedin	Yes	Yes	Yes
VKontakte	Yes	Yes	Yes
Twitter	Yes	Yes (limited)	Yes
Weibo	Yes	Yes (limited)	Yes
Snapchat	Yes	Yes (limited)	Yes
YouTube	No	Yes	Yes
Whatsapp	Yes	No	No
WeChat	Yes	No	No
Skype	Yes	No	No
Viber	Yes	No	No
WeChat	Yes	No	No

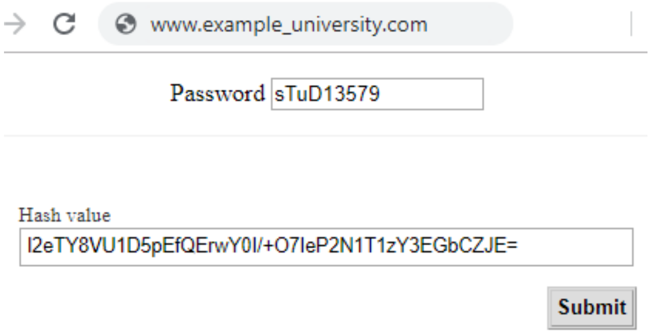
Table 14.2: Social networks versus requested features.

it has been widely used for developing applications in the research context [166]. Indeed, Facebook allows the exchange of private messages between two users with an encrypted connection, the publication of stories and posts that can be indexed by hashtags, and the search for stories by a given hashtag.

The main idea underlying our approach can be summarized as follows: first, the university authenticates the user and publishes on its Facebook profile the first part of the user's credential to access the e-learning site. When the user needs to access the site, she/he publishes on Facebook the second part of the credential. Clearly, each part of the credential appears randomly generated and does not disclose any private information of the user: however, the combination of the two parts allows the e-learning site to verify whether the user is authorized to access the service. The university can revoke the user's access by removing the published part of the credential. Moreover, the user can remove her/his Facebook post after accessing the service to delete the credential.

We describe how the steps of our solution are implemented.

1. *Setup*. For the sake of presentation and without loss of generality, we implement the two functions H and P by exploiting only the SHA-256 function [196], a well-known cryptographic hash function designed by the United States National Security Agency (NSA), widely used for its robustness against attacks. Thus, we define:
   - $H(x)$ =SHA-256(x);



Password sTuD13579

Hash value
I2eTY8VU1D5pEfQErwY0I/+O7IeP2N1T1zY3EGbCZJE=

Fig. 14.3: Web page used by students to send r_2^u.

- $P(x) = a_1, \ldots, a_n$, where a_1 = SHA-256(x) and a_j = SHA-256(a_{j-1}), with $2 \leq j \leq n$.

These definitions are made available to all the students, the university, and the e-learning site.

2. *OSN Registration*. In this step, the university creates and makes public its Facebook profile, which we assume to be `example_university`. Every student registers a Facebook profile, using information (e.g., screen-name, name, photos) unlinkable to the real identity. Moreover, the student sets the highest privacy degree, in such a way that no information is disclosed.
3. *Credential request.* The student needs to prove to be a member of the university. Consequently, the credential request is sent to the university. When the student submits the request for a credential, the university starts a Facebook login procedure. The student proves the possession of the Facebook profile, say `example_student`, using a single sign-on operation. After successful authentication, the student can type in a password (sTuD13579 in our example), which will be used as the random r_1^u. Then, a javascript calculates r_2^u =SHA-256(r_1^u), which is `I2eTY8VU1D5pEfQErwY0I/+07IeP2 N1T1zY3EGbCZJE=` by the base64 representation [140], used to convert a bit sequence into a text (to be included in a post). This value is sent to the university server: observe that only r_2^u, which is generated by a javascript, is sent to the server, not the password, which is not included in the HTML form. This step is depicted in Figure 14.3.
4. *User identification.* In this step, the user is requested to prove to be a student. First, the user performs the eID-based identification. Then, he/she uses the login and password of the university site to prove to be a student of that university.
5. *Credential generation* (AP *side*). In case of successful authentication, the university prepares a credential C containing the certified attribute, the temporal validity of the credential, and the reference to the user profile in SN. There exist several

ways to represent a credential, for example, by the standard SAML [125] (as done in eIDAS-compliant eIDs).

In Figure 14.4, we show an example of a credential represented by JSON, which is very easy to understand. The credential describes the type of attribute proved (to be a student), the name of the university, the expiration date, and the screen name of the user.

```
{
    "attribute": "student",
    "university": "example_university",
    "expiration": "30/08/2020",
    "profile": "example_student"
}
```

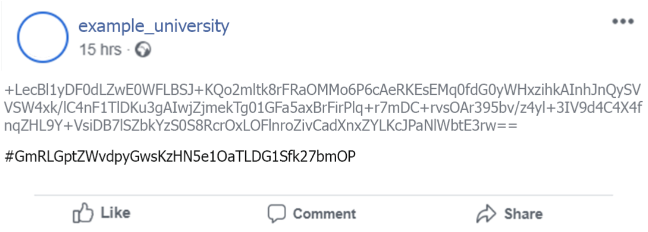
Fig. 14.4: Example of credential.

The university generates a random password r^{AP}, which is uNiV2468 in our example, and calculates $C^{AP} = C \oplus P(r_2^u) \oplus P(r^{AP})$. C^{AP} is posted in the Facebook profile of the university and indexed by SHA-256(r_2^u) = GmRLGptZWvdpyGwsKzHN5e10aTLDG1Sfk27bmOPMAdI. This post is reported in Figure 14.5, and the password digest is hashtagged (see the symbol # at the beginning of the second line): this post will be returned when searching for this hashtag.

6. *Credential use.* When the student needs to access the e-learning site and to prove to be a member of the university, then the student calculates and posts in the Facebook profile $C^U = P(r^{AP}) \oplus P(r_1^u)$ using $H(r_1^u)$ as a hashtag (see Figure 14.6). Then, the student goes to the e-learning site and logins by Facebook (as done in Figure 14.3). Then, the user fills in the password used earlier (i.e., sTuD13579) to give the service provider the possibility to restore the credential.
7. *Credential verification.* The service provider calculates r_2^u = SHA-256(r_1^u) and searches for the posts C_1 and C_2 indexed by I2eTY8VU1D5pEfQErwYOI/+07IeP2N1T1zY3EGbCZJE and GmRLGptZWv dpyGwsKzHN5e10aTLDG1Sfk27bmOPMAdI, respectively.

   Now, the service provider calculates $C_1 \oplus C_2 \oplus P(r_2^u) \oplus P(r_1^u)$, thus obtaining the initial credential C (reported in Figure 14.4). Since this credential satisfies all the checks, the student receives the grant to access the e-learning course for free.
8. *Credential revocation.* When the student wants to hide the possession of this attribute, it is sufficient to remove the generated post so that the credential C cannot be recovered.

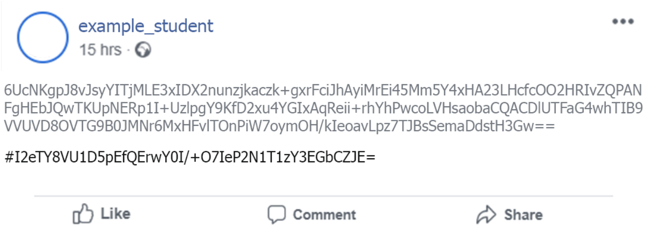
   On the other hand, also the university can revoke the attribute possession, for example, in case the student leaves the university before the credential expira-



+LecBl1yDF0dLZwE0WFLBSJ+KQo2mltk8rFRaOMMo6P6cAeRKEsEMq0fdG0yWHxzihkAInhJnQySV
VSW4xk/lC4nF1TlDKu3gAIwjZjmekTg01GFa5axBrFirPlq+r7mDC+rvsOAr395bv/z4yl+3IV9d4C4X4f
nqZHL9Y+VsiDB7lSZbkYzS0S8RcrOxLOFlnroZivCadXnxZYLKcJPaNlWbtE3rw==

#GmRLGptZWvdpyGwsKzHN5e1OaTLDG1Sfk27bmOP

Fig. 14.5: Post publishing (university side).



6UcNKgpJ8vJsyYITjMLE3xIDX2nunzjkaczk+gxrFciJhAyiMrEi45Mm5Y4xHA23LHcfcOO2HRIvZQPAN
FgHEbJQwTKUpNERp1I+UzlpgY9KfD2xu4YGIxAqReii+rhYhPwcoLVHsaobaCQACDlUTFaG4whTIB9
VVUVD8OVTG9B0JMNr6MxHFvlTOnPiW7oymOH/kIeoavLpz7TJBsSemaDdstH3Gw==

#I2eTY8VU1D5pEfQErwY0I/+O7IeP2N1T1zY3EGbCZJE=

Fig. 14.6: Post publishing (student side).

tion time. Again, removing the post associated with this credential is sufficient to make the verification procedure fail.

14.4 Quantitative analysis

In this section, we measure some parameters related to our proposal with the aim of showing the effectiveness of our solution.

The first parameter is efficiency. We quantify the time costs of the operations of our solution, which are: the number of requests over the network, the number of hash functions executed, and the number of pseudo-random numbers generated.

- Step A. In this phase, the user generates a random and calculates its hash. Then, she/he generates a connection request to the attribute provider.
- Step B. The attribute provider performs an authentication request and receives a response. Observe that we do not consider the cost of the user's authentication because it depends on the user's speed in the authentication.
- Step C. The attribute provider creates the credential by the generation of three pseudo-random numbers. Then, it calculates one hash function and sends a request to the social network.
- Step D. The user generates the credential by calculating two pseudo-random numbers and applying the hash function to one of them. Then, the user sends a request to the social network and another request to the service provider.
- Step E. The attribute provider calculates one hash function and performs two request connections to the social network. The service provider calculates four

pseudo-random numbers and applies one hash function. To verify such that credential, the service provider performs a connection request to the social network. If the verification succeeds, the service provider connects to the user.

- Step F. The attribute provider sends a request to the social network.
- Step G. The user sends a request to the social network.

Observe that Steps A, B, and C refer to operations that are done by users preliminarily (i.e., before they access a service), whereas Steps D and E are performed by a user and a service provider to grant a service. In contrast, Steps F and G refer to an infrequent operation (revocation). Thus, we focus our attention on Steps D and E, which are the most important operations related to service access; moreover, their time cost is the highest.

We ran some experiments to measure the time required by the different operations and used a 64-bit Windows 10 machine with Intel Core™ i7-7700K CPU 4.20GHz and 16GB RAM. We measured that the time of 1 million calculations of SHA-256 hashes is about 0.6 seconds and is 10 milliseconds for a single hash. Again, we measured that the time to generate 500 random numbers is about 1 millisecond. Concerning the network request time taken for operations *Send*, *Post*, and *Search*, we measured an average time of 500 milliseconds (the amount of data sent/received is very limited). As we found that the average time of hash computation and random generation is negligible with respect to the time of network request (they differ by 3 orders of magnitude), we omit in our analysis the time of the calculation of hashes and random numbers.

The second parameter we consider is related to user-friendliness, which refers to the ability of a service to be used easily by the users and is commonly adopted to evaluate a range of end-user computing technologies. It is well-known that if a social network website does not provide an efficient and user-friendly interface, then its users may be disappointed and switch to another social network [88].

The actions performed by the user are easy: the user visits the website of the service provider and is requested to authenticate by a social-network-based login procedure. After the authentication, the user inserts the password in the website and creates a post containing a simple text with a hashtag. In the occurrence, the user can delete such a post. Which and how many operations a user performs are reported in Table 14.3. It is evident that these operations are very friendly as social networks are daily used by billions of people. Thus, we conclude that the use of a social network positively affects perceptions of people, who are favorably disposed to work in an environment that they well know.

The third parameter considered is scalability, which refers to the ability of a system to maintain its functionalities despite the scaling of users' requests. This param-

Site browsing	Form compilation	Post creation	Hashtag search	Post removing
2	4	1	1	1

Table 14.3: Number of user's operations.

Service Provider	Identi-fication	SN site	Hashtag search	Post removing
1	5	2	4	2

Table 14.4: Number of requests for a full service access.

eter can be calculated by different types of performance measure attributes, such as the number of processed requests and network usage. In Table 14.4, we measure the number of requests for each service access: the number of requests related to the post removing refers to both the user and service provider side. We observe that the measured values are constant with respect to the number of users, which is an indicator of good scalability. Moreover, also the size of exchanged messages is limited and does not depend on the number of users (see Section 14.3). Consequently, we can state that the proposed solution offers high scalability and can support a very high number of users: the upper bound of the number of users is given by the number of users supported by the adopted social network. Consider that several strategies are used to ensure service availability [184] so that the probability of service interruption is negligible. Scalability is also favored because our solution does not require the use of heavy cryptography (hash functions are more efficient than encryption). This point is strongly related to the response time too.

Finally, we consider the cost of the solution related to the implementation and set-up costs. The price of our solution is limited because its architecture is mainly based on a social network. Social networks monetize their services with the precious information the users voluntarily reveal in their profiles, in their relationships, in their behavior [195], so that social networks can offer their services to registered users for free (differently from Blockchains). By relying on already existing social network systems, our costs are only those for set-up. Furthermore, it is worth noting that our solution, being compliant with the GDPR principles, can save up to 20M€ or up to 4% of the total worldwide annual turnover of the preceding financial year, as stated in [104]. On the other hand, solutions based on a blockchain have an increased cost of blockchain development and maintenance that are estimated between 40$-80K$.

14.5 Security analysis

In this section, we discuss how our solution reaches the expected goals and how possible attacks are contrasted.

We start by defining the *adversary model*. In our analysis, we assume that identity providers, attribute providers, and social networks are trusted parties, and they run the protocol correctly. Thus, the adversary can be a user, a service provider, or an entity external to the system. In our attack model, the adversary cannot compromise the behavior of the identity provider, the attribute provider, and the social network, and cannot modify the posts published by other users on their social profile. The adversary cannot break the cryptographic primitives (e.g., the adversary cannot generate a message that yields a given hash value) and cannot guess the user's password, secret information, or randomly generated values. Finally, we assume that users and service providers do not collude with each other.

The attacker aims to violate one of the security properties guaranteed by our solution, which are access-control and privacy. We describe how these properties are guaranteed.

In the verification phase, the service provider calculates $C^U \oplus C^{AP} \oplus P(H(r_1^u)) \oplus P(r_1^u)$, and the attribute provider generates C^{AP} after verifying the user owns the attributes to be certified. By construction, we have that $\overline{C}^U = P(r^{AP}) \oplus P(r_1^u)$ and $C^{AP} = C \oplus P(H(r_1^u)) \oplus P(r^{AP})$ so that $C^U \oplus C^{AP} \oplus P(H(r_1^u)) \oplus P(r_1^u) = C$. If C is a valid credential, then the access can be granted because C is obtained starting from C^{AP}, which is generated by the attribute provider. It is worth noting that the XOR function is well-known not only for its efficiency but also for some weaknesses documented in the literature and already exploited in some application contexts, like WEP [153]. The weakness is related to the possibility for a user to generate a new credential C_*^U such that $C_*^U \oplus C_2 = C_*$, where C_* is a fake credential (see Step 7 of Section 14.3). Specifically, to break the access control property, the adversary should be able to create a valid C. The most favorable case is that the user is the attacker so that he/she can generate C^U. In this case, the attacker has to use a suitable C^{AP}, which can be obtained in three ways: 1) by creating a new one, 2) by modifying an existing one, or 3) by using an existing one. Our threat model prevents cases 1 and 2 because the adversary cannot modify information uploaded by the attribute provider so that only the third possibility is available. In this case, given $C_1 = C_1^U \oplus C_1^{AP} \oplus P(H(x_1)) \oplus P(x_1)$ a valid credential, the attacker has to create a new credential $C_2 = X \oplus C_1^{AP} \oplus P(H(x_1)) \oplus P(x_1)$, where C^X is information handled by the attacker. We have that C^X should be equal to $C_2 \oplus C_1^{AP} = C_2 \oplus (C_1 \oplus P(H(r_1)) \oplus P(r^{AP}))$. Now, the attacker has to find x_1 and x_2 such that $P(x_1) \oplus P(x_2) = C^X$ (which is the information the user has to publish). Thus, $P(x_1) \oplus P(x_2) = C_2 \oplus C_1 \oplus P(H(x_1)) \oplus P(x_2)$, then $P(x_1) = C_2 \oplus C_1 \oplus P(H(x_1))$, and $P(H(x_1)) = P(x_1) \oplus C_2 \oplus C_1 = P(x_1) \oplus K$, where K is a constant. Now, the attacker has to find two seeds x_1 and x_2 such that the generated random number sequences are equal up to a constant K: this violates the assumption that P is a random number generator function. Moreover, x_2 should also be the hash value of x_1, and this violates the one-wayness property of the hash function H.

An attack we contrast is the replay attack, which is carried out by taking a credential used by a user to access a service and trying to use it. This attack is contrasted because the credential contains the screen-name of the social network profile so that another account cannot use it.

Concerning privacy, we observe that the credential does not contain any identifying information so that the service provider does not know the identity of the user accessing the service. As described in Section 14.2, the plain-text credential C is encrypted by the one-time pad function, an encryption technique that cannot be cracked, provided that the key used in the XOR function is random. In our case, the key is obtained by the function P, which has been defined to generate pseudo-random numbers. Thus, it is unlikely that an adversary can recover C by knowing C^U or C^{AP} if P is well-implemented (our implementation satisfies this requirement). Another observation is about the use of the randoms r_1 and r_2 in Step A (credential request) of Section 14.2. They are used so that the attribute provider is not aware of when the user exploits the credential (i.e., when the user publishes C^U) or terminates the use of this credential. Indeed, C^U is indexed by $H(r_1)$, and the attribute provider is not aware of r_1 so that it cannot know which hashtag has to be searched. This mechanism reduces the information about the user behavior known by the attribute provider and by the other actors in general.

Finally, we observe that also unlinkability can be achieved. Unlinkability means that the service provider is not able to guess that two different requests come from the same user. To achieve unlinkability, the user must require a new credential for each service request.

14.6 Related Work

In this section, we review the state of the art starting from Self-Sovereign Identity. A Self-Sovereign Identity must allow ordinary users to make claims, which could include personally identifying information or facts about personal capability or group membership. Ten principles of Self-Sovereign Identity are proposed in [72] A survey of solutions for Self-Sovereign Identity with and without the use of blockchain technology is presented in [257].

Blockchain technology has the potential to support emerging solutions on the data ownership and governance models. The study done in [161] categorizes these solutions into a taxonomy based on architecture, governance models, and other features.

EverID [95] is a user-centric solution which includes a scalable payment solution (EverChain) with a multi-currency wallet (EverWallet). Through the use of EverID, individuals control their database of identity elements, including their biometrics. This architecture is distributed and lays on an Identity Network (a private Ethereum Blockchain) and a Decentralized App (DApp), a software working on the decentralized network.

A decentralized identity based on hierarchically deterministic keys controlled and generated by the users is proposed in [139]. The architecture is based on a public blockchain in charge of providing a trusted storage layer and a mapping between each decentralized identifier (DID) and the corresponding DID document object (DDO). The lifeID Platform [170] is an identity service based on a permissionless blockchain able to run smart contracts. Users can create their identity by using a biometric-capable smartphone and the lifeID app.

The uPort technology is built on the Ethereum Blockchain [256] uses smart contracts, a mobile application, servers, data, and attestations to be proved. Sovrin is a global public utility for Self-Sovereign Identity and verifiable claims, and implements the concept "identity for all" [219]. It is built on a public blockchain and claims and credentials are represented by Sovrin's tokens which can be used for education, healthcare, and insurance.

A limitation of self-sovereign identity systems is the lack of qualified eID data [216]. For example, in the context of e-Government, employees may need to satisfy additional requirements about their identities that are mapped through qualified electronic identities (eIDs). The main advantage of blockchain-based solutions is that saved information is distributed and cannot get lost. However, if a private key is stolen, it is impossible to hide data. Differently, in our solution, even if the randoms or passwords used to generate the credential are made public, to hide the credential it is enough to remove the corresponding posts. As observed by [257], blockchain technology is an excellent foundation to face the problem, but it is not a necessity. Indeed, there exist other solutions that do not rely on blockchain.

An attribute-based credential [115, 70] is a cryptographic container of a few attributes and is signed by an authoritative party. In the IRMA identity platform [132], personal attributes are stored in the user's devices IRMA app: in the case of stolen or lost devices, users have to create all attributes one more time.

The Private Data System (PDS) proposed in [23] enables self-sovereign storage and sharing of private data among online users. This system is composed of spread online nodes, and their role is not based on distributed consensus. ReclaimID is a decentralized service for identity management [234]. The attributes used to access an online service are stored over a name system and under name-spaces of users. Furthermore, attributes are encrypted by the attribute-based encryption (ABE) method.

After describing the related literature, we compare our solution with the state of the art by considering the five aspects presented in Section 14.4, which are:

1. If the solution has been implemented (*Implementation*).
2. How long the service access takes for the user (*Efficiency*).
3. The degree to which the solution can be used by consumers with satisfaction (*Usability*).
4. How much the solution is scalable (*Scalability*).
5. The price of the solution (*Cost*).

Implementation is a boolean measure, whereas we use the values *low*, *medium*, and *high* for measuring the other parameters. Table 14.5 summarizes the results obtained by comparing the different techniques.

Table 14.5: Comparison with existing solutions.

	[23]	[234]	[132]	[256]	[219]	[95]	This solution
Impl.	No	Yes	Yes	Yes	Yes	Yes	Yes
Eff.	n.a.	High	High	Low	High	Low	High
Usab.	Medium	High	High	Low	Low	Low	High
Scal.	Medium	Medium	Low	Low	Medium	Low	High
Cost	n.a.	Low	Low	High	High	High	Low

Concerning [23], we note that although the authors provide a workflow of the different phases of the solution, a real use-case implementation is not presented (thus, efficiency cannot be measured). The use of executable choreographies and the division of roles among the nodes result in medium scalability. In the second examined solution [234], the attributes are encrypted, and the system is developed compliant with the OpenID standard, but it is suitable for small or medium applications. It has the same number of operations as our approach and is distributed. The third approach we consider is presented in [132]: this system has low scalability because it is centralized: the Privacy by Design Foundation has published the schema on GitHub for various issuers [131] and these schemas are used by the IRMA app using an auto-update mechanism [194]. Nevertheless, the idea of exploiting the user's device to manage credentials makes this system user-friendly. The solutions [95], [219], and [256] provide users with a wallet to generate transactions over a blockchain network. Since managing blockchain wallets is not easy, their usability is low. The transaction verification relies on the consensus mechanism of the network, which introduces delays at the expense of scalability. Furthermore, these solutions are based on a blockchain, which increases the solution price because each transaction has a fee. The different efficiency of the three solutions is due to the blockchain technology adopted: [95] and [256] exploit the Ethereum blockchain, which is based on the Proof of Work as a consensus algorithm. Instead, [219] implements the Plenum Consensus Protocol, which is considered faster than the Proof of Work. Concerning our approach, according to the results presented in Section 14.4, it presents the best behavior compared to all the considered metrics.

14.7 Conclusion

In this chapter, we proposed a new solution for the management of personal data that is based on the use of a social network in charge of providing users with the means of issuing and verifying claims and credentials. Our solution allows a user to control the information shared with other actors by using very efficient operations, which are the exclusive-or function, the hash function, and the pseudo-random number generator. A service provider can verify the attributes of a user by the support of the social network as a secure and transparent repository of the selected and hidden information. The effectiveness of our proposal has been shown in a real-life scenario using Facebook as a social network.

We showed that our solution offers several advantages, such as user-friendliness, cheapness, scalability, and availability. To remark the practical significance of the proposed approach, we highlight the importance of our solution concerning the European Union General Data Protection Regulation (GDPR) [104], and in particular, with the data minimization principle. Article 5(1)(c) says that personal data shall be *adequate, relevant, and limited to what is necessary in relation to the purposes for which they are processed*.

15

Enforcing User’s Preference in OSN Advertising

Social network advertising is currently one of the most effective advertising types available to promote a product or a brand. The problem discussed in this solution concerns the possibility to ensure that advertising reaches really interested users, and also to prove this. At this aim, we propose the use of blockchain to store users’ interest and to obtain an assertion that a user is interested in a product before the advertising is shown. The proposal has been implemented by a Solidity smart contract in Ethereum and has been shown to be effective and cheap.

15.1 Introduction

Social networks represent one of the revolutionary applications of the Internet and are currently the place where hundreds of millions of people reveal the most of their information, from personal data to visited places, friends, listen music, watched TV, and, in general, preferences and interests. Often, these data are publicly available, so that anyone can elaborate and obtain useful information: for example, there have been companies that have built their business by analyzing social network data to offer support in optimizing marketing campaigns. However, for legal reasons, this can be done on aggregated data.

In the last years, social networks have looked at the possibility to monetize the information coming from the membership of their users by third-party advertisers. One of the most famous examples is Facebook Adv [12], which supported by the fact that more than two billion people use Facebook every month, is able to create and run campaigns using simple self-service tools, allowing the client to choose an objective, select an audience, decide where to run the advertising, and set a budget. Moreover, Facebook Adv offers tools to check if one version of advertising is working better than another, or if it is being delivered efficiently, and make tweaks and adjustments as needed.

In facts, social network advertising is surely the most effective solution to draw attention to a product, a service, or an event. One of the major reasons of this effectiveness is that social networks can take advantage of the users' information and target their ads appropriately. Indeed, users provide social networks with demographic information, images, interests, purchasing patterns, device usage, and other activities and this piece of information is exploited to display advertising with characteristics that match those of users. This is important also from the user-side, since users see advertisements that might actually interest them, instead of general not personalized advertising. In these cases, social network is the only owner of user's interest and this piece of information is an important asset which cannot be shared, so that it is not possible for a third party to be sure that a social network has sent advertising only to really interested users.

A recent technology is that of Distributed Ledger, whose most famous example is blockchain. It is based on the concept of pseudo-anonymity [62], in order to preserve the linkability of the transactions made by users. Moreover, there exist solutions that integrate digital identity with the various advantages in using blockchain technology for a real scenario [59].

In this solution, we face this problem by proposing a technique allowing a social network to prove to a third party that a user reached by the advertising is really interested in that. The key idea is to rely on blockchain to store users' interest and to obtain an assertion that a user is interested in a product or a category before the advertising is shown. Everyday companies invest resources to obtain consistent revenues from advertising campaigns, our approach could improve the strategy used in advertising and marketing to prevent the waste of time and money derived from wrong targeting options. Furthermore, users could disclose interests in receiving targeting advertising and in having a proof of declared interest.

Specifically, we implemented our proposal on Ethereum, a global, open-source platform for decentralized applications, which control digital value, run exactly as programmed, and are available anywhere in the world. We wrote a smart contract that is exploited by social network users to publish their interests. Since users are referred by Ethereum address, which are pseudo-anonymous alphanumeric strings, no information leakage occurs, and users' privacy is kept. On the other part, social networks use the same smart contract to have a proof that the user receiving advertising matches the targeting: this proof is publicly available on the blockchain as an event. Finally, by analyzing the implementation costs of our proposal, we conclude that it is also a cheap solution.

15.2 Proposal

To introduce our proposal, we describe a scenario motivating our research. We have a company that wants to create a social advertising campaign in order to get more people to buy its products. Let SN the social network on which the campaign will be carried on. The company chooses the audience that it wants to reach with the advertising campaign (this is called *targeting*). There are a lot of targeting options, including choosing an audience based on demographics, age, social state: however, the company wants to reach people interested in what it offers. The social network allows its users to declare their interests and likes, so that the mapping between company requests and users' interests can be easily found.

However, since users' interests are private and known only to the social network, the company wants to be sure that the targeting is correctly identified, that is the company wants to contrast the possibility that the social network sends advertising to not interested people with the only purpose of increasing the money gain.

The system we propose aims at solving this problem by exploiting the power of blockchain, a distributed database storing a continuously-growing list of blocks. Each block contains a certain number of transactions and the digest of the previous block, so that an order of transactions is established and the older the block, the harder it will become to tamper it. In our proposal, we exploit smart contract, general purpose applications that take place on the blockchain and are able to generate transactions depending on certain conditions. Such applications implement self-enforcing business logic, and a non-repudiable and authoritative record of transactions is left on the blockchain. For space limitation, we cannot discuss such aspects: the interested reader can find useful information about blockchain, smart contract, and applications in [208].

In our scenario, we consider a generic social network user U and let $I = \langle i_1, \ldots, i_n \rangle$ be an ordered set of interests, such as Arts, Music, Movies, Reading, Games.

Let SM be a smart contract deployed in the blockchain, declaring two functions `setPref` and `checkPref`, whose use will be described below. Now we show the actions carried out by the social network and its users according to our protocol.

1. *Blockchain registration*. First, both the social network and its users register a blockchain account. Each one generates a pair of private and public blockchain keys and the corresponding blockchain address (typically, it is a portion of the cryptographic hash of the public key). The information about the user's blockchain address is added to the social network profile of the user, and it can be a piece of private or public information.

2. *Declaration of interests*. When the user declares the own interests on the social network, a bit-string $B = \langle b_1, \ldots, b_n \rangle$ is generated such that $b_j = 1$ with $1 \leq j \leq n$ if and only if the user is interested in i_j. Then, the user generates a blockchain transaction to call the function `setPref`, which has B as input. This function stores in the smart contract the interests declared by the user.

   For example, consider $I = \langle \text{Arts, Music, Movies, Reading, Games} \rangle$. If U is interested in Arts and Reading, then $B = \langle 1, 0, 0, 1, 0 \rangle$.

3. *Advertise sending*. After the advert has been created, the company defines the targeting by means of the interests: let B' be the bit-string $\{b'_1, \ldots, b'_n\}$ such that $b'_j = 1$ with $1 \leq j \leq n$ if and only if the advert should be shown to users interested in i_j. Again, with reference to the example above on the interests in I, if the campaign is for users interested in Arts, then $B' = \langle 1, 0, 0, 0, 0 \rangle$.

   Now, the social network extracts the list of users interested in the advert and, to prove such an interest, the social network generates, for each user, a blockchain transaction to call the function `checkPref`, which has A and B' as inputs, where A is the blockchain address of the user. This function verifies that there is an interest of this user in the advert (by comparing the bit-strings B and B') and emits an event that logs the result of this check.

Once the advertising campaign carries on, the company can measure the spread of the campaign by looking at the events on the blockchain. Moreover, the social network can prove to have sent the advertising only to people who declared such an interest.

In the next section, we show an implementation of our solution on Ethereum, which is the most used one for this kind of applications. Moreover, we give some detail about the cost of implementing our solution.

15.3 Implementation

In this section, we describe the detail regarding the implementation of our solution. We designed and deployed a smart contract, which implements the functions described in the previous sections. The smart contract, called `Advertising`, is implemented in Solidity, a statically-typed programming language designed for developing smart contracts, compiled to bytecode and executed on a suitable virtual machine (EVM).

The developed smart contract, which contains two functions, is reported in Listing 15.1. Every user is associated with an Ethereum address and can have a property `userMapping` indicating an advertising setting (Line 5). We decide to introduce an event `Setting`, which has as input the address of the social network, the address of the user and the result of the function, which is used to notify if an advertising is compliant with the user preference. The function setPref (Lines 8-10) is used by the user to set preferences: this choice is stored in the mapping userMapping. It is worth noting that the mapping is generated from msg.sender, the address of the user who calls the function setPref. The function checkPref (Lines 12-18) is called by an OSN interested in providing a targeted advertise for the user. This function compare the advertising request coming from the OSN with the one of the user (Line 14): the first comparison is done by an exclusive OR, which returns 1 where corresponding bits are different. This result is compared by an AND with not u_preference, in such a way that the result is zero if and only if the difference reported by the XOR is due to a bit equal to 1 of u_preference. In words, the result is true only if the user is interested in all the topic of the advertising.

The call to this function generates a Setting event reporting the reference to the user and the OSN, and the result of the call: in particular, the result will be true if the advertising request from the OSN is compliant with the user one, false otherwise. The use of events is necessary because the actual value returned by a function is always the hash of the transaction that's created: indeed, transactions do not return a value to the front-end because they are not immediately mined and included in the blockchain. As a solution, to obtain return value from the function, the front-end needs to keep watching for that event.

```
 1  pragma solidity ^0.5.9;
 2
 3  contract Advertising {
 4
 5     mapping(address=>uint) private userMapping;
 6     event Setting(address osn, address user, bool result);
 7
 8     function setPref(uint pref) public {
 9        userMapping[msg.sender]=pref;
10     }
11
12     function checkPref(address a, uint osn_request) public {
13        uint u_preference=userMapping[a];
14        if ((u_preference^osn_request)&($\sim$u_preference)>0)
15           emit Setting(msg.sender, a, false);
16        else
17           emit Setting(msg.sender, a, true);
18     }
19  }
```

Listing 15.1: Code of the smart contract.

In order to test this smart contract we used Etherscan [11], a Block Explorer and Analytics Platform for Ethereum, where it is possible finding all the Ethereum transactions. Particularly, when a function of the smart contract is called, the transaction starts and, after few seconds, it is visible on Etherscan. Moreover, in the section *Event Logs*, the events emitted during the execution of the function are also shown. The contract has been implemented by *Remix - Solidity IDE* [14] connected with Metamask [13], an extension for accessing Ethereum enabled distributed applications from the browser: the Ethereum testnet used is *Ropsten*.

The deploy of the contract, which is done one-tantum, costs 210 Micro(ETH) (in June 2019, this is about 0,051 $); the call to the function `setPref` costs 42 Micro(ETH) (about 0,010 $) and the function `checkPref` costs 23 Micro(ETH) (about 0,0056 $).

In summary, by paying 1 cent, a user can set the advertising preferences, whereas an OSN can check the compliance with users' choices of about 200 advertisings for about 1 dollar.

15.4 Related Work

In this section, we survey papers related to our proposal, which focuses on advertising in OSNs.

Targeted advertisement can enhance the success rate for selling products. The aim of [165] is presenting a model of advertising in OSNs. Particularly, the authors want to establish the following purchasing of a product from a given number of OSNs users and buyers. In their model, they consider several influence mechanisms and multiple rating levels.

In [242], the authors propose a study to analyze the advertising response in online social networks. This study defines a model based on collected data about the role of perceived enjoyment in advertising, that is perceived enjoyment is considered as a predicting variable of social influence and advertising variables.

In [175], OSN advertising services are explored as phenomena influencing online advertising markets. The authors suggest that most of OSNs are funded via advertising on their site. They concentrate on Facebook: they measure the Facebook Advertising Platform with suggested bids and then analyze bid data. They state that suggests bids to advertisers are calculated via sampling recent winning bids, furthermore, advertiser interest is focused on different parameters such as location, user interest, and age.

The paper [270] is focused on the targeting problem for advertisers. The authors identify two algorithms based on two prospective: OSN and advertiser. The former algorithms is a polynomial time algorithm, the latter reveals that advertisers reach more target audience by targeting subsets than directly targeting the users.

In [209], social relationships through online social networks are documented to validate the marketing value based on OSNs. The authors distinguish bonding and bridging connections in online social network and they analyze the impact of social capital, social status and sociability on response to advertising. Social capital has a positive effect regarding response to advertising, this concept brings about relationally-oriented and marketing-oriented management OSNs.

The paper [213] individuates the need for more effective advertising strategies in OSNs, because many of them are not able to generate consistent profits through advertising. The authors provide a probability-based method to forecast communication activity in Online Social Networks of users. They suppose that can be useful to identify more influential and active users, these ones can affect connections and contents online.

Several issues [172] arise regarding the implications of online advertising for the privacy of web users. These issues can expose users to attacks exploiting personalized advertising campaigns based on target bids; for example, a company with malicious aims, could extract information about the user.

The paper [148] analyzes the possible causes of privacy breaches and proposes several attacks generated from advertising systems with micro-targeting capabilities. The authors focus on the Facebook case study, in particular, on the risks of user privacy leakage. They argue that even the result of internal data mining, and not only using improper anonymization techniques, can attempt to breach the privacy of users.

In [107], the authors investigate how privacy regulation influences online advertising effectiveness in the European Union. This regulation defines restrictions for advertisers about collecting data on web users, the paper gives explanations about the loss of potency in not targeted advertising techniques. "Reactance" [253] is a motivational state when consumers resist something they find different by behaving in the opposite way to the one intended. This way, users can have the perception of unexpected control over personal information.

From the review of the state of the art, we conclude that targeting operation in the advertising coming from social network is relevant both for SN users and companies and that the aspect of users' privacy is very important. We believe that our proposal is the one that reaches the purpose of maintaining a correct and verifiable connection between the preferences of users and the advertising campaign of companies, yet keeping private user's information.

15.5 Conclusion

Blockchain technology is currently used in a lot of applications, from banking to industry, because it provides a registry and inventory system for the recording, tracking, monitoring, and transacting assets [247]. By relying on this technology, in this chapter, we proposed an approach to match user's preferences and the advertising campaign, in such a way that this matching can be proved by the social network.

A strong motivation for users to declare their interests is to receive personalized advertising: after they have published their interests, any social network can reach them with the most suitable products or services, provided that a user is registered in that social network. It is worth noting that in our approach user's interests are not declared on the online social network (as currently happens), but only on the blockchain. So, it is not necessary to ensure that the interests declared by the users match those they defined in the online social networks.

Our solution has been developed on Ethereum and has been shown to be cheap and effective. It enables a social network to comply with the general principle of accountability, also stated by the General Data Protection Regulation (GDPR) introduced recently. Indeed, leveraging blockchain, a social network is able to manage user preferences and can demonstrate that a user chose a particular interest in case of need (e.g., complaint from a user or a company).

Part V

Final Conclusions

In this thesis, we have presented new perspectives on balancing the demands of anonymity, privacy, and accountability in various application contexts. These points of view regard: *(i)* anonymity and digital identity, *(ii)* privacy and accountability and *(iii)* privacy and social network.

In the first part of this thesis, we described some basic concepts that are in common for every proposal included in this thesis.

In the second part of this thesis, we started by proposing a model that enables the execution of blockchain transactions linked with a public digital identity to enable digital-identity aware applications. Then, we applied this primitive to several challenges and contexts. In particular, we developed a solution that enables Ethereum transactions among secure digital identities not yet registered to a blockchain-based system. We also proposed a system that guarantees a certain degree of anonymity by not revealing users' whole identities while allowing users to prove the possession of some attributes to access an online service. Then, we explored the advantages of fog computing compared with the GDPR principle of data minimization in a healthcare scenario.

The third part of this thesis starts defining an access control scheme relying on blockchain technology that guarantees users' anonymity and the unlinkability of their different requests. Then, we presented a practical system for a real-life service delivery scenario that integrates the features of Ethereum's smart contracts with an Attribute-Based Encryption scheme. Then, we faced the challenges of an energy trading scenario by proposing a solution based on Ethereum and smart contracts that meets accountability and privacy requirements in smart grids. Finally, we defined a solution that allows the sharing of health records guaranteeing unlinkability between patient's identity and e-health records.

As for the fourth part, we first proposed an approach that combines blockchain technology with a highly adaptable model to define and verify users' privacy settings in social networks. Then, we explored how social networks can be exploited to issue and verify claims and credentials for accessing online services responding to the logic of selective disclosure of attributes. Finally, we developed a solution to match user preferences and advertising campaigns to manage user preferences and demonstrate, in case of need, the compliance with interests already declared and registered on the blockchain.

References

1. Base58Check. https://en.bitcoin.it/wiki/Base58Check_encoding.
2. ID-based Encryption. https://en.wikipedia.org/wiki/ID-based_encryption.
3. ISO/IEC 24760-1:2011 Information technology – Security techniques – A framework for identity management – Part 1: Terminology and concepts. http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html.
4. PiggyBee: CrowdShipping - Crowdsourced delivery. https://www.piggybee.com/.
5. Windows CardSpace. https://en.wikipedia.org/wiki/Windows_CardSpace.
6. Zipments: Same Day Delivery Service. https://zipments.com/.
7. Oraclize, 2016. http://www.oraclize.it/.
8. Bitnation Pangea | Your Blockchain Jurisdiction, 2018. https://tse.bitnation.co/.
9. OpenID – The Internet Identity Layer, 2018. https://openid.net/.
10. The eIDAS Regulation in 2017 – A pivotal year for digital services in the EU, 2018. https://www.gemalto.com/govt/identity/eidas-regulation-in-2017.
11. Etherscan. https://etherscan.io, 2019.
12. Facebook ads. https://www.facebook.com/business/ads, 2019.
13. Metamask. https://metamask.io, 2019.
14. Remix - Solidity IDE, 2019. https://remix.ethereum.org.
15. Solidity 0.5.7 documentation. https://solidity.readthedocs.io/en/v0.5.7, 2019.
16. Assad Abbas and Samee U Khan. A review on the state-of-the-art privacy-preserving approaches in the e-health clouds. *IEEE Journal of Biomedical and Health Informatics*, 18(4):1431–1441, 2014.
17. Saveen A Abeyratne and Radmehr P Monfared. Blockchain ready manufacturing supply chain using distributed ledger. *International Journal of Research in Engineering and Technology*, 2016.
18. Karim Abouelmehdi, Abderrahim Beni-Hessane, and Hayat Khaloufi. Big healthcare data: preserving security and privacy. *Journal of Big Data*, 5(1):1–18, 2018.
19. Andreas Abraham, Felix Hörandner, Thomas Zefferer, and Bernd Zwattendorfer. E-government in the public cloud: Requirements and opportunities. *Electronic Government*, 16, 2020.
20. Arif Ahmed, HamidReza Arkian, Davaadorj Battulga, Ali J Fahs, Mozhdeh Farhadi, Dimitrios Giouroukis, Adrien Gougeon, Felipe Oliveira Gutierrez, Guillaume Pierre, Paulo R

Souza Jr, et al. Fog computing applications: Taxonomy and requirements. *arXiv preprint arXiv:1907.11621*, 2019.

21. Nurzhan Zhumabekuly Aitzhan and Davor Svetinovic. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, 15(5):840–852, 2016.
22. Mustafa Al-Bassam. Scpki: A smart contract-based pki and identity system. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 35–40. ACM, 2017.
23. Sinică Alboaie and Doina Cosovan. Private data system enabling self-sovereign storage managed by executable choreographies. In *ICDAIS*, pages 83–98. Springer, 2017.
24. Maher Alharby and Aad van Moorsel. Blockchain-based smart contracts: A systematic mapping study. *arXiv preprint arXiv:1710.06372*, 2017.
25. Randa Almadhoun, Maha Kadadha, Maya Alhemeiri, Maryam Alshehhi, and Khaled Salah. A user authentication scheme of iot devices using blockchain-enabled fog nodes. In *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE, 2018.
26. Riham AlTawy, Muhammad ElSheikh, Amr M Youssef, and Guang Gong. Lelantos: A blockchain-based anonymous physical delivery system. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 15–1509. IEEE, 2017.
27. Merlinda Andoni, Valentin Robu, David Flynn, Simone Abram, Dale Geach, David Jenkins, Peter McCallum, and Andrew Peacock. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 100:143–174, 2019.
28. Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, pages 1–15, 2018.
29. Fabrizio Angiulli, Fabio Fassetti, Angelo Furfaro, Antonio Piccolo, and Domenico Saccà. Achieving service accountability through blockchain and digital identity. In *International Conference on Advanced Information Systems Engineering*, pages 16–23. Springer, 2018.
30. Simplice Asongu, Jacinta Nwachukwu, Stella-Maris Orim, and Chris Pyke. Crime and social media. *Information Technology & People*, 2019.
31. Simplice Asongu and Nicholas M Odhiambo. Tourism and social media in the world: An empirical investigation. *Journal of Economic Studies*, 2019.
32. Simplice A Asongu and Nicholas M Odhiambo. Governance and social media in african countries: An empirical investigation. *Telecommunications Policy*, 43(5):411–425, 2019.
33. Simplice A Asongu, Joseph I Uduji, and Elda N Okolo-Obasi. Homicide and social media: Global empirical evidence. *Technology in Society*, 59:101188, 2019.
34. Nuttapong Attrapadung and Hideki Imai. Dual-policy attribute based encryption. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, ed-

itors, *Applied Cryptography and Network Security*, pages 168–185, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

35. Oshrat Ayalon and Eran Toch. Not even past: Information aging and temporal privacy in online social networks. *Human–Computer Interaction*, 32(2):73–102, 2017.
36. Charles Baden-Fuller and Stefan Haefliger. Business models and technological innovation. *Long range planning*, 46(6):419–426, 2013.
37. Arnab Banerjee. Blockchain technology: Supply chain insights from erp. *Advances in Computers*, 2018.
38. Gaurang Bansal, Amit Dua, Gagangeet Singh Aujla, Maninderpal Singh, and Neeraj Kumar. Smartchain: a smart and scalable blockchain consortium for smart grid systems. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2019.
39. Jaume Barcelo. User privacy in the public bitcoin blockchain. *URL: http://www. dtic. upf. edu/jbarcelo/papers/20140704 User Privacy in the Public Bitcoin Blockc hain/paper.pdf)*, 2014.
40. Mohamed Baza, Mahmoud Nabil, Muhammad Ismail, Mohamed Mahmoud, Erchin Serpedin, and Mohammad Ashiqur Rahman. Blockchain-based charging coordination mechanism for smart grid energy storage units. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 504–509. IEEE, 2019.
41. Amos Beimel. *Secure schemes for secret sharing and key distribution*. Technion-Israel Institute of technology, Faculty of computer science Haifa, 1996.
42. Dario Belić. National Identification and Authentication System. http://infoz.ffzg.hr/INFuture/2015/images/papers/1-06\%20Belic,\%20National\%20Identification\%20and\%20Authentication\%20System.pdf, 2015.
43. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak specifications. *Submission to nist (round 2)*, pages 320–337, 2009.
44. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, May 2007.
45. Stefano Bistarelli, Marco Mantilacci, Paolo Santancini, and Francesco Santini. An end-to-end voting-system based on bitcoin. In *Proceedings of the Symposium on Applied Computing*, pages 1836–1841. ACM, 2017.
46. Kamanashis Biswas and Vallipuram Muthukkumarasamy. Securing smart cities using blockchain technology. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*, pages 1392–1393. IEEE, 2016.
47. Sandra J Blanke and Elizabeth McGrady. When it comes to securing patient health information from breaches, your best medicine is a dose of prevention: A cybersecurity risk assessment checklist. *Journal of Healthcare Risk Management*, 36(1):14–24, 2016.
48. Andreas Bogner, Mathieu Chanson, and Arne Meeuw. A decentralised sharing app running a smart contract on the ethereum blockchain. In *Proceedings of the 6th International Conference on the Internet of Things*, pages 177–178. ACM, 2016.

49. Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

50. Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

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

52. Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: an introduction. *R3 CEV, August*, 1:15, 2016.

53. Francesco Buccafurri, Vincenzo De Angelis, Gianluca Lax, Lorenzo Musarella, and Antonia Russo. An attribute-based privacy-preserving ethereum solution for service delivery with accountability requirements. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–6, 2019.

54. Francesco Buccafurri, Vincenzo De Angelis, Gianluca Lax, Lorenzo Musarella, and Antonia Russo. An attribute-based privacy-preserving ethereum solution for service delivery with accountability requirements. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–6, 2019.

55. Francesco Buccafurri, Lidia Fotia, and Gianluca Lax. Implementing advanced electronic signature by public digital identity system (spid). In *International Conference on Electronic Government and the Information Systems Perspective*, pages 289–303. Springer, 2016.

56. Francesco Buccafurri, Lidia Fotia, Gianluca Lax, and Rocco Mammoliti. Enhancing public digital identity system (spid) to prevent information leakage. In *International Conference on Electronic Government and the Information Systems Perspective*, pages 57–70. Springer, 2015.

57. Francesco Buccafurri, Gianluca Lax, Lorenzo Musarella, and Antonia Russo. Ethereum transaction and smart contracts among secure identities. In *Proc. of the Second Distributed Ledger Technology Workshop (DLT 2019)*, pages 5–16, Pisa, Italy, 2019.

58. Francesco Buccafurri, Gianluca Lax, Lorenzo Musarella, and Antonia Russo. Ethereum transactions and smart contracts among secure identities. In *DLT@ ITASEC*, pages 5–16, 2019.

59. Francesco Buccafurri, Gianluca Lax, Lorenzo Musarella, and Antonia Russo. Ethereum transactions and smart contracts among secure identities. In *DLT@ ITASEC*, pages 5–16, 2019.

60. Francesco Buccafurri, Gianluca Lax, and Antonia Russo. Allowing privacy-preserving fog computing with digital identity assurance in remote clinical services. *Electronic Government, an International Journal*, 1(1), 2022.

61. Francesco Buccafurri, Gianluca Lax, Antonia Russo, and Guillaume Zunino. Integrating digital identity and blockchain. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 568–585. Springer, 2018.

62. Francesco Buccafurri, Gianluca Lax, Antonia Russo, and Guillaume Zunino. Integrating digital identity and blockchain. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 568–585. Springer, 2018.

63. Francesco Buccafurri, Lorenzo Musarella, and Roberto Nardone. A routing algorithm increasing the transmission availability in smart grids. In *Proceedings of the 2019 Summer Simulation Conference*, page 47. Society for Computer Simulation International, 2019.

64. Franscesco Buccafurri, Gianluca Lax, and Antonia Russo. Exploiting digital identity for mobility in fog computing. In *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 155–160. IEEE, 2019.

65. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Annual International Cryptology Conference*, pages 56–72. Springer, 2004.

66. CEFDigital. Estonian eID scheme: Digi-ID. `https://ec.europa.eu/cefdigital/wiki/display/EIDCOMMUNITY/Estonia`, 2018.

67. Abdulla Chaer, Khaled Salah, Claudio Lima, Pratha Pratim Ray, and Tarek Sheltami. Blockchain for 5g: opportunities and challenges. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2019.

68. Yi Chen, Mingming Zha, Nan Zhang, Dandan Xu, Qianqian Zhao, Xuan Feng, Kan Yuan, Fnu Suya, Yuan Tian, Kai Chen, et al. Demystifying hidden privacy settings in mobile apps. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 570–586. IEEE, 2019.

69. Yun Chen, Hui Xie, Kun Lv, Shengjun Wei, and Changzhen Hu. Deplest: A blockchain-based privacy-preserving distributed database toward user behaviors in social networks. *Information Sciences*, 2019.

70. Lichen Cheng, Jiqiang Liu, Guangquan Xu, Zonghua Zhang, Hao Wang, Hong-Ning Dai, Yulei Wu, and Wei Wang. Sctsc: A semicentralized traffic signal control mode with attribute-based blockchain in iovs. *IEEE Transactions on Computational Social Systems*, 6(6):1373–1385, 2019.

71. Martin Christopher. *Logistics & supply chain management*. Pearson UK, 2016.

72. Christopher Allen. The Path to Self-Sovereign Identity. `http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html`, 2016. Online; accessed 3-July-2020.

73. Alan Cohn, Travis West, and Chelsea Parker. Smart after all: Blockchain, smart contracts, parametric insurance, and smart energy grids. *Georgetown Law Technology Review*, 1(2):273–304, 2017.

74. David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, W Timothy Polk, et al. Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile. *RFC*, 5280:1–151, 2008.

75. Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10):71, 2016.

76. Fangfang Dai, Yue Shi, Nan Meng, Liang Wei, and Zhiguo Ye. From bitcoin to cybersecurity: A comparative study of blockchain application and security issues. In *Systems and Informatics (ICSAI), 2017 4th International Conference on*, pages 975–979. IEEE, 2017.

77. Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.

78. Bhaskar DasGupta, Nasim Mobasheri, and Ismael G Yero. On analyzing and evaluating privacy measures for social networks under active attack. *Information Sciences*, 473:87–100, 2019.

79. Iman Dayarian and Martin Savelsbergh. Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. *Optimization Online*, 2017.

80. Angelo De Caro and Vincenzo Iovino. jpbc: Java pairing based cryptography. In *2011 IEEE symposium on computers and communications (ISCC)*, pages 850–855. IEEE, 2011. http://gas.dia.unisa.it/projects/jpbc/.

81. Brian Dean. Social network usage and growth statistics. https://backlinko.com/social-media-users, October 2021.

82. Ali Dorri, Fengji Luo, Salil S Kanhere, Raja Jurdak, and Zhao Yang Dong. Spb: A secure private blockchain-based solution for distributed energy trading. *IEEE Communications Magazine*, 57(7):120–126, 2019.

83. Jos Dumortier. Regulation (eu) no 910/2014 on electronic identification and trust services for electronic transactions in the internal market (eidas regulation). In *EU Regulation of E-Commerce*. Edward Elgar Publishing, 2017.

84. M Durgadevi and R Kalpana. A cooperative ga-sm-based prediction model for healthcare services. *Electronic Government, an International Journal*, 16(1-2):7–24, 2020.

85. eIDAS eID Technical Subgroup. eIDAS SAML Message Format. https://ec.europa.eu/cefdigital/wiki/download/attachments/82773108/eIDAS\%20SAML\%20Attribute\%20Profile\%20v1.2\%20Final.pdf, 2019. Accessed 13-January-2021.

86. eIDAS Observatory. eIDAS Regulation (Regulation (EU) N°910/2014). https://ec.europa.eu/futurium/en/content/eidas-regulation-regulation-eu-ndeg9102014, 2014.

87. eIDAS Observatory. SPID Sistema Pubblico di Identità Digitale. https://ec.europa.eu/futurium/en/content/eidas-regulation-regulation-eu-ndeg9102014, 2019. Online; accessed 17 August 2020.

88. Abida Ellahi and Rahat H Bokhari. Key quality factors affecting users' perception of social networking websites. *Journal of Retailing and Consumer Services*, 20(1):120–129, 2013.

89. Andreas Ellervee, Raimundas Matulevicius, and Nicolas Mayer. A comprehensive reference model for blockchain-based distributed ledger technology. In *ER Forum/Demos*, pages 306–319, 2017.

90. Eosio blockchain official website. https://eos.io/, 2020.

91. ePortugal. Digital mobile Key in Portugal. https://eportugal.gov.pt/en/servicos/ativar-a-chave-movel-digital, 2021.

92. Ethereum. Welcome to ethereum. https://www.ethereum.org, 2021.

93. Ethereum and IPFS APIs. Infura inc. https://infura.io/, 2020. Accessed 13-January-2021.

94. ethereumWiki. Problems. https://github.com/ethereum/wiki/wiki/Problems, 2016.

95. EverID. EverID Whitepaper. `https://coinosophy.files.wordpress.com/2018/05/everid-whitepaper.pdf`, 2018. Online; accessed 3-July-2020.

96. How can i adjust my facebook privacy settings? `https://www.facebook.com/help/193677450678703`, 2020.

97. Noémie Floissac and Yann L'Hyver. From aes-128 to aes-192 and aes-256, how to adapt differential fault analysis attacks on key expansion. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 43–53. IEEE, 2011.

98. Bogdan Cristian Florea. Blockchain and internet of things data provider for smart applications. In *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4. IEEE, 2018.

99. Federal Office for Information Security. German eID. `https://www.bsi.bund.de/EN/Topics/ElectrIDDocuments/German-eID/german-eID_node.html`, 2018.

100. Ministry for Primary Industries. Global Energy & CO2 Status Report 2019, (IEA), 2019. `https://www.iea.org/reports/global-energy-co2-status-report\\-2019`.

101. Keke Gai, Yulu Wu, Liehuang Zhu, Meikang Qiu, and Meng Shen. Privacy-preserving energy trading using consortium blockchain in smart grid. *IEEE Transactions on Industrial Informatics*, 15(6):3548–3558, 2019.

102. Keke Gai, Yulu Wu, Liehuang Zhu, Lei Xu, and Yan Zhang. Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks. *IEEE Internet of Things Journal*, 6(5):7992–8004, 2019.

103. Valentina Gatteschi, Fabrizio Lamberti, Claudio Demartini, Chiara Pranteda, and Víctor Santamaría. Blockchain and smart contracts for insurance: Is the technology mature enough? *Future Internet*, 10(2):20, 2018.

104. GDPR.EU. General Data Protection Regulation (GDPR), 2020.

105. Snigdha Gharami, B Prabadevi, and Anupama Bhimnath. Semantic analysis-internet of things, study of past, present and future of iot. *Electronic Government, an International Journal*, 15(2):144–165, 2019.

106. Michelle Goddard. The eu general data protection regulation (gdpr): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705, 2017.

107. Avi Goldfarb and Catherine E Tucker. Privacy regulation and online advertising. *Management science*, 57(1):57–71, 2011.

108. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.

109. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

110. Gideon Greenspan. Multichain private blockchain-white paper. *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, 2015.

111. Ke Gu, Na Wu, Bo Yin, and Weijia Jia. Secure data query framework for cloud and fog computing. *IEEE Transactions on Network and Service Management*, 17(1):332–345, 2019.

112. Zhitao Guan, Xin Lu, Naiyu Wang, Jun Wu, Xiaojiang Du, and Mohsen Guizani. Towards secure and efficient energy trading in iiot-enabled energy internet: A blockchain approach. *Future Generation Computer Systems*, 110:686–695, 2020.

113. Dongchao Guo, Jiaqing Dong, and Kai Wang. Graph structure and statistical properties of ethereum transaction relationships. *Information Sciences*, 492:58–71, 2019.

114. Rui Guo, Huixian Shi, Qinglan Zhao, and Dong Zheng. Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. *IEEE Access*, 6:11676–11686, 2018.

115. Brinda Hampiholi and Gergely Alpár. Privacy-preserving webshopping with attributes. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, pages 25–36. IEEE, 2017.

116. Haya R Hasan and Khaled Salah. Blockchain-based solution for proof of delivery of physical assets. In *International Conference on Blockchain*, pages 139–152. Springer, 2018.

117. Haya R Hasan and Khaled Salah. Blockchain-based solution for proof of delivery of physical assets. In *International Conference on Blockchain*, pages 139–152. Springer, 2018.

118. Haya R Hasan and Khaled Salah. Proof of delivery of digital assets using blockchain and smart contracts. *IEEE Access*, 6:65439–65448, 2018.

119. Xiaofan He, Juan Liu, Richeng Jin, and Huaiyu Dai. Privacy-aware offloading in mobile-edge computing. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

120. Christine V Helliar, Louise Crawford, Laura Rocca, Claudio Teodori, and Monica Veneziani. Permissionless and permissioned blockchain diffusion. *International Journal of Information Management*, 54:102136, 2020.

121. Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162), 2013.

122. Vincent C Hu, D Richard Kuhn, David F Ferraiolo, and Jeffrey Voas. Attribute-based access control. *Computer*, 48(2):85–88, 2015.

123. Dijiang Huang, Chun-Jen Chung, Qiuxiang Dong, Jim Luo, and Myong Kang. Building private blockchains over public blockchains (pop) an attribute-based access control approach. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 355–363, 2019.

124. Yuheng Huang, Haoyu Wang, Lei Wu, Gareth Tyson, Xiapu Luo, Run Zhang, Xuanzhe Liu, Gang Huang, and Xuxian Jiang. Characterizing eosio blockchain. *arXiv preprint arXiv:2002.05369*, 2020.

125. John Hughes and Eve Maler. Security assertion markup language (saml) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, pages 29–38, 2005.

126. Junbeom Hur and Dong Kun Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221, 2010.

127. Marco Iansiti and Karim R Lakhani. The truth about blockchain. *Harvard Business Review*, 95(1):118–127, 2017.

128. IBM. Ibm blockchain. `https://www.ibm.com/blockchain/solutions/identity`, 2020. Accessed 13-January-2021.

129. Maged Hamada Ibrahim. Octopus: An edge-fog mutual authentication scheme. *IJ Network Security*, 18(6):1089–1101, 2016.

130. Instagram privacy settings and information. `https://help.instagram.com/196883487377501`, 2020.

131. IRMA. IRMA scheme manager. `https://github.com/privacybydesign/pbdf-schememanager`, 2020. Online; accessed 3-July-2020.

132. IRMA-Privacy By Design Foundation. IRMA. `https://privacybydesign.foundation/irma-en/`, 2020. Online; accessed 3-July-2020.

133. Jim Isaak and Mina J Hanna. User data privacy: Facebook, cambridge analytica, and privacy protection. *Computer*, 51(8):56–59, 2018.

134. Ori Jacobovitz. Blockchain for identity management, 2016.

135. Chandan Kumar Jha and Oasis Kodila-Tedika. Does social media promote democracy? some empirical evidence. *Journal of Policy Modeling*, 2019.

136. Chandan Kumar Jha and Sudipta Sarangi. Does social media reduce corruption? *Information Economics and Policy*, 39:60–71, 2017.

137. Qi Jiang, Muhammad Khurram Khan, Xiang Lu, Jianfeng Ma, and Debiao He. A privacy preserving three-factor authentication protocol for e-health clouds. *The Journal of Supercomputing*, 72(10):3826–3849, 2016.

138. Yichuan Jiang and JC Jiang. Diffusion in social networks: A multiagent perspective. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(2):198–213, 2014.

139. Jolocom. Own your digital self. `https://jolocom.io/`, 2020. Online; accessed 3-July-2020.

140. Simon Josefsson. Rfc3548: The base16, base32, and base64 data encodings, 2003.

141. Ioannis Karamitsos, Maria Papadaki, and Nedaa Baker Al Barghuthi. Design of the blockchain smart contract: A use case for real estate. *Journal of Information Security*, 9(3):177–190, 2018.

142. Farzaneh Karegar, John Sören Pettersson, and Simone Fischer-Hübner. The dilemma of user engagement in privacy notices: Effects of interaction modes and habituation on user attention. *ACM Trans. Priv. Secur.*, 23(1), February 2020.

143. Florian Kelbert and Alexander Pretschner. Data Usage Control for Distributed Systems. *ACM Trans. Priv. Secur.*, 21(3):12:1–12:32, April 2018.

144. Saad Khan, Simon Parkinson, and Yongrui Qin. Fog computing security: a review of current applications and security solutions. *Journal of Cloud Computing*, 6(1):19, 2017.

145. Jongwoo Kim, Richard L Baskerville, and Yi Ding. Breaking the privacy kill chain: Protecting individual and group privacy online. *Information Systems Frontiers*, pages 1–15, 2018.

146. Jongwoo Kim, Eun Hee Park, Young Soon Park, Kyung Hee Chun, and Lynn L Wiles. Prosocial rule breaking on health information security at healthcare organisations in south korea. *Information Systems Journal*, 2021.

147. Markus Klems, Jacob Eberhardt, Stefan Tai, Steffen Härtlein, Simon Buchholz, and Ahmed Tidjani. Trustless intermediation in blockchain-based decentralized service marketplaces. In *International Conference on Service-Oriented Computing*, pages 731–739. Springer, 2017.

148. Aleksandra Korolova. Privacy violations using microtargeted ads: A case study. In *2010 IEEE International Conference on Data Mining Workshops*, pages 474–482. IEEE, 2010.

149. Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital supply chain transformation toward blockchain integration. In *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

150. Prasad Koti, P Dhavachelvan, T Kalaipriyan, Sariga Arjunan, J Uthayakumar, and Pothula Sujatha. An efficient healthcare framework for kidney disease using hybrid harmony search algorithm. *Electronic Government, an International Journal*, 16(1-2):56–68, 2020.

151. Frank Alexander Kraemer, Anders Eivind Braten, Nattachart Tamkittikhun, and David Palma. Fog computing in healthcare–a review and discussion. *IEEE Access*, 5:9206–9222, 2017.

152. Aparna Kumari, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. Fog computing for healthcare 4.0 environment: Opportunities and challenges. *Computers & Electrical Engineering*, 72:1–13, 2018.

153. Vishal Kumkar, Akhil Tiwari, Pawan Tiwari, Ashish Gupta, and Seema Shrawne. Vulnerabilities of wireless security protocols (wep and wpa2). *IJARCET Journal*, 1(2):34–38, 2012.

154. Gianluca Lax and Antonia Russo. A system to enforce user's preference in osn advertising. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1174–1178, 2019.

155. Gianluca Lax and Antonia Russo. Blockchain-based access control supporting anonymity and accountability. *Journal of Advances in Information Technology Vol*, 11(4), 2020.

156. Gianluca Lax and Antonia Russo. A lightweight scheme exploiting social networks for data minimization according to the gdpr. *IEEE Transactions on Computational Social Systems*, 8(2):388–397, 2021.

157. Gianluca Lax, Antonia Russo, and Lara Saidia Fascì. A blockchain-based approach for matching desired and real privacy settings of social network users. *Information Sciences*, 557:220–235, 2021.

158. Junqing Le, Di Zhang, Nankun Mu, Xiaofeng Liao, and Fan Yang. Anonymous privacy preservation based on m-signature and fuzzy processing for real-time data release. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

159. Jae Kyu Lee, Younghoon Chang, Hun Yeong Kwon, and Beopyeon Kim. Reconciliation of privacy with preventive cybersecurity: The bright internet approach. *Information Systems Frontiers*, pages 1–13, 2020.

160. YongJoo Lee and Keon Myung Lee. Blockchain-based rbac for user authentication with anonymity. In *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, pages 289–294, 2019.

161. Loic Lesavre, Priam Varin, Peter Mell, Michael Davidson, and James Shook. A taxonomic approach to understanding emerging blockchain identity management systems. *arXiv preprint arXiv:1908.00929*, 2019.

162. Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 568–588. Springer, 2011.

163. Lifang Li, Qingpeng Zhang, Xiao Wang, Jun Zhang, Tao Wang, Tian-Lu Gao, Wei Duan, Kelvin Kam-fai Tsoi, and Fei-Yue Wang. Characterizing the propagation of situational information in social media during covid-19 epidemic: A case study on weibo. *IEEE Transactions on Computational Social Systems*, 7(2):556–562, 2020.

164. Meng Li, Liehuang Zhu, and Xiaodong Lin. Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4573–4584, 2018.

165. Yongkun Li, Bridge Qiao Zhao, and John Lui. On modeling product advertisement in large-scale online social networks. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1412–1425, 2012.

166. Yongli Li, Peng Luo, and Paolo Pin. Utility-based model for characterizing the evolution of social networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

167. Zhuo-Rong Li, En-Chi Chang, Kuo-Hsuan Huang, and Feipei Lai. A secure electronic medical record sharing mechanism in the cloud computing platform. In *Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on*, pages 98–103. IEEE, 2011.

168. Kaitai Liang, Liming Fang, Willy Susilo, and Duncan S Wong. A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security. In *2013 5th International Conference on Intelligent Networking and Collaborative Systems*, pages 552–559. IEEE, 2013.

169. Xiaohui Liang, Zhenfu Cao, Huang Lin, and Jun Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 276–286, 01 2009.

170. LifeID. Welcome to lifeID. `https://lifeid.io/whitepaper.pdf`, 2020. Online; accessed 3-July-2020.

171. Wenmin Lin, Xuyun Zhang, Lianyong Qi, Weimin Li, Shancang Li, Victor S Sheng, and Surya Nepal. Location-aware service recommendations with privacy-preservation in the internet of things. *IEEE Transactions on Computational Social Systems*, 2020.

172. Yehuda Lindell and Eran Omri. A practical application of differential privacy to personalized online advertising. *IACR Cryptology EPrint Archive*, 2011:152, 2011.

173. Dongxi Liu. Efficient processing of encrypted data in honest-but-curious clouds. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 970–974. IEEE, 2016.

174. Yabing Liu, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 61–70. ACM, 2011.

175. Yabing Liu, Chloe Kliman-Silver, Robert Bell, Balachander Krishnamurthy, and Alan Mislove. Measurement and analysis of osn ad auctions. In *Proceedings of the second ACM conference on Online social networks*, pages 139–150. ACM, 2014.

176. Hal Lockhart and B Campbell. Security assertion markup language (saml) v2. 0 technical overview. *OASIS Committee Draft*, 2:94–106, 2008.

177. Rongxing Lu, Kevin Heung, Arash Habibi Lashkari, and Ali A Ghorbani. A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot. *IEEE Access*, 5:3302–3312, 2017.

178. Tom H Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*, 2015.

179. Michelle Madejski, Maritza Lupe Johnson, and Steven Michael Bellovin. The failure of online social network privacy settings, 2011.

180. Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control. In *IFIP international conference on distributed applications and interoperable systems*, pages 206–220. Springer, 2017.

181. Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer, 2018.

182. Awais Manzoor, Munam Ali Shah, Hasan Ali Khattak, Ikram Ud Din, and Muhammad Khurram Khan. Multi-tier authentication schemes for fog computing: Architecture, security perspective, and challenges. *International Journal of Communication Systems*, 2019.

183. Edoardo Marcucci, Michela Le Pira, Celine Sacha Carrocci, Valerio Gatta, and Eleonora Pieralice. Connected shared mobility for passengers and freight: Investigating the potential of crowdshipping in urban areas. In *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017 5th IEEE International Conference on*, pages 839–843. IEEE, 2017.

184. Newton Masinde and Kalman Graffi. Peer-to-peer based social networks: A comprehensive survey. *arXiv preprint arXiv:2001.02611*, 2020.

185. Hartwig Mayer. Ecdsa security in bitcoin and ethereum: a research survey. *CoinFaabrik, June*, 28, 2016.

186. Catherine A Meadows. Analyzing the needham-schroeder public key protocol: A comparison of two approaches. In *European Symposium on Research in Computer Security*, pages 351–364. Springer, 1996.

187. Stefan Mocanu, Ana Maria Chiriac, Cosmin Popa, Radu Dobrescu, and Daniela Saru. Identification and trust techniques compatible with eidas regulation. In *International Conference on Security and Privacy in New Computing Environments*, pages 656–665. Springer, 2019.

188. Mainack Mondal, Günce Su Yilmaz, Noah Hirsch, Mohammad Taha Khan, Michael Tang, Christopher Tran, Chris Kanich, Blase Ur, and Elena Zheleva. Moving beyond set-it-and-forget-it privacy settings on social media. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 991–1008, 2019.

189. Francisca Moreno and Shivangee Trivedi. Staying anonymous on the blockchain: Concerns and techniques, 2017. https://securingtomorrow.mcafee.com/mcafee-labs/staying-anonymous-on-the-blockchain-concerns-and-techniques/.

190. Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, and Vikas Kumar. Security and privacy in fog computing: Challenges. *IEEE Access*, 5(1):19293–19304, 2017.

191. Rida Zojaj Naeem, Saman Bashir, Muhammad Faisal Amjad, Haider Abbas, and Hammad Afzal. Fog computing in internet of things: Practical applications and future directions. *Peer-to-Peer Networking and Applications*, 12(5):1236–1262, 2019.

192. Nitin Naik and Paul Jenkins. Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect. In *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pages 163–174. IEEE, 2017.

193. Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system, 2008. https://bitcoin.org/bitcoin.pdf.

194. Jelle C Nauta and Rieks Joosten. Self-sovereign identity: A comparison of irma and sovrin. Technical report, TNO, 2019.

195. Blaise Ngonmang, Emmanuel Viennet, Savaneary Sean, Philippe Stepniewski, Françoise Fogelman-Soulié, and Rémi Kirche. Monetization and services on a real online social network using social network analysis. In *2013 ICDM Workshops*, pages 185–193. IEEE, 2013.

196. US NIST. Descriptions of sha-256, sha-384 and sha-512, 2001.

197. Alex Norta. Designing a smart-contract application layer for transacting decentralized autonomous organizations. In *International Conference on Advances in Computing and Data Sciences*, pages 595–604, 2016. http://www.ctan.org/pkg/subcaption.

198. OAuth Community Site. Open authorization. https://oauth.net/.

199. US Department of Health and Human Services. Health insurance portability and accountability act of 1996. *Public law*, 104:191, 1996.

200. Sejong Oh and Seog Park. Task–role-based access control model. *Information systems*, 28(6):533–562, 2003.

201. Openchain 0.7 documentation. https://docs.openchain.org/en/latest/, 2020.

202. Aafaf Ouaddah. A blockchain based access control framework for the security and privacy of iot with strong anonymity unlinkability and intractability guarantees. In *Advances in Computers*, volume 115, pages 211–258. Elsevier, 2019.

203. Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, 9(18):5943–5964, 2016.

204. Jaehong Park and Ravi Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 57–64, 2002.

205. Li Peng, Wei Feng, Zheng Yan, Yafeng Li, Xiaokang Zhou, and Shohei Shimizu. Privacy preservation in permissionless blockchain: A survey. *Digital Communications and Networks*, 2020.

206. Gareth W Peters and Efstathios Panayi. Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In *Banking Beyond Banks and Money*, pages 239–278. Springer, 2016.

207. Marc Pilkington. 11 blockchain technology: principles and applications. *Research handbook on digital transformations*, page 225, 2016.

208. Marc Pilkington. 11 blockchain technology: principles and applications. *Research handbook on digital transformations*, 225, 2016.

209. José Carlos Pinho and Ana Maria Soares. Response to advertising on online social networks: the role of social capital. *International Journal of Consumer Studies*, 39(3):239–248, 2015.

210. Otto Julio Ahlert Pinno, Andre Ricardo Abed Gregio, and Luis CE De Bona. Controlchain: Blockchain as a central enabler for access control authorizations in the iot. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

211. Serguei Popov. The tangle (2016). *Verfügbar: http://www. tangleblog. com/wpcontent/uploads/2016/11/IOTA_Whitepaper. pdf.[Zugriff am 22.05. 2019]*, 2018.

212. Walter Priesnitz Filho, Carlos Ribeiro, and Thomas Zefferer. Privacy-preserving attribute aggregation in eid federations. *Future Generation Computer Systems*, 92:1–16, 2019.

213. Florian Probst. Predicting users' future level of communication activity in online social networks: A first step towards more advertising effectiveness. In *AMCIS*, 2011.

214. The Provable blockchain oracle for modern DApps. `http://provable.xyz/`, 2020. Accessed 13-January-2021.

215. Lijin Quan, Lei Wu, and Haoyu Wang. Evulhunter: Detecting fake transfer vulnerabilities for eosio's smart contracts at webassembly-level. *arXiv preprint arXiv:1906.10362*, 2019.

216. Sebastian Ramacher. Privacy-preserving eid derivation for self-sovereign identity systems. In *ICICS 2019*, volume 11999, page 307. Springer Nature, 2020.

217. C Ramesh, K Venu Gopal Rao, and D Vasumathi. Comparative analysis of applications of identity-based cryptosystem in iot. *Electronic Government, an International Journal*, 13(4):314–323, 2017.

218. Shailendra Rathore, Pradip Kumar Sharma, Vincenzo Loia, Young-Sik Jeong, and Jong Hyuk Park. Social network security: Issues, challenges, threats, and solutions. *Inf. Sci.*, 421:43–69, 2017.

219. Drummond Reed, Jason Law, and Daniel Hardman. The technical foundations of sovrin. *The Technical Foundations of Sovrin*, 2016.

220. Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

221. Joel R Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T Graves, Fei Liu, Aleecia McDonald, Thomas B Norton, and Rohan Ramanath. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Tech. LJ*, 30:39, 2015.

222. RIPEMD160. Ripe message digest. `https://en.wikipedia.org/wiki/RIPEMD`, 2020. Accessed 13-January-2021.

223. Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

224. Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *International workshop on fast software encryption*, pages 371–388. Springer, 2004.

225. Manuel Rudolph, Denis Feth, and Svenja Polst. Why users ignore privacy policies–a survey and intention model for explaining user privacy behavior. In *International Conference on Human-Computer Interaction*, pages 587–598. Springer, 2018.

226. Manuel Rudolph, Svenja Polst, and Joerg Doerr. Enabling users to specify correct privacy requirements. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 39–54. Springer, 2019.

227. Sushmita Ruj, Milos Stojmenovic, and Amiya Nayak. Decentralized access control with anonymous authentication of data stored in clouds. *IEEE transactions on parallel and distributed systems*, 25(2):384–394, 2013.

228. Antonia Russo, Gianluca Lax, Baptiste Dromard, and Menad Mezred. A system to access online services with minimal personal information disclosure. *Information Systems Frontiers*, pages 1–13, 2021.

229. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005.

230. Ryuichi Sakai and Masao Kasahara. Id based cryptosystems with pairing on elliptic curve. *IACR Cryptology ePrint Archive*, 2003:54, 2003.

231. Khaled Salah, M Habib Ur Rehman, Nishara Nizamuddin, and Ala Al-Fuqaha. Blockchain for ai: Review and open research challenges. *IEEE Access*, 7:10127–10149, 2019.

232. Antonio Salis, Jens Jensen, Roberto Bulla, Glauco Mancini, and Paolo Cocco. Security and privacy management in a fog-to-cloud environment. In *European Conference on Parallel Processing*, pages 99–111. Springer, 2019.

233. Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer, 2000.

234. Martin Schanzenbach, Georg Bramm, and Julian Schütte. reclaimid: Secure, self-sovereign identities using name systems and attribute-based encryption. In *Trust-Com/BigDataSE*, pages 946–957. IEEE, 2018.

235. Daniel Servos and Sylvia L Osborn. Current research and open problems in attribute-based access control. *ACM Computing Surveys (CSUR)*, 49(4):1–45, 2017.

236. Don DH Shin. Blockchain: The emerging technology of digital trust. *Telematics and Informatics*, 45:101278, 2019.

237. Dong-Hee Shin. The effects of trust, security and privacy in social networking: A security-based approach to understand the pattern of adoption. *Interacting with computers*, 22(5):428–438, 2010.

238. Donghee Shin and Yujong Hwang. The effects of security and traceability of blockchain on digital affordance. *Online Information Review*, 2020.

239. Donghee Shin and Mohammed Ibahrine. The socio-technical assemblages of blockchain system: How blockchains are framed and how the framing reflects societal contexts. *Digital Policy, Regulation and Governance*, 2020.

240. Donghee Don Shin, Anestis Fotiadis, and Hongsik Yu. Prospectus and limitations of algorithmic governance: an ecological evaluation of algorithmic trends. *Digital Policy, Regulation and Governance*, 2019.

241. Pierluigi Siano. Demand response and smart grids—a survey. *Renewable and sustainable energy reviews*, 30:461–478, 2014.

242. Ana Maria Soares and José Carlos Pinho. Advertising in online social networks: the role of perceived enjoyment and social influence. *Journal of Research in Interactive Marketing*, 8(3), 2014.

243. Ye-Byoul Son, Jong-Hyuk Im, Hee-Yong Kwon, Seong-Yun Jeon, and Mun-Kyu Lee. Privacy-preserving peer-to-peer energy trading in blockchain-enabled smart grids using functional encryption. *Energies*, 13(6):1321, 2020.

244. Ivan Stojmenovic and Sheng Wen. The fog computing paradigm: Scenarios and security issues. In *2014 Federated Conference on Computer Science and Information Systems*, pages 1–8. IEEE, 2014.

245. Salvatore J Stolfo, Malek Ben Salem, and Angelos D Keromytis. Fog computing: Mitigating insider data theft attacks in the cloud. In *2012 IEEE symposium on security and privacy workshops*, pages 125–128. IEEE, 2012.

246. Cass R Sunstein. Valuing facebook. *Behavioural Public Policy*, pages 1–12, 2019.

247. Melanie Swan. *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

248. Take control of your Virtual Identity. `https://ec.europa.eu/commission/sites/beta-political/files/digital_avatar_280519_v5.pdf`, 2019.

249. Chandu Thota, Revathi Sundarasekar, Gunasekaran Manogaran, R Varatharajan, and MK Priyan. Centralized fog computing security platform for iot and cloud in healthcare

system. In *Fog computing: Breakthroughs in research and practice*, pages 365–378. IGI global, 2018.

250. Sergei Tikhomirov. Ethereum: state of knowledge and research perspectives. In *International Symposium on Foundations and Practice of Security*, pages 206–221. Springer, 2017.

251. Eran Toch, Yang Wang, and Lorrie Faith Cranor. Personalization and privacy: a survey of privacy risks and remedies in personalization-based systems. *User Modeling and User-Adapted Interaction*, 22(1-2):203–220, 2012.

252. Niko Tsakalakis, Sophie Stalla-Bourdillon, and Kieron O'hara. What's in a name: the conflicting views of pseudonymisation under eidas and the general data protection regulation. *Gesellschaft fur Informatik eV*, 2016.

253. Catherine E Tucker. Social networks, personalized advertising, and privacy controls. *Journal of marketing research*, 51(5):546–562, 2014.

254. How to protect and unprotect your tweets. `https://help.twitter.com/en/safety-and-security/how-to-make-twitter-private-and-public`, 2020.

255. UK National Cyber Security Center. Weekly threat report 2nd july 2021. `https://www.ncsc.gov.uk/report/weekly-threat-report-2nd-july-2021section2`.

256. uPort. uPort has evolved. `https://www.uport.me`, 2020. Online; accessed 3-July-2020.

257. Dirk Van Bokkem, Rico Hageman, Gijs Koning, Luat Nguyen, and Naqib Zarin. Self-sovereign identity solutions: The necessity of blockchain technology. *arXiv preprint arXiv:1904.12816*, 2019.

258. Paul Van Schaik, Jurjen Jansen, Joseph Onibokun, Jean Camp, and Petko Kusev. Security and privacy in online social networking: Risk perceptions and precautionary behaviour. *Computers in Human Behavior*, 78:283–297, 2018.

259. Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 735–737. ACM, 2010.

260. Naiyu Wang, Xiao Zhou, Xin Lu, Zhitao Guan, Longfei Wu, Xiaojiang Du, and Mohsen Guizani. When energy trading meets blockchain in electrical power system: The state of the art. *Applied Sciences*, 9(8):1561, 2019.

261. Shangping Wang, Yinglong Zhang, and Yaling Zhang. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access*, 6:38437–38450, 2018.

262. Shuai Wang, Liwei Ouyang, Yong Yuan, Xiaochun Ni, Xuan Han, and Fei-Yue Wang. Blockchain-enabled smart contracts: architecture, applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):2266–2277, 2019.

263. Shuai Wang, Yong Yuan, Xiao Wang, Juanjuan Li, Rui Qin, and Fei-Yue Wang. An overview of smart contract: architecture, applications, and future trends. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 108–113. IEEE, 2018.

264. Web3j SDK. Web3 labs ltd. `https://web3j.io/`, 2020. Accessed 13-January-2021.

265. Wei Wei, Houbing Song, Wei Li, Peiyi Shen, and Athanasios Vasilakos. Gradient-driven parking navigation using a continuous information potential field based on wireless sensor network. *Information Sciences*, 408:100–114, 2017.

266. Wei Wei, Xu Xia, Marcin Wozniak, Xunli Fan, Robertas Damaševičius, and Ye Li. Multi-sink distributed power control algorithm for cyber-physical-systems in coal mine tunnels. *Computer Networks*, 161:210–219, 2019.

267. Wei Wei, Bin Zhou, Dawid Połap, and Marcin Woźniak. A regional adaptive variational pde model for computed tomography image reconstruction. *Pattern Recognition*, 92:64–81, 2019.

268. Pamela J Wisniewski, Bart P Knijnenburg, and Heather Richter Lipford. Making privacy personal: Profiling social network users to inform privacy education and nudging. *International Journal of Human-Computer Studies*, 98:95–108, 2017.

269. Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

270. Chaolun Xia, Saikat Guha, and Shan Muthukrishnan. Targeting algorithms for online social advertising markets. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 485–492. IEEE Press, 2016.

271. Qi Xia, Emmanuel Boateng Sifah, Abla Smahi, Sandro Amofa, and Xiaosong Zhang. Bbds: Blockchain-based data sharing for electronic medical records in cloud environments. *Information*, 8(2):44, 2017.

272. Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *CoRR*, abs/1904.04098:1–34, 2019.

273. Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465, 2020.

274. Adamu Sani Yahaya, Nadeem Javaid, Fahad A Alzahrani, Amjad Rehman, Ibrar Ullah, Affaf Shahid, and Muhammad Shafiq. Blockchain based sustainable local energy trading considering home energy management and demurrage mechanism. *Sustainability*, 12(8):3385, 2020.

275. Rupeng Yang, Qiuliang Xu, Man Ho Au, Zuoxia Yu, Hao Wang, and Lu Zhou. Position based cryptography with location privacy: A step for fog computing. *Future Generation Computer Systems*, 78:799–806, 2018.

276. Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78. IEEE, 2015.

277. Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In *International conference on wireless algorithms, systems, and applications*, pages 685–695. Springer, 2015.

278. Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 2019.

279. Aiqing Zhang and Xiaodong Lin. Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain. *Journal of medical systems*, 42(8):140, 2018.

280. Chenghua Zhang, Jianzhong Wu, Yue Zhou, Meng Cheng, and Chao Long. Peer-to-peer energy trading in a microgrid. *Applied Energy*, 220:1–12, 2018.

281. Peng Zhang, Joseph K Liu, F Richard Yu, Mehdi Sookhak, Man Ho Au, and Xiapu Luo. A survey on access control in fog computing. *IEEE Communications Magazine*, 56(2):144–149, 2018.

282. Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

283. Kaile Zhou, Shanlin Yang, and Zhen Shao. Energy internet: the business perspective. *Applied Energy*, 178:212–222, 2016.

284. Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184. IEEE, 2015.