UNIVERSITAT ROVIRA i VIRGILI

Contributions to GDPR by Means of Smart Contracts

CRISTÒFOL DAUDÉN ESMEL



DOCTORAL THESIS
2025

Cristòfol Daudén Esmel

Contributions to GDPR by means of Smart Contracts

DOCTORAL THESIS

Supervised by Dr. Jordi Castellà-Roca

Co-Supervised by Dr. Alexandre Viejo

Doctoral program in Computer Science and Mathematics of Security

Department of Computer Engineering and Mathematics



UNIVERSITAT ROVIRA i VIRGILI

Tarragona

2025

I STATE that the present study, entitled "Contributions to GDPR by means of Smart Contracts", presented by Cristòfol Daudén Esmel for the award of the degree of Doctor, has been carried out under my supervision at the Department of Computer Engineering and Mathematics of this university, and that it fulfills all the requirements to be eligible for the International Doctorate Award.

Tarragona, 2025

Doctoral Thesis Supervisor	Doctoral Thesis Supervisor
Dr. Jordi Castellà-Roca	Dr. Alexandre Viejo

Acknowledgements

I would particularly like to thank my advisors Dr. Jordi Castellà and Dr. Alexandre Viejo for their guidance and support during the development of this thesis.

I also wish to show my appreciation to Josep Domingo-Ferrer and all CRISES group members, especially to Jesús Manjón, Carles Anglès Tafalla and Sergio Martinez for their help and advice. Finally, I am grateful to Dr. Vicenç Torra and the rest of the NAUSICA (PrivAcy-AWare traNSparent deCIsions) group of Umeå University for their hospitality and kindness during my 3 months research stay in Sweden.

Last but not least, I would like to thanks my family and friends for always being there when I needed them, especially to my parents and sister for all their support and love.

This work was partially supported by the Spanish Government under an FPU grant (ref. FPU20/03254), by the project "HERMES" funded by INCIBE and the European Union NextGenerationEU/PRTR, by project

PID2021-125962OB-C32 "SECURING/DATA" funded by MCIN/AEI/ 10.13039/501100011033/ FEDER, UE, and by the grant 2021SGR 00115 from the Government of Catalonia.

"Arguing that don't care about the right to privacy because you have nothing to hide is no different that saying you don't care about the free speach because you have nothing to say." Eduard Snowden

Contents

List of Figures **xiii**

List of Tables **xv**

Abstract **xvii**

Resum **xix**

Resumen **xxi**

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 7

1.3 Structure of this thesis 9

2 Related work **11**

2.1 Smart Contracts based Data Management Systems 12

2.2 Automatic and Transparent Management of Privacy Policies 17

2.3 GDPR-Compliant Solutions for Real-Time Bidding Protocol 20
2.4 Blockchain's cryptographic keys management systems . . 23

3 GDPR-compliant Personal Data Management System 29
3.1 Preliminaries 30
3.1.1 The GDPR in a nutshell 30
3.1.2 Blockchain technology and Bitcoin 32
3.1.3 Smart contracts and Ethereum 34
3.1.4 The XACML standard 36
3.2 Proposal Description 38
3.2.1 Requirements 38
3.2.2 Overview of the proposed architecture 42
3.2.3 Smart contracts 44
3.2.4 Architecture and procedures 49
3.3 Discussion 58
3.3.1 GDPR-Compliance analysis 60
3.3.2 Functionality analysis 64
3.3.3 Security and privacy analysis 68
3.3.4 Performance analysis 74
3.3.5 Smart contract costs 80
3.4 Integration of the new proposal in a real-world setting . . 87
3.4.1 User registration 89
3.4.2 Service access 90
3.4.3 Data processing 91

3.4.4 Data and consent management 91
3.4.5 Consent audit and legislation enforcement 93

4 PrivacySmart: Management of Privacy Policies 95
4.1 Proposal Description 96
4.1.1 System Architecture Overview 97
4.1.2 User Privacy Preferences 98
4.1.3 Pop-up interaction module 99
4.1.4 Consent Smart Contract 101
4.1.5 Workflow 101
4.2 Discussion 105
4.2.1 Implementation 105
4.2.2 Evaluation 107

5 Blockchain-Based GDPR-Compliant RTB 111
5.1 Preliminaries 112
5.1.1 Programmatic Advertising and Real Time Bidding 112
5.1.2 Pricing model 114
5.1.3 User consent in IAB's Europe OpenRTB 114
5.1.4 Audience Taxonomy 115
5.2 Proposal Description 118
5.2.1 Requirements 118
5.2.2 System architecture 119
5.2.3 Phase-1: Configuration 121
5.2.4 Phase-2: Web Browsing & Consent Enforcement . 125

5.2.5 Phase-3: Consent Management & Data Accountability . . . 127
5.3 Discussion . . . 127

6 Decentralized Multi-Platform Wallet 133

6.1 Preliminaries . . . 134
6.1.1 ECDSA . . . 134
6.1.2 Crypto wallets . . . 137
6.1.3 InterPlanetary File System (IPFS) . . . 146
6.2 Proposal Description . . . 147
6.2.1 Requirements . . . 147
6.2.2 System architecture . . . 149
6.2.3 Secure Key Management Smart Contract . . . 150
6.2.4 The proposed system in a nutshell . . . 153
6.3 Protocols . . . 156
6.3.1 Set-up . . . 156
6.3.2 Session key pair generation . . . 159
6.3.3 Key inventory . . . 162
6.3.4 Key recovery . . . 163
6.4 Functional analysis . . . 168
6.5 Security and privacy analysis . . . 178
6.6 Comparative analysis . . . 186
6.7 Preliminary prototype implementation . . . 190
6.7.1 Implementation details . . . 191
6.7.2 Interface overview . . . 193

6.7.3 Runtime performance analysis 196

7 Conclusions **201**

7.1 Contributions 201

7.2 Publications 204

7.3 Future Work 206

Bibliography **209**

List of Figures

3.1 XACML architecture and general workflow 38

3.2 Consent Management System architecture 45

3.3 Collection Consent Smart Contract generation and data collection 51

3.4 Data processing request and Processing Consent Smart Contract generation 52

3.5 Data access and retrieval 54

3.6 Consent audit process 59

3.7 Integration of the new proposal in a real-world setting . . . 88

4.1 Privacy Smart: System Architecture Overview 98

4.2 PrivacySmart: browser plug-in main interfaces. 107

4.3 PrivacySmart: smartphone application main interfaces. . . . 108

5.1 Programmatic Advertising Scheme 113

5.2 Subset of Interests segment of the IAB Tech Lab's Audience Taxonomy 116

5.3 Interests segment of the IAB Tech Lab's Audience Taxonomy 117

5.4 GDPR-Compliant RTB Proposal Overview 119

6.1 Decentralized Multi-Platform Wallet System's architecture . 151
6.2 Set-up the Smartphone App 157
6.3 Set-up the web browser Plug-in 160
6.4 Session key pair generation 162
6.5 Key Management 164
6.6 Key Recovery when Smartphone App is no longer accessible 166
6.7 Key Recovery when IPFS file is unavailable 167
6.8 Key Recovery when Smartphone App is no longer accessible and IPFS file is unavailable 169
6.9 Add and Remove Device Operation Evaluation 177
6.10 Modify Temporal Value Operation Evaluation 177
6.11 Store the Reference of a new Key Pair Operation Evaluation 178
6.12 Login 193
6.13 Sign-up 193
6.14 Mnemonic words and smart contract hash 194
6.15 Adding a Plug-in 194
6.16 Smartphone App's cryptographic keys 195
6.17 Plug-ins' cryptographic keys 195
6.18 dApps'cryptographic keys 195
6.19 Key recovery 195
6.20 Cost in milliseconds of the designed protocols 197

List of Tables

1.1 Notations used along the manuscript. 3

3.1 GDPR Articles in a Nutshell. 33
3.2 XACML terminology 37
3.3 Comparison of functionalities with State of the Art proposals. 75
3.4 Comparison of Get Processing Consent workflow steps. . . . 76
3.5 Comparison of Revoke Processing Consent workflow steps. . 79
3.6 Comparison of Access Data workflow steps. 79
3.7 Comparison summary 80
3.8 Comparison of EVM compatible Blockchains (March 2022). 84
3.9 Cost of the different functions of the provided Smart Contracts (CC referes to the Collection Consent Smart Contract and PC refers to the Processing Consent Smart Contract). . 87

4.1 Results of tool performance on websites that use a pop-up offered by the studied privacy and cookie policies banner providers. 110

6.1 ECDSA Parameters 137

6.2 Length relation between ENT, CS and MS 142
6.3 Comparison of EVM compatible Blockchains (February 2024) 175
6.4 Cost of the different functions of the *SKM SC* 178
6.5 Types of attacks and their coverage 179
6.6 Mapping of the considered requirements to specific properties. 187
6.7 Proposal comparison with existing mainstream schemes . . 188
6.8 Proposal comparison with state-of-the-art schemes present in the literature 190
6.9 Cost in milliseconds of the designed protocols 196

Abstract

The rapid growth of digital technologies is transforming society by introducing innovative services and products that simplify and enhance our daily lives. However, these conveniences often involve the generation and collection of vast amounts of personal data from users. This data is frequently processed by service providers to improve offerings or generate revenue, commonly by monetizing user data rather than charging directly for services. Such practices raise serious concerns regarding the potential misuse of sensitive information, as illustrated by scandals like Cambridge Analytica and Twitter's use of personal data for advertising purposes.

To address this situation, the European Union introduced the General Data Protection Regulation (GDPR) in 2018, designed to protect individuals' privacy. This legislation aims to ensure that personal data is gathered lawfully and used for legitimate purposes. While the GDPR has improved data protection awareness and strengthened consumer rights, its abstract nature and the lack of transparent implementation present ongoing challenges. Centralized verification systems employed by service providers are often opaque, leaving users with limited control over their data and inadequate tools for managing consent. In particular, users struggle to keep track of where they have granted consent and what specific permissions they have given, making it difficult to exercise their rights effectively.

This thesis tackles these challenges by introducing novel, GDPR-compliant protocols that improve user control and transparency in personal data management. The first contribution presents a new consent paradigm based on a Blockchain framework, empowering users to manage their personal data while providing immutable records to ensure regulatory compliance. The second and third contributions demonstrate the practical applicability of this framework in real-world scenarios. The PrivacySmart system, for instance, enables users to manage privacy preferences and cookie policies during web browsing. It automates interactions with cookie consent pop-ups and facilitates the exercise of privacy rights, reducing the users' burden on the management of such tedious tasks. Additionally, the third contribution proposes a decentralized consent management solution for aligning Real-Time Bidding (RTB) systems with privacy regulations. It introduces a new specificity metric within the working taxonomy, allowing users to understand the exposure of their profiles based on the data they choose to share. These innovations aim to enhance user autonomy and address pressing concerns related to data privacy and centralized control in the digital landscape. Finally, because these Blockchain-based solutions require users to manage cryptographic keys, the fourth contribution introduces a decentralized, multi-platform wallet. This wallet supports secure key management and underpins the other proposed tools, leveraging both Blockchain and InterPlanetary File System (IPFS) technologies to ensure user-friendly and secure interactions.

Resum

El ràpid creixement de les tecnologies digitals està transformant la societat amb la introducció de nous serveis i productes que simplifiquen i milloren la nostra vida quotidiana. No obstant això, aquestes comoditats sovint impliquen la generació i recollida de grans quantitats de dades d'usuari. Aquestes dades poden ser processades pels proveïdors de serveis per millorar les seves ofertes o generar ingressos, sovint mitjançant la seva monetització en lloc de cobrar directament pels serveis. Aquestes pràctiques generen certes preocupacions sobre el possible ús indegut d'informació delicada, com es va veure en escàndols com el de Cambridge Analytica o en l'ús que feia Twitter de dades personals amb finalitats publicitàries.

Per fer front a aquesta situació, la Unió Europea va introduir el Reglament General de Protecció de Dades (RGPD) l'any 2018, dissenyat per protegir la privacitat de les persones. Aquesta legislació té com a objectiu garantir que les dades personals es recullin de manera legal i s'utilitzin amb finalitats legítimes. Tot i que el RGPD ha contribuït a augmentar la consciència sobre la protecció de dades i ha reforçat els drets dels consumidors, la seva naturalesa abstracta i la manca d'una implementació clara i transparent continuen representant reptes. Actualment, els sistemes centralitzats de verificació utilitzats pels proveïdors de serveis sovint són opacs, cosa que deixa els usuaris amb poc control sobre les seves dades.

En particular, els usuaris tenen dificultats per controlar en quins llocs han atorgat el seu consentiment i quines autoritzacions específiques han concedit, la qual cosa complica l'exercici efectiu dels seus drets.

Aquesta tesi aborda aquests reptes proposant protocols innovadors i compatibles amb el RGPD que milloren el control i la transparència en la gestió de dades personals per part dels usuaris. La primera contribució presenta un nou paradigma de consentiment basat en un marc de treball amb tecnologia blockchain, que permet als usuaris gestionar les seves dades personals i proporcionar registres immutables que asseguren el compliment regulador. La segona i la tercera contribució demostren l'aplicabilitat pràctica d'aquest marc en escenaris reals. El sistema PrivacySmart, per exemple, permet als usuaris gestionar les preferències de privacitat i les polítiques de cookies durant la navegació web. Automatitza la interacció amb els pop-ups de consentiment de cookies i facilita l'exercici dels drets de privacitat, reduint la càrrega dels usuaris en la gestió d'aquestes tasques tedioses. A més, la tercera contribució proposa una solució descentralitzada de gestió del consentiment per alinear els sistemes d'oferta publicitària en temps real (*Real-Time Bidding*) amb les regulacions de privacitat. També s'introdueix un nou valor d'especificitat dins la taxonomia funcional, que permet als usuaris comprendre el grau d'exposició dels seus perfils segons les dades que decideixen compartir. Aquestes innovacions augmenten l'autonomia dels usuaris i aborden les preocupacions urgents relacionades amb la privacitat de les dades i el control centralitzat en l'actual entorn digital. Finalment, com que aquestes solucions basades en *Blockchain* requereixen que els usuaris gestionin claus criptogràfiques, la quarta contribució introdueix una cartera descentralitzada i multiplataforma. Aquesta cartera permet una gestió segura de claus i dona suport a les altres eines proposades, utilitzant tecnologies *Blockchain* i del Sistema de Fitxers InterPlanetari (IPFS) per garantir una interacció segura i fàcil d'usar.

Resumen

El rápido crecimiento de las tecnologías digitales está transformando la sociedad mediante la introducción de servicios y productos innovadores que simplifican y mejoran nuestra vida cotidiana. Sin embargo, estas comodidades suelen implicar la generación y recopilación de grandes cantidades de datos personales de los usuarios. Estos datos son procesados con frecuencia por los proveedores de servicios para mejorar sus ofertas o generar ingresos, normalmente mediante su monetización en lugar de cobrar directamente por los servicios. Estas prácticas generan preocupaciones sobre el posible uso indebido de información sensible, como se evidenció en escándalos como el de Cambridge Analytica o en el uso que hacía Twitter de los datos personales con fines publicitarios.

Para hacer frente a esta situación, la Unión Europea introdujo el Reglamento General de Protección de Datos (RGPD) en 2018, diseñado para proteger la privacidad de las personas. Esta legislación tiene como objetivo garantizar que los datos personales se recopilen de forma legal y se utilicen con fines legítimos. Aunque el RGPD ha contribuido a aumentar la conciencia sobre la protección de datos y ha reforzado los derechos de los consumidores, su naturaleza abstracta y la falta de una implementación clara y transparente siguen representando desafíos. Así, los sistemas de verificación utilizados por los proveedores de servicios suelen ser opacos,

lo que deja a los usuarios con poco control sobre sus datos y con herramientas insuficientes para gestionar el consentimiento. En particular, los usuarios tienen dificultades para saber en qué lugares han otorgado su consentimiento y qué autorizaciones específicas han concedido.

Esta tesis aborda estos retos proponiendo protocolos compatibles con el RGPD que mejoran el control y la transparencia en la gestión de los datos personales por parte de los usuarios. La primera contribución presenta un nuevo paradigma de consentimiento basado en un marco de trabajo con tecnología *Blockchain*, que permite a los usuarios gestionar sus datos personales y proporcionar registros inmutables que aseguran el cumplimiento normativo. La segunda y la tercera contribución demuestran la aplicabilidad práctica de este marco en escenarios reales. El sistema *PrivacyS-mart* permite a los usuarios gestionar sus preferencias de privacidad y las políticas de cookies durante la navegación web. Esta gestión se realiza de forma automática, facilitando el ejercicio de los derechos de privacidad y reduciendo la carga de los usuarios en la gestión de estas tareas tediosas. Además, la tercera contribución propone una solución descentralizada de gestión del consentimiento para alinear los sistemas de puja publicitaria en tiempo real (*Real-Time Bidding*) con las regulaciones de privacidad. También se introduce un nuevo valor de especificidad dentro de la taxonomía utilizada, que permite a los usuarios comprender el grado de exposición de sus perfiles según los datos que decidan compartir. Finalmente, dado que estas soluciones basadas en *Blockchain* requieren que los usuarios gestionen claves criptográficas, la cuarta contribución introduce una cartera descentralizada y multiplataforma. Esta cartera permite una gestión segura de claves y da soporte a las demás herramientas propuestas, utilizando tecnologías *Blockchain* y del Sistema de Archivos Interplanetario (IPFS) para garantizar una interacción segura y fácil de usar.

Chapter 1

Introduction

1.1 Motivation

The rapid advancement and development of new digital technologies have changed the dynamics of our daily lives by providing us with new services and devices. These advancements are helpful in many areas, for example, in social connectivity, we are able to maintain contact with those close through distance thanks to more powerful video-conference apps. Advances in information storage allow users to store huge amounts of information locally or remotely and move it around virtually instantaneously. Similarly, being able to precisely detect location, results in the development of technologies that reduces traffic, empowers people to move in cities where they have never been, allow hikers to avoid getting lost in the mountain and have allowed the development of smart cars. These improvements have also made possible the rise of new products such as

Smartwatches. These devices track a set of biomedical data: heart rate, sleep, activity and overall fitness level, among others, which can be useful in medical treatments and health control. Another commodity that is becoming more common due to its functionality and low cost are smart home devices with an Intelligent Personal Assistant (IPA) embedded. They allow users to program their security, lighting, and heating to exactly meet their daily needs with many features being controllable remotely. These services and devices generate huge amounts of information, which is processed by Service Providers (SPs) in order to improve and develop new services, but also to fund themselves making their products cheaper, even free [1, 2].

However, the collection and processing of these personal data can result in the extraction of detailed user knowledge and sensitive information, such as their health records, logs and lifelogging, which may jeopardize the individuals' privacy. A significant example of this situation is the Cambridge Analytica scandal in which the personal data of thousands of Facebook users was misused in order to influence voters in the US Elections 2016[1]. This incident raised serious concerns about the technical, commercial, political, and ethical aspects of personal data collection and analysis by platform owners such as Facebook and other third parties. Similarly, Twitter admitted the use of users' personal data for advertising purposes when they provided an email address or phone number for safety or security purposes[2]. Similarly, SPs can also share users' information with the government for social analysis [3, 4].

To mitigate the massive collection and processing of users' personal

[1]Facebook and Cambridge Analytica: What you need to know about the data scandal, `https://www.cnet.com/news/facebook-cambridge-analytica-data-mining-and-trump-what-you-need-to-know/`, Date of Publication: 18 April 2018.

[2]Personal information and ads `https://help.twitter.com/en/safety-and-security/data-and-privacy-controls/personalization-and-data-settings`, Accessed: 20 December 2023 (Deprecated).

data, the European Union took a decisive step with the General Data Protection Regulation (GDPR) [5] that came into effect from May, 2018. The main objective of this legislation is to guarantee specific privacy rights to users ensuring that personal data "can only be gathered legally, under strict conditions, for a legitimate purpose"; as well as to bring full control back to the data owners. The legislation considers four key players: i) the Data Subject (DS), this is, the identified or identifiable natural person(s) to whom the personal data belongs; ii) the Data Controller (DC), this is, the natural person or legal entity that determines the purposes for which and the means by which personal data is processed; iii) the Data Recipient (DR), this is, the natural person or legal entity to which the personal data is disclosed (it can be the same DC or another third party); and iv) the Data Processor (DP), this is, the natural person or legal entity that processes personal data on behalf of the DC. In order to enforce compliance with the regulation, organizations that violate these rights can be subject to administrative fines up to 20 million EUR.

Notation	**Description**
DS	Data Subject
DC	Data Controller
DP	Data Processor
DR	Data Recipient
SP	Service Provider
GDPR	General Data Protection Regulation
BC	Blockchain Technology
SA	Supervisory Authority
SC	Smart Contract
IPFS	InterPlanetary File System
dApp	Distributed Application

Table 1.1: Notations used along the manuscript.

The GDPR has brought several relevant benefits, such as boosting data security and protection awareness, or enabling consumers to control their preferences and to actively engage in the preservation of their rights. In

particular, under this regulation, companies need methods to efficiently demonstrate that the information collected and processed fulfills the regulation, first, to show the DSs that their personal data is lawfully processed, second, to prove the regulation compliance before a Supervisory Authority (SA). Nevertheless, works such as [6, 7] have shown that applying the GDPR in a practical scenario also raises significant issues that must be addressed. On the one hand, the GDPR's requirements are highly abstract, which in turn results in generic rules and principles that may be difficult to interpret and implement for both companies and consumers. On the other hand, traditional solutions follow a GDPR-compliance verification architecture in which all processes related to data consent management are carried out in Service Providers' own servers, out of reach from users' control, which, in turn, brings relevant concerns regarding their *lack of transparency* [8]. In addition to that, the use of these independent provider-side verification architectures, each managed by its hosting organization, makes it extremely difficult for DSs to ascertain where or to whom they have given consent for processing their personal data. This issue may be acknowledged by DSs as *lack of control* over their data.
As a result, nowadays, even in those scenarios in which the GDPR is actually being applied, DSs have no effective tools to know transparently and easily which data is being collected and processed and for which purposes. Instead, DSs are mostly limited to giving their consent beforehand, in a way that is based on abstract clauses. In order to bring the needed transparency to the data management, it is of paramount importance to provide mechanisms that enable all involved parties to verify both the data custody and data processing. From the DSs' point of view, these methods would allow them to be aware of which data is being collected, who is storing/processing it, and for which purposes; and they would also empower them, handing them an effective way to control and manage what happens with their personal data by means of granting/denying per-

missions to these Data Controllers/Processors. Regarding those entities, they would also be benefited greatly from the deployment of those tools. More specifically, DCs and DPs would be able to retrieve a certified proof that would serve as evidence of GDPR-compliance before any Supervisory Authority.
An example of where these tools can be applied is in managing privacy and cookie policy pop-ups that users are forced to interact with while navigating and accessing online services. Users can empower themselves to specifically select which personal data they are willing to share and control all the places where they have given consent. Additionally, these mechanisms can ensure compliance with current privacy regulations in Real-Time Bidding systems; the most widely used protocol in digital advertising delivery.

Taking this into account, the use of Smart Contracts (SCs) implemented on Blockchain Technology (BC) is a promising measure in GDPR-compliant personal data management thanks to its features: transparency, traceability, non-repudiation, integrity, immutability and decentralization. Some authors have already highlighted the potential BC technology and SCs for general-purpose data management and storage [6, 7, 9, 10, 11, 12, 13, 14].
However, the Blockchain ecosystem introduces its own vulnerability: individuals must use a Blockchain *wallet* to engage with the platform. This tool serves as a user interface bridging the gap between Blockchain networks and real-world applications. Specifically, a Blockchain wallet: i) facilitates the generation and storage of a user's private key; ii) authenticates a Blockchain user by confirming possession of the private key without relying on third-party intervention; and iii) grants a user access to their personal digital assets stored on the Blockchain (e.g., cryptocurrency ownership).

Given the considerations above, it becomes evident that the relevance of Blockchain wallets cannot be overlooked. So, there is a need for privacy-preserving wallet that addresses the challenges posed by the multitude of devices that users integrate into their daily lives without relying on any third party.

Regarding this, ensuring the security and synchronization of cryptographic key pairs across various devices is a challenging task. While using the Cloud for synchronization might seem like a straightforward solution, it introduces a centralization point that compromises the advantages of full decentralization, such as transparency and user control. As a result, the upcoming privacy-focused wallet design must navigate the complexities of today's multi-platform ecosystem while preserving its core decentralized benefits.

Additionally, when it comes to user privacy, reusing the same key pair for multiple Blockchain interactions poses significant risks. Adversaries could potentially link a wallet to an individual, exposing sensitive information. For instance, an attacker might trace a cryptographic key pair across several Distributed Applications (dApps) used by the same person and leverage them for *record linkage attacks* to uncover personal details. To reduce this risk, it's crucial to generate new key pairs for each dApp interaction or Blockchain activity. Thus, a robust privacy-preserving solution would involve a wallet that automatically and seamlessly creates and manages new key pairs for every transaction or interaction with a new service.

Last but not least, providing key recovery features in Blockchain wallets is crucial for maintaining user confidence, convenience, and security within the decentralized ecosystem. In this context, key recovery mechanisms are vital to mitigate the risk of losing access to digital assets due to forgotten or compromised keys. By allowing users to recover their keys if they lose access or their devices are damaged, Blockchain wallets can improve the

overall user experience and alleviate concerns about permanently losing valuable assets. Additionally, key recovery options can help drive broader adoption of Blockchain Technology by offering a safety net for users who might be hesitant to engage with decentralized systems because of key management worries.

1.2 Contributions

The aim of this thesis is to contribute with novel GDPR-complying protocols and technologies which provide effective measures for users to have control over their personal data while tackle the deployability, centralization and privacy issues found in the current literature works. Throughout the thesis we use a set of acronyms in order to make the text more readable, see Table 1.1 where all these notations are summarized for better understanding.

Under these premises, in the first contribution, we propose a Blockchain-based GDPR-compliant personal data management system that provides mechanisms for Data Subjects to control and manage their personal data, while providing publicly accessible and immutable evidence for a Service Provider to prove compliance with current legislation. These evidences might be then used by a Supervisory Authority that performs an auditing procedure on the SP to verify that this entity is processing the data processing exactly as agreed with the affected DSs.
Our proposal brings a new conceptual design and system architecture for human-centric access control and personal data management that uses Blockchain Technology, Smart Contracts, and the XACML access control standard to allow Service Providers to fulfill the GDPR requirements. Our work differentiates between the data collection and data processing concepts by identifying the Data Controller and Data Processor actors and treating them in a related but separate way. We also reduce the overhead

on Data Subjects. The target here is to minimize the required knowledge on Blockchain Technology that those actors may have and, also, reduce the number of operations that they must perform. We argue that both concepts, if not properly addressed, may seriously discourage the use of any personal data management system by the community.

In what regards our second and third contributions, we apply the developed technologies in two real-world use case scenarios. On the one hand, we propose PrivacySmart: an innovative system that manages privacy and cookie policy pop-ups for users as they navigate and access online services, all while taking their privacy preferences into account. This system empowers users by granting them control over the policies they have accepted and their corresponding websites, while helping them avoid the burden of policy management.
On the other hand, we address the challenges of ensuring compliance of Real-Time Bidding (RTB) systems with privacy regulations. In order to do so, we propose a novel framework that aligns RTB practices with these legal standards using the Blockchain Technology to decentralize user consent management, shifting control away from centralized systems and giving users direct control over their personal data. This system enables seamless web browsing by automatically handling consent management in accordance with user preferences.

Finally, in our fourth contribution, we satisfy the requirement of generating and managing the keys required in the proposed tools by providing a fully decentralized multi-platform wallet. It leverages on Blockchain and InterPlanetary File System (IPFS) technologies for managing asymmetric keys and enables key recovery. This novel wallet empowers users to interact with dApps built on Blockchain SCs while preserving their privacy and ensuring seamless key recovery in case of device theft or damage.

1.3 Structure of this thesis

The remainder of this thesis is organized as follows. First, in Chapter 2 the existing general-purpose data management systems are surveyed. We also present existing implementations that try to address the management of privacy policies and cookie pop-ups and the solutions that try to bring compliance to the RTB infrastructure. To conclude the chapter, we summarize existing proposals for Blockchain wallets. Chapter 3 presents our first contribution, an access control system for efficient and GDPR-compliant personal data management based on the Blockchain Technology. The second and third contributions are presented in Chapter 4 and Chapter 5 respectively, presenting two examples where the previous system can be applied: first, regarding the management and automatic acceptance of privacy and cookie policy pop-ups on behalf of the users, and second; bringing compliance with current legislation to the Real Time Bidding protocol, the most widely used technology in digital advertising. As all proposals are Blockchain-based and require the management of cryptographic keys, in Chapter 6 we present our last contribution, a multi-platform system for key management and recovery in decentralized applications. Finally, the concluding remarks and some guidelines for future research are given in Chapter 7.

Chapter 2

Related work

The contributions of this thesis can be categorized into three main areas: i) a framework for the management of users consent according to the GDPR through the use of the Blockchain Technology; ii) the application of the general framework into real world use cases, such as the management of privacy and cookie policy pop-ups when users access online services and the legal compliance of Real-Time Bidding protocol into the advertising ecosystem, and; iii) the design and development of a decentralized multi-platform wallet which allow users to manage all criptographic keys required by the previous systems.

On this basis, current published solutions on each of the presented areas are analyzed into different sections of this chapter. First, we consider Smart Contracts-based Data Management Systems that allow the different involved parties into user consent generation to prove the agreements made between them regarding the use and storage of the personal data. Then, we define works that apply our working lines in real world ap-

plications. On the one hand, analyze proposed works to facilitate the privacy policies and cookies usage management between users and Service Providers through the use of smart contracts and the Blockchain Technologies. On the other hand, we consider exploring alternative strategies that reduce reliance on personal data while maintaining advertising effectiveness. Finally, we show proposals in the literature to design and create effective and secure solutions concerning the generation, storage, recovery, and operation of Blockchain's cryptographic keys.

2.1 Smart Contracts based Data Management Systems

It has been emphasized that there is a need for general-purpose data management systems that enable the involved parties to verify the agreements made regarding the use and storage of personal data. To address this, recent research has explored the use of Smart Contracts and Blockchain Technology, which offer key features like transparency, traceability, non-repudiation, integrity, immutability, and decentralization.
These proposals can be categorized into two main scenarios: i) when a Data Subject wants to share their personal data with others and retains control over its management, and ii) when a Service Provider collects a Data Subject's personal data for further processing, typically in exchange for providing a product or service, such as one offered for free.

There are several proposals in the literature that fall into the first category, some relevant examples are [12], [15], [16], [13], [17] and [18]. These schemes follow a similar approach. In particular, they combine a decentralized storage system, the Ethereum Blockchain, and Smart Contracts, in order to provide a fair access control to those Data Processors that

want to collect and use the data shared by the DSs. An example can be found in [12], where authors combine technologies such as the interplanetary file system (as the decentralized storage system) and the Attribute Based Encryption (ABE) with the use of Smart Contracts running over the Ethereum Blockchain to provide a data sharing secure system. While those schemes present significant differences in the way the access policies are created, used, and, in some cases, revoked, all of them require the Data Subjects to deal with the burden of managing the use of the protected data. This implies that DSs must employ their own infrastructure and control all the data's life cycle, an scenario which is not aligned with our envisaged use case. For this reason, in the following, we will focus on the second use case in which a Service Provider collects personal data from the DSs for further processing by DPs.

Turning the spotlight in this second category, the next main aspect to be taken into account is whether those proposals are GDPR-compliant or not. In this way, wide-purpose schemes such as [19], [11], [20], [21], or health-care related specific proposals such as [9], [10],[22], and [23] do not take into account the GDPR requirements in their procedures. These schemes, in general, share a similar approach based on providing a decentralized personal data management system that ensures that users own and control their data, using the Blockchain as an automated access-control manager which, in some cases, is improved by means of Smart Contracts. For example, in [20] the authors use Smart Contracts to store the consumer access control polices that will be used in the access control process; also, in the health-care related system presented [10], Smart Contracts are used to link patients with their corresponding cryptographic material, hold a list of interactions among participants and, finally, establish a Patient-Provider Relationship Contract that identifies the records held by the care provider. Although these proposals address many issues, first, as already mentioned, they are not GDPR-compliant, which repre-

sents a significant shortcoming in the current context; last, but not less important, these works do not consider the different costs that may arise when being deployed on a large scale setting. In this way, the authors of those schemes only provide a high level description without any kind of implementation or evaluation of their proposed systems.

Focusing now on the GDPR-compliant proposals that can be found in the literature, the first scheme following this path is introduced in [6]. In this work, the authors present a scheme that uses Smart Contracts to embody the consent of the Data Subjects while their personal data is encrypted and directly held on the Blockchain. Although the data is encrypted, an external entity may identify the person behind a private key and link all the data stored in the Blockchain, hence this proposal does not provide anonymity but pseudonymity instead.

The authors in [14] propose a personal data and identity management system in which the consent given by the Data Subject is held by a controller by means of a Smart Contract, while her personal data is stored in an off-chain repository. The authors of this work only present a conceptual design. Technical details are not addressed.

Similarly, the solution presented in [24] uses Smart Contracts to maintain an Access Control List (ACL) that represents the rights to access a bundle of data; and it encrypts and stores the personal data into a decentralized file storage system. Again, the authors do not address relevant technical details such as how the ACL looks like or how the Smart Contracts are implemented.

Truong et al. [7] introduces a GDPR-compliant personal data management platform that allows Data Subjects to impose data usage consent, ensuring that only designated parties can process their personal data, and logging all data activities in an immutable distributed ledger using Smart

Contract and cryptography techniques. Mechanisms which are related to GDPR compliance are ported to a Blockchain network from a traditional centralised server by means of Smart Contracts. The proposed system has been successfully implemented on a permissioned Blockchain, however, its deployability on a public Blockchain has not been tested yet. Moreover, this scheme tackles the collection and data processing consents in a coarse-grained fashion not allowing the Data Subject to make micro decisions regarding whether a certain party can process or not only a part of the whole personal data which is available to others. Last but not least, this scheme stores a hash of the collected personal data in the distributed ledger for integrity purposes which, in turn, may result in a significant processing overhead when gathering dynamic data due to the necessity of keeping this hash up to date.

Davari and Bertino, in [25], combine the Blockchain Technology and Smart Contracts with the XACML (eXtensible Access Control Markup Language) access control specification to provide attribute-based access control and data management, while allowing Service Providers to fulfill the GDPR requirements. The proposed system consists of four main components: i) an ontology to formalize the notion of the GDPR consents, data flows, and decisions; ii) a decentralized personal data management that ensures compliance by leveraging the Blockchain; iii) a personal data repository which guarantees that only authorized Data Controllers and Data Processors can access the data; and iv) an XACML-based access control model containing consents, compliance, and a data management mechanism to enforce the legal privacy regulations specified by the GDPR. Even though this solution is promising, the authors do not provide any kind of implementation that proofs its deployability. Moreover, similarly to [7], this solution has issues when dealing with dynamic data, such as search logs or medical data, due the fact that they store the hash of the protected data in the Smart Contracts for integrity purposes.

The work presented by Barati and Rana in [26] shows how some GDPR rules can appear as opcodes in SCs. These opcodes allow to verify the operations that cloud providers carry out over users' personal data in a transparent and automatic way. The translation of GDPR rules into pseudo-code basically consists in writing questions related to data protection and privacy measures that should be supported by cloud services. This rules can then be assigned to each of the operations that can be carried out by providers (write, read, transfer, ...). The authors of this work explain how their scheme should be implemented and provide an extensive experimental section. In particular, they detail how four Smart Contracts are used to verify actor operations with respect to a set of GDPR rules, these are: the GDPR compliance contract (i.e., the policy that applies to the protected data); the Data Subject consent contract; the container contract that records all operations performed on personal data; and the Verification contract that allows to confirm whether the Data Subject consent has been obtained. Despite the significant positive qualities of this proposal, it also has two main issues to be considered: i) the use of four different SCs implies a significant computational cost that, in turn, may directly affect the efficiency of this proposal; and ii) a third party called the "Contract Activator" is required to generate all Smart Contracts, implying too many communications between all parties.

In [27], Merlec et al. propose a smart-contract-based dynamic consent management system backed by Blockchain Technology that targets personal data usage under the GDPR. More specifically, this scheme system allows individuals to control the collection and usage consent of their personal data throughout the data life-cycle. The proposed system cannot be considered fully decentralized due to the fact that a central entity behaving as regulator is required to allow entities to join the system and to approve their respective roles in the architecture. Moreover, the authors also expose other limitations such as the fact that an efficient and user-

friendly key management scheme is required to guarantee the security and privacy features; and the fact that deleting the individuals' stored personal data is complex and challenging due to the immutable nature of the Blockchain.

Finally, the authors in [28] use a private Blockchain to provide a delegation-based personal data processing request notarization framework under the GDPR. This framework allows DSs to delegate the requests to process their personal data to the Data Controllers. These data processing requests and the corresponding processing results are stored in a private Blockchain and notarized via a trusted institution of the linked Blockchain network. A main issue of this solution is that it directly omits how DSs should give their consent to DCs in order to collect their data. In the same way, it also omits how DPs should gather that data in order to process it.

2.2 Automatic and Transparent Management of Privacy Policies

Regarding real world problems that fall in our research lines, we find the management of privacy policies and cookies usage. This is a tedious and uncomfortable process for users who just want to access to some service in the web [29]. In order to address this, there are proposals that address the consent management between users and Service Providers through the use of smart contracts and the Blockchain Technologies.

For example, Vyas et al. propose in [30] a framework for automatically producing privacy policies for users' personal web contents based on a small amount of annotation information. However, this work is limited to the generation of individual policies for each content published into a social network, they do not consider general consent given when

accessing different web services that request users' data collection. In a more general ambit, authors in [31] and [32] present CookiePicker, a system that automatically validates the usefulness of cookies from a Web site and set the cookie usage permission on behalf of users. This system identifies those cookies that cause perceivable changes on a web page as useful, while simply classifying the rest as useless, thus enabling useful cookies and disabling the rest. However, this system does not take into account current paradigm where web service providers do not limit data collection from cookies but from other sources, making the users also to accept their privacy policies. Finally, authors in [33], present a concept for a privacy-friendly cookie setting interface that tries to help users during cookies preferences configuration. Their concept uses an assistant that guides the user to configure their privacy preferences via a series of questions. These preferences will be applied in the cookies configuration to all visited websites. However, the interface just manages the functionalities available in the cookie settings of common browsers, it does not consider available configurations on individual websites, neither the privacy policies that must be accepted on these websites. If a user wants a specific privacy and cookie policies configuration according to her privacy preferences on each website, it must be done manually.

A part from the scientific works trying to address the aforementioned issue with the automatic and transparent management of privacy and cookie policies, modern web browsers also provide users with some solutions. A user can set detailed cookie privacy options for web sites before or during visiting these sites. However, these cookie privacy options use to fail when are used in practice, as they are confusing for regular users [34].

On the other hand, we can find some commercial cookie management software for regular web browsers. While analyzing these tools, we have observed that although they all manage some cookie pop-ups and are convenient to use, they are not so versatile in terms of the number of web

pages with which they can interact. For example, Polish Cookie Consent [1] automatically accepts cookie policy/GDPR on websites. On the other hand, Ninja Cookie [2] and Auto Cookie Output [3] have a more restrictive policy by removing cookie banners and rejecting the use of non-essential cookies. Finaly, Super Agent [4] automatically fills out website's cookie consent forms on user's behalf based on their preferences. This tool splits optional cookies into three categories (Advertising, Functional and Performance), thus automatically opt out of cookies that are not clearly within any of these categories and let users control the remaining three. Although a combination of these tools could be efficient in terms of cookies acceptance/rejection, they do not offer cookies management, as none offers the possibility of consulting the consent they manage neither take into account privacy policies that also must be managed along with cookies.

After evaluating the available proposals pertaining to privacy policies and the administration of cookie usage, we have come to recognize two main trends. On one hand, there are proposals aimed at addressing consent management between users and service providers, but these often lack practicality. On the other hand, there are tools designed to streamline the management of privacy policies and cookie usage by employing automatic acceptance or rejection mechanisms. However, these tools limit users' ability to maintain comprehensive control over the consents they grant. In light of these findings, there is a clear necessity for a practical solution that enables users to effortlessly and transparently handle privacy policies

[1] Polish Cookie Consent, `https://addons.mozilla.org/en-GB/firefox/addon/polish-cookie-consent/`, Last updated: 11 February 2022.

[2] Ninja Cookie, `https://addons.mozilla.org/en-GB/firefox/addon/ninja-cookie/`, Last updated: 14 August 2020.

[3] Auto Cookie Optout, `https://addons.mozilla.org/en-GB/firefox/addon/auto-cookie-optout/`, Last updated: 4 August 2022.

[4] Super Agent - Automatic cookie consent, `https://addons.mozilla.org/en-GB/firefox/addon/super-agent/?utm_source=addons.mozilla.org&utm_medium=referral&utm_content=search`, Last updated: 20 June 2025.

and cookie usage while retaining full control over all granted consents.

2.3 GDPR-Compliant Solutions for Real-Time Bidding Protocol

Another real world problem addressed during the development of this thesis is related to the legal compliance of digital advertising. Research on this line has primarily focused on three key areas: i) improving ad targeting and revenue generation from the advertiser's perspective; ii) analyzing the impact of digital ads on users, including consumer behavior, perception, and privacy concerns; and iii) exploring alternative strategies that reduce reliance on personal data while maintaining advertising effectiveness.

The first area, which has been the focus of much of the existing research, explores multiple strategies to enhance ad delivery and personalization. Some studies analyze user behavior through click-through logs to improve ad click-through rates [35, 36], while others examine the broader ad exchange ecosystem, investigating time-dependent patterns in impressions, bids, and conversion rates to refine RTB mechanisms [37]. Additional approaches include recommendation systems that balance ad revenue and user experience [38], dynamic budget allocation strategies that optimize campaign spending [39], and analyses of how website quality influences ad performance and user engagement [40]. Although these studies have advanced targeted advertising and monetization, they rely heavily on extensive user data collection and often require user consent for data processing. However, obtaining meaningful and GDPR-compliant user consent remains a challenge, highlighting the need for alternative privacy-preserving approaches that reduce reliance on personal data.

Regarding the impact of digital advertising on users, research has highlighted significant privacy concerns, particularly related to user tracking, data leakage, and compliance with legal frameworks. Studies have shown that major advertising trackers can identify users with over 99.5% certainty after just 30 search result clicks [41] and that RTB can expose up to 27% of a user's browsing history to bidders [42]. These privacy risks have led to increased scrutiny over the adequacy of user consent in programmatic advertising. While regulatory efforts such as the GDPR and the California Consumer Privacy Act (CCPA) aim to strengthen user rights, research suggests that current consent mechanisms often fail to meet legal standards due to their complexity and lack of transparency [43]. The Belgian Data Protection Authority's ruling against IAB Europe further underscored these issues, concluding that the TCF does not ensure valid GDPR-compliant consent, given the challenges in informing users and managing Data Controller responsibilities [44]. These challenges call for new privacy-preserving approaches that can ensure compliance while minimizing reliance on personal data.

The third and final area of research examines the risks that digital advertising poses to user privacy and explores alternative approaches that reduce reliance on personal data. Traditional ad-matching schemes, such as the one proposed in [45], initially rely only on webpage content and ad features, avoiding direct user tracking. This limitation in data collection provides a basic form of privacy protection, as no personal user information is gathered. However, the approach also lacks adaptability since it does not incorporate user feedback. To improve performance, the authors later extend their model by incorporating ad impression and click data, which introduces tracking mechanisms. While this enhances targeting efficiency, it also increases data collection and reduces privacy protection. This transition reflects a common pattern in digital advertising research where efficiency improvements often come at the cost of greater data col-

lection. These challenges highlight the need for GDPR-compliant privacy-preserving solutions that ensure effective advertising while respecting user consent.

In contrast, Provost et al. [46] propose a privacy-friendly approach to audience selection for online brand advertising by leveraging browsing behavior to infer quasi-social networks. Unlike traditional methods that rely on explicit user tracking, this approach constructs an anonymous network based on shared visits to user-generated content pages, such as social networking sites. The framework identifies potential brand audiences by selecting social-network neighbors of users who have previously exhibited brand affinity. A ranking mechanism based on brand proximity ensures that selected audiences show higher brand engagement while preserving user anonymity. The authors demonstrate that this method can effectively target relevant audiences without collecting personal identifiers or content data. However, it does not address broader issues of user consent and transparency, which are central to the objectives of this paper.

Finally, to address GDPR compliance issues in RTB and the TCF following their declaration as non-compliant by European authorities in February 2022, IAB Europe proposed two new initiatives: the *Vendor Compliance Program* and the *Global Accountability Platform*. The Vendor Compliance Program consists of both automated and manual monitoring on end-user devices to identify what information is collected by companies participating in the TCF. On the other hand, the Global Accountability Platform is an audited self-reporting system designed to provide accountability for the movement of RTB data between servers. However, as noted in [47], both initiatives face significant challenges, and the study questions whether the use of the TCF for RTB can be effectively monitored, audited, or secured.

As a result, there is still a need for a non-intrusive solution that en-

sures legal compliance in digital advertising while preserving the existing paradigm. Rather than replacing the system, targeted modifications should enhance user consent management, data privacy, and GDPR compliance without disrupting the industry.

2.4 Blockchain's cryptographic keys management systems

Previous presented Blockchain-based GDPR consent management frameworks require a system to handle the cryptographic keys they use. Blockchain wallets play a crucial role in this technology, leading to various proposals in the literature aimed at developing secure and efficient solutions. These efforts generally focus on aspects such as key generation, storage, recovery, and operation. Based on these factors, Blockchain wallets can be grouped into three main types: i) trusted third-party-dependent wallets, ii) multi-signature wallets, and iii) seed-derived wallets.

The first category, *trusted third-party-dependent wallets*, represents perhaps the most extensively explored realm, primarily focusing on the storage of cryptographic keys (or core data facilitating their recovery) in external repositories. These repositories may consist of servers owned by service providers or other third parties chosen directly by users themselves. Some relevant proposals that fall within this category are [48, 49, 50, 51, 52, 53, 54, 55, 56]. It is interesting to analyze the evolutionary trajectory of these solutions. For instance, [48] introduces a two-server password-authenticated secret sharing (2PASS) scheme, wherein users initially distribute secret shares of their password and secret key to both servers. Later, they can retrieve these shares by sending new secret shares of their password to the same servers, which collaboratively deter-

mine whether all received secrets constitute shares of the same password. Similarly, [52] presents a secret key backup and recovery protocol that relies on the threshold secret sharing algorithm, through which the wallet splits user's cryptographic keys into multiple shares and distributes them among carefully selected third parties. Along the same lines, [54] proposes HasDPSS, a Blockchain-based key management system tailored for decentralized storage. Acknowledging the diverse reliability of participants, this system incorporates a hierarchical access structure to enable fine-grained key management, building upon the principles of secret sharing. Finally, works such as [55] and [56] enhance security by incorporating biometrics to encrypt the shares of private keys. However, all proposals in this category inherently rely on external entities to safeguard users' cryptographic materials, a significant limitation discussed in the introduction concerning widely used online wallets, impeding their ability to meet the requirements of fully decentralization and anonymity from users point of view.

Works falling into the second category, *multi-signature wallets*, share similarities with those introduced earlier, as they partition their secrets and enlist external parties to store the resulting fragments. Nonetheless, their primary emphasis lies in safeguarding cryptographic keys against misuse rather than prioritizing recoverability, which is presumed due to the distribution of keys or key shares among a group of participants within the system. Some proposals within this category are [57, 58, 59, 60, 61]. Notably, [58] exemplifies this approach with a threshold-optimal signature algorithm, where the secret key is distributed among n parties, and at least t of them are required to sign a transaction. Additionally, [59] presents an advancement over [58] by leveraging homomorphic encryption to reduce computational costs. While the focus on safeguarding against misuse is a relevant aspect of these solutions, two significant drawbacks emerge: firstly, correct key recovery depends on the involvement of a sufficient number of participants, potentially resulting in temporary unavailability

or prolonged recovery times if participants are intermittently offline; and secondly, there exists a clear reliance on external entities.

In the solutions that fall into the third and last category, *seed-derived wallets*, cryptographic material is generated by deriving it from an initial seed. A common method involves generating this seed from mnemonic words, which users must commit to memory to enable recovery of their cryptographic keys. This approach is notably employed in popular web browser extension wallets like Metamask, as introduced earlier in the paper. However, should a user forget these mnemonic words, all associated data on the Blockchain linked to the corresponding secret keys would be irrevocably lost. Moreover, writing down these mnemonic words introduces a potential security vulnerability. To mitigate these concerns, alternative approaches leveraging biometric data or personal information have been devised by the scientific community. These proposals offer key recovery without the need for third-party storage of cryptographic keys or related core information used for recovery. Noteworthy schemes within this category are [62, 63, 64, 65], which are discussed in further detail.

In [62], the authors propose an innovative approach to key generation and recovery in a healthcare Blockchain, utilizing biosensors alongside *fuzzy vault* techniques. Prior to transmitting physiological signals to the healthcare Blockchain, a biosensor node generates a key using additional physiological signals and fuzzy vault technology. This key is then employed to encrypt the respective signals. Later, when a user seeks to retrieve her physiological data from the Blockchain, she can use her biosensor node to recover the encryption key and utilize it for data restoration. However, a significant drawback of this method arises if a user experiences drastic changes in physiological signals, such as those resulting from accidents or illnesses, rendering key recovery impossible. Furthermore, there exists a potential vulnerability wherein an adversary capable of intercepting the

physiological signals used in seed generation could gain access to the user's keys.

Authors in [63] introduce the "Partial Knowledge Recovery Scheme" (PKRS), which facilitates the retrieval of an encrypted private key by leveraging personal security questions. When a user intends to secure a private key, she is prompted with a series of questions. Next, the answers to these questions are utilized to encrypt fragments of the private key, resulting in the creation of a secured private key. This secured private key comprises an array of tuples, each containing a question alongside a private key fragment encrypted with the corresponding answer. Upon requesting the recovery of the associated private key, the user must provide answers to a subset of the stored security questions. These answers, stored within the secured private key, activate and combine the fragments, ultimately enabling key recovery. Notably, this scheme requires no external entity during the recovery process, as all necessary information is contained within the secured private key itself. However, it is imperative to acknowledge that a third party is still necessary for storing the secured private key. In addition to that, an adversary may use social engineering techniques to gather user information and deduce the answers to the security questions.

Seo et al. [64] present a comparable method but, in this case, images are used instead of questions. In particular, the authors propose a novel approach to private key generation and recovery, based on the concept that users retain lasting memories from distinct images. Cryptographic keys are generated through a process involving these images and their associated locations, chosen by the user to evoke natural memories. Later, the user's private key is derived based on the specified picture locations. During key recovery, the user must recall the locations of the chosen pictures from the initial key generation phase. However, a significant drawback of this

approach surfaces: adversaries with knowledge of the picture locations may exploit this information to retrieve the corresponding private key. Moreover, the effectiveness of this method heavily relies on the user's long-term memory capabilities, thereby posing challenges similar to those encountered by wallets utilizing mnemonic words for seed generation.

The authors of [65] advocate for the implementation of hardware wallets to safeguard cryptographic material. Their proposal involves employing an Elliptic-Curve Diffie-Hellman (ECDH) key agreement protocol to back up hardware wallets. Through this method, the root of private keys is transferred between hardware wallets, addressing potential Man-In-The-Middle attack vulnerabilities associated with ECDH through side-channel human visual verification facilitated by the display screen on a hardware wallet. Consequently, the user possesses two hardware wallets with identical private keys, with one serving as the primary wallet and the other as a backup. However, in our operational context, where a new key pair is generated for each service access, adopting this scheme would impose a significant overhead on the user. In particular, each key pair would need manual securing, rendering the process cumbersome and impractical.

Chapter 3

Blockchain-based Access Control System for Efficient and GDPR-compliant Personal Data Management System

In this chapter we present a new lightweight blockchain-based GDPR-compliant personal data management platform in order to solve the identified problematic in Section 2.1, regarding the lack of control that users have to manage what happens with their personal data. The objective of our proposal is to offer public access immutable evidences that reflect the reached agreements between Data Subjects and Service Providers. In this way, Service Providers can effectively demonstrate that they are ful-

filling the regulation, and Data Subjects are able to control and manage their personal data according to their legitimate rights. We have implemented the new system, and we have performed a detailed study which includes: GDPR-compliance, provided functionality, security and privacy issues, and the cost in terms of gas and US dollars of the different operations to be run on the Blockchain.

The contributions of this Chapter have been published in [66] and an extended version of this work has been published in a Journal in [67].

The remainder of this Chapter is structured as follows. First, Section 3.1 gives a brief description of the main technologies and concepts used by the proposed system. Then, entering more in detail, Section 3.2 describes the proposed system and the protocols on which it is based. Section 3.3 provides a functional, security and privacy analysis along with a comparison with other proposals in the literature. Finally, Section 3.4 presents how the presented proposal is integrated in a real-world environment.

3.1 Preliminaries

A brief description about the main technologies and concepts used in our contribution is next introduced.

3.1.1 The GDPR in a nutshell

The GDPR was created to give the citizens of the EU (i.e., Data Subjects) control over their personal data. This legislative text consists of 99 articles covering all of the technical and admin principles around how commercial and public organizations must handle clients and customers personal data [5]. We differentiate two main aspects of the GDPR; “personal data” and “processing of personal data”:

- "personal data" pertains to "any information relating to an identified or identifiable natural person" [Art 4.1.]. For example, the name, online identifier, email, or even one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;

- whereas "processing of personal data" refers to "any operation or set of operations which is performed on personal data or on sets of personal data" [Art 4.2.]. Therefore, a simple operation of storing a username or an IP address on a web server constitutes processing of personal data of a user.

So, the main goal of the GDPR legislation is to provide citizens with full control over their personal data. To fulfill this, this regulation differentiates between the roles: Data Subject, Data Controller and Data Processor; and it explicitly specifies associated rights and obligations under the EU data protection law. In this way, this regulation requires that personal data must be managed by a DC ensuring the rights of the DS, in such way that the DS can impose consents and withdraw those consents whenever she wants. The DS must also be able to know at any time *who*, *what*, *why*, *when* and *how* her data is being processed. For this data being able to be collected and processed, DCs and DPs need DS's valid legal consent. Furthermore, apart from being compliant with the GDPR, DCs have to be able to demonstrate it in front of a Supervisory Authority in an audit.

After a deep analysis of the legislative text, we have extracted some articles from which we obtain the requirements for our proposal. However, from the extracted ones, there are some articles we have not considered in this work because they are out of scope such as Art 5. (Principles relating to Processing of Personal Data); or because we will address them in a

future work, such as Art. 16 (Right to Rectification), Art. 19 (Notification Obligation) or Art. 20 (Right to Data Portability). These articles are summarized in Table 3.1.

3.1.2 Blockchain technology and Bitcoin

The concept of Blockchain architecture was first mentioned by S. Haber and W.S. Stornetta in 1991 [68]. In this work, they propose computationally practical procedures for digital time-stamping of documents (text, audio, picture, and video files). The idea behind this scheme is to make infeasible for a user either to back-date or to forward-date her document, even with the collusion of a time-stamping service. This represented a computationally practical solution for the order and management of digital documents, so that they could not be modified or manipulated.

Some years later, some developers working under the pseudonym *Satoshi Nakamoto* released a white paper establishing the model for a proof-of work based Blockchain [69]. This Blockchain consists of a distributed growing list of records, called blocks, that are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle tree [70]). This design allows to timestamp blocks without requiring them to be signed by a trusted party and introduces a difficulty parameter to stabilize the rate in which blocks are added to the chain. This system was implemented the following year by Nakamoto, becoming the core component of the cryptocurrency Bitcoin, where it serves as the public ledger for all transactions on the network.

The security of the Blockchain relies on the hardness of the hash function used to link the blocks which, in the case of Bitcoin, is SHA-256. Once a block is added to the Blockchain, it cannot be changed, as later blocks are chained after it and the work to change the block would include

GDPR Article	Description
5. Principles relating to Processing of Personal Data	Data Controller must fulfill with the properties of lawfulness, fairness, transparency, purpose limitation, accuracy, storage limitation, integrity and confidentiality. It also must be able to demonstrate compliance with all of them.
6. Lawfulness of Processing	Data Subject must give explicit consent for the processing of his personal data for one or more specific purposes and the processing must comply the legal obligation to which the Data Controller is subject.
7. Conditions for Consent	Controller shall be able to demonstrate that the Data Subject has given the explicit consent to process his personal data. This consent can be withdrawn at any time.
12. Transparent Information	Controller shall take appropriate measures to provide any information relating to processing to the Data Subject in a concise, transparent intelligible and easily accessible form.
13. Information to be provided	Controller shall provide the Data Subject with certain information where her personal data is collected.
15. Right of Access	Allows access by the Data Subject to any personal data held by a company.
16. Right to Rectification	Ensures that any data held by Data Controllers can be rectified if inaccurate.
17. Right to Erasure	Ensures that any data held by Data Controllers can be erased if the subject desires.
18. Right to restriction of processing	Ensures that any data gathered will only be used for the purposes for which consent was given.
19. Notification Obligation	Controller must notify, to each recipient to whom the personal data have been disclosed, regarding rectification or erasure of personal data or restriction of processing.
20. Right to Data Portability	Empowers Data Subjects to take their data with them if they leave the organization.
21. Right to Object	Empowers Data Subjects to object to what how their data is being processed.
22. Automated Individual Decision Making	Prevents automated individual decision making, including profiling, of Data Subjects and selling this information to marketing firms.
25. Data Privacy by Design	Mandates that privacy and the trust it engenders must be built into the design using appropriate technical and organizational measures.

Table 3.1: GDPR Articles in a Nutshell.

redoing all the blocks after it. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. So, Blockchain Technology has the properties of immutability and persistence [71].

Payments in Bitcoin are made through structures called transactions, which can be seen as a payment made with a bank check. For a transaction to be valid, it must fulfill with some requirements, such as:

- *Non Double Spending*: inputs must be unspent outputs of a previous transaction.
- *Digital Signature*: each input must be digitally signed.
- *Authenticity*: the provided signature must match with the owner (receiver) of the inputs (transactions from where the money transferred in this new transaction come).
- *Correctness*: the sum of the values of all inputs must be higher or equal than the sum of all values of all outputs.

Digital signatures are used to verify that the user who issues a transaction is the one she claims she is. So, public cryptography is used in order to avoid that no one, except for the receiver of a transaction, could spend it in the future (use the money transferred to her as the input of another transaction). In the case of Bitcoin, the Elliptic Curve DSA (ECDSA) scheme is used to sign each transaction. The use of digital signatures adds the properties of authenticity, integrity and non-repudiation [72].

3.1.3 Smart contracts and Ethereum

Later on, Blockchain Technology separated from the currency and its potential for other financial, inter-organisational transactions was ex-

plored. Blockchain 2.0 was born, referring to applications beyond currency, in which Ethereum has been one of the most significant ones. The Ethereum Blockchain system was proposed in 2014 by Vitalik Buterin [73]. This system introduces computer programs into the blocks, currently known as Smart Contracts.

A Smart Contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. It has the property of automatically verifying the contract and execute the agreed terms, so it allows the performance of credible transactions without third parties, providing a security that is superior to traditional contract law and reducing other transaction costs associated with contracting. Smart contract transactions are traceable, transparent and irreversible [74].

In general, the use of Smart Contracts over the Blockchain Technology provides the following properties:

- *Immutability*: impossibility of changing the state of an object after it is created.
- *Persistence*: once an object is created, it cannot be removed or forgot.
- *Authenticity*: ability to prove that a user or application is genuinely who that person or what that application claims to be.
- *Integrity*: maintenance of, and the assurance of the accuracy and consistency of data over its entire life-cycle.
- *Non-Repudiation*: assurance that someone cannot deny the validity of something.
- *Traceability*: capability of keeping track of a given set of information to a given degree.
- *Transparency*: ability to easily access and work with data no matter

where they are located or what application created them.

- *Irreversibility*: once a transaction is added into the Blockchain, it cannot be turned back to the previous state.

3.1.4 The XACML standard

XACML stands for "eXtensible Access Control Markup Language". The standard defines a declarative fine-grained, attribute-based access control policy language, an architecture, and a processing model describing how to evaluate access requests according to certain rules defined in policies [75].

It is primarily an Attribute-Based Access Control system (ABAC), also known as a Policy-Based Access Control (PBAC) system, where these attributes and/or polices associated with a certain user, action or resource are used to decide if certain entity gets access to a resource in a particular way.

The XACML model differentiates the stages of access control identifying them as different points (see Table 3.2), and supports and encourages their separation as individual identities. One of the goals of this system is to promote common terminology and interoperability between access control implementations by multiple vendors.

The current workflow of this model is described in Figure 3.1:

1. A user sends a request which is intercepted by the Policy Enforcement Point (PEP).

2. The PEP transforms the user request into a XACML authorization request.

Abbreviation	Term	Description
PAP	Policy Administration Point	Manages access authorization policies.
PDP	Policy Decision Point	Evaluates an access request against the authorization policies before issuing an access decision.
PEP	Policy Enforcement Point	Intercepts user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e., access to the resource is approved or rejected), and acts on the received decision.
PIP	Policy Information Point	Source of attribute values (i.e., a resource, subject, environment).
PRP	Policy Retrieval Point	Where the XACML access authorization policies are stored, typically a database or the filesystem.

Table 3.2: XACML terminology

3. The PEP forwards the authorization request to the Policy Decision Point (PDP).

4. The PDP evaluates the authorization request against the policies in use. The policies are acquired via the Policy Retrieval Point (PRP) which is, in turn, managed by the Policy Administration Point (PAP). If indicated by the policies, the PDP also retrieves any required attribute value from underlying Policy Information Points (PIP).

5. The PDP reaches a decision (Permit / Deny / NotApplicable / Indeterminate), and returns it to the PEP.

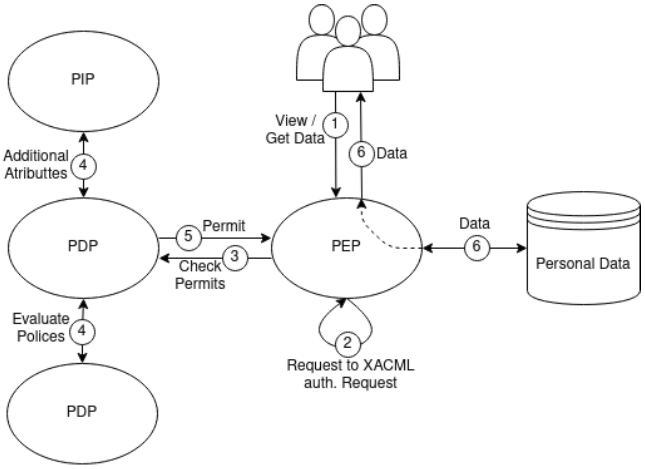


Figure 3.1: XACML architecture and general workflow

3.2 Proposal Description

In this section, we describe in detail our proposed data management platform, which is built on top of a Blockchain-based access control system, and allows the fulfillment of the GDPR to the Service Providers using it.

First, the set of requirements considered in the design of the new proposal are introduced; then, an overview of the proposed system is presented; after that, the use of Smart Contracts to represent the consents is discussed; next, both the architecture and the system's workflow are deeply explained; finally, the integration of the new proposal in a real-world setting is discussed.

3.2.1 Requirements

According to the regulations, DSs should have access to all consents given to DCs and DPs at any time. This can be easily achieved by making them freely accessible to everyone. However, making them publicly acces-

sible would imply that any non-authorized party may find a way to tamper them with. In order to prevent this undesirable situation, the proposed system must ensure that: i) all actions linked to an agreement (i.e., consent) can only be performed by authorized actors; ii) these actions must be immutable, i.e., once those actions have been validated they can not be modified or deleted; and iii) the result of validating these actions, i.e., the agreement/consent achieved by the different parties, must be persistent and it may be modified only by subsequent validated actions.

In summary, firstly, the proposed system has to provide public access to the immutable evidences that show the agreements/consents achieved between DSs and DCs regarding DS's personal data. Those agreements/-consents can only be updated by means of further validated actions which, in turn, are represented by additional immutable evidences. Secondly, the proposed system should try to mitigate identity and/or attribute disclosure risks by means of preventing adversaries from linking different agreements/consents to a certain DS. For example, if a DS grants her consent to her hometown web page, a technical web, and some webs related to her hobbies, an adversary should not be capable of re-identifying the DS, or retrieving any significant piece of information from her.

In order to achieve that, the new proposal and the evidences that it generates must fulfill the following requirements:

GDPR requirements

According to the rights and duties we have addressed in section 3.1.1, here we extract the main requirements our proposal must fulfill in order to provide GDPR compliance to Data Controllers, Data Processors and Data Subjects:

R1.1. DC needs DS's explicit consent in order to collect her personal data

(Art. 6, 12).

R1.2. Data-collecting consent is specific for a certain period of time (Art. 17) and for a specific set of personal data, so it must contain the following information (Art. 13): i) identity of the DC; ii) consent lifetime; iii) set of personal data; and iv) identities of all Data Recipients.

R1.3. DP needs DS's consent in order to process her personal data. DP also needs the DC's consent due to the fact that the processing is done under her responsibility (Art. 6, 12).

R1.4. Data-processing consent is specific for a certain period of time (Art. 17), purpose and for a specific set of personal data, so it must specify the following information (Art. 13): i) identity of the DC; ii) identity of the DP; iii) consent lifetime; iv) set of personal data; v) identities of all Data Recipients; and vi) the processing purpose.

R1.5. User consent must be available for the DS in order to control and manage her data, in such a way that she can:

R1.5.1. Modify which personal data can be collected (Art. 18).

R1.5.2. Request their data not to be longer collected (Art. 7, 18).

R1.5.3. Request their data to be erased (Art. 17).

R1.5.4. Revoke the processing consent for a specific processor at any time. (Art. 7, 21, 22).

R1.5.5. Revoke the processing consent for a specific purpose at any time. (Art. 7, 21, 22).

R1.6. User consent must be available for DCs and DPs in order to have evidence about the lawfulness of the processing in the case of an audit (Art. 7).

R1.7. DCs determine the purposes and means of the processing of personal data and, hence, they must be able to revoke the consent to process DS's personal data given to any DP.

Functional requirements

The proposed system must also fulfill a set of functional requirements in order to overcome the current literature:

R2.1. It has to be fast and efficient, becoming as lightweight as possible.

R2.2. It must provide a single resilient point of access to all consents/agreements given by a DS to process her personal data. This requirement addresses the current unfeasibility for DSs to keep track of all places in which they have given these consents.

R2.3. It must work in a distributed way, hence, consent management must not be centralized at the Service Providers' IT systems. This requirement addresses trust concerns on the traditional model in which any consent reached between a Service Provider and a DS is stored in the Service Provider's servers.

R2.4. It must allow DSs, DCs and DPs to interact with and modify the agreements reached between them by means of certain actions.

R2.5. It must keep track of all the actions related to a certain, allowing then to reconstruct that consent life-cycle.

Security and privacy requirements

Finally, for the system to be robust enough, the agreement-evidences that it provides and the actions that the different actors may do over them must fulfill a set of security requirements which are next summarized:

R3.1. Actors cannot perform an action over the agreements for which they have no permissions according to the GDPR.

R3.2. Actors must be able to identify themselves, in such a way that they can prove who they claim to be.

R3.3. No actor can falsely claim that she has not performed a certain action.

R3.4. The proposed system must provide immutable and permanent evidences of all actions carried out over the consents/agreements.

R3.5. No personal data of the DS must be stored in the system. It cannot provide any way of identifying the DS either.

R3.6. Only authorized actors must have access to DS's collected personal data.

3.2.2 Overview of the proposed architecture

In order to address the different problems and issues presented in previous sections, the proposed solution ports the current GDPR-compliance verification process to a Blockchain network, decentralizing it, and making the consents/agreements with the different Service Providers easily available for Data Subjects.

In the introduction, we have described a scenario in which a Service Provider collects personal data from a Data Subject for further processing in exchange for using a product that it provides free of charge. Performing the aforementioned port to the GDPR regulation first requires us to relate the GDPR actors (i.e., DS, DC, DR and DP) with the entities of the depicted scenario. In this way, there is a first direct relation between the GDPR's DS and the Data Subject that uses the Service Provider product.

Next, focusing on the Service Provider and the three GDPR actors left, there is a wide range of different options in that case: for example, the DC, DP and DR may be fully independent third parties and be hired by the Service Provider to perform their assigned duties; the Service Provider may simultaneously be those three actors; or it could even be something in between, this is, a Service Provider acting as the DC and the DR, and a fully independent third party acting as the DP.

Once the different main actors of the proposal have been stated, we next differentiate the two types of consents/agreements that DSs and DCs may reach:

- *Data-collection Consent*: Consent between a DS and a DC that encloses all the required information and conditions that the DC must fulfill in order to collect DS's personal data in exchange for using a service.
- *Data-processing Consent*: Consent between a DP and a DC that encloses all the required information and conditions that the DP must fulfill in order to access the data collected by the DC and process it.

The workflow of the presented framework relies on three main steps: *Consent Generation* and *Data Management* steps are briefly represented in Figure 3.2 (i.e., blue and green sections respectively). The third step, *Consent Management*, consists of the interactions of the different actors with the generated Smart Contracts in order to manage the given consent.

In the Consent Generation procedure, first, a DS requests access to certain service; then, the DC asks the DS for permission to collect her personal data in exchange for accessing the requested service. If both

agree with the conditions, a new *Data-collection Consent* is generated, the DS gets access to the offered service, and the DC starts collecting DS's personal data. Next, if a DP request the DC to process some of the collected data, the DC will create a new *Data-processing Consent* for that DP, specifying which data can be processed and for which purpose. Note that, if a *Data-Processing Consent* between a DC and a DP already exists, it can be extended with new collected data and new purposes at any time.

In the Data Management step, a DP that owns a valid *Data-Processing Consent* uses the proposed XACML extension to request the DR to supply the stored DS's personal data. Similarly, DSs can also use that method to get access to their collected data.

Finally, in the Consent Management step, all actors can interact with the different consents that have been created according to their respective rights over them. For example, a DS can revoke any previously established consent for processing or collecting her data at any time. In the same way, a Supervisory Authority can check these consents at any time in order to audit that a DC is fulfilling the legislation.

3.2.3 Smart contracts

In order to fulfill the requirements indicated in 3.2.1, the new proposal implements the *Data-collection* and *Data-processing* consents introduced above by means of Smart Contracts. In particular, the following two Smart Contracts are defined:

- *Collection Consent Smart Contract*: It embodies the authorization that a DS gives to a DC to collect her personal data.

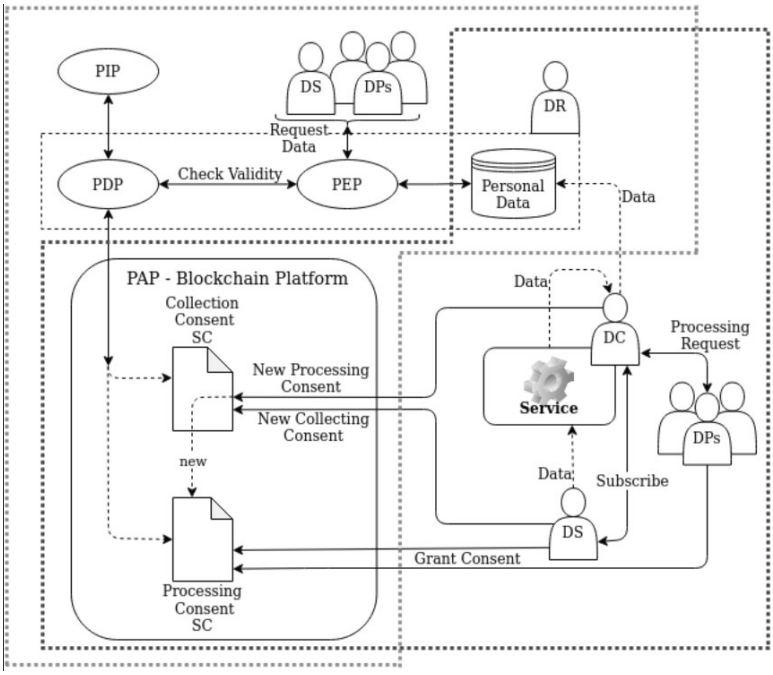


Figure 3.2: Consent Management System architecture

- *Processing Consent Smart Contract*: It contains the authorization that a DC and the DS give to a DP to process a subset of DS's personal data for a specific purpose.

Collection Consent Smart Contract

This Smart Contract consists of a single class that holds:

- Identities: public keys of the Data Subject, Controller and Recipients (*dataSubject*, *dataController* and *dataRecipients* arguments). These public keys are used as pseudonyms by the different actors. In this way, a DS may use a different key pair for each DC in order to improve her anonymity.
- Data categories: the data categories that can be collected are stored

in the *Data* argument. Note that this argument does not contain the data itself.

- Consent lifetime: period of time during which the data can be collected and stored (*beginningDate* and *expirationDate* arguments).
- Default purposes: list of processing purposes for which a DS allows the processing of her persona data (*defaultPurposes* argument).
- Blacklist: list with all the DPs that the Data Subject has revoked her consent to process her personal data (*processorsBlacklist* argument).
- Processing Consents: link to all the *Processing Consent Smart Contracts* that depend on this consent (*processingConsentContracts* argument);
- Valid flag: flag that indicates the validity of the Smart Contract (*valid* argument).
- Erasure flag: flag that, when it is set to true, compels the DC to erase all the personal data collected from the DS (*erasure* argument).

When a new *Collection Consent Smart Contract* is instantiated, the DS must provide the following arguments: i) *dataSubject*; ii) *dataController*; iii) *dataRecipients*; iv) *defaultPurposes*; v) *beginningDate*; and vi) *expirationDate*. Automatically, *valid* argument is set to one while *erasure* argument is set to zero.

Once the *Collection Consent Smart Contract* instance is created, the different actors may interact with it by means of the following methods:

- *grantConsent()* and *revokeConsent()* methods are used by the DS to modify the *valid* flag, enabling or disabling the SC.

- *verify()* method checks whether the Smart Contract is valid or not. In order to be valid, the *valid* flag value must be true and the linked lifetime period must have not expired.

- *newPurpose()* method is used by the DC to create a consent for a DP to allow it to process the personal data of the bounded DS. More specifically, two different situations apply: i) if in the Smart Contract already exists a *Processing Consent Smart Contract* instance linked to this DP, this method adds a new processing purpose to that contract; and ii) otherwise, a new *Processing Consent Smart Contract* related to the DS, the DC, and the new DP is created and its address is stored in the *processingConsentContracts* argument. Note that this method requires the *Collection Consent Smart Contract* to be valid (i.e., the *verify()* method must return "true").

- *modifyData()* method is used by the DS to update the *Data* argument. If the new data argument is more restrictive than the previous one (allowing the collection of less personal data than before), those *Processing Consent Smart Contracts* that state the process of data which cannot be collected anymore will also have this argument updated. This is achieved by using the *modifyData()* method of the corresponding *Processing Consent Smart Contract*.

- *eraseData()* method is used by the DS to set the *erasure* flag to "true". This compels the DC to erase all personal data collected from that DS.

- *revokeConsentPurpose()* method is used by the DS to revoke the processing consent for all processors that has requested to process DS's personal data for a specified purpose; and remove that purpose from the *defaultPurposes* list (if included). This method requires as an input argument the purpose for which its processing consent will be revoked.

- *revokeConsentProcessor()* method is used by the DS to revoke all processing consents given to a Data Processor; and add the specified Data Processor to the *processorsBlacklist* list (if she is not included).

Processing Consent Smart Contract

The DC responsible of a *Collection Consent Smart Contract* creates a *Processing Consent Smart Contract* for each processor that requests to process DS's personal data. This Smart Contract holds different policies for each processing purpose, specifying which subset of data can be processed and for which period of time.

This Smart Contract consists of a single class and holds the following information:

- Identities: the identities of the Data Subject and controller plus the address of the processor (*dataSubject*, *dataController* and *dataProcessor* arguments).
- Data: which data can be processed (*Data* argument) and for which specific period of time (*beginningDate* and *expirationDate* arguments).
- Purpose: the specific purpose for which the data will be processed (*purpose* argument).
- Valid Flag: flag that indicates if the Smart Contract is valid (*valid* argument).

This contract encapsulates the *Data, beginningDate, expirationDate, purpose*, and *valid* arguments in blocks. There are as many blocks as processing purposes the bounded DP has requested.

There are two main cases when a DP requests to process DS's personal data for a certain purpose: i) it is the first time that the DP request to process the data, so the DC needs to instantiate a new *Processing Consent Smart Contract*; or ii) there is an existing *Processing Consent Smart Contract* for that specific DP, so the DC just has to add a new processing purpose to the that Smart Contract.

When a new *Processing Consent Smart Contract* is instantiated, the DC indicates the *dataSubject, dataController, dataProcessor* arguments, and all the information related to the processing purpose: the processing purpose (*purpose* argument) for which the data will be processed, the personal data to which this purpose is applicable, and the duration of the consent (*beginningDate* and *expirationDate* arguments). On the other hand, when a DP wants to add a new processing purpose, the DC only adds an additional block of information.

Regarding how the different actors may interact with this Smart Contract, similarly to the *Collection Consent Smart Contract*, the *Processing Consent Smart Contract* implements the *grantConsent()*, *revokeConsent()*, *modifyData()* and *verify()* methods. Those methods been already explained above.

3.2.4 Architecture and procedures

We next explain in detail the architecture of the proposed system and the different procedures that are used to generate and manage the consents and the data.

Collection Consent Smart Contract generation and data collection

Figure 3.3 depicts the workflow followed by the new system when a new consent for data collection must be generated and the collection of the corresponding personal data is started. In particular, the following steps apply:

1. A DS request access to certain service provided by a DC.
2. The DC emits a data collection consent request. This request specifies which DS's personal data can be collected while using the aforementioned service and for how long this data can be kept.
3. If the DS agrees with DC's conditions, she generates a new *Collection Consent Smart Contract*. Otherwise, both parties must renegotiate the collecting polices, or the DS may finally refuse using the service offered by the DC.
4. If the *Collection Consent Smart Contract* fulfills DC's requirements, this entity validates the Smart Contract. As a result, DS gets access to DC's service and DC starts collecting DS's personal data.
5. This personal data collected by the DC is stored off-chain by the DR. In this way, there is no personal information stored in the Smart Contracts. They only hold the information regarding which categories of data can be collected. Note also that the DR can be the DC itself or an additional third party.

During the aforementioned procedure, DS generates a new *Collection Consent Smart Contract*. In the new proposal, generating Smart Contracts and interacting with them is addressed by means of transactions

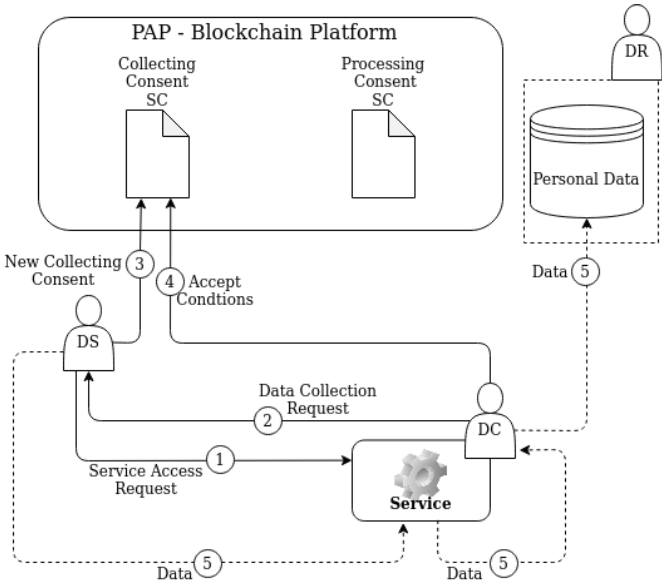


Figure 3.3: Collection Consent Smart Contract generation and data collection

which are digitally signed using public-key cryptography and the corresponding pair of secret/public keys. In order to improve DS's anonymity, for each new generated *Collection Consent Smart Contract*, the DS uses a new pair of keys. In this way, the chance of linkage attacks is mitigated. All the used key pairs are stored in a key storage system provided by the proposed service and directly controlled by the user.

Data processing request and Processing Consent Smart Contract generation

Once the *Collection Consent Smart Contract* is valid and DS's personal data has been collected and stored, a DP can request to process that data for certain purposes. Figure 3.4 depicts this procedure. In particular, the following steps apply:

1. A DP performs a data processing request to the DC regarding the personal data gathered from the DS.

2. The DC generates a new *Processing Consent Smart Contract* for that specific DP. If a *Processing Consent Smart Contract* bounded to DP already exists, DC only adds a new processing purpose to the actual Smart Contract. Whatever the case is, in the contract, the DC specifies which DS's personal data can be processed, the processing purpose and the period of time of this consent

3. The DC asks the DS to agree on the aspects of the new processing purpose and to validate the Smart Contract.

4. If the DP agrees also on the conditions of the new generated processing purpose, it can then request DS's personal data to the DR in order to process it.

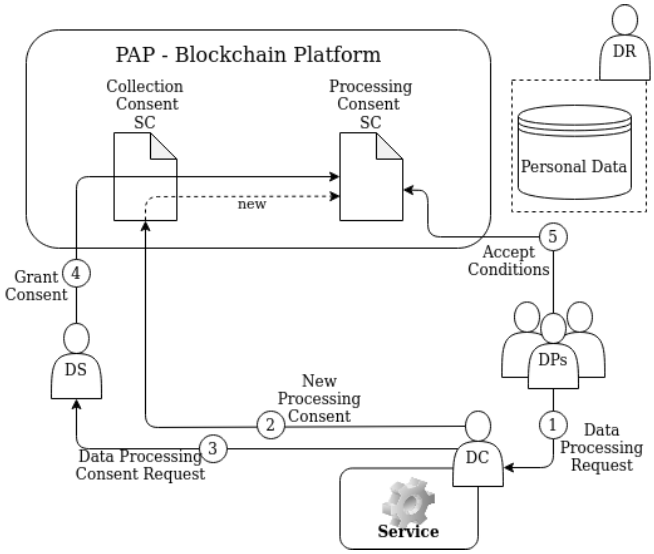


Figure 3.4: Data processing request and Processing Consent Smart Contract generation

Note that a *Collection Consent Smart Contract* may have a list of default purposes (i.e., *defaultPurposes* argument). In such manner, if a DP performs a request to process DS's personal data for a purpose which is already included in the *Collection Consent Smart Contract*, and the DP is not included in the *blacklist*, it is not needed that the DS validates the *Processing Consent Smart Contract*, therefore, steps 3 and 4 of this procedure are not required.

Data access and retrieval

Once the DS's personal data is collected, stored and guarded by the DR, different actors may be interested in accessing and retrieving it for different purposes. For example, a DP may be willing to process that data, or the DS, to whom the data belongs, may be interested in checking her own collected data. Whatever the case is, the procedure used to access DS's personal data is based on a XACML extension that enforces the access control over the protected data. Figure 3.4 shows this procedure. In particular, the following steps apply:

1. When an entity sends a data access request to the DR, it is intercepted by the Policy Enforcement Point (PEP) of the XACML-based access control system.

2. The PEP converts the data access request into a XACML authorization request that is forwarded to the Policy Decision Point (PDP).

3. The PDP evaluates the authorization request against the policies (i.e, Smart Contracts) it is configured with. Depending on who the entity requesting the data is, the *Collection Consent Smart Contract* of a *Processing Consent Smart Contract* will be used. In particular,

if the request is performed by the DS, the *Collection Consent Smart Contract* will be used (step 3a); on the contrary, if the request is performed by a DP, the *Processing Consent Smart Contract* linked to that DP will be used (step 3b). Smart contracts are acquired via the Policy Administration Point (PAP) by searching them in the Blockchain.

4. If needed, the PDP uses any underlying Policy Information Points (PIP) available to retrieve any additional attribute value which could be relevant to reach a final decision regarding the data access request.

5. Finally, the PDP reaches a decision (Permit / Deny / NotApplicable / Indeterminate), and returns it to the PEP, who will then give access or not to the requested data.

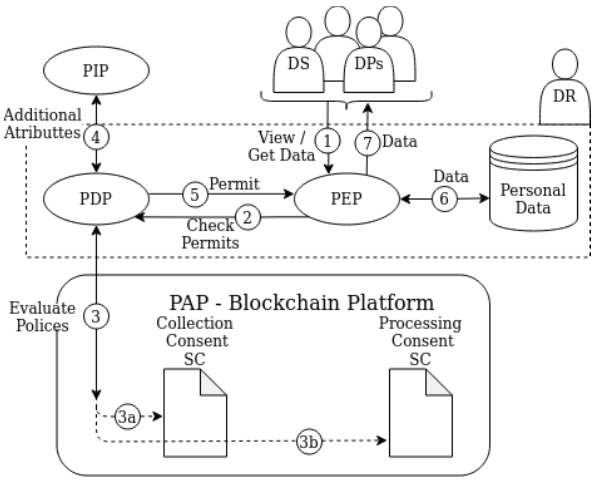


Figure 3.5: Data access and retrieval

Consent management

According to the requirements described in section 3.2.1, DSs have rights over their collected data, thus, at a certain point of time, they may be willing to modify their given consents, revoke them, request the collected data to be erased, etc. In the same way, DCs also have the right to revoke a given processing-consent to a DP, as the data processing is done under their responsibility. In the following, the set of actions that can be done over the different generated Smart Contracts are described:

- *Revoke Consent:* A DS can revoke her consent for collecting and processing her personal data at any time. In order to revoke a processing consent given to a certain DP, the DS must use the *revokeConsent()* method implemented in the *Processing Consent Smart Contract*. This method can also be used by the DC in charge. Similarly, for revoking a data-collecting consent given to a certain DC, the DS must use the *revokeConsent()* method implemented in the *Collection Consent Smart Contract*. When this method is called, the Smart Contract performs recursive calls to the *revokeConsent()* method of all the *Processing Consent Smart Contract* which are bound to it. Therefore, when a data-collecting consent is revoked, all the linked data-processing consents are also revoked.

- *Erase Data:* DSs can use the *Collection Consent Smart Contract*'s *eraseData()* method to remove their collected personal data. More specifically, this method sets the *Erasure* flag to "true", forcing the DC to erase the collected personal data without undue delay. The corresponding DC is fully responsible of this action, therefore, if the data is stored using a third party as DR, the DC must guarantee

that the data destruction process is performed correctly. Note that running this method implies calling the revoke consent method too.

- *Modify Data:* DSs can modify at any time which of their personal data can be collected or processed for certain purposes by using the *Collection Consent and Processing Consent Smart Contracts*' *modifyData()* method. In particular, when a DS uses the *modifyData()* method of a *Collection Consent Smart Contract* to make the subset of personal data that can be collected less permissive than the previous one, in turn, the attached *Processing Consent Smart Contracts* get a recursive call to their *modifyData()* method to modify their respective subsets of personal data that can be processed.

Consent audit

A main objective of this new proposal is to provide Service Providers with a mechanism that enables them to demonstrate that they have the proper consents to collect and process DS's data in the case of an audit. These audits are carried out by Supervisory Authorities (SAs), this is "independent public authorities responsible of monitoring the application of the GDPR, in order to protect the fundamental rights and freedoms of natural persons in relation to processing and to facilitate the free flow of personal data within the Union" [Art. 51].

The proposed system fulfills this objective by means of three main points: i) all the agreements reached between DSs, DCs, and DPs are declared inside Smart Contracts; ii) Smart Contracts are published in the Blockchain, hence, they are publicly available; and iii) due to the Blockchain's persistence and immutability properties, Supervisory Authorities are able to check the life-cycle of a certain consent/agreement in the case

it is required.

For example, if a SA suspects that a certain DC is collecting more personal data than she is allowed or she is collecting that data without consent, the SA may follow the next procedure (see Figure 3.6(a)):

1. The SA checks which data is being collected by the DC. This is achieved by requesting this information to the corresponding DR.

2. The SA looks for the related *Collection Consent Smart Contract* in the Blockchain and verifies that it exists and that it is currently valid (using the Smart Contract's *verify()* method).

3. If the Smart Contract is not currently valid, the SA checks the life-cycle of the agreement in order to validate whether she had a previous consent that allowed the collection of the data when it was still valid.

4. If the *Collection Consent Smart Contract* is valid, the SA checks whether the DC has collected more data than it was authorized to. This is done by matching the stored data by the DR with the values stored in the *Collection Consent Smart Contract*'s *data* field.

5. Finally, if the SA detects that the DC has collected personal data without consent or that this entity is keeping data for which it does not have a valid consent any more, the SA will impose the corresponding administrative fines.

Similarly, if a Supervisory Authority suspects that a certain DP is misbehaving when processing some personal data, this entity may follow the next procedure (see Figure 3.6(b)):

1. The SA checks which data is being processed by the DP and for which purpose.

2. The SA looks for the corresponding *Processing Consent Smart Contract* in the Blockchain and verifies that it exists and that it is currently valid (using the Smart Contract's *verify()* method).

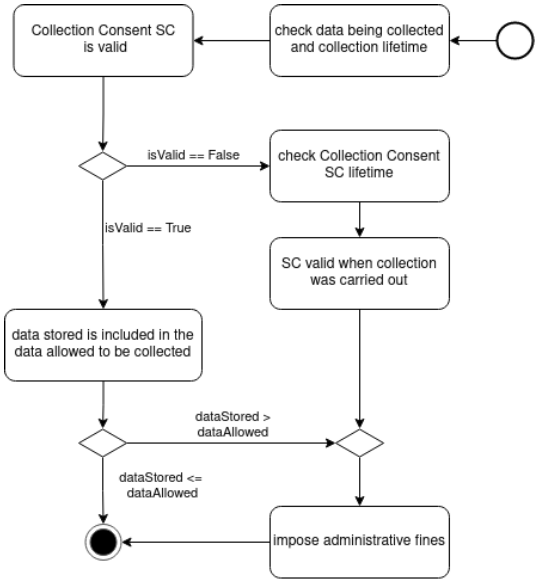
3. If the Smart Contract is not currently valid, the SA checks the lifecycle of the agreement in order to validate whether the processing process was carried out when it was still valid.

4. If the *Processing Consent Smart Contract* is or was valid when the processing was carried out, the SA checks whether the DP has processed more data than it was authorized to. This is done by matching the processed data with the values stored in the *Processing Consent Smart Contract*'s *data* field.

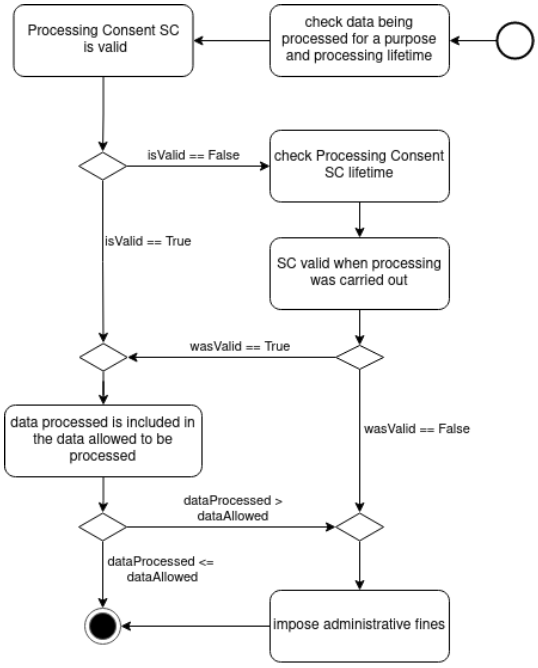
5. Finally, If the SA detects that the DP has processed personal data without the proper consent, it will impose the corresponding administrative fines to this misbehaving entity.

3.3 Discussion

This section provides an analysis and discussion of the described platform. This includes analyzing GDPR-compliance, functionality, security and privacy, performance and, finally, studying the costs that the use of Smart Contracts will introduce.



(a) Data Collection Consent Audit Process



(b) Data Processing Consent Audit Process

Figure 3.6: Consent audit process

3.3.1 GDPR-Compliance analysis

In section 3.2.1, we defined the main requirements that the proposed system should successfully fulfill in order to be GDPR-compliant. Now, in this section, those requirements are put to the test. The provided discussion consists of four propositions with several claims that support their fulfillment.

Proposition-1: A data-collection consent represents the agreement between DS and DC regarding the collection of DS's personal data

When a DS requests access to a certain service, a data-collection consent contract is created between this entity and the DC that represents the service provider. This contract specifies: i) which personal data can be collected; ii) the period of time it can be kept; iii) the identity of the DC; and iv) the identities of the DRs. This contract is valid as proof of the DS's explicit consent given to the DC to collect her personal data.

This proposition is directly related to the GDPR requirements R1.1 and R1.2, which state that a DC must have the DS's explicit consent in order to collect her personal data. It is supported by the following claim.

Claim 1. *A new Collection Consent Smart Contract is created every time a DS requests access to a service which collects her personal data.*

Proof. When a DS request access to a certain service, the service provider (i.e., the DC) may request that some of DS's personal data be collected in exchange. In this case, the proposed system allows the DS to generate a new Smart Contract that specifies which personal data the DC will be able to collect and for how long it will be kept. The Smart Contract also holds

the identities of the DC and all the DRs that will store the collected data. In this way, the created Smart Contract represents the explicit consent given by the DS to a certain DC to collect her personal data. □

Proposition-2: A data-processing consent represents the agreement between DS, DC, and DP regarding the processing of DS's personal data

When a DP request to process DS's collected personal data, the DC in charge of collecting that data can create a new data processing consent contract specifying: i) which of DS's personal data can be processed and for which purpose; ii) the period of time it can be kept; and iii) the identities of the DC and the DPs. This contract requires that the DS gives her explicit or implicit consent to be valid.

This proposition is directly related to the GDPR requirements R1.3 and R1.4, which state that a DP must have the consent of the DSs in order to process their personal data. It is supported by the following two claims.

Claim 2. *A new Processing Consent Smart Contract is created every time a new DP requests to process some DS's personal data collected by a DC. A reference to this new smart contract is stored into the Collection Consent Smart Contract that encompasses the whole data management process.*

Proof. When a DP requests to process DS's personal data for the first time to a DC, the DC creates a *Processing Consent Smart Contract* that holds the consent for processing a subset of DS's personal data for a specific purpose and period of time. This Smart Contract is deployed through the *Collection Consent Smart Contract* that was previously agreed between the DS and the DC by using the *newPurposeConsent()* method. This

method adds the address linked to this new Smart Contract into the *Purposes* list of the *Collection Consent Smart Contract*. Once a *Processing Consent Smart Contract* binding a certain DP with the *Collection Consent Smart Contract* exists, future data processing requests made by the same DP will be added to the existing Smart Contract. □

Claim 3. *A Processing Consent Smart Contract requires DS's explicit or implicit approval in order to be valid.*

Proof. When the DC creates or adds a new processing purpose in a *Processing Consent Smart Contract*, that specific processing purpose may be already recorded in the *defaultPurposes* list of the linked *Collection Consent Smart Contract*. If that is the case, the DP gets DS's implicit consent for processing her personal data for that specific purpose. Otherwise, the DP requires that the DS explicitly approves the data processing request by running the *grantConsent* method of the *Processing Consent Smart Contract*. □

Proposition-3: A DS can interact with her data-collection and data-processing consents according to the GDPR's rights

The proposed system allows DSs to interact with the generated data-collection and data-processing consents, which are backed by means of smart contracts, in the following ways:

- A DS may request to remove all her collected personal data at any time (Art. 17 Right to erasure).
- A DS may limit which of her personal data can be collected, which can be processed, and for which purposes (Art. 18 Right to restriction of processing).

- A DS may revoke her consent at any time (Art. 21 Right to Object).

According to the presented proofs in Claims 1 and 2, data-collection and data-processing consents are stored into Smart Contracts. Furthermore, these Smart Contracts implement a set of methods that allow DSs to interact with them allowing to exercise their rights over their personal data.

This proposition is directly related to the GDPR requirements R1.5.1 and R1.5.5, which state that DSs must be able to apply their rights over their personal data anytime. It is supported by the following claim.

Claim 4. *Collection and Processing Consent Smart Contracts implement a set of methods that allow DSs to exercise their rights over the reached agreements.*

Proof. As shown in sections 3.3.1 and 3.3.2., *Collection and Processing Consent Smart Contracts* implement a set of methods allowing DSs to:

- Revoke an specific data-collection or data-processing consent at any time (*revokeConsent()*).
- Modify which personal data can be collected or processed (*modifyData()*).
- Request the DC to erase their collected personal data (*eraseData()*).
- Revoke the processing consent for an specific purpose and remove it from the *defaultPurposes* list (*revokeConsentPurpose()*).

□

Proposition-4: A DC can revoke at any time all the data processing consents that this entity has issued

According to the presented proof in Claim 2, when a DP request to process DS's personal data, the DC that has collected that data generates a processing consent which is stored into a Smart Contract. In the same way, at anytime, the DC can revoke her consent given to the DP to process DS's data and overrule the previous agreement.

This proposition is directly related to the GDPR requirement R1.7, which states that DC must be able to revoke their consents to DPs to process DS's personal data anytime. It is supported by the following claim.

Claim 5. *Processing Consent Smart Contract implements an specific method that allows the DC in charge to revoke her consent for processing the collected data.*

Proof. The DC that has generated a certain *Processing Consent Smart Contract* may use the *revokeConsent()* method to revoke its consent. □

3.3.2 Functionality analysis

Functional requirements R2.2, R2.3 and R2.4 presented in section 3.2.1 are studied in this subsection. Remaining functional requirements R2.1 and R2.5 will be tackled in further subsections. The following discussion consists of a set of propositions, each one addressing a certain considered requirement. Each proposition has a set of claims to support its fulfillment.

Proposition-5: A DS can access all the data-collection and data-processing consents that she has agreed from a single resilient access point

A DS can use the asymmetric key pairs generated and stored in the key storage system to find and access all her published Collection Consent Smart Contracts and all the corresponding Processing Consent Smart Contracts. In this way, the proposed system allow DSs to find and retrieve all the collection and processing consents that they have given. This proposition is directly related to the functional requirement R2.2, which states that the proposed scheme must provide a single resilient point of access to all the consents that the DS has given. In this case, this access point is the Blockchain. This proposition is supported by the following four claims.

Claim 6. *A public Blockchain is used to store all the data-collection and data-processing consents generated by the proposed system. Public Blockchains are resilient by design, provide access transparency, and individuals can interact with them by means of asymmetric key pairs.*

Proof. Each DS has a key storage system that generates and stores all the asymmetric key pairs ($SK_{DS_{i,j}}$,$PK_{DS_{i,j}}$) required to interact with the public Blockchain. Blockchains are fully distributed and, hence, resilient by design. Moreover, they work as a single access point by means of the access transparency they provide to users. □

Claim 7. *All the data-collection and data-processing consents which are agreed between the different parties are stored in the public Blockchain by means of public and immutable smart contracts.*

Proof. In the proposed system, data-collection and data-processing consents are represented by *Collection Consent Smart Contracts* and *Processing Consent Smart Contract* respectively. Smart contracts are de-

ployed into the public Blockchain by means of transactions. Transactions are immutable records in the Blockchain digitally signed by means of a public-key algorithm with signing capabilities and an asymmetric secret key $SK_{DS_{i,j}}$. No party can change or tamper with a transaction after it has been recorded into the shared ledger. □

Claim 8. *A DS can find and access a certain smart contract published on the Blockchain by knowing the corresponding public key $PK_{DS_{i,j}}$ of the asymmetric secret key $SK_{DS_{i,j}}$ which was used to perform its deployment.*

Proof. All Smart Contracts deployed on the Blockchain are public and anyone can access them. As stated previously, Smart Contracts are published by means of digitally signed transactions (see Claim 7). DSs employ asymmetric key pairs ($SK_{DS_{i,j}}$, $PK_{DS_{i,j}}$) to interact with the Blockchain. In particular, the private key $SK_{DS_{i,j}}$ is used to digitally sign a transaction to be deployed, while the linked public key $PK_{DS_{i,j}}$ can be used to find the corresponding smart contract on the ledger. □

Claim 9. *Collection Consent Smart Contracts are bound to Processing Consent Smart Contracts by means of the "Purposes" field of the former. In this way, when a certain Collection Consent Smart Contract is found an retrieved, it is straightforward to find and retrieve all the Processing Consent Smart Contracts which are bound to it.*

Proof. Collection Consent Smart Contracts keep track of all *Processing Consent Smart Contracts* bound to them. An entity that retrieves a certain *Collection Consent Smart Contract* is then able to read the *Purposes* list where the addresses of the *Processing Consent Smart Contracts* are stored. All *Processing Consent Smart Contracts* are published in a public Blockchain, therefore, by knowing their addresses, they can be straightforwardly found and accessed. □

Proposition-6: The proofs linked to the consents agreed between DSs and DCs/DPs are stored in a fully distributed way

According to the proofs presented in Claims 1 and 7, all the agreements made between DSs, DCs and DPs are encoded into Smart Contracts that are then published on a public Blockchain. The Blockchain technology is hosted by a set of anonymous and fully distributed nodes. In this way, all these agreements are not kept in a unique central server. Instead of that, they are shared between all the nodes that maintain the network. This proposition is directly related to the functional requirement R2.3, which states that the proposed scheme must work in a distributed way.

Proposition-7: DSs, DCs, and DPs can interact with all the published data-collection and data-processing consents

All agreements made between DSs and DCs/DPs are encoded into Smart Contracts which are then published in a public Blockchain. These Smart Contracts implement methods which can be invoked by the different entities by using transactions digitally signed transactions on the Blockchain. This proposition is directly related to the functional requirement R2.4, which states that the proposed scheme must allow DSs, DCs and DPs to interact with and modify all the agreements already made between them. This proposition is supported by the following claim.

Claim 10. *Entities can interact with the published agreements by sending digitally signed transactions to the Blockchain.*

Proof. As explained in Section 3.3.1, Smart Contracts provide methods to interact with them. By design, interacting with a method offered by a smart contract deployed into a public Blockchain requires the use digitally signed transactions. In particular, to run a certain method, the calling

entity must publish a digitally signed transaction on the Blockchain that, in consequence, will be recorded in the ledger and become immutable. □

3.3.3 Security and privacy analysis

This section focuses on studying the security and privacy requirements of the provided system. Moreover, the functional requirement R2.5 is also addressed here. Like in previous sections, the discussion is organized as a set of propositions, where each proposition may have several claims to support its fulfillment.

Proposition-8: Only authenticated and authorized entities can interact with the consents made between DSs, DCs, and DPs

Entities can interact with the Smart Contracts by using transactions. These transactions are digitally signed and the Smart Contract stores the identifying public keys of the involved entities. As a result, smart contracts can verify whether the entity that is willing to perform a certain action has the proper rights, according to her role and the GDPR rules, to run the corresponding method.

This proposition is directly related to requirement R3.1, which states that an entity must not be able to carry out an action over an agreement for which it has not permission; and to requirement R.3.2, which states that entities must be properly authenticated, which, in turn, is achieved by digitally signing the Blockchain transactions. It is supported by the following three claims.

Claim 11. *Smarts contracts safely store the public keys of all the entities involved in the represented agreements.*

Proof. When a new *Collection Consent Smart Contract* representing a certain data collection agreement is created, the public keys of the DS

($PK_{DS_{i,j}}$) and the DC (PK_{DC_j}) involved are stored in its internal state. In the same way, when a new *Processing Consent Smart Contract* is created, the public keys of the DS, DC, and the corresponding DP (PK_{DP_k}) are also stored in its internal state. □

Claim 12. *Each public key belongs to a unique entity and cannot be tampered.*

Proof. Each public key belongs to an asymmetric public/secret key pair that is uniquely linked to a specific entity. For example, the key pair of DS i used to sign the *Collection Consent Smart Contract* with the DC j is the pair ($PK_{DS_{i,j}}$, $SK_{DS_{i,j}}$). Only the party that knows the secret key $SK_{DS_{i,j}}$ of the aforementioned key pair can prove to be the owner of the corresponding public key. Moreover, due to the fact that the transactions recorded on the Blockchain are immutable, public keys stored in smart contracts cannot be tampered with. □

Claim 13. *Only authorized entities can perform actions over a smart contract.*

Proof. Each method of a smart contract has a set of authorized roles that limit who can perform that specific action. In this way, before running any action, the smart contract verifies whether the corresponding entity has the proper rights to perform that action according to its role. This process is twofold: first, the smart contract checks that the key pair which is being used to sign the transaction of the specific method call (see Claim 10) matches the stored public key; second, the smart contract checks that this public key is linked to a role that grants access to the required action. □

Proposition-9: An entity that participates in the system cannot falsely claim that they have not performed a certain action (i.e., non-repudiation)

As it has been explained previously, actions carried out over smart contracts are done by means of digitally signed transactions (see Claim 10). Digital signatures, if they are generated correctly using a sound asymmetric cryptosystem and the cryptographic material is kept safe, are assumed to ensure the non-repudiation property (explained in Section 2.3). In this way, a DS may digitally sign a transaction by means of her secret key $SK_{DS_{i,j}}$ and a public-key algorithm with signing capabilities, for example, the ECDSA algorithm, which is included in the current Digital Signature Standard (DSS) issued by NIST[1]. ECDSA is assumed to be secure provided that the signer uses a proper random nonce to generate the corresponding digital signature[76]. Regarding the security of the signing key, it is only known by the DS, who is responsible of keeping it secure in the key storage system in order that no adversary can retrieve it. According to this, an entity willing to perform a certain action over a smart contract must digitally sign a Blockchain transaction linked to that specific action. Due to the fact that digital signatures ensure the non-repudiation property, it can be stated that no entity can falsely claim that it has not performed a certain action.

This proposition is directly related to requirement R3.3.

[1]Digital Signature Standard (DSS): `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`

Proposition-10: All the consents and all the actions applied to them that may modify their state are generated as permanent and publicly accessible proofs that allow the public tracking of all events

As it has been explained previously, all agreements between DSs, DCs and DPs are encoded into Smart Contracts which are published on a public Blockchain. According to the persistence property of this technology, explained in Section 3.1.3, for extension, these agreements become permanent and publicly accessible. So, according to proofs presented in Claim 10, all interactions with agreements are registered on a public Blockchain by means of transactions, which make them permanent and publicly accessible too. This fact allows any entity to reconstruct the life-cycle of any agreement.

This proposition is directly related to requirement R3.4, which states that the system system must provide immutable evidences of all the events which have been carried out. Moreover, this proposition is also directly related to the functional requirement R2.5, which states that the system must keep all actions related to a consent allowing any external entity to reconstruct that consent life-cycle.

Proposition-11: No entity can link a certain consent agreement with a certain DS (i.e., anonymity)

No DS's personal information is stored in the Smart Contracts so it is unfeasible to directly re-identify a certain DS by means of the agreements generated by the proposed system. However, an identity disclosure attack may be possible by using multiple generated agreements as quasi-identifiers. In order to avoid this issue, the proposed scheme uses a new asymmetric key pair each time a DS sets a new agreement with a DC. As a result of that, an adversary cannot link different agreements together and

she cannot obtain any additional information that allows her to re-identify the DS.

This proposition is directly related to requirement R3.5., which states that the system must not publicly store DSs' personal data neither provide any way to re-identify them. It is supported by the following three claims.

Claim 14. *DS's personal data is never stored in the Smart Contracts generated by the proposed system.*

Proof. The *Collection and Processing Consent Smart Contracts* employed in the proposed system only store the public key of the asymmetric key pair used to digitally sign the corresponding Blockchain transaction. Those Smart Contracts do not store any piece of personal data belonging to the involved DS. □

Claim 15. *The agreements made between DCs and DSs are considered to be Quasi-Identifiers.*

Proof. The agreements between a DS i and a DC j may contain information about DS's interests (e.g., location, political views, etc.). On the one hand, it is not possible to re-identify a certain DS just by analyzing a single agreement made by her. On the other hand, a capable adversary may be successful re-identifying a certain DS if she has access to multiple agreements made between this DS and different DCs. Due to this situation, the agreements made by DSs are considered to be quasi-identifiers. □

Claim 16. *The proposed scheme uses a new asymmetric key pair each time a DS sets a new agreement with a DC.*

Proof. As explained in Claim 6, DSs use a key storage system to generate and store all the crytographic material required to interact with the public

Blockchain. By using this key storage system, a DS may generate a new asymmetric key pair whenever she wants. In particular, the proposed scheme generates a new pair for each new agreement with a DC. □

Proposition-12: Only authorized entities can obtain DS's collected personal data

The DR is the entity in charge of managing the access requests to the protected personal data. This entity uses a XACML extension to drive all access control process. By using this extension, the DR, before granting access to the requesting actor, verifies that the data requester owns the proper access rights on the Blockchain. Those access rights are encoded in the Blockchain by means of smart contracts which represent the agreements made between DSs, DCs and DPs (see Proposition-6).

This proposition is directly related to requirement R3.6., which states that the system must guarantee that only properly authorized actors should have access to DS's collected personal data. It is supported by the following claim.

Claim 17. *The DR manage the access to the DSs' collected data by means of the proposed XACML extension.*

Proof. As shown in the architecture of the proposed scheme, the DR is the entity in charge of keeping safe the personal data collected by the DC and managing any access to it by means of the proposed XACML extension. In order to achieve this, the XACML extension uses the Blockchain Technology as the *Policy Administration Point (PAP)* in the access control architecture. Then, when the *Policy Enforcement Point (PEP)* receives a data access request from an entity, it asks the *Policy Decision Point (PDP)* for an grant/deny access decision. The data access request contains a digitally signed temporal proof using the entity's secret key SK_e

that validates the right ownership of the corresponding public key PK_e. Next, the PDP uses the PAP to get from the Blockchain the corresponding consent smart contract (i.e., a *Collection Consent Smart Contract* if the requesting entity is a DS/DC or a *Processing Consent Smart Contract* if it is a DP). After that, in a two-step operation, the PDP first checks whether the obtained smart contract contains the corresponding PK_e; and, finally, it ascertains whether the requesting entity has the proper rights to access the protected data. □

3.3.4 Performance analysis

Functional requirement R2.1 states that the new proposal must be fast and efficient. In particular, it should be more lightweight than existing ones. In order to validate this requirement, the new scheme is compared with those solutions in the literature that: i) are properly documented so performing a comparison is possible; and, ii) whose main objectives and functionality are similar enough to the goals considered in this paper.

Table 3.3 focuses on the functionalities offered by the current schemes in the literature that effectively consider the GDPR compliance. In particular, this table summarizes whether the different aspects of the GDPR regulation are tackled by each proposal or not. As it can be seen, only two works differentiate between data-collection and data-processing consents. These are Truong et al. and Nessie et al. Next, from these two, only the former fulfills all the functionalities that we have established as essential in this paper. Regarding Nessie et al., it is not clear whether the collected data can be erased or not, due to the fact that the data is sent to each processor who has the consent to process it and no further explanations are given in this regard. For this reason, the corresponding property is left as unknown in this case.

Focusing now on the performance analysis, it is worth mentioning that

Functionality	**Our Proposal**	**Truong et al. [7]**	**Nessie et al. [11]**	**Wirth and Kolain [6]**	**Barati et al. [26]**	**Wang et al. [12]**
Grant Collection Consent	✓	✓	✓	X	X	X
Revoke Collection Consent	✓	✓	✓	X	X	X
Grant Processing Consent	✓	✓	✓	✓	✓	✓
Revoke Processing Consent	✓	✓	✓	X	X	✓
Access Data	✓	✓	✓	✓	✓	✓
Modify Data	✓	✓	✓	X	✓	✓
Delete Data	✓	✓	?	X	X	✓
Update Policies	✓	✓	✓	X	X	✓

Table 3.3: Comparison of functionalities with State of the Art proposals.

most of the schemes in the literature do not provide an experimental performance analysis at all, or, if they do, it differs considerably from the study that we have carried out on the new proposal. Therefore, in order to fairly compare the performance of our scheme with the rest of the solutions present in the literature, our performance analysis focuses only on the three most relevant operations among the eight indicated above. More specifically, we compare the number of steps required to perform each one of these three main operations.

The three integral operations being considered are: i) the "get processing consent" process, from which a DP obtains the authorization of the DS and the DC to process DS's personal data; ii) the "revoke processing consent" operation, which is used by a DS or DC to revoke a processing consent previously given; and iii) the "access data" method used by the DP and the DS to get access to the collected data. These three procedures have been selected due to the fact that they are covered by most

systems in the literature and, also, these are the procedures explained with a significantly better level of detail in those proposals.

Note that, in our work, we have stressed the importance of the data-collection consent which is established between a DS and a DC, a contract that must be valid in order to start collecting DS's personal data. Nevertheless, as it can be shown in Table 3.3, most schemes do not distinguish between the data-collection and data-processing consents, and they do not differentiate between a DC and a DP either. Due to this situation, in this analysis we do not evaluate the Grant and Revoke data-collection consent processes, among others.

Get Processing Consent	**Our Proposal**	**Truong et al. [7]**	**Nessie et al. [11]**	**Wirth and Kolain [6]**	**Barati et al. [26]**	**Wang et al. [12]**
DS-DP	0	2	2	1	0	2
DS-DC	0-1	2	1	0	2	0
DS-BC	0-1	0	1	2	1	5-6
DP-BC	1	0	0	1	1	0
DC-BC	1	4	2	0	2	0
DC-DP	1	0	2	2	4	0
DS-DR	0	0	0	0	0	2
DC-DR	0	0	0-2	0	0	0
DP-DR	0	0	0-2	0	0	0

Table 3.4: Comparison of Get Processing Consent workflow steps.

Centering the study on the *Get Processing Consent* operation, Table 3.4 shows the number of steps that each entity (i.e., DS, DC, DR and DP) has to perform in order to complete this procedure.

In order to fill this table, certain assumptions regarding some proposals

have been made. Those are next detailed:

- Nessie et al. [11]: In this work, DCs are supposed to store DSs' personal data. Therefore, when a DP request to process the collected data, this data is transferred to the DP, the DS is informed, and a new identity and a new usage contract is created by the DS. If an external DR were involved in this process, this would imply adding four more steps for each run of this operation. Since the authors do not specify this point, Table 3.4 depicts both possibilities.

- Wang et al. [12]: This work belongs to the category of proposals in which DSs are willing to share her personal data with other parties. On the other hand, our new proposal runs under the assumption that a service provider collects the personal data of a DS for further processing in exchange for using a product that it offers. Due to that particularity, Wang et al. requires DSs to deploy a first Smart Contract through which they will add new DPs, add new data or revoke consents in order to control the access to their data. This situation does not happen in our new proposal. In order to compare both proposals, Table 3.4 depicts both the original solution presented by the authors, and it also depicts the case in which this first Smart Contract is not generated.

- Barati et al. [26]: This scheme introduces a new entity named "Contract Activator", which is charge of deploying all Smart Contracts and interact with the other entities. In our study, we consider that this entity acts as DC, due to its similarities with the DC depicted in our proposal. Moreover, in this work the personal data to be processed is not sent to DPs, it is processed in containers instead. In consequence, in our study we consider that the entity that runs these containers acts as DR.

As shown in Table 3.4, in all the analyzed proposals the DS is performing three or more interactions with the system (see the three first columns) than in our new scheme. In particular, the new proposal only requires two interactions in the worst case (i.e., when a DP requests to process DS's personal data for a purpose not included in the default purposes list of the *Collection Consent Smart Contract*), and zero interactions in the best case (i.e., when the processing purpose already is in the aforementioned list). This aspect is of paramount importance due to the fact that the lower the number of interactions, the more lightweight the system is from the point of view of the user.

On the other hand, economical costs are directly related to the number of interactions between any entity and the Blockchain. Taking this into account, [6] and [11] require three Blockchain operations. Our new proposal also requires three Blockchain operations in the worst case, but it requires one less Blockchain operation in the best case. The rest of the studied schemes require four or more Blockchain operations.

Summarizing all the points above, this analysis shows that the *Get Processing Consent* operation designed for our new scheme is the most efficient in terms of overhead at the DS's side and theoretical economical costs among the current schemes in the literature.

Regarding the *Revoke Processing Consent* operation, this method only requires the intervention of one entity. This entity can be a DS who does not want her personal data to be processed anymore, or it can be the DC in charge of the collected data, which does not want that a certain DP processes it anymore. According to that, table 3.5 depicts the number of steps an entity must follow in order to complete by their own the revoke processing consent process.

As it can be seen in this table, only [7] and our new proposal allow

Revoke Processing Consent	Our Proposal	Truong et al. [7]	Nessie et al. [11]	Wirth and Kolain [6]	Barati et al. [26]	Wang et al. [12]
DS-BC	1	4	1	-	-	1
DC-BC	1	4	-	-	-	-

Table 3.5: Comparison of Revoke Processing Consent workflow steps.

both, the DC and the DS, to revoke a processing consent. The rest of the studied schemes do not consider this operation. Going deeper into this point, [7] requires four steps to complete this operation, while our new proposal requires only one step.

Access Data	Our Proposal	Truong et al. [7]	Nessie et al. [11]	Wirth and Kolain [6]	Barati et al. [26]	Wang et al. [12]
DP-DR	2	2	0-2	2	2	2
DP-DC	0	0	0	2	3	0
DP-BC	0	2	2	0	1	6
DC-BC	0	0	0	0	1	0
DR-BC	1	2	0-2	0	0	0
DC-DS	0	0	0	0	1	0

Table 3.6: Comparison of Access Data workflow steps.

Regarding the *Access Data* operation, Table 3.6 shows the number of steps each entity has to follow in order to allow a DP to get access to the collected personal data. As it can be seen, half of the proposals require

at least 6 steps to complete the full process. Among the schemes that require less steps, there are: [11] with 2 steps (note that, if an external DR is involved, it then requires 6 steps); [6] with 4 steps; and our new proposal with only 3 steps.

Functionality	**Our Proposal**	**Truong et al. [7]**	**Nessie et al. [11]**	**Wirth and Kolain [6]**	**Barati et al. [26]**	**Wang et al. [12]**
Get Processing Consent	3-5	8	8-12	6	10	8-9
Revoke Processing Consent	1	4	1	-	-	1
Access Data	3	6	2-6	4	8	8

Table 3.7: Comparison summary

Finally, in order to summarize all the findings of this study, Table 3.7 depicts the corresponding comparison results. As this table reflects, the total number of steps required by our new scheme is lower than in the rest of the analyzed proposals. This shows that the new proposal is the most lightweight in terms of entity interactions and theoretical efficiency.

3.3.5 Smart contract costs

This section evaluates the costs linked to the use the Smart Contracts. Analyzing those costs is of paramount importance due to the fact that they have a direct incidence on the feasibility of the proposed system when deployed in a real scenario.

In order to perform this evaluation in a plausible setting, the new scheme has been implemented, and the personal Blockchain *Ganache*[2],

[2]Truffle Suite Ganache, `https://trufflesuite.com/ganache/`

which is widely used for fast and easy Ethereum distributed application development, has been used. Ganache allows developers to deploy Smart Contracts, run commands, and inspect the state of the data while controlling how the chain operates.

The different Smart Contracts in use by the proposed system have been coded using the Solidity language[3] and they have been compiled by means of the *Truffle suite*[4]. This is a well-known ecosystem for Web3 development that provides a development environment, asset pipeline, and testing framework for developing Smart Contracts using the Ethereum Virtual Machine (EVM). In particular, the aforementioned personal Blockchain *Ganache* is part of the *Truffle suite*. Also, note that Web3[5] is a modular, reactive, type safe Java and Android library for working with Smart Contracts and integrating with clients (nodes) on the Ethereum network. This tool is used to interact with the Smart Contracts and the Blockchain.

In the following, we test and validate the correctness of the new proposed scheme on the introduced Blockchain infrastructure. This infrastructure was deployed in a single laptop. The performed evaluation comprises: i) studying some relevant EVM compatible networks in terms of efficiency and cost to run the proposed scheme on one of them; and ii) evaluating the cost in gas of deploying the Smart Contracts and the cost of performing their related operations.

[3]Solidity Programming Language, `https://soliditylang.org/`
[4]Truffle Suite, `https://trufflesuite.com/truffle/`
[5]Web3j library, `https://docs.web3j.io/`

Analysis of EVM compatible networks

Every transaction that is used to deploy a contract or invoke a function of a Smart Contract requires the payment of a fee to compensate the mining node for running the transaction and saving it on the Blockchain. EVM compatible networks use gas to represent this fee. Users can purchase gas from the mining nodes by paying with the network token (in the Ethereum network, this token is the *Ether*). The gas and the network token are two distinct terms. In particular, the gas indicates a constant cost of performing an operation on a Blockchain network. On the other hand, the network token is a volatile virtual currency, which is used to pay for the network resources.

This section studies some of the most relevant EVM compatible networks in terms of latency, throughput, and gas cost. Due to the difference in the token price and gas cost between the different networks, in order to evaluate the transaction cost among them, we use the transaction "base fee" cost, which is of 21000 gas[6]. This gas covers the cost of an elliptic curve operation to recover the sender address from the signature as well as the disk and bandwidth space for storing the transaction.

Table 3.8 shows the comparison of the studied Blockchain platforms considering their key features. As it can be seen, the most efficient Blockchain, in terms of cost, is the *BitTorrent Chain*. However, it is still a young network (it was deployed on 9th December, 2021), and, currently, this network suffers from a clear lack of information and usage statistics. Due to that, this network is not considered a valid option in this study. In the same way, *Ethereum Mainnet*, *Binance Smart Chain*, *Avalanche Chain*, and *Phantom Opera* are not considered valid options either, due to the

[6]Ethereum Gas and fees, `https://ethereum.org/en/developers/docs/gas/`

fact that they generate very high costs when deploying Smart Contracts.

Regarding the two Blockchain networks left, *Polygon Chain* and *HECO Chain*, according to *Polygon Chain*'s documentation[7], this network supports up to 10,000 tps, having a peak of 106 tps (16th of June, 2021). On the other hand, *HECO Chain*, according to its documentation[8], only supports up to 2000 tps, having a peak of 50 tps (10th of May, 2021). Also, the block mining time of *Polygon Chain* is 2s, while *HECO Chain*'s is 3s, making *Polygon Chain* the most efficient network.

[7]Polygon Chain, `https://docs.polygon.technology/docs/home/polygon-basics/what-is-polygon`, Accessed: 16 June 2021 (Deprecated)

[8]HECO Chain, `https://www.hecochain.com/developer.133bd45.pdf`, Accessed: 10 May 2021 (Deprecated)

Network	Consensus Protocols	Token	Mining Block Time	Transaction Throughput	Gas Cost	Token Price ($)	Tx Base Fee Cost ($)
Ethereum Mainnet[1]	PoW	ETH	13s	14 tps	35 Gwei	3146.36	2.31
Binance Smart Chain[2]	DPoS	BNB	3s	54 tps	6.5 Gwei	416.23	0.06
Polygon Chain[3]	PoS	MATIC	2s	30 tps	60 Gwei	1.61	0.002
Fantom Opera Chain[4]	Lachesis	FTM	1s	9.5 tps	360.5 Gwei	1.40	0.01
BitTorrent Chain[5]	PoS	BTT	2s	0.04 tps	304015 Gwei	$1.94x10^{-6}$	$1.24x10^{-5}$
Avalanche Chain[6]	Snowman	AVAX	2s	9.4 tps	42.5 nAVAX	86.0	0.077
HECO Chain[7]	HPoS	HT	3s	5.32 tps	4.40 Gwei	9,0	0.00083

Table 3.8: Comparison of EVM compatible Blockchains (March 2022).

[1] `https://etherscan.io` (accessed on 26th of March, 2022)
[2] `https://bscscan.com` (accessed on 26th of March, 2022)
[3] `https://polygonscan.com` (accessed on 26th of March, 2022)
[4] `https://ftmscan.com` (accessed on 26th of March, 2022)
[5] `https://bttcscan.com` (accessed on 26th of March, 2022)
[6] `https://snowtrace.io` (accessed on 26th of March, 2022)
[7] `https://hecoinfo.com` (accessed on 26th of March, 2022)

Gas Consumption

This section evaluates the cost, in terms of gas, of deploying on the personal Blockchain *Ganache* the different Smart Contracts considered in the new scheme, and the cost of performing the different allowed actions on them. According to this information, we then use the *Polygon Chain* network to calculate the cost in terms of tokens and USD of the aforementioned operations. During the analysis, an average gas value of 60 Gwei (60x10^-9 MATIC) and 1 MATIC = 1.61 USD is observed.

Table 3.9 shows that the proposed *Collection Consent and Processing Consent Smart Contracts* require 3239226 and 1629698 gas for deployment respectively (by means of "newCollectionConsent" and "newProcessingConsent" operations in each case). Therefore, creating and deploying them on the *Polygon Chain* network require 0.313 and 0.157 USD respectively.

Regarding the costs of the operations to be applied on the deployed Smart Contracts, as Table 3.9 depicts, all functions, except the *newPurpose* one, have a cost of less than 0.005 USD. The *newPurpose* method has a cost of 0.02 USD because it generates a new structure on the *Processing Consent Smart Contract* (as explained in section 3.2.3). On the other hand, the other operations simply modify a single variable of the state of the Smart Contracts, or, in the case of the *revokeConsentPurpose* and *revokeConsentProcessor* functions, they make recursive modifications of a single variable depending on the number of processors that process data for a certain purpose, or the number of purposes for which a processor processes the collected data respectively.

In order to put these costs into perspective, they are compared with the results provided by three works in the literature that provide empirical results. These are [12, 16, 26]. First, the costs of performing actions on already deployed Smart Contracts are reviewed; next, the costs of

deploying the Smart Contracts are discussed too.

Regarding the operations to be run on deployed Smart Contracts, [12, 16] show numbers in the range between 13000 and 50000 of gas, while [26] reports an average cost of 600000 gas. In comparison, the operations performed on the proposed *Collection Consent and Processing Consent Smart Contracts* are in the range between 18685 and 42946 (221707 if the costly "newPurpose" operation is considered), with an average cost of 28798 of gas (again, 44874 of gas if the "newPurpose" operation is considered). Whatever the case is, the costs of the new proposal are equivalent to the costs reported by its counterparts in the literature.

Finally, regarding the average cost of deploying a Smart Contract, in [12, 16, 26] this operation requires around 1500000 of gas. In comparison, the proposed *Processing Consent Smart Contract* matches this value with a cost of 1629698 of gas. Nevertheless, the proposed *Collection Consent Smart Contract* doubles this average with a cost of 3239226 of gas. This higher cost was expected, due to the fact that the number of methods, their complexity and, also, the data structures used in Smart Contracts directly increase their deployment costs. In this way, the proposed *Collection Consent Smart Contract* contains more methods and a higher complexity than other works in the literature in order to provide additional functionalities. It is worth mentioning that the aforementioned cost in gas implies an affordable cost in USD (i.e., 0.31 USD), due to the fact that the real economical cost linked to the deployment operation directly depends on the practical Blockchain being used and the price of its token. As it has been previously mentioned, in our case we have used the *Polygon chain* network.

According to the findings of this study, we argue the following points: i) the enhanced functionalities of our proposal justify spending more gas in deploying the *Collection Consent Smart Contract*; ii) the rest of operations

of the two Smart Contracts used in our proposal are expected to be used more frequently than the costly one, and they have a cost in gas equivalent to the other works in the literature; and iii) the cost in USD of running those operations depends on the Blockchain in use, in this way, the results that we have obtained in the *Polygon Chain* network show that all of them are affordable.

Smart Contract : Operation	Actor	Gas used	Cost (Token)	Cost (USD-$)
CC : newCollectionConsent	DS	3239226	0.1944	0.3129
CC : grantConsent	DC	29813	0.0018	0.0029
CC : revokeConsent	DS	27397	0.0016	0.0026
CC : modifyData	DS	42946	0.0026	0.0041
CC : eraseData	DS	29868	0.0018	0.0029
CC : revokeConsentPurpose	DS	23748	0.0014	0.0023
CC : revokeConsentProcessor	DS	21715	0.0013	0.0021
PC : newProcessingConsent	DC	1629698	0.0978	0.1574
PC : newPurpose	DC	221707	0.0133	0.0214
PC : grantConsent	DS	28880	0.0017	0.0028
PC : grantConsent	DP	33751	0.0020	0.0033
PC : modifyData	DS	27647	0.0017	0.0027
PC : revokeConsent	DS	31964	0.0019	0.0031
PC : revokeConsent	DC	30182	0.0018	0.0029
PC : revokeConsent	DP	18685	0.0011	0.0018

Table 3.9: Cost of the different functions of the provided Smart Contracts (CC referes to the Collection Consent Smart Contract and PC refers to the Processing Consent Smart Contract).

3.4 Integration of the new proposal in a real-world setting

The new scheme presented in this work is the core component of a larger system intended to allow DSs to control and manage all the consents given to the SPs to collect and process their personal data, along with the collected data itself. Even though this extensive system is still to be

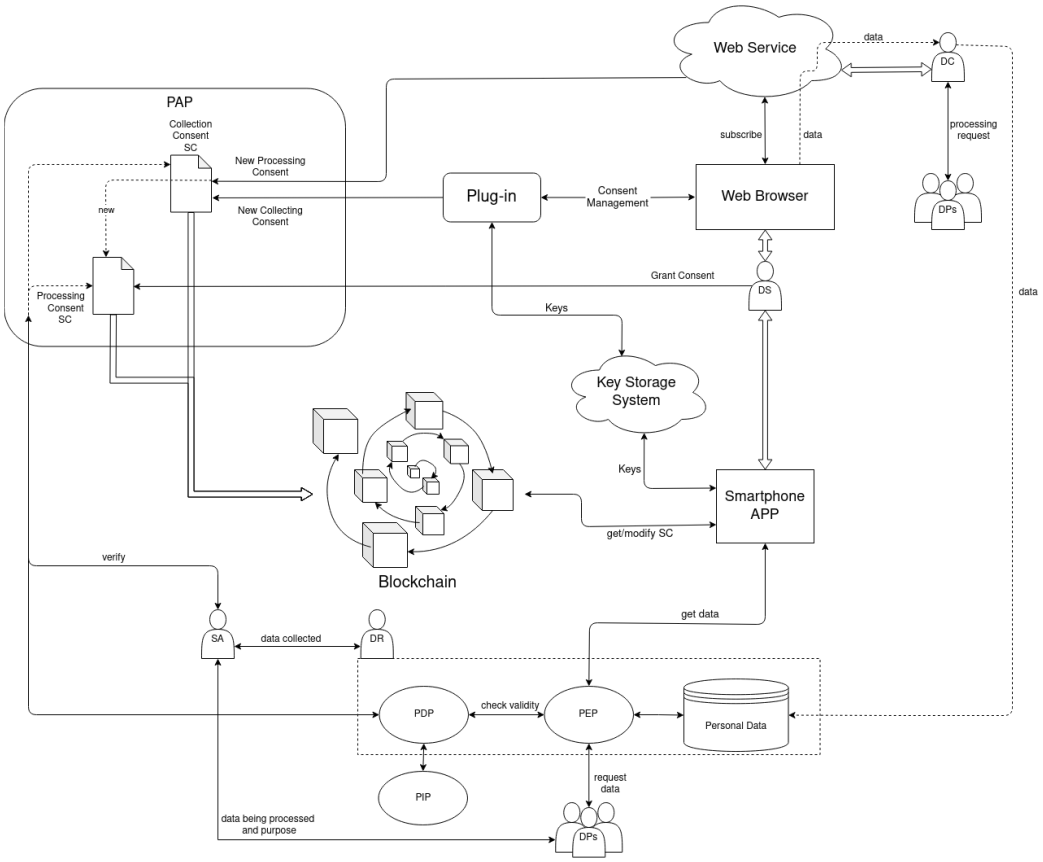


Figure 3.7: Integration of the new proposal in a real-world setting

defined, we next briefly introduce how we envisage it to be, in order to better contextualize the new proposed scheme.

A main point to be considered is that, in the expected large system, two software components are responsible of managing all the DSs' interactions. In Figure 3.7, we depict the specific case in which those two software components are a web browser plug-in and an smartphone application. More specifically, a DS is expected to: i) use her web browser (equipped with our plug-in) to consume the SPs' services; and ii) use her smartphone application to control and manage the access to her data and

to her consents (i.e., smart contracts). Note that this behaviour is aligned with other common web-based scenarios such as buying a certain good in an electronic commerce while browsing the web and authorizing the corresponding payment by means of a bank's smartphone app.

According to this scenario, the following subsections detail the different steps that a DS must perform in order to use the new proposal. In particular, the user registration, service access, data processing and data/consent management steps are next discussed.

3.4.1 User registration

A DS willing to use this system must first install and set-up the smartphone application. During this process, the DS obtains a set of credentials, this is, a root secret and public key pair (SK_{DS_0}, PK_{DS_0}), which are used to derive new key pairs and to authenticate herself in the following interactions with the system. This root key pair is unique and might be stored in the smartphone's secure storage (e.g., Samsung's Knox Vault[9]).

Next, the DS must install the proposed plug-in in the different web browsers she plans to use. During this installation, a new master key pair (SK_{DS_i}, PK_{DS_i}) is generated from the root credentials (SK_{DS_0}, PK_{DS_0}) for each different web browser i. All these master key pairs (SK_{DS_i}, PK_{DS_i}) are stored and managed by a decentralized multi-platform *Key Storage System*, such as the one presented in [77]. In the last step of the plug-in set-up, the DS is requested to fix a set of pre-defined privacy preferences that are used to automatize the generation of collection consents with the different DCs. At the current point, the method that should be used to gather those privacy preferences is fully open and it requires further work in order to find the best option. In any case, a plausible possibility would

[9]Knox Vault: `https://docs.samsungknox.com/admin/whitepaper/kpe/knox-vault.htm`

be to show a list of data categories together with check-boxes, and allow the DS to select those type of elements that she wants to share with the DCs.

3.4.2 Service access

When a DS uses her web browser to consume a certain service offered by a DC, the installed plug-in, in a transparent way to the user, runs the *Collection Consent Smart Contract generation* procedure described in Section 3.2.4.

By means of this procedure, the DC asks first for permission to collect DS's personal data in exchange for accessing the requested service. This data collection consent request is filled with a set of conditions that the plug-in checks with the set of DS's pre-defined privacy preferences. If those conditions match, the plug-in generates a new key pair $(SK_{DS_{i,j}}, PK_{DS_{i,j}})$ from the stored master key pair (SK_{DS_i}, PK_{DS_i}). The new key pair is stored in the *Key Storage System*.

Next, the plug-in uses the newly generated secret key $SK_{DS_{i,j}}$ to spawn a fresh *Collection Consent Smart Contract*. Then, the DC verifies whether this smart contract fulfills its requirements. If that verification succeeds, the DC starts collecting DS's personal data, and the DS starts consuming the service.

In the case that the data collection consent conditions are not matched with the DS's pre-defined privacy preferences, the plug-in shows those conditions to the DS, and she is required to accept them explicitly before continuing with the process.

3.4.3 Data processing

Once a *data-collection consent* has been agreed between a DC and a DS (and the linked *Collection Consent Smart Contract* has been published in the Blockchain), a certain DP can request the corresponding DC to process some of the collected data. This data processing request issued by the DP is driven by the *Processing Consent Smart Contract generation* procedure described in Section 3.2.4.

By means of this procedure, the DC builds a *Processing Consent Smart Contract* for that specific DP that states which data can be processed and for which purpose. If the smart contract linking the DC with the DP already exists, it only adds a new processing purpose to it. Whatever the case is, this *Processing Consent Smart Contract* verifies that the processing purpose of the *data processing request* issued by the DP is included in its *default-purposes* list before allowing the DP to gather any data. If this is the case, the *valid flag* of the smart contract is automatically set to "true".

In the case that the DP's purpose is not found in the smart contract's *default-purposes* list, the DC requires the affected DS to validate the new processing purpose by means of her smartphone application. A possible way of achieving this validation would be to set the smartphone app to periodically perform push requests to the Blockchain in order to find all the *Processing Consent Smart Contracts* that are linked to the DS and have their *valid flag* set to "false". For each non-valid *Processing Consent Smart Contract* found, the smartphone application would show a message to the DS asking her to explicitly grant or deny that *processing consent*.

3.4.4 Data and consent management

The smartphone application is expected to allow the DSs to keep track of all the consents that they have agreed with third parties, and the data

that those parties have collected and processed while backed by the corresponding consents. The procedures that apply to these two cases are described in Sections 3.2.4 and 3.2.4.

Focusing on the data management procedure, a DS uses her smartphone application to identify herself as the data owner and request the DR to retrieve all her personal data which was collected by a certain DC. In order to do that, the following steps are followed: i) the smartphone application obtains from the *Key Storage System* the cryptographic keys ($SK_{DS_{i,j}}, PK_{DS_{i,j}}$) that were used for deploying the *Data-Collection Consent Smart Contract*; ii) the smartphone application uses the retrieved secret key $SK_{DS_{i,j}}$ to compute a digitally signed temporal proof; iii) the smartphone builds a data access request containing the digitally signed temporal proof and sends it to the DR through the proposed XACML extension; iv) the *Policy Enforcement Point (PEP)* receives the data access request and asks the *Policy Decision Point (PDP)* for a grant/deny access decision; v) the PDP uses the *Policy Administration Point (PAP)* to get from the Blockchain the corresponding *Collection Consent Smart Contract*; and vi) the PDP, in a two-step operation, checks whether the smart contract contains the corresponding $PK_{DS_{i,j}}$, and it ascertains whether the requesting entity has the proper rights to access the protected data.

Regarding the consent management procedure, a DS can use her smartphone application to interact with the different consents/smart contracts that she has created. For this purpose, the smartphone application is able to retrieve all the required cryptographic keys from the *Key Storage System*, use those keys to find the corresponding smart contracts that have been published on the Blockchain, and interact with them by means of the methods they offer.

3.4.5 Consent audit and legislation enforcement

When a DS (or the SA itself) suspects that a certain DC/DR or DP has misbehaved with the consent given, she can request the SA to perform an auditing process. The SA will then follow the steps described in Section 3.2.4.

If the suspicious entity is a DP, the auditing process will first lead the SA to check which data has been processed by the DP and for which purpose. Then, it will retrieve and use the related *Processing Consent Smart Contract* to verify whether the consent was valid when the processing of the data was performed. In the same way, if the DR is believed to be illegally keeping some personal data on behalf of a DC, the SA will check the validity and extend of the given consent by means of the related *Collection Consent Smart Contract*.

In both cases, if after completing the auditing process the SA detects that personal data has been collected or processed without the proper consent, she will impose the corresponding administrative fine to the DC and/or the DP.

Chapter 4

PrivacySmart: Automatic and Transparent Management of Privacy Policies

Our contribution in Chapter 3 successfully tackles the centralization, functional and implementation issues found in the literature regarding the management of the GDPR-consent in order to collect and process users' personal data by service providers.

In Chapter 4 we apply the developed technology in a real-world use case such as the management of privacy policies and cookies on websites while users access to these services. In this regards, we present PrivacySmart: an innovative system that streamlines privacy and cookie policy

management for users while respecting their preferences. It consists of a browser plug-in and a smartphone app, which together simplify policy acceptance and tracking. The plug-in manages pop-ups according to users' preferences during browsing, while the app consolidates accepted policies across devices. This system enhances user control and reduces the hassle of managing policies manually. Tested as a Firefox extension, PrivacySmart has shown promising results in evaluations.

This contribution is supported by a publication in [78].

The remainder of this Chapter is organized as follows. Section 4.1 introduces the new proposal and details the protocol steps of the new model. Next, Section 4.2 explains how we have implemented the presented scheme and evaluates its functionality against the most common privacy and cookie policies banner providers.

4.1 Proposal Description

In this section we describe our system for the automatic management of cookie and personal data usage policies which comprises the agreements between Data Subjects and Service Providers about DSs' personal data, automatically and transparently to the users while they navigate and access online services.

The plug-in offers several key functionalities: Firstly, it automatically accepts cookies and privacy policies based on user-specified privacy levels, making web browsing easier and faster. Secondly, it registers user consent on each website using a Smart Contract, ensuring a secure record of the provided consent. The plug-in is compatible with cookie pop-ups from major websites, it effectively identifies cookie and privacy policies configuration pop-ups, and it determines the appropriate selections that

align with the user's privacy preferences. This process includes deploying a Smart Contract on the Blockchain to record the provided consent, allowing the user to control and manage accepted privacy and cookie policies for each visited website.
In addition, we provide a smartphone application that consolidates all privacy and cookie policies managed by the plug-in, empowering the user with complete control over their personal data while giving them a clear overview of the management of their privacy preferences.

4.1.1 System Architecture Overview

As shown in Figure 4.1 we can identify four components on our working environment: our proposed plug-in, the provided smartphone application through which the user has full control of the accepted policies, the Blockchain where SCs are deployed, and the web browser - where the plug-in is installed - that the user uses to access different websites. The plug-in consists of three main modules: the core module, the pop-up interaction module and the configuration module. The core module identifies the privacy pop-up of the web site visited, it initializes the module responsible of directly interacting with the pop-up and deploys a new SC with the cookies and privacy policies accepted. Then, the pop-up interaction module - which consists of a set of scripts, each designed specifically for a concrete pop-up - accepts or rejects the cookies and privacy policies on the pop-up and returns the configuration set to the main program. Finally, the configuration module allows the user to set-up her privacy preferences, to configure the keys used for interacting with the Blockchain, and to manage all the consents given by the plug-in trough a UI. The smartphone application plays a crucial role in enhancing the user's control and convenience. It seamlessly downloads all the consent Smart Contracts deployed on the Blockchain, effectively summarizing all policies that have been accepted by the plug-ins installed across any users' browsers. This consolidated view

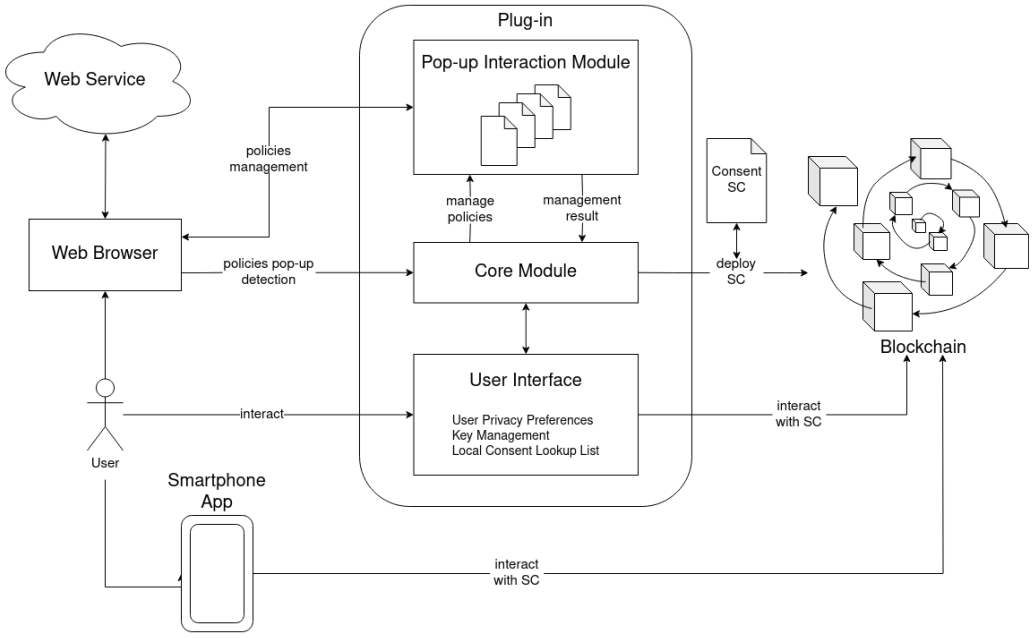


Figure 4.1: Privacy Smart: System Architecture Overview

offers a comfortable and user-friendly review of the collection of accepted policies.

4.1.2 User Privacy Preferences

The Privacy policies state who will be able to use the user's collected data, for which purpose can be used and for how long it can be kept. These policies can be represented at different levels of specification and data granularity, like a formal notice or like a more structured notice using templates. Formal notice allows Service Providers to set up a more customized privacy policies thus being able to arbitrarily create rules that describe how data will be managed. These policy notices are difficult to interpret, to read and to match automatically due to the variability in the terms and phrasing used across websites when describing the rules. A clear example of this kind of policies are the Terms and Conditions a user

must accept when signing up to certain services. Some Service Providers opt for more structured notices, such as cookies and privacy policies configuration pop-ups. These pop-ups consist of templates provided by some "*cookies providers*" which are easier to interpret and manage automatically. According to this, work focuses on the management of these web policy pop-ups according to the user's pre-established privacy preferences.

To ease the policy specification process, we consider using policy templates, as suggested in [79]. These templates represent different privacy levels, defined according to the configuration alternatives that offer the studied pop-ups. These privacy levels can be divided into four main categories – from the most restrictive level one to the most permissive level four – which are:

1. Reject all: all cookies or all non-essential cookies are rejected.
2. Reject profiling cookies and third parties: this configuration rejects the use of cookies for profiling interest. Furthermore, it rejects the use of the collected data by third parties or web site partners.
3. Reject third parties: this option just rejects the use of the collected data by third parties or web site partners.
4. Accept all: all cookies are accepted.

In order to set up this privacy preferences value, the user must be requested to select one of the provided options when the plug-in is installed in the web browser.

4.1.3 Pop-up interaction module

In the current model, websites offer management of their privacy and cookie policies through pop-ups. In order to cover this need, there are com-

panies, privacy and cookie policies banner providers, that are dedicated to the design of these cookie management pop-ups and provide these tools to other companies which can use them in their websites. In this way, website providers can comply with the GDPR thus presenting the users their privacy policies and a way to grant them.

Each website provider can modify the visual section and the content of the pop-up, by specifying which types of cookies will be used, which data can be collected and which third parties will process the collected data. The main interoperability issue here is that each cookies provider makes its own standard, so the solutions to manage the forms on each website can be very different.

In order to decide which privacy and cookie policies banner providers take into account, we have considered the most used ones according to a study performed by Wrappalyzer[1], where they have analyzed over 1.200.000 websites. The selected ones on this study are Didomi[2], CookieYes[3], CookieNotice[4] and OneTrust[5]. For each of these providers we have implemented a script in the pop-up interaction module responsible of interacting with the banner and manage the consent on users' behalf. The result of the operation is sent back to the Core module, which will record it into an SC. In the case some privacy and cookie policies pop-up does not allow the consent management using a privacy level lower than the one set by the user, the pop-up interaction module, hides the banner without accepting anything, in such a way that the user can access to the web content without being annoyed. This modularity of the system allows the tool to be able to work with more privacy and cookie policies banners of other

[1]Cookie compliance technologies market share, `https://www.wappalyzer.com/technologies/cookie-compliance/`, Accessed: 15 January 2023

[2]Didomi: `https://www.didomi.io/es/`

[3]CookieYes: `https://www.cookieyes.com/`

[4]Cookie Notice: `https://cookiesnotice.com/`

[5]OneTrust: `https://www.onetrust.es/`

providers by simply adding specific scripts for them to this module.

4.1.4 Consent Smart Contract

The Consent SC is responsible for storing the consent managed by the plug-in when it interacts with the web browser pop-up. The SC consists in a single class that holds:

- Web service url: Uniform resource locator of the web page where consent has been managed through the plug-in (*url* argument).
- Privacy level: Integer value that indicates the privacy level established by the user and the one used by the plug-in to perform the acceptance of policy cookies (*privacyLevel* argument).
- Consent validity period: This indicates the dates from which the consent is valid(*beginningDate* and *expirationDate* arguments).

Once the *Consent Smart Contract* instance is created, the following methods can be used to interact with it:

- *whichWeb()* method returns the url value stored in the SC.
- *whichPreference()* method returns the privacy level value stored in the SC.
- *isValid()* method checks that the consent lifetime period has not expired.

4.1.5 Workflow

The proposed plug-in workflow can be divided into three main phases: the set-up phase, where the plug-in is installed on the web browsers and

the user indicates her privacy policy preferences; ii) the navigation phase, when the user accesses to different web pages, and; iii) the control phase, where the user can manage all cookies and privacy policies accepted by the plug-in.

1. Set-up phase

   User

   (a) Install the plug-in on the web browser.

   (b) Configure the plug-in.

      i. Set privacy policy preferences in the plug-in using the UI.

      ii. Chose wheter to use and existing key pair for the deployment of the SCs or to generate a new one. In the first case, the user must input the mnemonic words associated to that key pair and move to step 1d; otherwise, the workflow continues as stated.

   Plug-in

   (c) Execute the BIP-39 protocol [80], such generating 24 mnemonic words and a 512 bits seed, the *BIP-39 seed*.
   NOTE: The user must keep these mnemonic words in a safe place, without them she will not be able to recover the keys or use the same account into another web browser or the smartphone application.

   (d) Execute the BIP-32 protocol [81] using the *BIP-39 seed* as input. This process generates user's Secret Key (SK_u).

   (e) Derives user's Public Key (PK_u) by means of SK_u.

   (f) Create a local consent lookup list. This list is internally used by the plug-in to check if the privacy and cookie policies on a particular website have already been managed.

2. Navigation phase

User

(a) Access a web page with cookies and privacy policies configuration banner.

Plug-in - Core module

(b) If the cookies and privacy policies have already been managed on this web and the consent is valid - *isValid()* method of the Consent SC returns True - no banner will appear and the process will end. Otherwise, if the cookies and privacy policies have already been managed on this web but the consent is not valid - *isValid()* method of the Consent SC returns False - the pertinent cookies are removed from the browser and the process follows as if the consent had not yet been managed.

(c) Detect the emerging pop-up and identifies the script in the pop-up interaction module responsible of managing it.

(d) Initialize the pop-up interaction module script, indicating the user's privacy policy preferences.

Plug-in - Pop-up interaction module

(e) The script interacts with the pop-up, performs the cookies and privacy policies configuration management, and returns the result to the core module.

Plug-in - Core module

(f) The core module deploys a new Consent SC on the Blockchain - using the key pair (PK_u, SK_u) - indicating the consent granted and the lifetime of the consent. This process returns an identifier of the SC.

(g) The core module adds the web url along with the SC's identifier to the consent lookup list.

3. Management phase

   User

   (a) Requests the consents managed by the plug-in through the UI.

   Plug-in

   (b) The plug-in displays the consent lookup list with all consents managed.

   (c) In the case the user wants to see the details of a given consent, she selects it on the display.

   (d) The plug-in gets the details of the selected consent from the Blockchain and displays them for the user.

The provided smartphone application empowers the users with complete control over all privacy and cookie policies managed by all the associated plug-in installed on any browsers by the user. To ensure a comfortable and user-friendly experience, the application offers these simple functionalities:

1. Add s new key pair: This allow users to add a new key pair (PK_i, SK_i) by introducing the mnemonic words provided by the different plug-ins during the set-up phase. By using these mnemonic words, the app will execute BIP-39 and BIP-32 protocols, as in the plug-in Set-up phase, thus recovering the cryptographic keys. After adding a new key pair the app will have full access to all SCs containing the managed policies.

2. Access and Review: Users can conveniently access and review the entire collection of accepted policies through the application. This allows an easy monitoring and management of privacy preferences.

By providing these functionalities, the smartphone application ensures that the user has a seamless and user-friendly experience, enabling them to exercise complete control over their privacy and cookie policies across different plug-ins installed on their browsers.

4.2 Discussion

After implementing the described proposal, we have validated its efficacy through live experiments.

4.2.1 Implementation

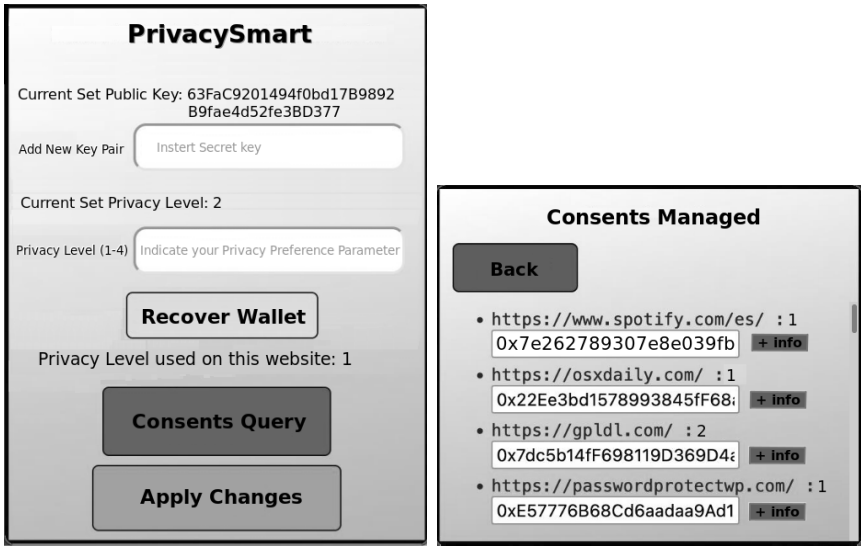
We have implemented our plug-in as a Firefox extension coded in JavaScritp[6], HTML[7] and CSS[8] (source code available in Github [9]). We have chosen Firefox because it is one of the most used browsers, it is very extensible, and it allows programmers to add new features or modify existing ones. The scripts in the pop-up interaction module - each to manage a different banner provider - are coded in JavaScript. Concerning the deployment of the Consent SCs, we have used web3.js library [10], so the plug is able to connect into an existing Blockchain. Summarizing, our plug-in is installed on Firefox as an add-on, without modifying the browser's source code. We omit further details on the tool implementation and only describe plug-in's main interfaces:the plug-in Configuration module main interface and the consent display interface. The configuration module interface, shown in Figure 4.2(a), allows users to i) manually modify the key

[6]JavaScript: `https://developer.mozilla.org/en-US/docs/Web/javascript`
[7]HyperText Markup Language: `https://developer.mozilla.org/en-US/docs/Web/HTML`
[8]Cascading Style Sheets: `https://developer.mozilla.org/en-US/docs/Web/CSS`
[9]Source Code: `https://github.com/EduardBel/PrivacySmart`
[10]Web3.js: `https://web3js.org/`

pair used to deploy the SCs where the consent is registered; ii) recover an old key pair using the mnemonic words set; iii) change user's privacy preference parameter, and; iv) manage the consents given by the plug-in on the different web sites the user has visited. In order to update the new values for the new key pair or the privacy preference parameter, after adding the new values, the user has to click the "Apply Changes" button. On the same way, if the user wants to see all consents given by the tool, she must click the "Consents Query" button, and the consent display interface will pop-up. Finally, if the user needs to recover an old key pair, she must click the "Recover Wallet" button, and a new interface will appear to introduce the mnemonic words. This UI also shows the user the consent given in the current web site, if any.

The consent display interface shows all privacy and cookie policies managed by the plug-in on the different websites the user has visited. As shown in Figure 4.2(b), it displays each of the sites, along with the privacy level used on that site and a button in the case the user wants more detailed information of the consent given.

Concerning the smartphone application, it has been coded using Java and web3j library for Blockchain interactions. Figure 4.3, shows the main display interfaces of the application. On the one hand, Figure 4.3(a) shows how the application displays on which websites the policies have been managed by a selected a plug-in (the plug-in has been selected in a previous layout, and these plug-ins are added to the app by specifying the key pair used for deploying the Consent SC). On the other hand, after a user selects a specific website, all information about the consent given is shown as in Figure 4.3(b).



(a) Configuration module main interface.

(b) Consent display interface.

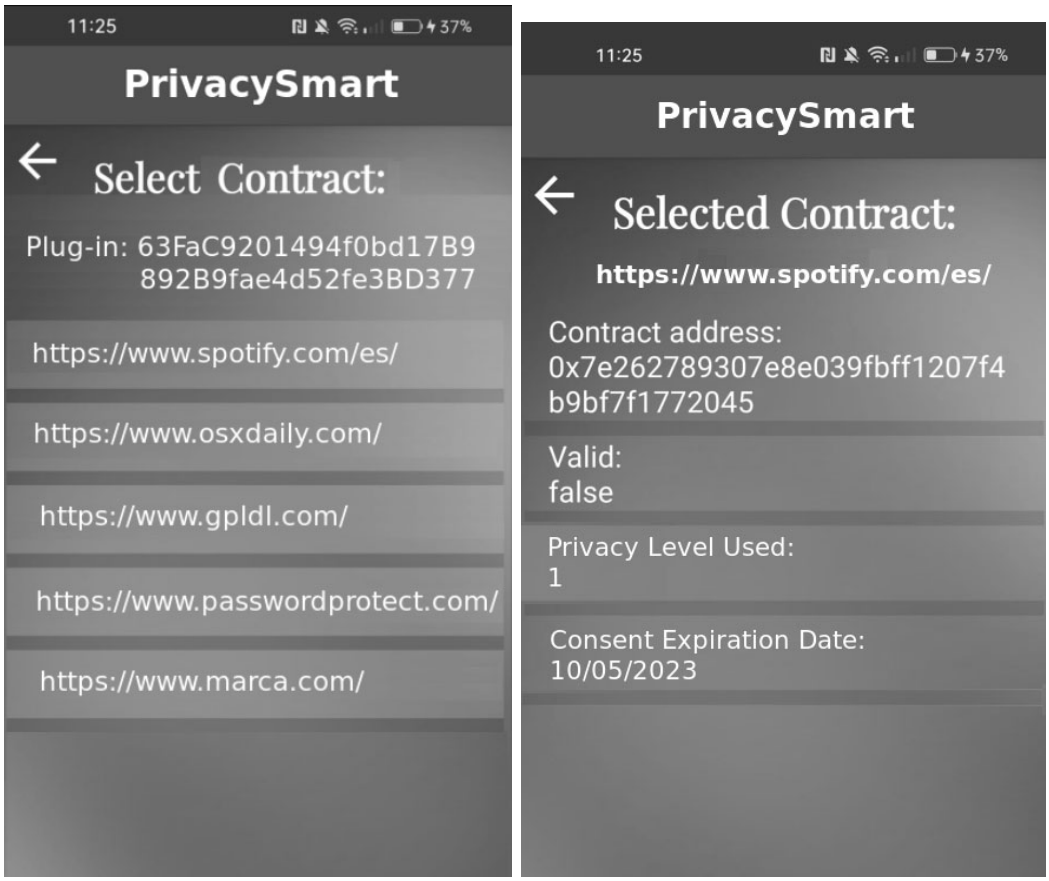
Figure 4.2: PrivacySmart: browser plug-in main interfaces.

4.2.2 Evaluation

We installed our proposed plug-in on a Firefox web browser and used Ganache[11] for the simulation of a public Blockchain in order to perform our experiments. On the experiments carried out, although key management system is already implemeted, we have used the keys provided by Ganache for the deployment of the ConsentSCs in the Blockchain.
In order to determine the correctness of the proposed plug-in, we have evaluated its performance on some websites that use a pop-up offered by the most used privacy and cookie policies banner providers according to the results presented in Wappalyzer[12]. Here we have used different user's privacy parameter values and we have taken into account i) if the plug-in is able to manage the pop-up transparently and on users behalf, so the

[11] Truffle Suite Ganache: `https://trufflesuite.com/ganache/`

[12] Cookie compliance technologies market share, `https://www.wappalyzer.com/technologies/cookie-compliance/`, Accessed: 15 January 2023



(a) Interface for selecting a consent managed from a browser plug-in.

(b) Interface for viewing managed consent information.

Figure 4.3: PrivacySmart: smartphone application main interfaces.

user does not even notices it, and; ii) if the consent has been performed according to user privacy preferences. In the first parameter (*Transparent Management*, third column in Table 4.1), our evaluation primarily revolves around the plugin's capability to handle the pop-up discreetly, ensuring that the user remains unaware of its presence. We consider the scenario wherein consent cannot be managed in accordance with the user's specified privacy parameters, and in such cases, the plugin should adeptly conceal the pop-up, preventing any disruption or annoyance to the user. According to the second parameter we evaluate if the consent has been managed according to the user's privacy preference, and the Consent SC has been deployed correctly.

Results obtained, summarized in Table 4.1, show that our plug-in works perfectly with pop-ups provided by Didomi and OneTrust, allowing users to decide between different privacy preference levels. On the other hand, results obtained in relation to the management of the CookieYes and CookieNotice banners, show that there are pages that do not offer consent management beyond accepting all privacy and coolies policies. If user's privacy preference is set to level 4, it accepts the policy, on any other case the plug-in just hides the pop-up without accepting anything.

Provider	Website	Transparent Management	Privacy level used	Consent Correct
Didomi	marca.com	✓	1	✓
	as.com	✓	1	✓
	elpais.com	✓	2	✓
	giphy.com	✓	2	✓
	reverso.net	✓	3	✓
	elmundo.es	✓	3	✓
	bfmtv.com	✓	4	✓
	societe.com	✓	4	✓
CookieYes	wpmet.com	✓	1	✓
	sliderrevolution.com	✓	2	✓
	tympanus.net	✓	3	✓
	wpdatatables.com	✓	4	✓
	bezkoder.com	✓	3	✓
	wpthemedetector.com	✓	4	✓
			2	X
	styde.net	✓	4	✓
			1	X
CookieNotice	gpldl.com	✓	1	✓
			2	✓
	mailtrap.io	✓	1	X
	osxdaily.com	✓	2	X
	searchengineland.com	✓	3	X
	booster.io	✓	3	X
	wpbuffs.com	✓	3	✓
			4	✓
OneTrust	fiverr.com	✓	1	✓
	gitlab.com	✓	2	✓
	app.slack.com	✓	3	✓
	freepik.com	✓	4	✓
	udemy.com	✓	1	✓
	upwork.com	✓	2	✓
	open.spotify.com	✓	3	✓
	elementor.com	✓	4	✓

Table 4.1: Results of tool performance on websites that use a pop-up offered by the studied privacy and cookie policies banner providers.

Chapter 5

Blockchain-Enhanced User Consent for GDPR-Compliant Real-Time Bidding

Along with the real-world use case presented in Chapter 4, in this Chapter we ensure the Real-Time Bidding compliance with GDPR and the ePrivacy Directive. Current solutions, like the IAB's Transparency and Consent Framework and OpenRTB, lack transparency and fail to secure legally valid user consent. We propose a Blockchain-based framework that decentralizes consent management, giving users direct control. The system automates consent handling, provides immutable compliance proof, and includes a smartphone app for users to manage consent per website. Additionally, we introduce a specificity value for IAB Audience

Taxonomy elements, helping users assess the privacy impact of sharing data. By enhancing autonomy, transparency, and accountability, our approach strengthens trust in programmatic advertising while maintaining GDPR compliance without disrupting the industry.

This contribution is supported by a publication in [82].

The remainder of this Chapter is organized as follows. First, in Section 5.1, provides a brief explanation of the Real-Time-Bidding protocol, its pricing model, the IAB's Europe implementation of the protocol and the taxonomy used. Section 5.2 introduces the new proposal and details the protocol steps of the introduced model. Finally, Section 5.3 analyzes the fulfillment of the GDPR compliance requirements.

5.1 Preliminaries

This section provides an overview of the key technologies and concepts used in our contribution.

5.1.1 Programmatic Advertising and Real Time Bidding

Programmatic advertising [83] automates the purchase and sale of ad space through *Ad Exchanges*. It involves two main entities: *Supply-Side Platforms (SSPs)*, used by publishers to manage and sell ad inventory, and *Demand-Side Platforms (DSPs)*, used by advertisers to automate ad buying and campaign management.

Advertisers define their campaign objectives, target audience, and bid amounts via a DSP, while publishers use an SSP to manage available ad space and pricing. The Ad Exchange serves as a marketplace where SSPs and DSPs interact, with notable examples including Google Ad Exchange,

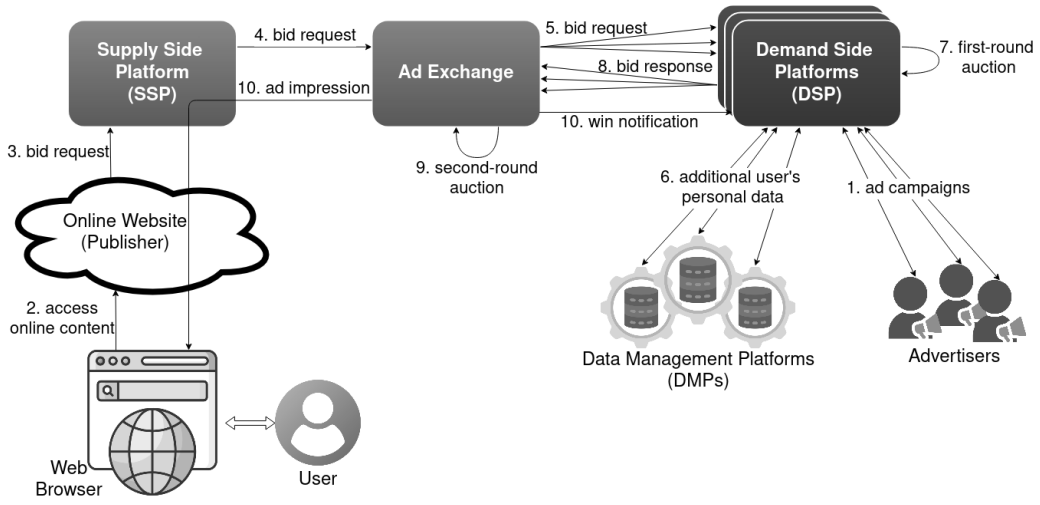


Figure 5.1: Programmatic Advertising Scheme

Facebook Ad Exchange, and Microsoft Advertising Ad Exchange. Additionally, *Data Management Platforms (DMPs)* aggregate and analyze user data, enabling advertisers to refine their targeting strategies.

At the core of programmatic advertising is Real-Time Bidding (RTB), where ad inventory is bought and sold in real-time during a web page's loading process. When a user visits a website, an auction occurs within milliseconds, determining which ad will be displayed based on advertisers' pre-targeting criteria.

The RTB process, illustrated in Figure 5.1, follows these steps:

1. Advertisers specify campaign details, including ads, target profiles, and bid amounts via DSPs.
2. A user visits a publisher's website, which has reserved ad space for programmatic sales.
3. The publisher sends a bid request to the SSP, including page details and user data.

4. The SSP forwards the request to the Ad Exchange.
5. The Ad Exchange relays the bid request to the connected DSPs.
6. DMPs may attach additional user-related metadata to the bid request sent to the RTB system. Then, the DSPs retrieve this additional user information and analyze the bid request.
7. Each DSP conducts an internal auction among relevant ad campaigns.
8. DSPs submit bids, specifying the amount and ad content.
9. The Ad Exchange selects the highest bid and runs a second-price auction, where the highest bidder pays slightly above the second-highest bid.
10. The winning ad is sent to the SSP and displayed on the publisher's webpage.

5.1.2 Pricing model

The primary pricing model in real-time display advertising is CPM (Cost Per Thousand Impressions), where advertisers pay per 1,000 ad displays. Alternative models include CPC (Cost Per Click), charging per ad click, and CPA (Cost Per Acquisition), where payment occurs only after a user completes a specific action[1].

5.1.3 User consent in IAB's Europe OpenRTB

In Programmatic Advertising, the IAB's TCF embeds user consent into OpenRTB bid requests [84], enabling DSPs and adtech vendors to verify

[1]Start.io: Advertising Pricing Models, `https://www.start.io/glossary/advertising-pricing-models`, Accessed: 29 January 2025.

consent before processing personal data for targeted ads. The bid request also includes user data for targeting and measurement.

Consent is stored as a Base64 string generated by the *Consent Management Platform (CMP)*, indicating: i) approved processing purposes, ii) authorized vendors, iii) legal basis (e.g., consent or legitimate interest), and iv) permitted data uses (e.g., personalized ads, analytics).

Upon receiving a bid request, DSPs and vendors check the consent string to determine processing permissions. If restrictions apply, they must exclude certain data or refrain from bidding. Advertisers must ensure compliance with a valid legal basis.

Despite these measures, OpenRTB fails to meet legal requirements due to insufficient transparency and user control. Additionally, bid requests expose user data—such as location, device details, and browsing history—raising privacy concerns over how this information is handled and shared.

5.1.4 Audience Taxonomy

Traditionally, data vendors, DMPs, and analytics providers have used custom taxonomies to segment audiences, leading to inconsistencies in data descriptions. Variability arises from diverse segmentation approaches, including demographic, interest-based, purchase intent, and psychographic classifications, making data comparability challenging.

To address this, the IAB Tech Lab introduced Audience Taxonomy 1.0[2], a standardized framework for labeling audience segments across different providers. This taxonomy ensures consistent classification of both

[2]Audience Taxonomy: `https://iabtechlab.com/standards/audience-taxonomy/`

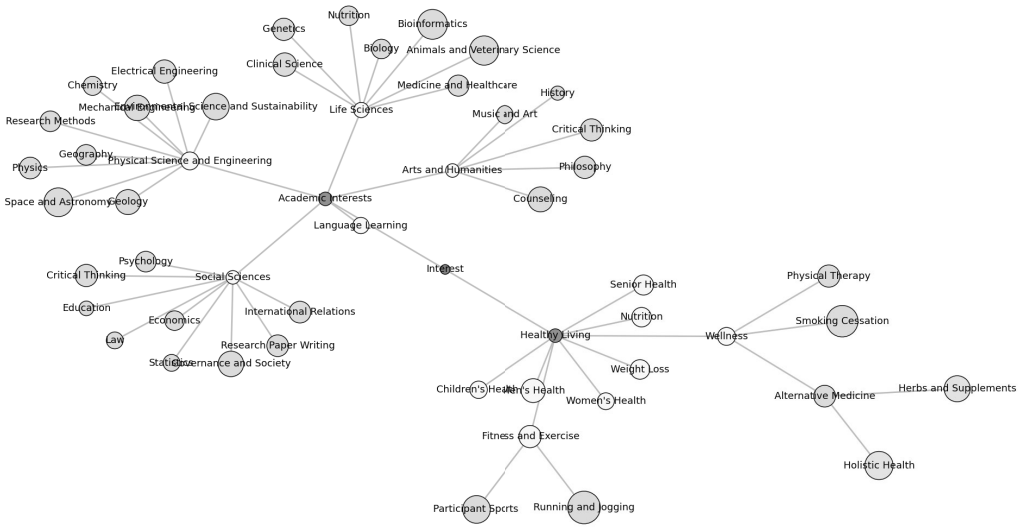


Figure 5.2: Subset of Interests segment of the IAB Tech Lab's Audience Taxonomy

first-party and third-party audience data. In this work, we adopt this taxonomy to define user profiles, using the 496 values under the *Interests* segment. These are organized in an architectural structure as can be seen in Figure 5.3. In this figure, each node represents an element of the taxonomy and the color and size of the nodes indicate their height and their specificity (see Section 5.2.3 for further details). In order to calculate this specificity, first, we have performed a search of the different values on the Google search engine and then, we have applied the equation defined in Section 5.2.3 (Equation 5.6). See Figure 5.2, for a zoom of the branches *Academic Interests* and *Healthy Living* for a more detailed example of the labels.



Figure 5.3: Interests segment of the IAB Tech Lab's Audience Taxonomy

5.2 Proposal Description

In this section, we present the proposed solution for ensuring GDPR compliance in RTB while maintaining transparency and user control. First, we define the requirements that the system must meet to align RTB practices with privacy regulations. Next, we describe the system architecture, explaining how Blockchain Technology and a browser plug-in enable decentralized consent management. Finally, we detail the three-phase workflow of the framework: Configuration, where users define their privacy preferences; Web Browsing & Consent Enforcement, where consent is automatically applied during online activity; and Consent Management & Data Accountability, which allows users and regulatory authorities to monitor and verify data usage.

5.2.1 Requirements

In order to reconcile the TCF and RTB systems with GDPR, the following requirements must be fulfilled:

R1 User consent must be explicit, informed, and freely given, ensuring clarity in purpose, data usage, and user choice without implicit assumptions or misleading design.

R2 The integrity and authenticity of consents must be preserved.

R3 Users must be able to access, erase, and withdraw consent as easily as they give it.

R4 Users should have accessible records of visited publishers and involved trackers.

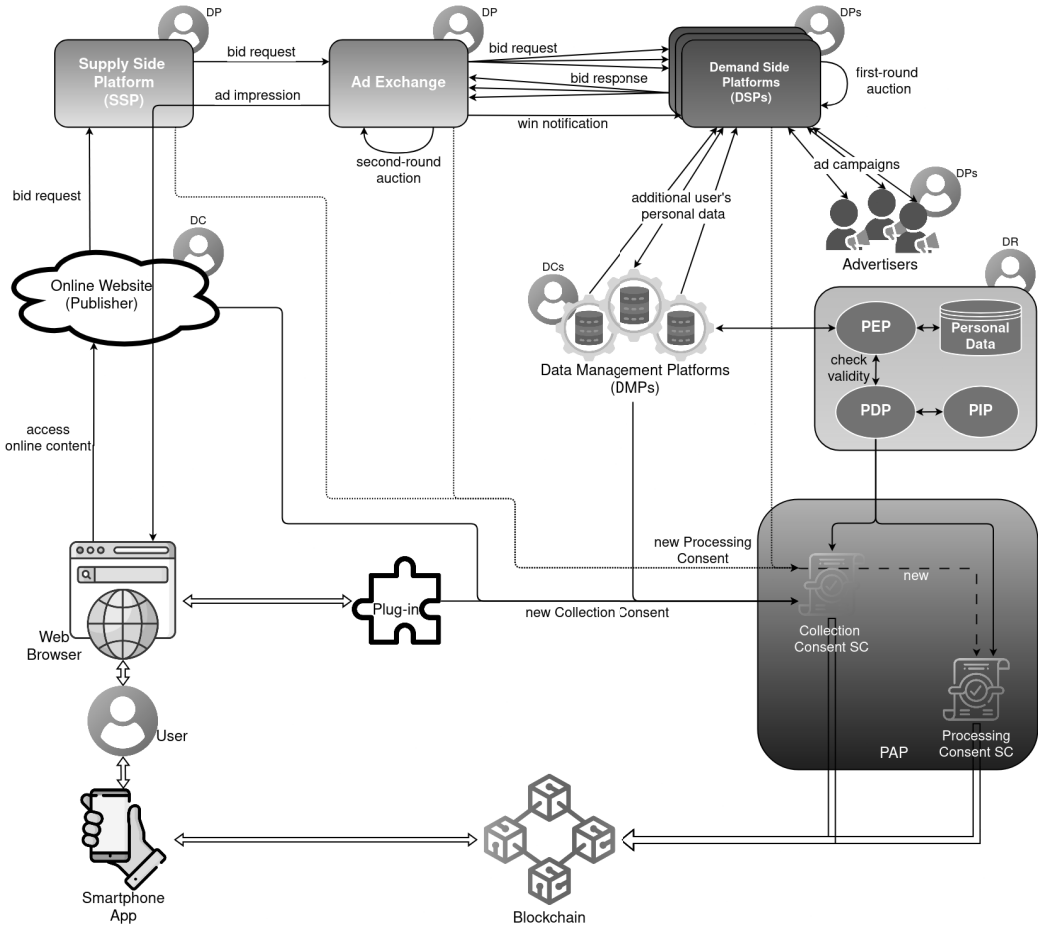


Figure 5.4: GDPR-Compliant RTB Proposal Overview

R5 Compliance with user preferences must be enforced across the adtech ecosystem.

5.2.2 System architecture

To meet the legal requirements of OpenRTB, our solution decentralizes GDPR compliance verification and data management by leveraging a Blockchain network, following the scheme presented in Chapter 3. By

storing user consents on-chain, we provide both users and adtech entities with immutable proof of granted permissions for data processing.

Figure 5.4 presents an overview of the system, where the browser extension (plug-in) plays a key role in consent management, serving as the user's access point. The plug-in, integrated into the OpenRTB model, ensures compliance without modifying the existing ecosystem. Once configured, it automates consent management based on the user's preferences, transparently handling data collection and processing. Additionally, it records visited sites and displayed ads for user review.

Under GDPR, adtech entities process personal data in two primary roles: Data Controllers, which determine the purpose and means of data processing, and Data Processors, which process data on behalf of a DC. In this context, Publishers and Data Management Platforms (DMPs) act as DCs because they collect and provide users' personal data for profiling in ad campaigns. Conversely, SSPs, DSPs, Ad Exchanges, and Advertisers function as DPs since they process this data during the advertising process. Additionally, the role of Data Recipient refers to the entity responsible for storing users' collected data. A DMP can also act as a DR if it directly stores the data. However, if storage is outsourced to a third party, then that third party assumes the role of DR.

As already detailed in Section 3.2.2, our system distinguishes between:

- *Data-Collection Consent:* Agreement between a user and a DC, specifying conditions for data collection in exchange for web content access.
- *Data-Processing Consent:* Agreement between a DP, DC, and user, outlining conditions under which the DP processes user data collected by the DC.

Additionally, a smartphone application provides users with a centralized interface to monitor, modify, and revoke consents in real-time. It also enables users to request data deletion, ensuring full control over their personal information in compliance with GDPR.

The system follows a three-phase workflow:

1. *Configuration:* Users define their privacy preferences, specifying which data can be collected and how it can be processed.
2. *Web Browsing & Consent Enforcement:* As users navigate the web, consent is automatically applied, and the RTB process operates within the predefined privacy settings.
3. *Consent Management & Data Accountability:* Users can monitor their consent history, track displayed ads, modify permissions, and request data deletion as needed.

5.2.3 Phase-1: Configuration

During the initial configuration, the user installs the plug-in, which enables control over personal data sharing with adtech entities. As part of this setup, the user specifies: i) the types of data she is willing to share; ii) the processing purposes she consents to by default; and iii) whitelisted or blacklisted vendors from the IAB Global Vendor List.

If a new entity, not previously approved, seeks access to user data for a specific purpose, it must first request explicit manual consent.

To help users assess their privacy exposure, the plug-in evaluates the worst-case scenario profile that could be constructed from the selected data. It then generates a privacy risk score ranging from 0 to 1, offering a clear, quantifiable measure of potential data exposure.

Since adtech entities use user profiling to categorize audiences for targeted advertising, understanding how shared data influences profiling is essential. The proposed system leverages this information to compute exposure levels, helping users make informed privacy decisions. The next subsections detail how user interests are structured, how the specificity of shared data is measured, and how this data is processed within the system to assess potential privacy risks.

User Profiling in Programmatic Advertising.

User profiles in programmatic advertising are structured as topic distributions that reflect interests derived from browsing behavior. These profiles are built using elements such as keywords, website titles, and metadata, mapped to IAB's Audience Taxonomy (Section 5.1.4).

Our approach, based on Information Theory, measures the specificity of each taxonomy element using Inverse Document Frequency (IDF)[3]. The web, as the largest publicly available corpus of human knowledge, serves as a reference point to estimate the relative rarity of terms. The frequency with which a term appears across indexed web pages determines its specificity—less frequent terms provide more precise insights into user interests, while widely used terms offer broader categorizations [85].

For example, general terms such as "health" appear frequently across web content, making them less specific and revealing only broad user interests. In contrast, rare terms such as "lung cancer" occur far less often, making them more specific and offering deeper insights into user preferences.

[3]TF-IDF in NLP: Term Frequency-Inverse Document Frequency, `https://medium.com/@abhishekjainindore24/tf-idf-in-nlp-term-frequency-inverse-document-frequency-e05b65932f1d`, Accessed: 5 October 2024.

To compute specificity, we analyze the hits (H) from web search engines. The IDF value for a term k is calculated as:

$$IDF(k) = -log(\frac{HC(k)}{total_webs}) \tag{5.1}$$

where $HC(k)$ is the number of search results for term k ($HC(k) = |H(k)|$), and $total_webs$ is the estimated number of indexed websites (e.g., 400 billion in Google[4]).

To normalize specificity across different taxonomy levels, we define the Specificity (SP) metric as:

$$SP(k) = \frac{IDF(k)}{-log(\frac{1}{total_webs})} = \frac{-log(\frac{HC(k)}{total_webs})}{-log(\frac{1}{total_webs})} \tag{5.2}$$

This normalization ensures that specificity values remain within a consistent range, making it possible to compare different taxonomy elements effectively.

Hierarchy-Based Adjustments.

The taxonomy used in this work follows a hierarchical structure, where higher-level categories are more general and lower-level categories are more specific. Consequently, the specificity of a non-leaf category depends on its own frequency as well as the aggregated contributions of its subcategories.

Since we are using hit counts to measure specificity, the HC value of a non-leaf category must be adjusted to reflect its overall presence in search

[4]Google's Index Size Revealed, 400 Billion Docs: `https://zyppy.com/seo/google-index-size/`

results. Ideally, this would involve computing the union of unique search hits from all its subcategories along with its own hits:

$$HC(k) = \left| \left[\bigcup_{k' \in sonsOf(k)} H(k') \right] \cup H(k) \right| \tag{5.3}$$

However, accurately computing this union is computationally infeasible, as search results often overlap across subcategories or redirect to identical web content. To approximate this value efficiently, we define the adjusted hit count of a non-leaf category as:

$$HC(k) = |H(k)| + \max_{k' \in sonsOf(k)} |H(k')| \tag{5.4}$$

which follows the assumption:

$$HC(k) = \left| \left[\bigcup_{k' \in sonsOf(k)} H(k') \right] \cup H(k) \right| \geq \max_{k' \in sonsOf(k)} |H(k')| \tag{5.5}$$

This approximation ensures that higher-level categories retain a meaningful specificity score while remaining computationally efficient. The adjusted HC values are then used to compute IDF, ensuring that hierarchical dependencies are accounted for in the final specificity calculation.

Privacy Awareness and Exposure Calculation.

The plug-in uses this specificity metric to help users assess their profile exposure when selecting allowed data categories. Once a user defines a

personalized subset (P), their overall profile specificity (SP') is recursively computed:

$$SP'(k) = \begin{cases} \sum\limits_{k' \in sonsOf(k) \cap P} SP'(k'), & \text{if } NOT \text{ areLeafs}(sonsOf(k)) \\ \sum\limits_{k' \in sonsOf(k) \cap P} SP(k'), & \text{if areLeafs}(sonsOf(k)) \wedge \\ & |sonsOf(k) \cap P| < |sonsOf(k)| \\ SP(k), & \text{if areLeafs}(sonsOf(k)) \wedge \\ & |sonsOf(k) \cap P| = |sonsOf(k)| \vee \\ & |sonsOf(k) \cap P| = 0 \end{cases} \tag{5.6}$$

The final value is normalized between 0 and 1, enabling users to understand their privacy exposure level before consenting to data sharing.

5.2.4 Phase-2: Web Browsing & Consent Enforcement

When a user accesses a website with ad slots served through programmatic advertising, the publisher must obtain user consent before collecting and transmitting personal data to adtech entities.

In the proposed solution, instead of requiring the user to manually accept the publisher's terms via a TCF pop-up, the plug-in automatically provides the publisher with the user's preset privacy preferences. If the publisher agrees to these conditions, the plug-in generates a *Data-Collection Consent*, granting the user access to the content while allowing the publisher to collect personal data.

To share collected data with other RTB entities (DSPs, Ad Exchanges, and SSPs), the publisher must establish a *Data-Processing Consent* for

each. If an entity is whitelisted in the plug-in's IAB Global Vendor List, the publisher generates the corresponding consent automatically. Otherwise, the publisher must request explicit user consent through the plug-in's API.

Both *Data-Collection* and *Data-Processing Consents* are encoded as Smart Contracts (SCs) on the Blockchain (see Section 3.2.3). The plug-in manages all consents on behalf of the user, ensuring compliance. These SC references replace the TCF's TC string, which has been deemed non-compliant with current legislation.

Additionally, SSPs may enrich user data using Data Management Platforms (DMPs). In such cases, DMPs must have pre-obtained user consent to lawfully collect data, and SSPs must establish a *Data-Processing Consent* before using it, as detailed in Section 3.2.4.

Once all necessary consents are in place, the RTB protocol proceeds without modification, except when an SSP requests additional user data beyond what is included in the bid request. In this scenario, the SSP must verify a valid *Data-Processing Consent* between the user and the DMP supplying the data. The DMP, in turn, uses the proposed XACML extension to confirm consent validity before requesting the Data Recipient (DR) to provide the user's stored personal data.

Throughout this process, the plug-in records bid impressions for all displayed ads. The logged data includes: i) the winning advertiser's ID; ii) the user data shared by the publisher; iii) the amount paid for the impression; and iv) the reference to the Data-Collection Consent between the user and the publisher.

5.2.5 Phase-3: Consent Management & Data Accountability

During this phase, users can access and manage all generated consents, monitoring which entities collect and process their personal data and for what purposes. They can also modify or revoke these consents through a smartphone application, which interacts with Smart Contracts deployed on the Blockchain to ensure transparency and enforcement.

For data accountability, the plug-in records all displayed ads as the user browses the web. This enables users to verify whether advertisers had proper consent to use their data for targeted advertising.

Additionally, Supervisory Authorities can audit these records at any time to ensure adtech entities comply with legal requirements and operate within GDPR regulations.

5.3 Discussion

Section 2.3 defined key GDPR compliance requirements. This section evaluates their fulfillment through four propositions, each backed by supporting claims.

Proposition-1: User consent for data collection and processing is explicit, informed, and specific to user preferences

Upon installing the plug-in, users set their privacy preferences and assess their data exposure. These preferences generate a *Data-Collection Consent* when accessing services that request personal data for ads. A *Data-Processing Consent* ensures additional entities (e.g., SSPs, DSPs, Ad Exchanges) obtain explicit user authorization before accessing the data.

This proposition directly supports requirement R1, which mandates that consent be explicit, informed, and freely given. It is validated through the following claims:

Claim 1. *User consent preferences are explicitly defined during the plug-in setup.*

Proof. Upon installation, the plug-in requires users to configure their privacy settings via a user-friendly interface. Users specify: i) which personal data categories may be collected; ii) the duration for which data may be retained; and iii) which vendors from the IAB's Global Vendor List are pre-approved for data processing. Additionally, the plug-in provides transparency by evaluating and displaying the specificity of selected data categories, helping users understand their potential exposure based on the methodology in Section 5.2.3. □

Claim 2. *A new Data-Collection Consent is generated whenever a user accesses an online service that collects personal data for advertising purposes.*

Proof. When a user visits a website that requires personal data for ad targeting, the plug-in ensures that the publisher adheres to the user's predefined privacy preferences. If the publisher agrees to these conditions, the plug-in generates a *Data-Collection Consent* contract between the user and the publisher. This contract specifies: i) the types of personal data the publisher may collect; ii) the retention period; and iii) the identity of the DR responsible for storing the collected data. This contract serves as explicit proof that the user has consented to data collection under the specified conditions. □

Claim 3. *A new Data-Processing Consent is created whenever an additional entity requests to process user data collected by the publisher.*

Proof. If an SSP, DSP, Ad Exchange, or advertiser requests access to user data collected by the publisher, a *Data-Processing Consent* contract must be established. This contract specifies: i) the subset of personal data requested; ii) the processing purpose; and iii) the duration for which the data will be processed. The consent is linked to the original *Data-Collection Consent* to ensure that data usage remains traceable and compliant with user preferences. □

Claim 4. *Data-processing consent requires explicit or implicit user approval.*

Proof. When a publisher creates a *Data-Processing Consen*t request for an entity, the plug-in checks whether the entity is whitelisted and whether the requested data aligns with the user's predefined preferences. If both conditions are met, the consent is automatically validated. Otherwise, the user must manually approve or reject the request through a notification generated by the plug-in. This ensures that no personal data is shared beyond the user's explicit authorization. □

Proposition-2: Generated Consents Maintain Integrity and Authenticity Throughout Their Lifecycle

The *Data-Collection* and *Data-Processing Consents* referenced in Claims 2 and 3 are stored as Smart Contracts on a public Blockchain. Public Blockchains are inherently tamper-resistant, ensuring transparency, integrity, and authenticity. Additionally, users interact with these contracts through asymmetric cryptographic key pairs, reinforcing security and non-repudiation.

This proposition directly supports requirement R2, which mandates that the integrity and authenticity of consents must be preserved. It is validated by the following claim:

Claim 5. *All Data-Collection and Data-Processing Consents established between the involved parties are securely stored on a public Blockchain using immutable smart contracts.*

Proof. In the proposed system, *Collection Consent Smart Contracts* and *Processing Consent Smart Contracts* encode the respective agreements. Once deployed on the Blockchain, these contracts inherit its immutability, authenticity, integrity, and irreversibility properties, ensuring that consents cannot be altered or forged [74]. □

Proposition-3: Users retain full control over their consents and GDPR rights

The proposed system allows users to manage consents via smart contracts, enabling them to: i) access their data; ii) request deletion; iii) control data collection and processing; iv) revoke consent anytime. As stated in Claim 5, consents are stored in Smart Contracts, providing users with methods to exercise their rights. The DR enforces access control through an XACML extension, verifying permissions recorded on the Blockchain. *Collection-Consent Smart Contracts* encode agreements between users (data owners) and publishers (data collectors), ensuring secure and transparent access.

This proposition supports requirement R3, ensuring users can easily access, delete, and withdraw consent. The following claims support this:

Claim 6. *Smart contracts enable user rights management.*

Proof. The *Collection and Processing Consent Smart Contract*s implement methods that allow users to: i) revoke a *Data-Collection* or *Data-Processing Consent* (i.e., revokeConsent()); ii) modify which personal data

can be collected or processed (i.e., modifyData()); and iii) request the DC to erase stored data (i.e., eraseData()). □

Claim 7. *The DR enforces access control to user data through the XACML extension.*

Proof. The DR regulates access to user data using an XACML extension, ensuring compliance with granted permissions. The Blockchain functions as the Policy Administration Point (PAP), maintaining access rights immutably. When a request is made: i) the Policy Enforcement Point (PEP) forwards it to the Policy Decision Point (PDP) for validation; ii) the PDP retrieves the corresponding *Collection-Consent SC* from the Blockchain, verifies the user's public key (PK_{user}), and determines whether the requesting entity has the necessary permissions. This process guarantees that data access is only granted in accordance with the user's established consent. □

Proposition-4: Users have direct access to their consents and ad tracking records

The proposed system allows users to review and manage all granted consents and collected data via the smartphone app (Claims 2, 3 and 7), ensuring compliance with their rights (Claim 6). Additionally, the plug-in logs ad impressions, summarizing displayed ads, involved entities, and personal data used for targeting. This transparency enables users to monitor data usage and detect potential misuse. If unauthorized processing is suspected, they can request an audit with a Supervisory Authority under GDPR.

This proposition supports requirements R6 and R7 by ensuring users have access to records of visited publishers, trackers, and adherence to

their preferences across the adtech ecosystem. It is supported by the previous claims as well as the following:

Claim 8. *The Plug-in Captures All Bid Impressions and Related RTB Data.*

Proof. When a user visits a website displaying ads via RTB, the plug-in records bid impressions and associated data, including the advertiser, participating entities, ad category, and user data used in the selection process. This information is summarized and made accessible through the plug-in's interface, allowing users to review ad delivery details in a clear and structured manner. □

Chapter 6

Multi-Platform Wallet for Privacy Protection and Key Recovery in Decentralized Applications

Although our contributions in Chapters chapters 3 to 5 successfully tackles the management of user consent according to current legislation on different areas, all proposals require generation and management of cryptographic keys to interact with the different systems. In this chapter, we propose a fully decentralized multi-platform wallet designed to meet the key management requirement. This innovative wallet enables users to securely interact with Decentralized Applications built on Blockchain smart contracts while maintaining privacy and ensuring easy key recovery in the event of device theft or damage. The system utilizes Blockchain

Technology and the InterPlanetary File System to manage asymmetric keys and facilitate key recovery. It is economically feasible, with a detailed cost analysis, and demonstrates strong resilience against security and privacy threats. A comparative analysis shows how this new approach outperforms existing mainstream and cutting-edge solutions. Finally, we present an initial prototype implementation to demonstrate the system's practicality and feasibility.

This proposal is supported by a publication showing an initial version of the system in [86], which has been reformed and extended in a Journal in [87].

The remainder of this chapter is organized as follows. Section 6.1 introduces the different wallet technologies that actually exist, the protocols used by the development of our proposal and and a brief explanation of how does the IPFS works. Section 6.2 introduces the new proposal and its novelty. Section 6.4 analyzes how the new proposal fulfills the security and privacy requirements considered in our scenario. Finally, Section 6.7 details how we have implemented the proposed system.

6.1 Preliminaries

This section introduces a brief overview of the primary technologies and concepts employed in the contribution of this chapter.

6.1.1 ECDSA

Elliptic Curve Digital Signature Algorithm (ECDSA) [88] consists of a variant of the Digital Signature Algorithm (DSA) using elliptic-curve

cryptography.

The signature process begins by having the two parties, Alice and Bob, agree on the curve parameters ($CURVE$, G , n). In addition to the field and equation of the curve, they need G, a base point of prime order on the curve; being n the multiplicative order of the point G. Assuming that every nonzero element of the ring $\mathbb{N}/n$ is invertible, $\mathbb{N}/n$ must be a field. It implies that n must be prime according to Bézout's identity.

Once the parties have agreed on the parameters, Alice creates a key pair. This consists of a private key integer d_A, randomly selected in the interval $[1, n-1]$; and a public key curve point $Q_A = d_A \times G$ ($\times$ denotes the elliptic curve point multiplication by a scalar).

Then, for Alice to sign a message m, she performs the following steps:

1. Calculate $h = H(m)$, where $H()$ is a cryptographic hash function with output converted into an integer.
2. Let z be the L_n leftmost bits of h, where L_n is the bit length of the group order n (z can be greater than n, but it cannot be longer).
3. Select a cryptographically secure random integer k from $[1, n-1]$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$, go back to step 3.
6. Calculate $s = k^{-1}(z + rd_A) \bmod n$. If s = 0, go back to step 3.
7. The signature is the pair (r, s). (And ($r, -s \bmod n$) is also a valid signature.)

Once the signature process has been completed and Alice has sent $m, (r, s)$ to Bob, he can authenticate the signature by using a copy of her

public-key curve point Q_A. But first, Bob must verify that Q_A is a valid curve point as follows:

1. Check that Q_A is not equal to the identity element O, and its coordinates are otherwise valid.
2. Check that Q_A lies on the curve.
3. Check that $n \times Q_A = O$.

After that, Bob authenticates the signature as follows:

1. Verify that r and s are integers in $[1, n-1]$. If not, the signature is invalid.
2. Calculate $h = H(m)$, where $H()$ is the same function used in the signature generation.
3. Let z be the L_n leftmost bits of h.
4. Calculate $u_1 = zs^{-1} \bmod n$ and $u_2 = rs^{-1} \bmod n$.
5. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$. If $(x_1, y_1) = O$ then the signature is invalid.
6. The signature is valid if $r \equiv x_1 \bmod n$, invalid otherwise.

Note that an efficient implementation would compute inverse $s^{-1} \bmod n$ only once. Also, using Shamir's trick, a sum of two scalar multiplications $u_1 \times G + u_2 \times Q_A$ can be calculated faster than two scalar multiplications done independently.

On the signature process, it is not only required for k to be secret, but it is also crucial to select different k for different signatures. Otherwise, an

attacker can find d_A as: given two signatures (r, s) and (r, s'), employing the same unknown k for different known messages m and m', an attacker can calculate z and z'. Since $s - s' = k^{-1}(z - z')$, the attacker can find $k = \frac{z-z'}{s-s'}$. Finally, using the equation in step 6, $s = k^{-1}(z + rd_A)$, the attacker can calculate the private key $d_A = \frac{sk-z}{r}$. All operations in this paragraph are done modulo n.

From now one we denote the key pair (d_a, Q_A) as (SK, PK) for the private and the public key respectively.

CURVE	the elliptic curve field and equation used
G	elliptic curve base point, a point on the curve that generates a subgroup of large prime order n
n	integer order of G, means that $n \times G = O$, where O is the identity element
d_A	the private key (randomly selected)
Q_A	the public key $d_A \times G$ (calculated by elliptic curve)
m	the message to send

Table 6.1: ECDSA Parameters

6.1.2 Crypto wallets

In the domain of Blockchain transactions, as previously stated, the individual initiating the transaction validates it by signing with a private key, while other nodes utilize public keys for transaction verification. These private keys serve as the “passwords" for the transaction address and must be securely stored to prevent unauthorized access and misuse of linked assets. Crypto wallets serve as the means to achieve this security.

Crypto wallets are hardware or software applications used for storing private keys essential for transactions while also functioning as client software enabling Blockchain operations. By means of these wallets, individuals can seamlessly manage their assets published on the Blockchain,

including cryptocurrencies and smart contracts, leveraging their private keys to authorize transactions. Furthermore, selected crypto wallets extend their utility by granting access to dApps and assorted decentralized services, thereby offering users a comprehensive ecosystem for managing and interacting with their digital resources.

A wallet comprises private keys, serving as both the repository for these keys and the primary interface of crypto clients. This interface traditionally integrates various functionalities beyond key storage, including transaction history, address book management, and currency exchange rate tracking.

According to [49] and [89], there are various wallet implementations leveraging different technologies. This diversity permits us to categorize them into the following groups:

- *Web (online) wallets*: Services accessible to users through a browser interface, eliminating the need for downloading or installation. One advantageous aspect of these services is their universal accessibility, enabling users to connect from any location and device via the internet. However, while many web wallets offer users control over their keys, the management and storage of cryptographic keys typically rest with a third party. Consequently, the security of these web wallets relies on the provider's handling of these keys and the level of trust users place in them.
- *Mobile wallets*: Mobile applications available on Android and iOS platforms where cryptographic keys are generated and securely stored. These wallets are often considered more secure than online counterparts, offering users greater control over their keys. However, they require regular data backups to prevent the complete loss of cryptographic keys in the event of device malfunction, loss, or security

breach.

- *Desktop wallets*: Applications installed on personal computers provide users with complete control over their cryptographic keys. These keys are typically stored within an encrypted file, safeguarded by a private password known only to the user. Maintaining the secrecy and safety of this password is paramount, as its loss would entail losing access to all private keys and associated assets stored within the Blockchain.
- *Hardware wallets*: Offline devices, such as USB drives or physical sheets with printed private keys, offer enhanced security. By storing private keys offline, they mitigate the risk of hacking. Additionally, many of these devices are equipped for online transactions, featuring compatibility with various internet interfaces.

On the other hand, [90] and [91] further categorize wallet implementations into more specific classifications, considering not only the devices used to store keys but also the methods by which these keys are managed:

- *Keys in local storage*: Software manages multiple private keys by storing them in the device's local storage, often within a file or database. When initiating a new transaction, a Blockchain client can access these keys and broadcast the transaction across the network. This approach offers several advantages, including the elimination of cognitive overhead for users, as the software manages key access. Moreover, this system can generate and store a virtually unlimited number of keys due to their small size. However, the reliance on local storage poses various security threats to consider, such as unauthorized access by other applications, potential malware attempting to breach the keys file, and risks associated with physical theft, loss, or equipment malfunction.

- *Password-protected (encrypted) wallets*: Private keys are stored locally, similar to the preceding type of wallet, yet the wallet file performs encryption using a key derived from a user-selected password or passphrase. This measure mitigates the risk of physical theft, although it requires the user's recall of the password used for encrypting the keys. If the password is forgotten, access to all Blockchain assets associated with the aforementioned keys is lost. Additionally, it is worth mentioning that a user cannot access the keys on a new device by simply entering the password, she must also transfer the encrypted wallet file to this new device.

- *Offline storage of keys*: Keys are stored offline on some form of portable media, such as a USB or a piece of paper (as in previously explained *Hardware wallets*). While this approach offers the advantage of resilience against malware attacks, it renders the wallet inaccessible for immediate software use, preventing users from accessing their assets in a easy and comfortable way.

- *Air-gapped key storage*: Similar to the previous implementation, wallets are stored on a secondary device capable of generating, signing, and exporting transactions without ever being connected to a network. This method enhances resistance to key theft, as the keys are never directly exposed on an internet-connected device.

- *Password-derived keys*: Cryptographic keys are generated based on a password chosen by the user. However, a limitation of this approach is that it only creates one keypair, requiring the user to choose a new password for each new keypair or adopt a more robust solution, such as the *Bitcoin Improvement Proposal 32 (BIP-32)* [81], i.e., hierarchical deterministic wallets.

- *Hosted wallet*: Private keys are managed by a third-party entity, allowing users to access their keys or transactional functionalities

via a hosted-wallet web service using standard web authentication methods like passwords or two-factor authentication. Additionally, providers of hosted wallet web services may offer smartphone applications as clients, which offer the advantage of simplicity compared to web-based interfaces. However, similar to Web wallets, the primary concern with this approach lies in the security of cryptographic keys, which relies on the provider's practices and the level of trust users place in them.

Bitcoin Improvement Proposal 39 (BIP-39)

BIP-39 [80] describes the implementation of a mnemonic code, which is a series of easily memorable words, for generating deterministic wallets. Employing mnemonic codes or phrases enhances the usability of managing a wallet seed for humans compared to dealing with raw binary or hexadecimal representations. The protocol consists of two main steps: i) generating a mnemonic code; and ii) converting these words into a binary seed. This seed can later be used to generate deterministic wallets using BIP-32 or similar methodologies.

The aforementioned steps work as follows:

1. For the generation of the mnemonic code a 32 bits multiple entropy variable ENT is generated (size between 128 and 256 bits): $Gen(128 - 256) \rightarrow ENT$.

2. Next, a checksum is generated by calculating the *SHA256* hash of ENT and taking the first size-of-ENT divided by 32 bits: $CS = Parse_{(|ENT|/32)}(Hash256(ENT))$

3. This checksum is appended to the end of the initial entropy and the resulting concatenated bits are split into groups of 11 bits: $[MS] = split_{11bits}(ENT|CS)$. Each one of these groups encodes a number from 0-2047, serving as an index into a wordlist.

4. Finally, these numbers are converted into words and used to form a mnemonic sentence. The mnemonic can be protected with a passphrase; if not, an empty string is used instead.

Table 6.2 illustrates the relation between the initial entropy length (ENT), the checksum length (CS), and the resulting length of the generated mnemonic sentence (MS) in words.

CS = ENT / 32
MS = (ENT | CS) / 11

ENT	CS	ENT+CS	MS
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24

Table 6.2: Length relation between ENT, CS and MS

To generate a binary seed, the PBKDF2 function is used with a mnemonic sentence (in UTF-8 NFKD) as the password, while using the mnemonic along with the passphrase (also in UTF-8 NFKD) as the salt. The iteration count is set to 2048, and HMAC-SHA512 serves as the pseudo-random function. The resulting derived key has a length of 512 bits (equivalent to 64 bytes).

Bitcoin Improvement Proposal 32 (BIP-32)

BIP-32 [81] establishes a standard for hierarchical deterministic wallets, enabling seamless interoperability across various clients while fully leveraging all generated key pairs. This specification consists of two parts: i) the methodology for deriving a hierarchical structure of key pairs from a single seed; and ii) a comprehensive illustration of constructing a wallet framework on top of this hierarchical structure.

In the protocol explanation, we assume the use of elliptic curve cryptography, specifically adhering to the field and curve parameters delineated in secp256k1 [92]. The summation of two coordinate pairs is defined as application of the EC group operation, while the employment of the HMAC-SHA512 function adheres to the specifications outlined in RFC 4231 [93]. Moreover, we adopt the following standard conversion functions:

- $point(p)$: returns the coordinate pair resulting from EC point multiplication (repeated application of the EC group operation) of the secp256k1 base point with the integer p.
- $ser_{32}(i)$: serialize a 32-bit unsigned integer i as a 4-byte sequence, most significant byte first.
- $ser_{256}(p)$: serializes the integer p as a 32-byte sequence, most significant byte first.
- $ser_P(P)$: serializes the coordinate pair $P = (x, y)$ as a byte sequence: $(0x02\ or\ 0x03)||ser_{256}(x)$ (i.e., EC1's compressed form), where the header byte depends on the parity of the omitted y coordinate.
- $parse_{256}(p)$: interprets a 32-byte sequence as a 256-bit number, most significant byte first.

The protocol comprises a function tasked with deriving a set of child keys from a parent key. To mitigate dependence solely on the parent key during derivation, both private and public keys are extended with an additional 256 bits of entropy. This extension, denoted as the chain code, remains consistent for corresponding private and public keys, spanning 32 bytes. The extended private key is denoted as (SK, c), while the extended public key is represented as (PK, c), where SK denotes the standard private key, $PK = point(SK)$, and c signifies the chain code. Each extended key accommodates 2^{31} normal child keys and 2^{31} hardened child keys, each with an assigned index. Normal child keys are allocated indices ranging from 0 to $2^{31} - 1$, whereas hardened child keys utilize indices from 2^{31} to $2^{32} - 1$. To simplify notation for hardened key indices, i_H symbolizes $i + 2^{31}$.

Regarding the child keys, a vulnerability has been identified within BIP-32-compliant wallets. More specifically, if an attacker gains access to the master public key along with any child private key, she can then reconstruct the master private key. However, BIP-32 introduces a safeguard against this vulnerability by permitting "hardened" child private keys, which can be compromised without jeopardizing the master private key. Additionally, these hardened keys lack the property of the master public key, which would otherwise enable anyone to derive a public child key from it. This property exposes a vulnerability wherein an attacker could discern the owner of a public child key through a brute force attack on the master public key.

The Child Key Derivation (CKD) algorithm depends on whether the child key is hardened or not (alternatively, whether $i \geq 2^{31}$), as well as whether we are referring to private or public keys. From a parent extended private key, a child extended private key and/or a child extended public key can be derived using CKD. Additionally, CKD can derive a child

extended public key from another parent extended public key. However, our focus in this study lies on deriving private child keys from private parent keys exclusively, generating only hardened child private keys to circumvent the aforementioned vulnerability and to sidestep the master public key property. Accordingly, child private keys are computed via the function $CKDpriv((SK_{par}, c_{par}), i) \rightarrow (SK_i, c_i)$ as follows:

1. Check whether $i \geq 2^{31}$ (i.e., whether the child key is a hardened).
   (a) If it is hardened: let $I = HMAC{-}SHA512(Key = c_{par}, Data = 0x00||ser_{256}(SK_{par})||ser_{32}(i))$ (the 0x00 pads the private key to make it 33 bytes long).
   (b) If it is not hardened (it is a normal child key): do $i+ = 2^{31}$ and go back to step 1.
2. Split I into two 32-byte sequences, I_L and I_R.
3. The returned child key SK_i is $parse_{256}(I_L) + SK_{par}\ mod(n)$.
4. The returned chain code c_i is I_R.
5. In case $parse_{256}(I_L) \geq n$ or $SK_i = 0$, the resulting key is invalid, and it is required to proceed with the next value for i. Note that this situation can happen with a probability lower than 1 in 2^{127}.

To start the execution of the derivation protocol, an initial key pair known as the master key is required. The generation of the master key involves the use of a seed value, employed in the following manner:

1. Generate a seed byte sequence S of a chosen length (between 128 and 512 bits; 256 bits is advised) from a pseudo-random generator. In this paper, we use the seed obtained from the BIP-39 protocol, as explained in Section 6.1.2.

2. Calculate $I = HMAC - SHA512(Key = \text{"}Bitcoinseed\text{"}, Data = S)$.

3. Split I into two 32-byte sequences, I_L and I_R.

4. Use $parse_{256}(I_L)$ as master secret key, and I_R as master chain code. If $parse_{256}(I_L) = 0$ or $parse_{256}(I_L) \geq n$, the master key will be invalid.

6.1.3 InterPlanetary File System (IPFS)

The InterPlanetary File System[1], introduced by Juan Benet in [94], is a protocol that defines a file sharing peer-to-peer network for storing and sharing data in a distributed file system. Employing content-addressing, IPFS assigns a unique identifier to each file within a global namespace, interlinking IPFS hosts. Unlike conventional centralized servers, IPFS operates on a decentralized model, wherein user-operators collectively host portions of the overall data, thereby offering resilience in file storage and sharing. Any participant within the network can store a file using its content address, enabling any other peer to locate and request the content from any node possessing it via a Distributed Hash Table (DHT).

In addition to its primary function as a data storage system, IPFS is frequently integrated with various other technologies, including Blockchain. For instance, in [95], Kumar et al. introduce an IPFS-based Blockchain storage model. Here, transactions are stored within the IPFS distributed file system, with only the transaction hash recorded in the Blockchain block. This approach effectively addresses the storage challenges arising from the increasing volume of Blockchain transactions.

However, in conventional P2P file-sharing network systems like IPFS, data stored in nodes remains immutable, as it can only be removed by

[1]InterPlanetary File System (IPFS): `https://docs.ipfs.tech/`

other nodes and not by the data owner. Such a characteristic may raise concerns when managing certain types of data. To address this challenge, Yeh et al. proposed a solution in [96]: a monitorable P2P file-sharing system built upon a consortium Blockchain, enabling file revocation within a decentralized environment. Their approach leverages a Trusted Execution Environment (TEE), such as Intel Software Guard Extensions (SGX). This TEE ensures the integrity of the P2P file-sharing system's executables and generates a file authentication code for each IPFS node, thereby ensuring correct synchronization of the system.

6.2 Proposal Description

In this section, we describe in detail our proposed fully decentralized multi-platform wallet, leveraging Blockchain and IPFS technologies to deliver its functionalities.

Initially, we introduce the set of requirements pivotal to the design of the new scheme. Next, we present the architecture of the proposed system. Following this, we describe the smart contract utilized for storing cryptographic keys. Finally, we provide a high-level overview of how the proposed system operates.

6.2.1 Requirements

In our comprehensive review of existing literature and in light of the identified weaknesses, we have identified some points that our multi-platform wallet must fulfill to effectively address these challenges.

Primarily, the new scheme must facilitate the generation and access

of key pairs across various devices while ensuring convenient management for users. Furthermore, emphasizing the paramount importance of trust, our objective is to develop a solution that operates autonomously, thereby eliminating dependence on third-party entities. Additionally, our approach prioritizes user privacy and security, carefully addressing concerns regarding key recovery procedures and potential cyber-attacks. This holistic strategy not only ensures seamless functionality but also provides a robust defense against vulnerabilities, fostering a secure and trustworthy environment for users.

Aligned with these objectives, we have crafted a series of functional, security, and privacy requirements, which are introduced in the subsequent sections.

Functional requirements

The proposed system must fulfill a set of functional requirements in order to address the challenges highlighted in the current literature:

R1.1. Users should have the capability to generate keys from various devices utilizing the provided wallet.

R1.2. Key pairs should always be accessible to users, including the option to export them from the wallet for use in other applications if desired.

R1.3. Users should have the capability to recover all cryptographic keys in the event of loss of access to the wallet.

R1.4. The solution must be economically viable for users.

Security and privacy requirements

The proposed system must fulfill a set of security and privacy requirements in order to be robust enough. These requirements are built on the premise that the software components of the proposed system are correctly implemented and run on personal devices (i.e., smartphones, laptops, etc.) whose operating systems natively apply protection measures. These measures prevent adversaries with physical access to those devices from logging in, retrieving or altering sensitive data, or modifying the application code in any way.

R2.1. Keys must be kept confidential from adversaries.

R2.2. Keys should be safe against tampering by attackers.

R2.3. Keys should not be kept by a third party, thus ensuring that the system does not rely on external trust.

R2.4. Keys should remain unlinked to each other or to a specific user, ensuring that all generated keys are independent.

6.2.2 System architecture

Figure 6.1 depicts the general overview of the proposed system, where the *Plug-in* and the *Smartphone App* components are especially relevant and serve as access points for the users.

The *Plug-in* component is required to be installed on web browsers through which users access Blockchain-based dApps. This plug-in is responsible for seamlessly generating cryptographic keys and deploying/utilizing smart contracts automatically and transparently for the user, facilitating the use of dApps.

The *Smartphone App* component serves as a smartphone application through which users can add and configure a new *Plug-in* installed on their device's browser. Additionally, this app allows users to recover all generated keys and access all deployed smart contracts, enabling seamless interaction with them.

The cryptographic key pairs generated by the *Plug-in* components are securely stored within the *Key Storage* component. This storage system leverages IPFS technology, making it fully decentralized. The IPFS solution enables key recovery by offering a distributed and secure storage layer that eliminates reliance on a central authority, without imposing any cost on users.

Interactions among the *Smartphone App*, the different *Plug-ins*, and the *Key Storage* system are facilitated through Blockchain-based smart contracts. Specifically, the setup of the *Key Storage* and all related key management interactions are governed by the *Secure Key Management Smart Contract*. Further details about this smart contract will be provided in the subsequent section.

6.2.3 Secure Key Management Smart Contract

The *Secure Key Management Smart Contract (SKM SC)* is a smart contract deployed on the Blockchain by each user of the proposed system. Its purpose is to enable user access to cryptographic keys stored on IPFS and facilitate information exchange between the browser plug-in and the smartphone during initial setup. To provide these functionalities, the smart contract comprises a single class that includes:

- *Smartphone app identifier*: This refers to the root public key of the smartphone ("smartphoneID" argument). This value is automatically configured during the deployment of the smart contract,

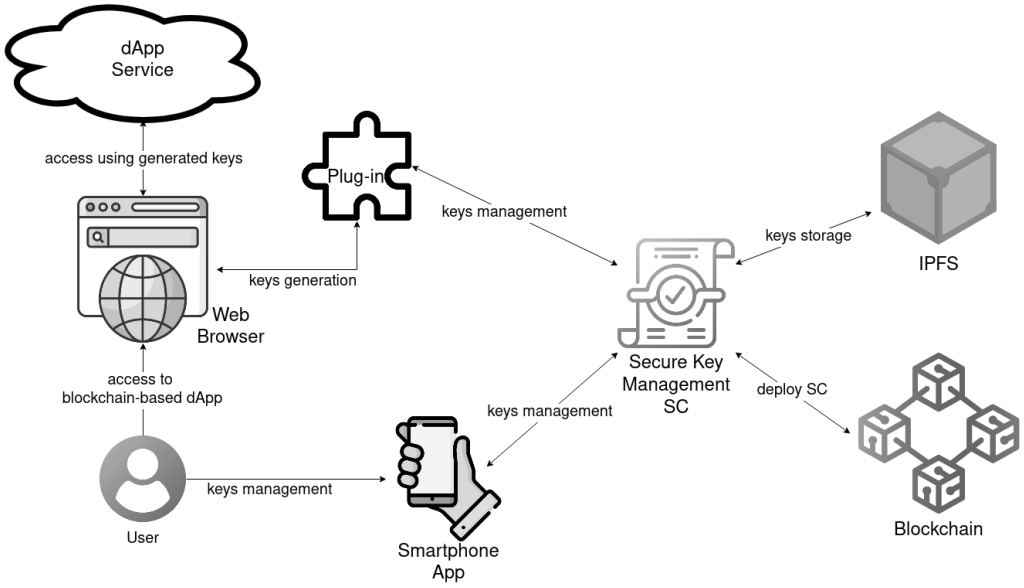


Figure 6.1: Decentralized Multi-Platform Wallet System's architecture

with the public key associated with the private key used to sign the deployment transaction. The public key serves as an encryption mechanism for all cryptographic keys generated by the various plug-ins.

- *Authorized plug-ins*: This comprises a list of public keys belonging to plug-ins authorized to interact with the smart contract, thereby adding references to new cryptographic keys stored in IPFS ("whiteList" argument).

- *References to cryptographic keys*: This mapping correlates the public keys of the plug-ins in the "whiteList" list with hash references to files stored in IPFS. These files contain encrypted cryptographic keys generated by each plug-in ("refsList" argument).

- *Temporal value*: This temporal data field serves as a medium for exchanging information between the Smartphone app and the plug-

ins ("temp" argument).

Once the *SKM SC* instance is created (Algorithm 1), the following methods are available for interaction:

- The *addDevice()* and *removeDevice()* methods (Algorithms 2 and 3) facilitate the addition or removal of a new public key to the "whiteList" list. These methods are exclusively accessible via the smartphone application, employing the private key associated with the public key specified in "smartphoneID" for transaction signing.
- The *storeRef()* method (Algorithm 4) updates the value in the mapping corresponding to the public key associated with the private key used for transaction signing. This value represents a reference to a file in IPFS where the keys generated by a plug-in are stored. Access to this method is limited to devices whose public keys are stored in the "whiteList" list or to the smartphone application.
- The *modTemp()* method (Algorithm 5) updates the data within the "temp" field. Access to this method is restricted to the smartphone application.

The size of the smart contract grows with the addition of plug-ins into the system. Specifically, for each registered plug-in, the "whiteList" argument acquires a new 20-byte public key, while the "refsList" argument includes a boolean value and an IPFS reference, totaling 33 bytes. In aggregate, these components require two 32-byte memory slots (i.e., 64 bytes) within the smart contract. Note that, the contract's size remains unaffected by the number of cryptographic keys generated by these plug-ins. It is considered that this increase in size does not pose scalability concerns because each user has their own smart contract, which in turn

manages only a limited number of devices used for internet browsing. According to Statista[2], the global average of devices and connections per person in 2023 was approximately 3.6.

Algorithm 1: *Constructor* generates a new Secure Key Management Smart Contract instance

Input: smartphone ID PK_{sp}, digitalSignature s_{sp}
Output: res
Initialisation: $res \leftarrow error$
1 **if** $verify(PK_{sp}, s_{sp})$ **then**
2 $smartphoneID \leftarrow PK_{sp}$;
3 $whiteList \leftarrow [\,]$;
4 $refsList \leftarrow < K, V >$;
5 $temp \leftarrow null$;
6 $res \leftarrow success$;
Return: res

Algorithm 2: *addDevice* adds a new public key to the "whiteList" list

Input: device ID PK_{device}, digitalSignature s_{sp}
Output: res
Initialisation: $res \leftarrow error$
1 **if** $verify(smartphoneID, s_{sp})$ **then**
2 $whiteList.add(PK_{device})$;
3 $res \leftarrow success$;
Return: res

6.2.4 The proposed system in a nutshell

The proposed wallet operates through four distinct protocols, which are briefly summarized below and will be explained in detail in the following section.

[2]Statista: `https://www.statista.com/statistics/1190270/number-of-devices-and-connections-per-person-worldwide/`, Accessed: 20 January 2024

Algorithm 3: *removeDevice* removes an existing public key from the "whiteList" list

Input: device ID PK_{device}, digitalSignature s_{sp}
Output: res
Initialisation: $res \leftarrow error$

```
1 if verify(smartphoneID, s_sp) then
2     if PK_device in whiteList then
3         whiteList.remove(PK_device);
4         res ← success;
```

Return: res

Algorithm 4: *storeRef* updates the IPFS reference in the mapping corresponding to the device that has invoked the method, or the position associated with the specified device if the method is invoked by the smartphone app

Input: device ID PK_{device}, digitalSignature s, IPFS reference $IPFSref$
Output: res
Initialisation: $res \leftarrow error$

```
1 if verify(PK_device, s) OR verify(smartphoneID, s) then
2     if PK_device in whiteList then
3         refsList[PK_device] ← IPFSref;
4         res ← success;
```

Return: res

Algorithm 5: *modTemp* modifies the data contained in the "temp" field

Input: temporal value $newTemp$, digitalSignature s_{sp}
Output: res
Initialisation: $res \leftarrow error$

```
1 if verify(smartphoneID, s_sp) then
2     temp ← newTemp;
3     res ← success;
```

Return: res

1. *Set-up*: The setup of the proposed wallet involves two steps. First, users must install and configure the provided *Smartphone App* on their smartphones. During this process, users receive a personal password and 24 mnemonic words that they must remember. Additionally, the Root cryptographic keys are generated, which are crucial for creating all subsequent cryptographic keys. Following this, users must install and configure the *Plug-in* in the browsers where they intend to use the wallet.

2. *Session key pair generation*: After completing the setup, users who wish to access Blockchain-based *dApps* must follow this protocol to generate the necessary key pairs. During this process, the browser's *Plug-in* automatically detects the intent to interact with a *dApp*, and transparently generates and stores the key pairs. These key pairs are securely stored in IPFS, enabling their recovery in case of loss. By utilizing a fully distributed storage technology like IPFS, the wallet eliminates the need for third-party services that may require trust.

3. *Key inventory*: Users utilize the *Smartphone App* installed on their smartphones to execute this protocol and manage all cryptographic key pairs generated by the wallet to date. Specifically, the *Smartphone App* displays each key pair associated with a particular *Plug-in* and *dApp*. Additionally, the wallet offers options to export a selected key pair in PKCS#12 format, send it via email, or transfer it to an external device.

4. *Key recovery*: This protocol enables users to recover all their cryptographic keys if their smartphones running the *Smartphone App* are compromised, lost, or damaged. To do this, users must reinstall the *Smartphone App* and use their personal password and 24 mnemonic words to regenerate their original Root cryptographic keys. With these Root keys, they can access the fully distributed IPFS to re-

cover all the derived keys used to interact with each *dApp*. If the files stored in IPFS are unavailable, the proposed wallet can use the Root keys to generate new keys and store them again in distinct locations within IPFS.

6.3 Protocols

In this section, we establish the formal protocols governing the proposed system, providing enough detail to facilitate their implementation. These protocols are: *Set-up*, *Session key pair generation*, *Key inventory*, and *Key recovery*.

6.3.1 Set-up

This protocol aims to install and configure both the *Smartphone App* and the web browser *Plug-in*. It is divided into two distinct subprotocols, each devoted to one of these components. It is worth mentioning that a *Smartphone App* is expected to handle multiple *Plug-in* components.

Setting-up the Smartphone App

Installing the *Smartphone App* involves the *User* and the app component itself to follow these steps (see Figure 6.2):

1. The *User* installs the *Smartphone App* in her/his smartphone and creates a new secure log-in, by specifying a username and a password *pw*.
2. The *Smartphone App* executes the BIP-39 protocol using *pw* as input, generating 24 mnemonic words and a 512-bit seed, known as the *BIP-39 seed*.

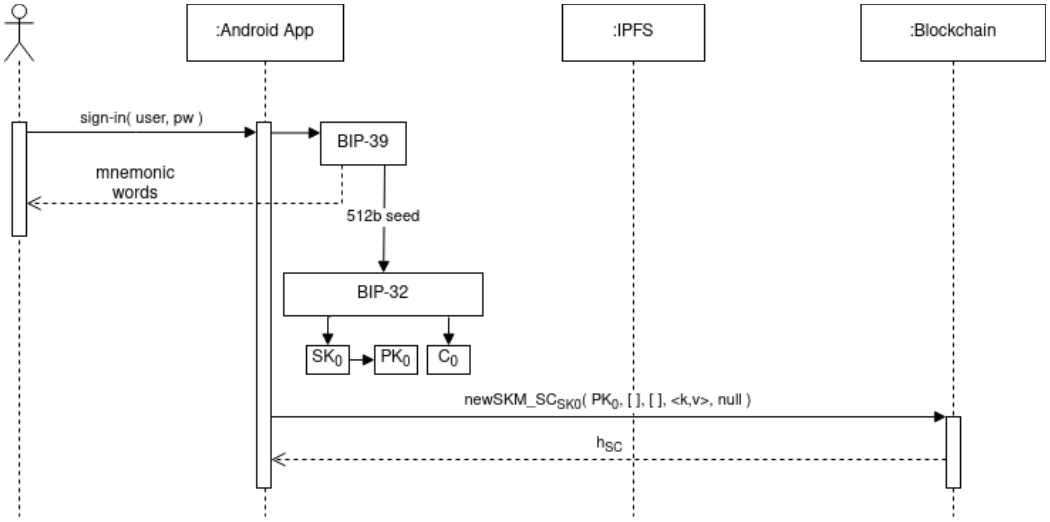


Figure 6.2: Set-up the Smartphone App

3. The *User* securely stores the 24 mnemonic words, as they are essential for key recovery in the event of compromise to either the smartphone or the key storage.

4. The *Smartphone App* executes the BIP-32 protocol using the *BIP-39 seed* as input. This process generates the Root Secret Key SK_0 and the Root Chain Code C_0.

5. The *Smartphone App* stores SK_0 in the smartphone's secure storage.

6. The *Smartphone App* derives the Root Public Key PK_0 as follows: $PK_0 = SK_0 \times G$ (see Section 6.1.1).

7. The *Smartphone App* uses the Root Key pair (SK_0, PK_0) to deploy a new *SKM SC* onto the Blockchain, resulting in the generation of a hash h_{SC} that uniquely identifies the published smart contract.

Setting-up the web browser Plug-in

Installing and configuring the *Plug-in* component in the *User*'s chosen browser requires the collaboration of these entities along with the already deployed *Smartphone App* to complete the following steps (see Figure 6.3):

1. The *User* installs the provided *Plug-in* in a browser of her/his choice.
2. The *User* introduces the h_{SC} (i.e., the *SKM SC* identifier) into the *Plug-in*.
3. The *Plug-in* retrieves PK_0 from the smart contract.
4. The *Plug-in* uses a secure symmetric cryptosystem (e.g., AES [97]) to generate a new session key K_i.
5. The *Plug-in* generates a QR image containing the computed session key K_i and the browser's identifier i.
6. The *Smartphone App* scans the QR image shown in the web browser by the *Plug-in* and retrieves K_i and i.
7. The *Smartphone App* employs SK_0, C_0, and the hashed browser's identifier, denoted as $SHA256(i)$, as inputs to the BIP-32 protocol. This process generates the Master Secret Key SK_i and the Master Chain Code C_i specific to that particular *Plug-in*.
8. The *Smartphone App* derives the corresponding Master Public Key PK_i as follows: $PK_i = SK_i \times G$ (see Section 6.1.1). This element unequivocally identifies that specific *Plug-in* component.
9. The *Smartphone App* adds PK_i to the authorized *Plug-in* components list of the *SKM SC* (referred to as the *whileList* argument) using the *addDevice()* method.

10. The *Smartphone App* uses the session key K_i to encrypt the set (SK_i, PK_i, C_i). This is, $Enc_{K_i}(SK_i, PK_i, C_i)$.

11. The *Smartphone App* stores the pair ($Enc_{K_i}(SK_i, PK_i, C_i)$, i) in the temporal data field of the *SKM SC* (referred to as the "temp" argument) using the *modTemp()* method.

12. The *Smartphone App* deletes the set (SK_i, PK_i, C_i) from its memory.

13. The *Plug-in* retrieves the pair ($Enc_{K_i}(SK_i, PK_i, C_i)$, i) from the "temp" argument of the *SKM SC*.

14. The *Plug-in* uses the session key K_i to decrypt $Enc_{K_i}(SK_i, PK_i, C_i)$, obtaining the set ($SK_i$, PK_i, C_i).

Note that each web browser the *User* intends to utilize requires its specific *Plug-in* component. Consequently, this sub-protocol must be performed for each necessary web browser *Plug-in*.

6.3.2 Session key pair generation

Upon completion of the *Set-up protocol*, users can seamlessly access various Blockchain-based *dApps*. To facilitate these accesses, the installed *Plug-in* in the web browser must automatically and transparently generate and manage the needed cryptographic keys. The steps involved in this process are next detailed (see Figure 6.4):

1. The *User* accesses a new Blockchain-based *dApp*.

2. The *Plug-in* computes a hash value based on the domain name j of the *dApp* service provider, which serves to uniquely identify

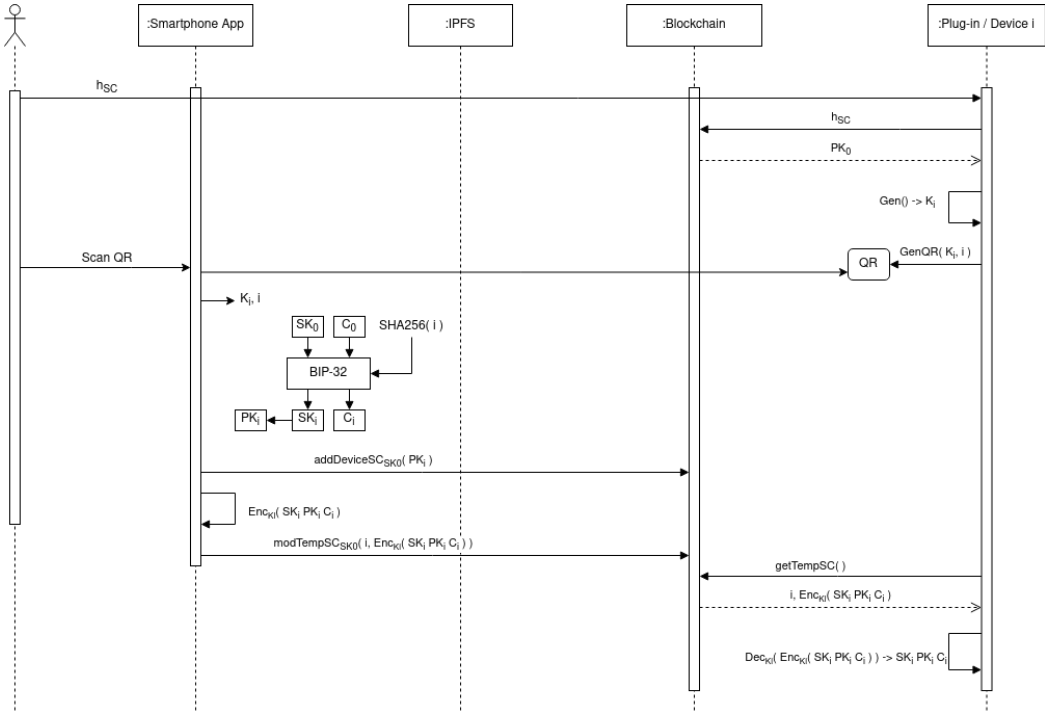


Figure 6.3: Set-up the web browser Plug-in

the specific *dApp*. This hash value is generated using the function $SHA256(j)$.

3. The *Plug-in* uses the browser's Master Private Key SK_i, Master Chain Code C_i, and the *dApp* identifier $SHA256(j)$ as inputs to the BIP-32 protocol. This process generates a Secret Key SK_{ij} and a Chain Code C_{ij} specific to that particular *dApp*. Then, the *Plug-in* derives the corresponding Public Key PK_{ij} as follows: $PK_{ij} = SK_{ij} \times G$ (see Section 6.1.1).

4. The *Plug-in* provides the key pair (SK_{ij}, PK_{ij}) to the *User*, facilitating seamless interaction with the dApp. While the detailed implementation of this aspect falls outside the scope of this paper, one possible approach could emulate the user interface of browser

plugins such as Metamask, presenting the key pair to the *User* and offering the choice between exporting it in [3] or uploading it to the *Key Storage* system.

5. The *Plug-in* uses a secure symmetric cryptosystem (e.g., AES [97]) to generate a new session key K_{ij}.

6. The *Plug-in* uses the session key K_{ij} to encrypt the set (SK_{ij}, PK_{ij}, j). This is, $Enc_{K_{ij}}(SK_{ij}, PK_{ij}, j)$.

7. The *Plug-in* uses the Root Public Key PK_0 to encrypt the session key K_{ij}. This is, $Enc_{PK_0}(K_{ij})$.

8. The *Plug-in* stores the set ($Enc_{K_{ij}}(SK_{ij}, PK_{ij}, j)$, $Enc_{PK_0}(K_{ij})$) in the IPFS. This process entails a series of automated sub-steps that facilitate subsequent access to the cryptographic keys by the *Smartphone App*.

   (a) The *Plug-in* obtains the hash reference h_i of the file stored in the IPFS, which contains all the encrypted cryptographic keys generated by the *Plug-in*. This retrieval is fulfilled by leveraging its identifier PK_i within the "refsList" mapping contained in its associated *SKM SC*. If the "refsList" field is empty, the procedure goes directly to step 8c, where the plug-in generates a new file with a new IPFS reference.

   (b) The *Plug-in* gathers the $file_{h_i}$ from the IPFS.

   (c) The *Plug-in* updates $file_{h_i}$ by appending the set

$$Enc_{K_{ij}}(SK_{ij}, PK_{ij}, j), Enc_{PK_0}(K_{ij})$$

[3] Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1: `https://datatracker.ietf.org/doc/html/rfc3447`

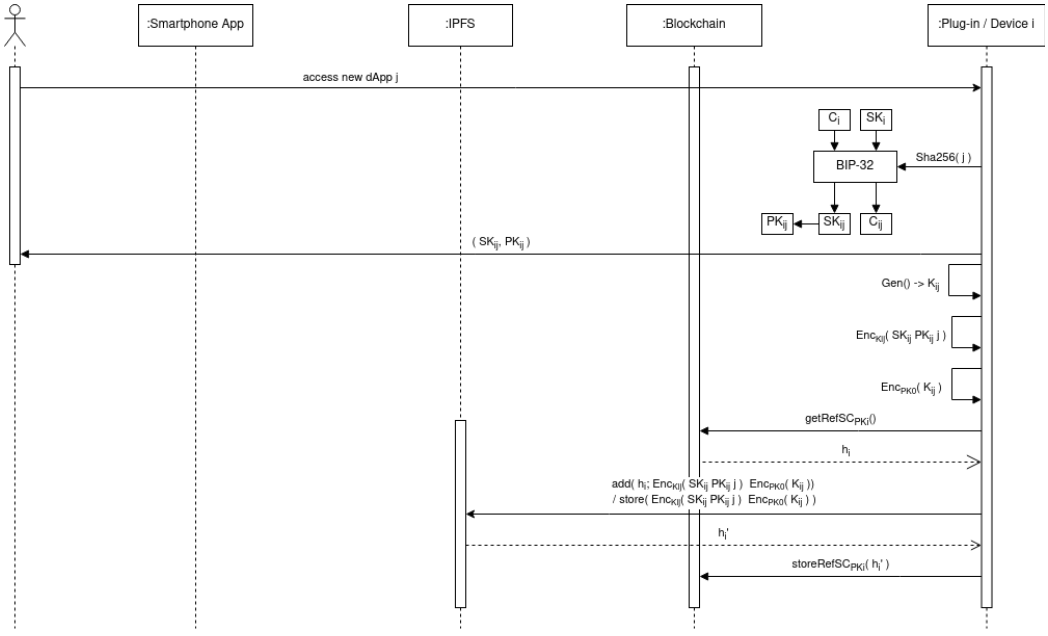


Figure 6.4: Session key pair generation

This process generates an updated reference h_i' for the stored file.

(d) The *Plug-in* uses the *storeRef()* method of its associated *SKM SC* to update the "refsList" mapping with the new reference.

6.3.3 Key inventory

This process enables the *User* to manage and monitor all cryptographic key pairs she has generated. To achieve that, the *User* uses the *Smartphone App* component, which follows the next steps (see Figure 6.5):

1. The *Smartphone App* periodically retrieves the "refsList" value from the *SKM SC*, which includes the hash references h_i to the files stored

in IPFS, each one linked to a certain *Plug-in* i and storing all encrypted cryptographic keys generated by this entity.

2. If a certain IPFS h_i has been modified, such as the reference h_i changing or a new file being added (when a *Plug-in* generates a new key pair for the first time), the *Smartphone App* retrieves the corresponding file $file_{h_i}$ from the IPFS. Else, the protocol goes to the first step.

3. The *Smartphone App* gets the encrypted keys linked to the updated $file_{h_i}$. This is, $Enc_{K_{ij}}(SK_{ij}, PK_{ij}, j)$, $Enc_{PK_0}(K_{ij})$.

4. The *Smartphone App* decrypts the session key K_{ij} using the Root Secret Key SK_0. This is, $Dec_{SK_0}(Enc_{PK_0}(K_{ij}))$.

5. The *Smartphone App* decrypts the set (SK_{ij}, PK_{ij}, j) using the session key K_{ij}. This is, $Dec_{K_{ij}}(Enc_{K_{ij}}(SK_{ij}, PK_{ij}, j))$. The decrypted set is stored in the *Smartphone App*'s file system.

6. The *Smartphone App* shows to the *User* each generated key pair (SK_{ij}, PK_{ij}) associated with a particular *Plug-in* i and a specific *dApp* j. Additionally, the system provides options for exporting a selected key pair in PKCS#12 format, sending it via email, or transferring it to an external device.

6.3.4 Key recovery

The *User* should be capable of recovering her/his cryptographic keys if her/his device is compromised, lost, or damaged. Accordingly, this section details the recovery process of the proposed new wallet covering three scenarios: i) the *User*'s smartphone is no longer accessible (e.g. stolen,

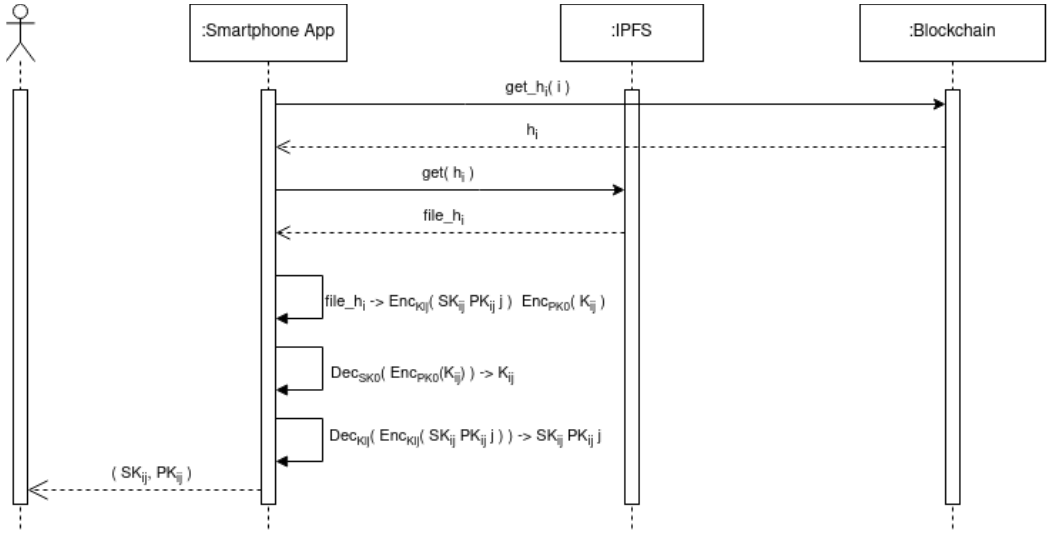


Figure 6.5: Key Management

lost, or broken); ii) the IPFS files that store the keys generated by the web browser plug-ins are deleted; and iii) both situations occur simultaneously.

Smartphone App is no longer accessible

When a *User*'s smartphone is compromised, lost, or damaged, access to her/his *Smartphone App* is forfeited. Consequently, she also loses access to the Root Key pair (SK_0, PK_0) necessary for decrypting and accessing all cryptographic keys generated by the wallet. In such an event, the recovery key process entails regenerating this Root Key pair through the following steps (see Figure 6.6):

1. The *User* installs the *Smartphone App* on a new smartphone device and establishes a new secure login, by setting up her/his username and password pw.

2. The *User* inputs the 24 mnemonic words generated during the *Set-up protocol* into the *Smartphone App*, along with the initial password *pw* used during the wallet's initial configuration

3. The *Smartphone App* runs the BIP-39 protocol using the provided 24 mnemonic words and the initial password *pw*, thereby generating the original 512-bit seed known as the BIP-39 seed.

4. The *Smartphone App* executes the BIP-32 protocol using the BIP-39 seed as input. This process generates the original Root Secret Key SK_0 and Root Chain code C_0.

5. The *Smartphone App* stores SK_0 in the smartphone's secure storage.

6. The *Smartphone App* derives the Root Public Key PK_0 by means of SK_0 as follows: $PK_0 = SK_0 \times G$ (see Section 6.1.1).

7. The *Smartphone App* fetches the *SKM SC* published on the Blockchain using the Root Key pair (SK_0, PK_0) and stores its hash h_{SC} in the app's storage.

8. The *Smartphone App* follows the procedure explained in Section 6.3.3 to retrieve all cryptographic key pairs previously generated.

IPFS file is unavailable

In the event that the IPFS nodes hosting the key pairs file generated from a specific browser i become inaccessible or delete the file, the *User* directs the *Smartphone App* to execute the following steps to regenerate the file (see Figure 6.7):

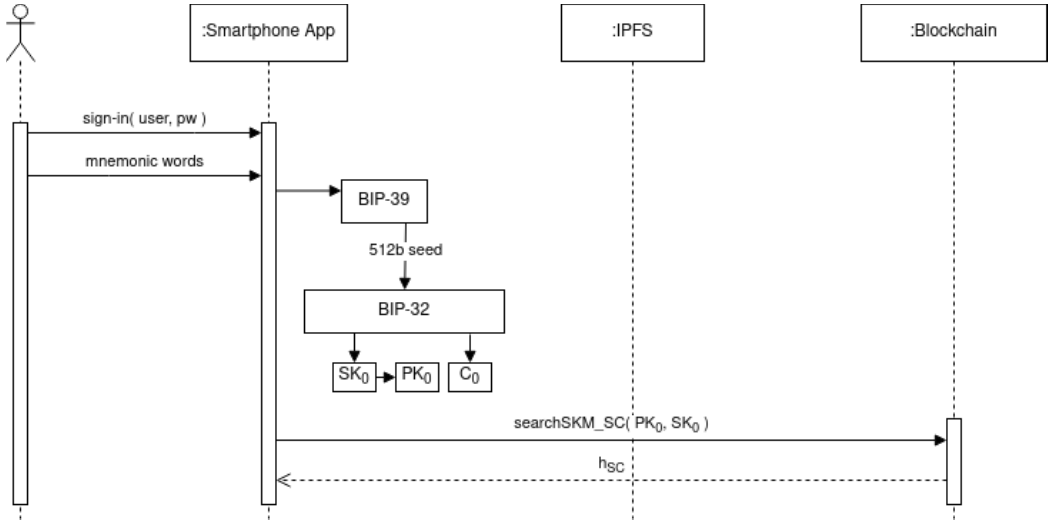


Figure 6.6: Key Recovery when Smartphone App is no longer accessible

1. The *User* activates the process to regenerate the session key pairs stored in the IPFS through the *Smartphone App*.

2. The *Smartphone App* generates a new session key K'_i.

3. The *Smartphone App* retrieves from its file system all the stored key pairs that the *Plug-in i* generated to interact with each distinct *dApp j*.

4. The *Smartphone App* uses the new session key K'_i to encrypt all the key pairs retrieved in the former step. This is, $Enc_{K'_i}(SK_{ij}, PK_{ij}, j)$

5. The *Smartphone App* encrypts K'_i using the Root Public Key PK_0. This is, $Enc_{PK_0}(K'_i)$.

6. The *Smartphone App* adds all the newly re-encrypted key pairs and the session key into a new file, denoted as $file'_i$.

7. The *Smartphone App* stores $file'_i$ in the IPFS. This process returns the reference h'_i to the stored file.

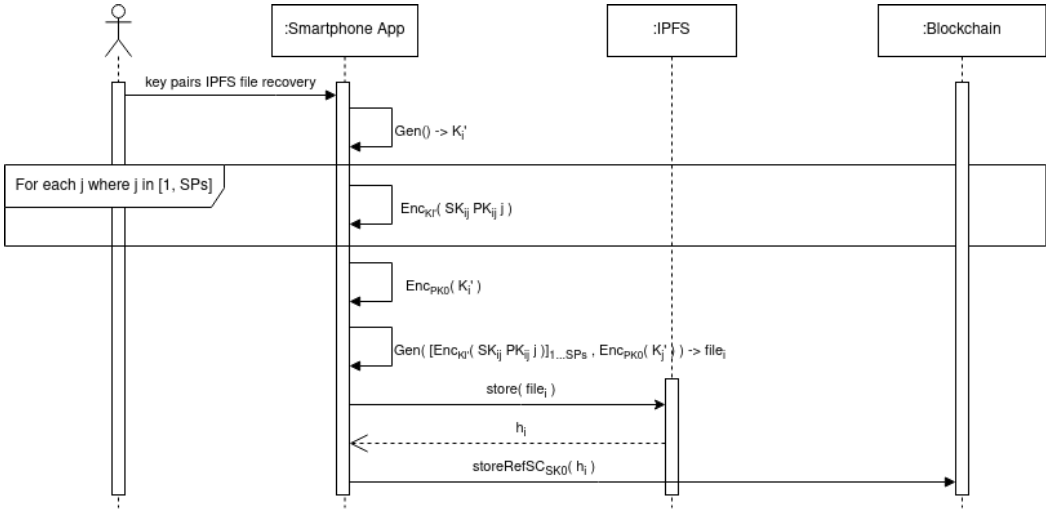


Figure 6.7: Key Recovery when IPFS file is unavailable

8. The *Smartphone App* utilizes the *storeRef()* method within the SKM smart contract to update the "RefsList" field linked to the *Plug-in i* with the fresh reference h'_i. This method is run through a transaction signed by the Root Secret Key SK_0, with the *Plug-in*'s public key PK_i provided as an argument.

Lost access to Smartphone's app and IPFS file is unavailable

In the event of the *User* losing access to the *Smartphone App* and simultaneous unavailability or removal of the key pairs file stored by a specific *Plug-in i* on IPFS nodes, the recovery process for each key pair generated for each *dApp j* proceeds as follows (see Figure 6.8):

1. The *User* initiates the *Key recovery* process detailed in Section 6.3.4 (i.e., "*Smartphone App* is no longer accessible") to reinstall the *Smartphone App* and regain access to the Root key pair (SK_0, PK_0).

2. The *Smartphone App* retrieves the identifier i associated with the *Plug-in* i whose IPFS key pairs file is unavailable from the *SKM SC*.

3. The *Smartphone App* retrieves the Master Secret Key SK_i and Master Chain code C_i for the specific *Plug-in* i via the BIP-32 protocol. This procedure uses the Root Private Key SK_0, the Root Chain code C_0, and the hash of the *Plug-in* identifier $SHA256(i)$ as input parameters.

4. The *Smartphone App* uses SK_i to derive the *Plug-in*'s Master Public Key PK_i as follows: $PK_0 = SK_0 \times G$ (see Section 6.1.1).

5. The *Smartphone App* generates a new Secret Key SK_{ij} and a new Chain code C_{ij} for each *dApp* j via the BIP-32 protocol. This procedure uses the *Plug-in*'s Master Private Key SK_i, the *Plug-in*'s Master Chain code C_i, and the hash of the *dApp* identifier $SHA256(j)$ as input parameters.

6. The *Smartphone App* uses SK_{ij} to derive the corresponding Public Key PK_{ij} as follows: $PK_0 = SK_0 \times G$ (see Section 6.1.1).

7. The *Smartphone App* shows to the *User* each generated key pair (SK_{ij}, PK_{ij}) associated with a particular *Plug-in* i and a specific *dApp* j. Additionally, the system provides options for exporting a selected key pair in PKCS#12 format, sending it via email, or transferring it to an external device.

6.4 Functional analysis

This section examines the functional requirements outlined in Section 3.2.1. The following discussion comprises a series of propositions, each ad-

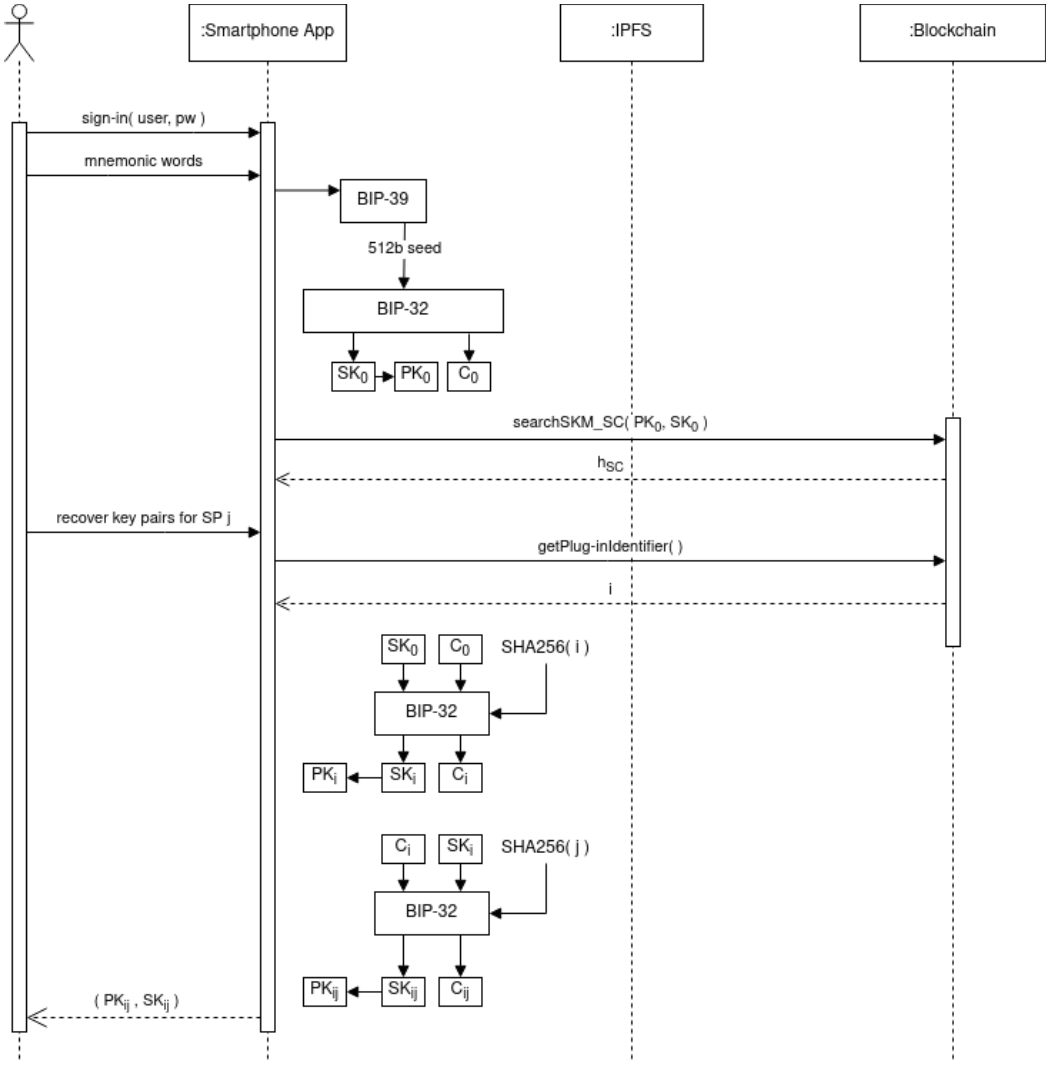


Figure 6.8: Key Recovery when Smartphone App is no longer accessible and IPFS file is unavailable

dressing a specific requirement, with multiple claims provided to support its fulfillment.

Proposition-1: Users must be capable of using different devices to generate their cryptographic keys

This proposition addresses requirement R1.1, which states that *Users* should have the capability to generate keys from various devices utilizing the proposed wallet. The following two claims support this assertion.

Claim 1. *The web browser Plug-ins are the components that generate the cryptographic keys needed to interact with the different dApps*

Proof. A *User* can install and configure a web browser *Plug-in i* on one of her/his devices by following the steps specified in 6.3.1. As a result of this process, *Plug-in i* acquires a Master key set (SK_i, PK_i, C_i). Next, when the *User* accesses a *dApp j*, *Plug-in i* uses SK_i, C_i and the hash of the *dApp*'s identifier $SHA256(j)$ as inputs in the BIP-32 protocol to compute a new key pair (SK_{ij}, PK_{ij}). This key pair is then used to interact with the specific dApp j. □

Claim 2. *Users have the flexibility to install and set-up a Plug-in within the web browser of each of their capable devices, allowing for multiple installations across different devices.*

Proof. As specified in the "Setup" protocol (see Section 6.3.1), each *User* owns a single *Smartphone App* operating within her/his smartphone. The *Smartphone App* is responsible for generating the 24 mnemonic words, the BIP-39 seed, the Root Keys (SK_0, PK_0), and the Root Chain Code C_0. Next, this cryptographic material is used during the installation of the distinct *Plug-ins* to compute the corresponding Master key sets (SK_i, PK_i, C_i) mentioned in the previous claim.

As shown in "Key inventory" protocol (see Section 6.3.3), the cryptographic keys generated by all *Plug-ins* linked to a particular *Smartphone*

App are directly managed by the *User* through the *Smartphone App* itself. □

Proposition-2: Users must have the capability to access their cryptographic keys at any given moment and utilize them on alternative platforms

This proposition addresses requirement R1.2, which states that key pairs must be available for *Users* at any time, also in the case they want to export them from the wallet and import them in another application. The following three claims support this assertion.

Claim 3. *All key pairs (SK_{ij}, PK_{ij}) generated by a User through her/his installed Plug-ins to interact with dApps are stored in the IPFS, with corresponding references stored and accessible via the SKM smart contract.*

Proof. As specified in the "Session key pair generation" protocol (see Section 6.3.2), each *Plug-in i* stores distinct key pairs (SK_{ij}, PK_{ij}) generated for interaction with various dApps j in a file $file_{h_i}$ on IPFS. Each key pair is stored in the format of the set ($Enc_{K_{ij}}(SK_{ij}, PK_{ij}, j)$, $Enc_{PK_0}(K_{ij})$). The reference h_i of $file_{h_i}$ in IPFS is stored in the "refsList" field of the publicly accessible SKM smart contract. □

Claim 4. *A User can access the key pairs generated by the different Plug-ins using her/his Root Key pair (SK_0, PK_0).*

Proof. As specified in the "Key inventory" protocol (see Section 6.3.3), a *User* can use her/his *Smartphone App* to access the "refsList" field of the deployed SKM smart contract. From there, she can retrieve the $file_{h_i}$ file stored in IPFS, obtain the encrypted key pairs ($Enc_{K_{ij}}(SK_{ij}, PK_{ij}, j)$, $Enc_{PK_0}(K_{ij})$), and decrypt them using the Root Secret key SK_0 which is only known by the *Smartphone App*. □

Claim 5. *A User can export a selected key pair in PKCS#12 format using her/his Smartphone App, with options to send it via email or transfer it to an external device.*

Proof. As seen in the previous claim, a *User* can retrieve the key pairs generated by the different *Plug-ins* through her/his *Smartphone App*. Furthermore, this tool provides options for exporting a selected key pair in PKCS#12 format, sending it via email, or transferring it to an external device (refer to the "Key inventory" protocol in Section 6.3.3). □

Proposition-3: Users must have the capability to recover their cryptographic keys in the event of loss of access to the wallet

This proposition addresses requirement R1.3, stating that the proposed system should enable *Users* to recover all cryptographic keys generated by the various *Plug-ins* in the event of smartphone loss or change, resulting in the loss of access to the *Smartphone App*. The following two claims supports this assertion.

Claim 6. *A User can recover her/his Root Key pair (SK_0, PK_0) in the event of smartphone loss or damage.*

Proof. According to the "Key recovery" protocol (see Section 6.3.4), a *User* who has lost access to her/his *Smartphone App* (for instance, due to smartphone loss or damage) can install a new instance of the *Smartphone App* on a new device. Then, by inputting the 24 mnemonic words and the password *pw*, she can re-generate her/his Root Key pair (SK_0, PK_0) using the BIP-39 and BIP-32 protocols. □

Claim 7. *A User can recover all the key pairs generated by the different Plug-ins using her/his Root Key pair (SK_0, PK_0).*

Proof. After regenerating the Root Key pair, the *User* can utilize it to recover all key pairs generated by the various Plug-ins, as depicted in the previous Claim 4 □

Proposition-4: The new system must be economically viable for users

This proposition addresses requirement R1.4, stating that the proposed system should incur low costs in terms of Blockchain gas consumption, ultimately leading to reduced economic costs. This factor is crucial, as it significantly impacts the feasibility of deploying the system in real-world scenarios.

To evaluate these costs, a realistic environment for the new wallet has been established. This setting is based on the following points:

- The personal Blockchain *Ganache*[4], a widely adopted platform for Ethereum distributed application development, has been employed. Ganache empowers developers to deploy Smart Contracts, execute commands, and inspect data state, all while controlling chain operations and associated costs.

- The SKM smart contract, coded in Solidity[5], has been compiled using the Truffle suite[6], a well-known ecosystem for Web3[7] development. This suite offers a comprehensive development environment, asset pipeline, and testing framework for Smart Contracts using the Ethereum Virtual Machine (EVM).

[4]Truffle Suite Ganache: `Ganache:https://trufflesuite.com/ganache/`
[5]Solidity Programming Language: `https://soliditylang.org/`
[6]Truffle suite: `https://trufflesuite.com/truffle/`
[7]web3j Library: `https://docs.web3j.io/`

- The new proposal has been deployed on the *Polygon Chain*[8] network, chosen after an analysis of some of the most relevant EVM-compatible networks concerning latency, throughput, and gas costs. Note that every interaction involving the deployment of a smart contract function requires a fee to compensate the mining node for processing and recording the transaction on the Blockchain. Gas serves as the unit representing this fee in EVM-compatible networks. Users obtain gas from mining nodes by exchanging it for the network token (such as *Ether* in the Ethereum network). It is important to differentiate between gas and the network token: gas denotes a fixed cost for actions on the Blockchain, while the network token is a fluctuating virtual currency used to acquire network resources. Table 6.3 provides a summary of the aforementioned analysis, highlighting the Polygon Chain as the most efficient network.

[8]Polygon Chain: `https://docs.polygon.technology/`

Network	Consensus Protocols	Token	Mining Block Time	Transaction Throughput	Gas Cost	Token Price ($)	Tx Base Fee Cost ($)
Ethereum Mainnet[1]	PoS	ETH	13s	13.1 tps	48 Gwei	2,637.81	2.66
Binance Smart Chain[2]	DPoS	BNB	3s	51.6 tps	3 Gwei	322.49	0.02
Polygon Chain[3]	PoS	MATIC	2s	45.7 tps	136.9 Gwei	0.86	0.002
Fantom Opera Chain[4]	Lachesis	FTM	1s	2.5 tps	9.97 Gwei	0.39	0.01
BitTorrent Chain[5]	PoS	BTT	2s	0.04 tps	304015 Gwei	$9.7x10^{-7}$	$6.19x10^{-6}$
Avalanche Chain[6]	Snowman	AVAX	2s	17.3 tps	25 nAVAX	39.39	0.021
HECO Chain[7]	HPoS	HT	3s	5.32 tps	2.3 Gwei	1.44	$6.96x10^{-5}$
Cardano[8]	PoS	ADA	20s	7 tps	0.17 ADA (per Tx)	0.53	0.09

Table 6.3: Comparison of EVM compatible Blockchains (February 2024)

[1] `https://etherscan.io` (accessed on 2nd of February, 2024)
[2] `https://bscscan.com` (accessed on 2nd of February, 2024)
[3] `https://polygonscan.com` (accessed on 2nd of February, 2024)
[4] `https://ftmscan.com` (accessed on 2nd of February, 2024)
[5] `https://bttcscan.com` (accessed on 2nd of February, 2024)
[6] `https://snowtrace.io` (accessed on 2nd of February, 2024)
[7] `https://hecoinfo.com` (accessed on 2nd of February, 2024)
[8] `https://cardano.org/` (accessed on 2nd of February, 2024)

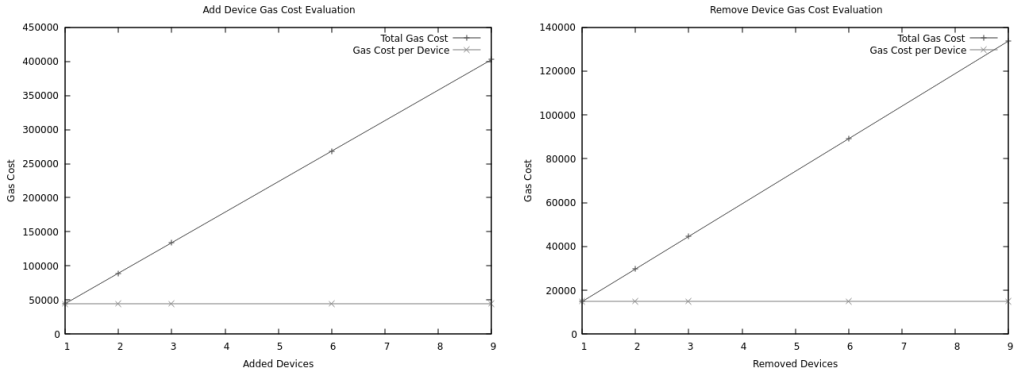
The following claim supports this proposition.

Claim 8. *When deployed on the Polygon Chain network, the new system demonstrates reduced costs in terms of gas, tokens, and dollars.*

Proof. The cost analysis of the functions provided by the SKM smart contract considers *Users* with 1 to 9 installed *Plug-ins* and 1 to 20 generated keys per *Plug-in*.

Table 6.4 illustrates that deploying the SKM smart contract requires 768490 gas, equivalent to 0.09 USD when utilizing the *Polygon Chain* network. Concerning the operational costs of the SKM smart contract, Figures 6.9(a) and 6.9(b) demonstrate that the *addDevice* and *removeDevice* functions entail a constant gas cost of 44713 and 14856, respectively. Similarly, Figure 6.10 indicates that the *modTemp* function carries a constant cost of 29426 gas. Finally, Figure 6.11 reveals that the *storeRef* function incurs an initial cost of 43049 gas due to the generation of the data structure for references. From that point, the cost remains constant at 28058 gas, notwithstanding of the number of loaded devices or stored key pairs in the system.

As a concluding remark, focusing on the cost in USD of running the smart contract functions as a measure of the economic feasibility of the new system, our findings on the *Polygon Chain* network indicate that all operations are economically viable. □



(a) Add Device Operation Evaluation (b) Remove Device Operation Evaluation

Figure 6.9: Add and Remove Device Operation Evaluation

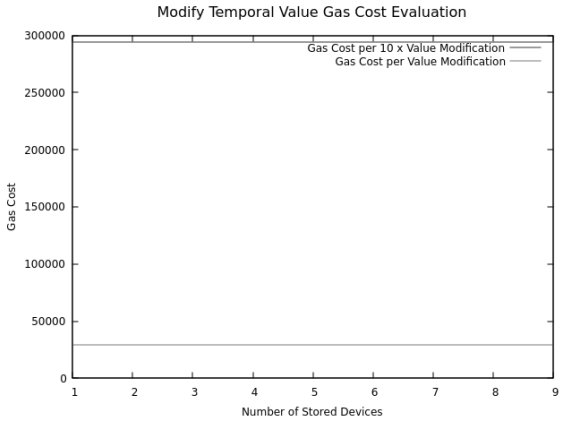


Figure 6.10: Modify Temporal Value Operation Evaluation

Operation	Gas used	Cost (Token)	Cost (USD $)
newSKMSmartContract	768490	0.1052	0.0905
addDevice	44713	0.0061	0.0053
removeDevice	14856	0.0020	0.0017
modTemp	29426	0.0040	0.0035
storeRef	43049- 28058	0.0059 - 0.0038	0.0051 - 0.0033

Table 6.4: Cost of the different functions of the *SKM SC*

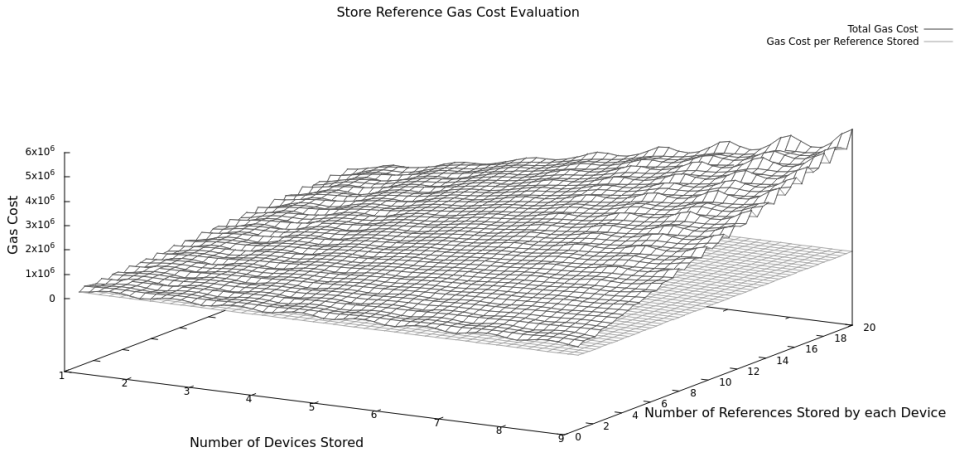


Figure 6.11: Store the Reference of a new Key Pair Operation Evaluation

6.5 Security and privacy analysis

This section focuses on studying the security and privacy requirements of the provided system (see Section 6.2.1). The discussion is organized as a set of propositions, where each proposition may have several claims to support its fulfillment.

To provide context for this analysis, Table 6.5 maps the various anticipated attack types to the propositions and claims that form the basis of this security evaluation. Additionally, the attacker model is based on the following assumptions:

Attack target	Attack description and coverage	
Key confidentiality	Attacker physically steals the smartphone and attempts to access all keys.	Addressed by the multi-factor authentication system in use (as stated in the attacker model).
	Attacker obtains the mnemonic words and tries to reconstruct all keys.	Covered in Proposition 5, Claim 9.
	Attacker attempts to steal the *Plug-in*'s master key pair during its transmission between the *Smartphone App* and the *Plug-in* or while stored in IPFS.	Covered in Proposition 5, Claims 10 and 11.
	Attacker tries to steal the *Plug-in*'s master key pair while stored on the smartphone or the *Plug-in* device.	Addressed by the native protection measures implemented by their respective operating systems (as stated in the attacker model).
	Attacker acts as a trusted third party and attempts to exploit this trust to access all keys.	Covered in Proposition 7, Claim 16.
Key integrity	Attacker tries to tamper with the *Smartphone App*'s keys stored on the smartphone.	Addressed by the native protection measures implemented by their respective operating systems (as stated in the attacker model).
	Attacker attempts to tamper with the *Plug-in*'s keys while stored on the smartphone or the *Plug-in* device.	Addressed by the native protection measures implemented by their respective operating systems (as stated in the attacker model).
	Attacker tries to tamper with the *Plug-in*'s keys during their transmission between the *Smartphone App* and the *Plug-in*.	Covered in Proposition 6, Claim 12.
	Attacker attempts to tamper with the reference to the *Plug-in*'s keys stored in IPFS or the uploaded file itself.	Covered in Proposition 6, Claims 13 and 14.
User privacy	Attacker tries to compromise the privacy of the *Users* by linking different keys or associating a specific key with a particular *User*.	Covered in Proposition 8, Claim 16.

Table 6.5: Types of attacks and their coverage

- The computational power available to the attacker is insufficient to compromise modern computationally secure cryptosystems.
- An attacker who gains physical access to the *User*'s devices running the *Smartphone App* and the distinct web browser *Plug-ins* cannot log into the compromised devices due to the native protection measures implemented by their respective operating systems. These measures prevent adversaries from retrieving or altering sensitive data or modifying the application code in any way.
- All the underlying algorithms are implemented correctly, and the parameters meet the necessary security requirements.
- The digital wallet, installed on the smartphone, effectively authenticates the user using a multi-factor authentication system comprising a user ID, password, and biometric third factor. It stands as the sole trusted entity empowered to generate, backup, recover, and store all cryptographic keys. During the setup process, users are prompted to establish a secure passphrase in compliance with NIST standards [98].

Proposition-5: Keys must be kept confidential from adversaries

This proposition addresses requirement R2.1, which focuses on protecting the confidentiality of the generated cryptographic keys. This assertion is based on the premise that the personal devices (i.e., smartphones or laptops) running the software components of the proposed system are natively secure, as stated in the attacker model. It is supported by the following three claims, demonstrating that even if potential adversaries gain access to the mnemonic words used to generate the Root secret key, the SKM smart contract, or the files stored in IPFS, they are unable to retrieve the distinct computed secret keys.

Claim 9. *Adversaries cannot deduce the Root secret key solely from knowledge of the mnemonic words used during its generation.*

Proof. The Root secret key SK_0 is derived through the use of the BIP-32 protocol alongside a *BIP-39 seed* as an input parameter. This seed, in turn, is generated by employing the BIP-39 protocol, with input parameters consisting of a *User*-provided password pw and the mnemonic words. Consequently, the Root secret key SK_0 remains secure as long as the password pw is protected against adversaries.

Note that, with knowledge of the mnemonic words, a computationally capable attacker can initiate a brute-force attack to deduce pw. The success of such an attempt is directly correlated with the complexity of pw. Therefore, to mitigate these risks, *Users* must set strong pw according to the NIST standards [98]. □

Claim 10. *The confidentiality of the Plug-in's Master key pair during its exchange between the Smartphone App and the Plug-in is securely protected against any adversary.*

Proof. The exchange of information between the *Smartphone App* and the *Plug-in* occurs through the designated "temp" field of the *SKM SC*. All data transmitted within this field is encrypted using a secure symmetric cryptosystem and a unique session key K_i specific to *Plug-in* i. Consequently, during the exchange of the *Plug-in*'s Master key pair, it is encrypted using the corresponding K_i. This session key in use is encoded into a QR code, requiring physical access for reading it. By ensuring the concealment of the QR code, the session key remains protected, preventing any attempts by adversaries to eavesdrop it and, hence, decrypt the *Plug-in*'s Master key pair. □

Claim 11. *The key pairs (SK_{ij}, PK_{ij}) generated by Plug-in i to interact with distinct dApps j are publicly stored in IPFS. Nevertheless, their confidentiality against any potential adversary is securely protected.*

Proof. In the proposed system, all key pairs (SK_{ij}, PK_{ij}) are stored in a file published on IPFS, with its reference stored in the *SKM SC*'s "refsList" field, thus making it publicly accessible. In order to protect against unauthorized access by attackers who might obtain this file, the key pairs (SK_{ij}, PK_{ij}) are encrypted using a secure symmetric encryption scheme and a session key K_{ij} before being stored on the IPFS. Then, the session key is further encrypted using the Root public key PK_0, ensuring that only the holder of the corresponding Root secret key SK_0 possesses the ability to decrypt the aforementioned key pairs. □

Proposition-6: Keys must remain intact against tampering attempts by attackers

This proposition addresses requirement R2.2, which focuses on protecting the integrity of the *Plug-in*'s Master key pair (SK_i, PK_i) and Chain code C_i during transfer, as well as the end key pairs (SK_{ij}, PK_{ij}) stored in IPFS. This assertion is based on the premise that the personal devices (i.e., smartphones or laptops) running the software components of the proposed system are natively secure, as stated in the attacker model. It is supported by the following three claims.

Claim 12. *The proposed system ensures the integrity of the Plug-in's Master key pair (SK_i, PK_i) and Chain code C_i against any adversary during its transfer from the Smartphone App to the Plug-in.*

Proof. As mentioned in the "Setting-up the web browser Plug-in" protocol (see Section 6.3.1), the Master key pair (SK_i, PK_i) and the Master Chain

code C_i are transmitted from the *Smartphone App* to each *Plug-in* via the "temp" field of the SKM smart contract.

To manipulate this field, an entity must execute a digitally signed transaction on the Blockchain, invoking the *modTemp()* function of the SKM smart contract. Each function within the SKM smart contract is tied to a designated set of authorized keys, dictating which entities can execute them. In the case of the *modTemp()* function, when the *Smartphone App* creates and deploys this smart contract, it sets its Root Public key PK_0 as the only authorized party that can perform this operation.

As a result, only the *Smartphone App*, as the sole owner of the corresponding Root Secret key SK_0, retains the capability to use this function and alter the contents of the "temp" field. This strict control ensures that while the Root Secret key SK_0 remains secure, the integrity of the cryptographic keys stored in the "temp" field remains secure throughout their transit from the *Smartphone App* to each *Plug-in*. □

Claim 13. *The proposed system ensures the integrity of the reference to the key pairs (SK_{ij}, PK_{ij}) stored in IPFS, protecting it against unauthorized modifications.*

Proof. As mentioned in the "Session key pair generation" protocol (see Section 6.3.2), key pairs (SK_{ij}, PK_{ij}) are stored in the IPFS and the corresponding reference to their location is recorded in the "refsList" field of the SKM smart contract.

To manipulate this field, an entity must execute a digitally signed transaction on the Blockchain, invoking the *storeRef()* function of the SKM smart contract. Only entities possessing the Master Secret keys SK_i linked to the Master Public keys PK_i, specified in the "whitelist"

field of the SKM smart contract, are permitted to use the *storeRef()* function. Moreover, the smart contract only allows an entity that knows a certain Master Secret key SK_i to modify its particular position in the "refsList" field. In order to add or remove values from the "whileList" field, an entity must send a digitally signed transaction invoking the *addDevice()/removeDevice()* methods. These operations, in turn, require the entity to possess the Root Secret key SK_0, which is only know by the *Smartphone App*.

As a result, as long as the *Smartphone App*'s Root Secret key SK_0 and the *Plug-ins*' Master Secret keys SK_i are kept confidential, no adversary can tamper with the references to the IPFS. Also, if an adversary can disclose the Master Secret key of a certain *Plug-in*, she will then be able to modify only the IPFS reference of this specific *Plug-in*. □

Claim 14. *The proposed system ensures the integrity of the key pairs (SK_{ij}, PK_{ij}) stored in IPFS, protecting it against unauthorized modifications.*

Proof. Regarding the integrity of the files uploaded to the IPFS, it is noteworthy to mention the *immutability property* [9] inherent in IPFS technology. This property establishes that once a file is added to the IPFS network, its content remains unalterable without modifying the corresponding reference. Consequently, any attempt by an adversary to manipulate the key pairs (SK_{ij}, PK_{ij}) stored within IPFS would require altering their original references. □

Proposition-7: Keys must not be held by third parties

This proposition addresses requirement R2.3, which focuses on offering a fully decentralized multi-platform wallet that does not rely on exter-

[9] IPFS Immutability: `https://docs.ipfs.tech/concepts/immutability/`

nal trust to provide its functionalities. The following claim supports this assertion.

Claim 15. *The proposed system does not depend on any trusted third party to store the generated cryptographic keys.*

Proof. The proposed system leverages the fully distributed technologies Blockchain and IPFS to facilitate communication between its various components and securely store generated cryptographic keys. Specifically, the SKM smart contract deployed on the Blockchain orchestrates communication between the *Smartphone App* and the *Plug-ins* while managing the keys, which are securely stored in the IPFS instead of relying on any third-party storage system. □

Proposition-8: Keys must remain unlinked between them and unassociated with individuals

A Blockchain wallet is considered anonymous when it is impossible to connect addresses (key pairs) with their owners' identities. In this context, this proposal addresses requirement R2.4, aiming to ensure that a derived public key remains indistinguishable from another unless additional information is leaked.

The following claim supports this assertion, showing how the proposed system implements the BIP-32 protocol to generate unlinkable cryptographic keys.

Claim 16. *The proposed system generates key pairs that are unlikable between them.*

Proof. In the proposed system, the BIP-32 protocol is used to generate all the cryptographic keys. As explained in Section 6.1.2, this method

contains a vulnerability wherein an attacker can deduce a parent private key SK_{par} given the parent public key PK_{par} and a child private key SK_i. This is possible because parent public keys are used in the key derivation function $CKD_{pub}((PK_{par}, c_{par}), i)$. In order to avoid this weakness, our implementation of the BIP-32 protocol exclusively employs the key derivation function $CKD_{priv}((SK_{par}, c_{par}), i)$ to generate‘ ‘hardened” child private keys using the parent private key as input. In this way, even in the event of an attacker knowing a child private key, the recovery of the parent private key SK_{par} becomes infeasible. □

6.6 Comparative analysis

This section compares the new proposal with both the well-established mainstream schemes introduced in Section 6.1.2 and the state-of-the-art solutions present in the literature and detailed in Section 2.4.

For this analysis, we have mapped each of the considered requirements to specific properties that a wallet may or may not fulfill. This correspondence is illustrated in Table 6.6. Note that we have divided requirement 1.2 into two properties: one addressing direct access to the user's generated keys, and the other focusing on the ability to use those keys across different platforms or devices.

Table 6.7 summarizes the comparison between the new proposed solution and existing mainstream schemes based on the aforementioned properties (taking into account their implementation specification and the study of [91]). The table indicates whether a wallet fully satisfies a property (✓), partially satisfies it (+/-), does not satisfy it (X), or we lack sufficient information to reach a conclusion (EMPTY SPACE).

Requirement	Property
R1.1. Users should have the capability to generate keys from various devices utilizing the provided wallet.	Multiple Key Pairs Generation
R1.2. Key pairs should be accessible to users at all times, including the option to export them from the wallet for use in other applications if desired.	Immediate Access to Keys
	Cross-Device Portability
R1.3. Users should have the capability to recover all cryptographic keys in the event of loss of access to the wallet.	Key Recovery
R1.4. The solution must be economically viable for users.	Economically available
R2.1. Keys must be kept confidential from adversaries.	Resistant to Theft
R2.2. Keys should be safe against tampering by attackers.	Key Integrity and Authenticity
R2.3. Keys should not be kept by a third party, thus ensuring that the system does not rely on external trust.	No Trusted Third Parties
R2.4. Keys should remain unlinked to each other or to a specific user, ensuring that all generated keys are independent.	Key Unlikability

Table 6.6: Mapping of the considered requirements to specific properties.

As illustrated in Table 6.7, the new proposal particularly excels in terms of interoperability (immediate access to keys and/or cross-device portability), key unlinkability (i.e., privacy), key recovery, and resistance to theft. This superiority is anticipated, as these features were fundamental to the design of the new proposal, making the difference with respect

[10]Bitcoin Core: `https://bitcoin.org/en/bitcoin-core/`
[11]Android Bitcoin Wallet: `https://bitcoin.org/en/wallets/mobile/android/bitcoinwallet/`
[12]MultiBit: `https://github.com/Multibit-Legacy/multibit`
[13]Bitaddress: `https://www.bitaddress.org/`
[14]Bitcoin Paper Wallet: `https://bitcoinpaperwallet.io/`
[15]Bitcoin Armory: `https://www.bitcoinarmory.com/`
[16]Trezor: `https://trezor.io/`
[17]Brainwallet: `https://brainwalletx.github.io/`
[18]Coinbase: `https://www.coinbase.com/`
[19]Binance: `https://www.binance.com/`

Wallet	Multiple Key Pairs Generation	Immediate Access to Keys	Cross-Device Portability	Key Recovery	Economically Available	Resistant to Theft	Key Integrity and Authenticity	No Trusted Third Parties	Key Unlikablility
Our Proposal	✓	✓	✓	✓	+/-	✓	✓	✓	✓
Keys in Local Storage (e.g., Bitcoin Core[10], Android Bitcoin Wallet[11])	✓	✓	X	X	✓	X		✓	
Password-protected (Encrypted) Wallets (e.g., MultiBit[12])	✓	✓	X	X	✓	X		✓	
Offline Storage of Keys (e.g., Bitaddress[13], Bitcoin Paper Wallet[14])	✓	X	✓	X	✓	X		✓	
Air-gapped Key Storage (e.g., Bitcoin Armory[15], Trezor[16]).	✓	X	X	X	+/-	X		✓	
Password-derived Keys (e.g., Brainwallet[17])	✓	✓	✓	✓				✓	
Hosted Wallet (e.g., Coinbase[18], Binance [19])		+/-	+/-	✓	+/-	+/-	✓	X	

Table 6.7: Proposal comparison with existing mainstream schemes

to mainstream schemes.

Regarding the state-of-the-art schemes in the literature, Table 6.8 reveals that multi-signature wallets (i.e., [57, 58, 60, 61]) fall behind in terms of immediate Access to keys, as their primary focus is on safeguarding cryptographic keys against misuse rather than ensuring their availability. These schemes are assumed to offer some degree of key recovery through the distribution of keys or key shares among a group of participants. However, they do not account for scenarios where participants lose their shares

or become malicious.

The *seed-derived wallet* presented in [62] employs biosensors for key generation. Consequently, the number of keys that can be generated is limited by the range of signals these biosensors can measure, and the system's interoperability is confined to its designed purpose. Additionally, significant changes in a user's physiological signals, such as those caused by accidents or illnesses, may impede key recovery. Furthermore, there is a potential vulnerability: an adversary who intercepts the physiological signals used in seed generation could gain access to the user's keys

The proposals by Singh et al. [63] and Seo et al. [64] utilize the individuals' memory for key protection. Specifically, they employ answers to questions and specified picture locations to encrypt fragments of the private key for future recovery. While these schemes do not require an external entity during the recovery process, they do rely on a third party for storing the secured private key. Furthermore, an adversary might exploit social engineering techniques to gather user information, deduce the answers to security questions, or identify the images and their locations.

Rezaeighaleh et al.'s proposal [65] utilizes hardware wallets to safeguard cryptographic material, wherein users possess two hardware wallets with identical private keys, one acting as the primary wallet and the other as a backup. However, should users wish to generate multiple key pairs, this scheme imposes a significant overhead. Additionally, using these keys across different devices or services entails further inconvenience, requiring users to carry and connect the device each time they are used. Moreover, storing keys on these devices exposes them to the risk of physical theft. If both devices are lost or one is damaged unexpectedly, access to the cryptographic keys could be irretrievably lost.

In conclusion, this comparison highlights the superiority of the new

wallet over existing literature, particularly in terms of key recovery, and it also demonstrates notable strengths in properties such as immediate Access to keys, cross-device portability and key unlinkability. On top of that, it is important to note that none of the previously mentioned solutions consider the economic costs they entail, a crucial factor in assessing the feasibility of these schemes.

Wallet	Multiple Key Pairs Generation	Immediate Access to Keys	Cross-Device Portability	Key Recovery	Economically Available	Resistant to Theft	Key Integrity and Authenticity	No Trusted Third Parties	Key Unlikablility
Our Proposal	✓	✓	✓	✓	+/-	✓	✓	✓	✓
[57]		X		+/-		✓	✓	X	
[58]		X		+/-		✓	✓		
[60]		X		+/-		✓	✓	✓	✓
[61]		X		+/-		✓	✓	✓	
[62]	+/-	✓	X	+/-		+/-	✓	✓	✓
[63]	✓	✓	✓	+/-		+/-	✓	X	
[64]	✓	✓	+/-	+/-		+/-	✓	X	
[65]	+/-	+/-	✓	+/-		+/-	✓	✓	

Table 6.8: Proposal comparison with state-of-the-art schemes present in the literature

6.7 Preliminary prototype implementation

To evaluate the viability of the new proposal, we developed a preliminary prototype of the wallet. This section begins with a discussion of the technical aspects of the implementation, followed by an overview of the

user interface, and concludes with an analysis of the runtime performance of the various protocols.

6.7.1 Implementation details

The implemented prototype consists of three essential components: the *Smartphone App*, the *Secure Key Management Smart Contract*, and the web browser *Plug-in*. The source code for the *Smartphone App* and the *Secure Key Management Smart Contract* can be found at `https://github.com/imiguelrodriguez/TFGwallet`, while the code for the *Plug-in* is accessible at `https://github.com/imiguelrodriguez/TFGwalletPlugIn`. Below, we provide an overview of the most significant aspects of their development:

- The *Smartphone App* was designed to run on an Android device, while the *Plug-in* was developed to operate on the Google Chrome web browser.
- The *Smartphone App* was developed using the Android Studio IDE, with the code written in Kotlin and structured according to the Model-View-Controller (MVC) architecture.
- The *Plug-in* was developed using web-based technologies, including HTML, CSS, and JavaScript.
- Protocols BIP32 and BIP39, discussed in Section 6.1, were implemented with an object-oriented approach to ensure modularity in both Kotlin and JavaScript, facilitating their integration into the *Smartphone App* and *Plug-in*, respectively.
- To deploy and interact with the *Secure Key Management Smart Contract* from both the *Smartphone App* and the *Plug-in*, the web3j[20]

[20]Web3j Library: `https://docs.web3j.io/`

and web3js[21] libraries were utilized, respectively. Additionally, to streamline interaction with smart contracts via the web3j library, a Java wrapper was generated using the Truffle suite[22].

- The *Secure Key Management Smart Contract* that manages the wallet was implemented in Solidity[23], an object-oriented programming language specifically designed for the Ethereum Virtual Machine (EVM). Next, it was deployed on the Sepolia Test Network[24], a Blockchain environment tailored for testing, enabling the evaluation of Blockchain applications in a realistic setting.

- To connect to the IPFS network in our test environment, we downloaded and executed the IPFS daemon from the official website[25]. Running the daemon integrates the designated computer into the IPFS network, facilitating the publication and retrieval of files within this decentralized system. For deploying the prototype in a real-world setting, utilizing a public IPFS gateway[26] would be a more practical alternative.

- The ECDSA cryptosystem, with a 256-bit key size, was utilized for generating signatures. For encryption, the Elliptic Curve Integrated Encryption Scheme (ECIES) [99] was employed, using AES in GCM mode.

- To perform cryptographic operations within the digital wallets, the EVM key specifications were followed. Specifically, ECDSA was employed with a 256-bit private key and a point on the secp256k1 ECDSA curve, represented as an (x, y) point, for the corresponding

[21] Web3js Library: `https://docs.web3js.org/`
[22] Truffle suite: `https://archive.trufflesuite.com/`
[23] Solidity Programming Language: `https://soliditylang.org/`
[24] Ethereum Sepolia Faucet: `https://sepoliafaucet.com/`
[25] InterPlanetary File System (IPFS): `https://ipfs.tech/`
[26] IPFS gateway: `https://docs.ipfs.tech/concepts/ipfs-gateway/`

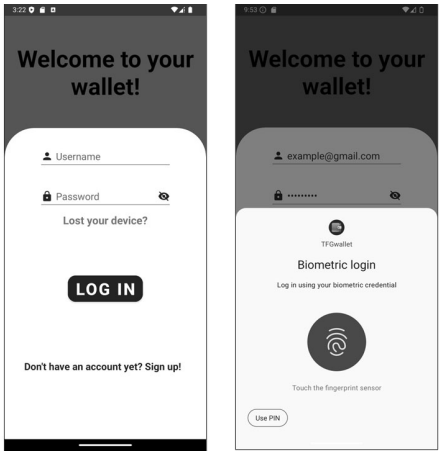


Figure 6.12: Login

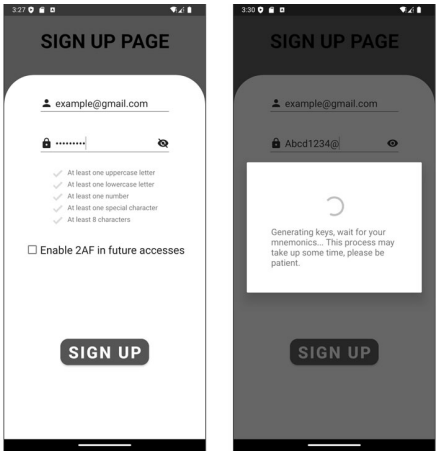


Figure 6.13: Sign-up

public key. The public Ethereum addresses were generated by taking the lower 160 bits of the Keccak-256 (also known as SHA-3) digest of the public key.

6.7.2 Interface overview

Upon launching the *Smartphone App*, the *User* is presented with a login screen, which can be configured to utilize *Biometric Two-Factor Authentication*, as illustrated in Figure 6.12.

If the *User* does not have an account, one must be created by setting up the *Smartphone App*. This involves registering with a valid email address and a secure password. Upon successful registration, a new local account is established, keys are generated, and the Blockchain account is funded. The *User* must wait for this process to be completed, as shown in Figure 6.13.

Following key generation, the *Smartphone App* presents the *User* with her 24 mnemonic words, which are crucial for key recovery. Once this

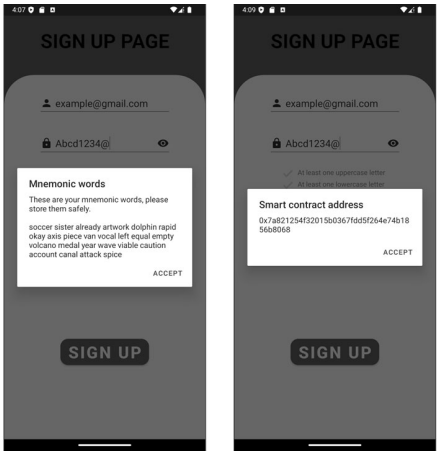


Figure 6.14: Mnemonic words and smart contract hash

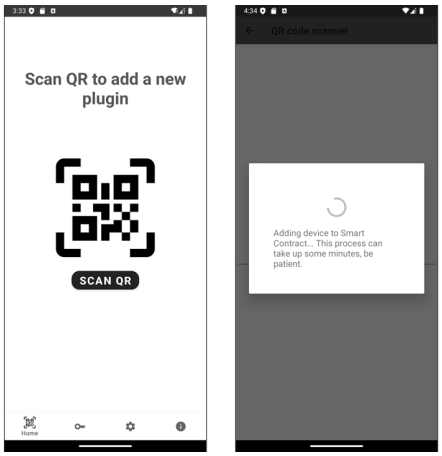


Figure 6.15: Adding a Plug-in

step is completed, the smart contract is deployed, and its address (i.e., the smart contract's hash) is displayed. Figure 6.14 depicts this process.

After successfully logging into the *Smartphone App*, the *User* can add new *Plug-ins* by scanning a QR code. By clicking the scan button on the designated screen, the device's camera will activate, enabling the user to capture the QR code. Once scanned, a loading message will appear as the *Smartphone App* communicates with the Blockchain. Figure 6.15 illustrates this procedure.

Figure 6.16 depicts the screen where the *User* can view the *Smartphone App*'s master key and the list of added *Plug-ins* along with their corresponding cryptographic keys. Initially, the *Smartphone App* conceals the master key details, which the *User* can reveal by tapping on the corresponding item.

As it can be seen in Figure 6.17, the same approach applies to the list of registered *Plug-ins*. Initially, only the names of the devices where the *Plug-ins* are installed are displayed. By tapping on a specific *Plug-in*, the

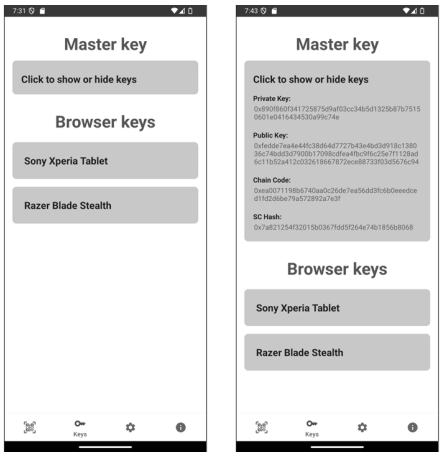


Figure 6.16: Smartphone App's cryptographic keys

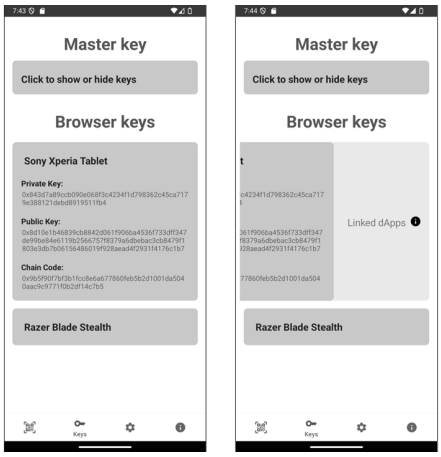


Figure 6.17: Plug-ins' cryptographic keys

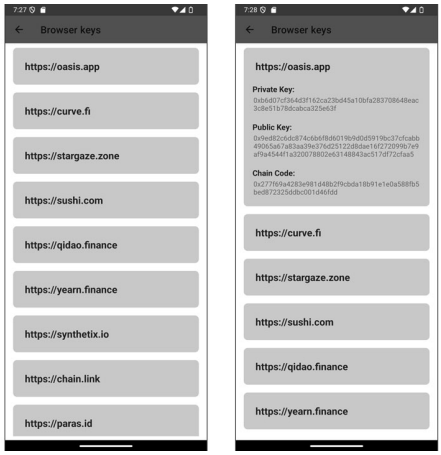


Figure 6.18: dApps'cryptographic keys

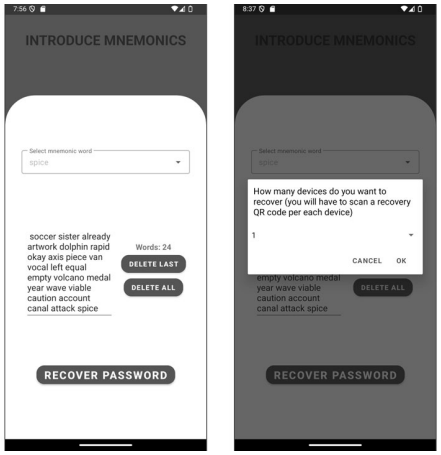


Figure 6.19: Key recovery

app reveals its cryptographic keys. Additionally, by swiping left, the *User* can access a new screen displaying all the accessed *dApps* associated with that *Plug-in*, as shown. This procedure is shown in Figure 6.18.

Finally, Figure 6.19 illustrates the key recovery process. The *Smartphone App* first displays a recovery screen, resembling the sign-up interface, where the system verifies that the provided email is valid, and that the password meets the required criteria. Once both fields are validated, the app transitions to a screen where the mnemonic words can be entered. Upon tapping the recovery button, the user is prompted to enter the number of previously registered devices. The user must then scan the corresponding number of QR codes. After scanning, the recovery process is initiated, allowing the user to sign in as usual.

Protocol	Average time (ms)	Median (ms)	Deviation (ms)
P1 - Setting-up the *Smartphone App*	26594.90	19073.50	10706.98
P2 - Setting-up the web browser *Plug-in*	44953.40	42409.57	6787.60
P3 - Session key pair generation	18815.67	17192.9	8117.89
P4 - Key inventory	181.90	71	271.71
P5 - Key recovery at the *Smartphone App*	689.70	695	95.28

Table 6.9: Cost in milliseconds of the designed protocols

6.7.3 Runtime performance analysis

Leveraging the implemented prototype, this section examines the execution time of each designed protocol, measured in milliseconds, to assess the performance of the new wallet and its practical feasibility in a realistic environment. In the case of the "Key recovery" procedure, we tested the case in which the *Smartphone App* is no longer accessible. Each protocol was executed 10 times for this analysis.

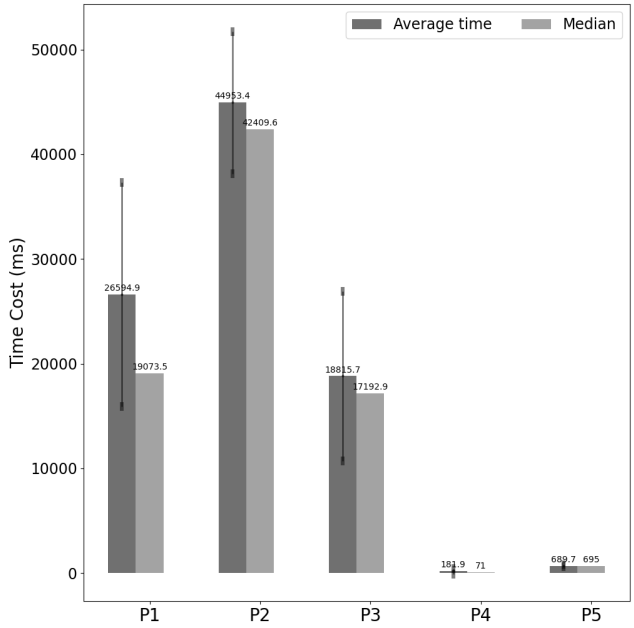


Figure 6.20: Cost in milliseconds of the designed protocols

As outlined in Section 6.7.1, these tests were conducted using the Sepolia Test Network due to its close resemblance to a realistic environment. This choice is significant for two reasons. First, all interactions with the Blockchain require Sepolia ETH tokens, which are limited and earned through network mining, thereby constraining the number of tests that can be performed. Second, the status and current traffic of the Sepolia Test Network can fluctuate, directly influencing the execution times of operations involving Blockchain interactions.

Table 6.9 presents a summary of the execution times, measured in milliseconds, for the evaluated protocols. Figure 6.20 provides a graphical representation of these values for clearer assessment. Key findings from this analysis include the following points:

- Setting up the *Smartphone App* under standard Blockchain load

takes approximately 19 seconds, while configuring a *Plug-in* requires about 42 seconds. Since these operations are infrequent, the times involved seem to be manageable.

- The "Session key pair generation" protocol is crucial, as it is executed when a user first interacts with a new *dApp* while browsing the Internet. Under typical Blockchain conditions, this process takes approximately 17 seconds, which is a significant cost in this context. However, it is important to emphasize that this duration represents the total time for the entire protocol. Notably, the most time-consuming part of this execution is the Step 8d, an automated process transparent to the user, in which the *Plug-in* utilizes the *storeRef()* method of its associated *SKM SC* to update the "refsList" field with the new reference. During this step, the user is already engaged with the *dApp*, so it does not impact the user experience. Consequently, excluding this step, the user-perceived waiting time for this protocol is approximately 270 milliseconds. It is also worth noting that this operation only occurs the first time the *Plug-in* interacts with a particular *dApp*. In summary, we consider this cost acceptable for such a critical procedure.

- Interactions with the Blockchain are by far the most time-consuming, accounting for 95% to 99% of the total execution time in the first four protocols. In the "Key recovery" protocol, these interactions represent approximately 75% of the total time. This indicates that using a faster Blockchain could significantly reduce the times reported in this study.

- The "Key inventory" and "Key recovery" protocols require negligible time overall.

- The time listed in Table 6.9 for the "Key recovery" protocol refers exclusively to the steps executed by the *Smartphone App*. An ad-

ditional step is carried out by each involved *Plug-in*, which must download a file containing keys from the IPFS. The size of this file is directly proportional to the number of dApps linked to the *Plug-in*, with each *dApp* corresponding to one key pair. In the tests conducted, downloading a file with 10 key pairs from the IPFS took an average of 245.4 milliseconds, with a deviation of 25.73 milliseconds. For a file with 20 key pairs, the download time averaged 361.8 milliseconds, with a deviation of 64.32 milliseconds. All those reported times seem to be affordable.

Chapter 7

Conclusions

7.1 Contributions

The General Data Protection Regulation (GDPR) has marked a decisive step in the mitigation of the massive collection and processing of users' personal data, by guaranteeing specific privacy rights to users as well as bringing full control back to the data owners. However, applying this legislation in the real world raises significant issues due to the abstract nature of the requirements in the legal text. This results in generic rules and principles that may be difficult to interpret and implement for both companies and consumers. Furthermore, as shown in Section 1.1, current solutions that try to apply the GDPR follow an architecture in which all processes related to data consent management are carried out in Service Providers' own servers, out of users' control.

In the light of the above, there is a need for general-purpose data management systems that allow the involved parties to prove the agreements

made between them regarding the use and storage of the personal data in various real-world use-case environments. In order to achieve this, in the last years, researchers have proposed different works based on Smart Contracts and the Blockchain Technology, which offer desired features such as transparency, traceability, non-repudiation, integrity, immutability, and decentralization. However, these proposals fail to correctly identify all entities involved in the process, impose a high overhead on end users or do not provide a usable implementation as stated in 2.

For these reasons, in this thesis we have contributed with some tools and frameworks in different environments that address the deployability and application of the current legislation to data management and control issues that current works in the literature present.

In our first contribution, presented in Chapter 3, we introduce a novel system that leverages smart contracts and Blockchain Technology to create publicly accessible, immutable records of agreements between Data Subjects and Service Providers regarding the collection and processing of personal data. These records play a crucial role in ensuring compliance with GDPR by enabling effective management of personal data. The proposed system allows Data Controllers and Data Processors to demonstrate authorization for data collection and processing during audits. Additionally, it empowers Data Subjects by providing real-time transparency into what data is being collected, who is processing it, and for what purposes. They can also modify their consent details at any time. Our system has been implemented and evaluated, with a comprehensive analysis covering GDPR compliance, functionality, security, and privacy considerations, and the cost of Blockchain operations in terms of gas fees and USD.

In Chapter 4, we introduce PrivacySmart, a system that builds upon the novel framework proposed in the previous chapter. PrivacySmart streamlines the handling of privacy and cookie policy pop-ups, provid-

ing users with a seamless and transparent browsing experience across various online services. The system consists of two key components: a browser plug-in installed on each user's browser, and a companion smartphone application that centralizes the management of accepted policies. The plug-in automatically processes privacy and cookie prompts, allowing users to browse the web uninterrupted. Meanwhile, the smartphone app offers a centralized interface where users can monitor and manage their consent decisions per browser and website, maintaining full control over their privacy preferences. We implemented PrivacySmart as a Firefox extension and evaluated its effectiveness on websites that deploy pop-ups via commonly used banner providers. While the plug-in occasionally faces challenges in enforcing user preferences due to limitations imposed by certain service providers, our results demonstrate that the system reliably manages pop-ups in a transparent manner on behalf of the user.

Similarly, in Chapter 5, we extend our consent management framework to address GDPR and ePrivacy compliance within the context of Real-Time Bidding (RTB), focusing on the decentralization of user consent. Our approach harmonizes the IAB's Transparency and Consent Framework (TCF) with OpenRTB protocols, ensuring alignment with legal standards while granting users direct and granular control over their personal data. Additionally, it provides verifiable, immutable records to support regulatory compliance. A browser plug-in enforces user-defined consent preferences automatically, enabling a frictionless browsing experience and fostering transparency in data handling. Complementing this, a smartphone application empowers users to oversee, adjust, or revoke their consents at any time. Furthermore, we introduce a "specificity" metric within the IAB Audience Taxonomy to evaluate the privacy risks associated with the data being shared.

Finally, as our fourth contribution, we address a fundamental require-

ment shared across the previous solutions: the secure and user-friendly management of cryptographic keys necessary for interacting with Blockchain technologies. Chapter 6 presents a fully decentralized, cross-platform wallet that utilizes Blockchain and IPFS to manage asymmetric cryptographic keys and support reliable key recovery. This innovative solution enables users to interact with decentralized applications (dApps) built on smart contracts while safeguarding their privacy from potential attackers targeting sensitive data. Our design ensures that users can recover their keys seamlessly in cases of device loss or damage, without relying on any trusted third parties. The system has been developed with economic feasibility in mind—smart contracts implementing the core functionality were carefully evaluated to determine the costs associated with their use. A thorough security and privacy analysis demonstrates the robustness of the approach: i) the system resists unauthorized access to cryptographic keys, ii) generated keys are protected against tampering, and iii) key generation remains unlinkable to specific users, preserving anonymity. Additionally, we compare our wallet with mainstream and cutting-edge solutions from the literature, highlighting the unique benefits of our approach. To validate its practical viability, we developed a preliminary prototype, which includes a detailed implementation overview, user interface design, and performance evaluation of the core protocols.

7.2 Publications

The publications supporting the content of this thesis are stated below:

1. C. Daudén-Esmel, J.Castellà-Roca and A. Viejo, *"Blockchain-Based Access Control System for Efficient and GDPR-Compliant Personal Data Management"*, Computer Communications, Volume 214, pp. 67-87, 2024, ISSN: 0140-3664, doi: `https://doi.org/10.1016/j.comcom.2023.11.017`.

2. C. Daudén-Esmel, J. Castellà-Roca, A. Viejo and Ignacio Miguel Rodriguez, "Multi-Platform Wallet for Privacy Protection and Key Recovery in Decentralized Applications", Blockchain: Research and Applications, 6, Issue 1, pp. 100243, 2025, ISSN: 2096-7209, doi: https://doi.org/10.1016/j.bcra.2024.100243.

3. C. Daudén-Esmel, J. Castellà-Roca, A. Viejo and J. Domingo-Ferrer, *"Lightweight Blockchain-based Platform for GDPR-Compliant Personal Data Management"*, 2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP), Zhuhai, China, 2021, pp. 68-73, doi: https://doi.org/10.1109/CSP51677.2021.9357602.

4. C. Daudén-Esmel, J. Castellà-Roca, A. Viejo and E.J. Bel-Ribes, *"PrivacySmart: Automatic and transparent management of privacy policies"*, in: S. Katsikas, et al. Computer Security. ESORICS 2023 International Workshops, Springer Nature Switzerland, Cham, 2024, pp. 216–231, doi: https://doi.org/10.1007/978-3-031-54204-6_12.

5. C. Daudén-Esmel, J. Castellà-Roca, A. Viejo and Vicenç Torra, *Blockchain-Enhanced User Consent for GDPR-Compliant Real-Time Bidding*, in: Katsikas, S., Shafiq, B. (eds) Data and Applications Security and Privacy XXXIX (DBSec 2025), Lecture Notes in Computer Science, vol 15722, Springer Nature Switzerland, Cham, 2025, pp.137-155, doi: https://doi.org/10.1007/978-3-031-96590-6_8.

6. C. Daudén-Esmel, J.Castellà-Roca and A. Viejo, *"Sistema para la gestión automática de las políticas de privacidad y uso de las cookies"*, XVII Reunión Española sobre Criptología y Seguridad de la Información - RECSI 2022, Santander, Spain, 2022, pp. 83-87, ISBN:

978-84-19024-14-5, url: https://recsi2022.unican.es/wp-content/uploads/2022/10/LibroActas-978-84-19024-14-5.pdf.

7.3 Future Work

Regarding future work, our primary objective is to design and implement the complete ecosystem outlined throughout the various chapters of this thesis. While we have already developed and tested the core components of our decentralized protocols —particularly those involving strict timing constraints— a full-scale implementation of the entire framework is essential to validate its applicability in real-world scenarios. This will enable us to move beyond theoretical and small-scale experimental settings, providing a more comprehensive evaluation of our approach.

A full deployment will also allow for a detailed assessment of the computational overhead associated with the SC logic. In particular, we intend to analyze gas consumption, an important metric that quantifies the computational resources required to execute Blockchain operations. By measuring gas usage across different functions and workflows, we can determine the cost and analyze the efficiency of our design, identify any potential bottlenecks, and explore optimizations to improve overall performance. This analysis is vital for evaluating the practical deployability and economic feasibility of the proposed smart contracts on public Blockchain platforms.

In parallel, our research has highlighted a crucial non-technical challenge in the domain of privacy: user behavior. We have come to understand that one of the most significant obstacles to improving digital privacy lies not just in technical solutions, but in user awareness and decision-making. Despite the availability of privacy-preserving tools and mechanisms, many users unknowingly compromise their personal data due to a lack of understanding or awareness.

Therefore, future work should also consider initiatives aimed at user education and empowerment. This includes developing strategies, tools, and user-friendly interfaces that help individuals make more informed choices about what personal information they share and how it is used by online services. Collaborations with UX designers, digital literacy educators, and policymakers could play a key role in shaping interventions that foster a more privacy-conscious digital culture.

Ultimately, for privacy-preserving technologies to have a meaningful and widespread impact, both robust technical frameworks and a well-informed user base are essential. Our future efforts aim to address both dimensions in an integrated and holistic manner.

Bibliography

[1] A. Esteve, "The business of personal data: Google, Facebook, and privacy issues in the EU and the USA," *International Data Privacy Law*, vol. 7, pp. 36–47, 03 2017.

[2] K. Houser and W. Voss, "Gdpr: The end of google and facebook or a new paradigm in data privacy?," *SSRN Electronic Journal*, 07 2018.

[3] A. Harris, "Spy agency sought us call records before 9/11, lawyers say," *Bloomberg. com*, vol. 30, 2006.

[4] J. Bamford, "The nsa is building the country's biggest spy center (watch what you say)," *Wired, March*, vol. 15, 2012.

[5] "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (text with eea relevance)," 5 2016.

[6] C. Wirth and M. Kolain, "Privacy by blockchain design: A blockchain-enabled gdpr-compliant approach for handling personal data," in *Reports of the European Society for Socially Embedded Technologies (EUSSET)*, 2018.

[7] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, "Gdpr-compliant personal data management: A blockchain-based solution," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1746–1761, 2020.

[8] S. Bu-Pasha, A. Alen-Savikko, J.-S. Mäkinen, R. Guinness, and P. Korpisaari, "Eu law perspectives on location data privacy in smartphones and informed consent for transparency," *European Data Protection Law Review*, vol. 2, pp. 312–323, Sept. 2016.

[9] L. A. Linn and M. B. Koo, "Blockchain for health data and its potential use in health it and health care related research," in *ONC/NIST Use of Blockchain for Healthcare and Research Workshop. Gaithersburg, Maryland, United States: ONC/NIST*, pp. 1–10, 2016.

[10] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *2016 2nd International Conference on Open and Big Data (OBD)*, pp. 25–30, Aug 2016.

[11] R. Neisse, G. Steri, and I. Nai Fovino, "A blockchain-based approach for data accountability and provenance tracking," in *ARES '17: Proceedings of the 12th International Conference on Availability, Reliability and Security*, pp. 1–10, August 2017.

[12] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018.

[13] M. Chowdhury, A. Colman, A. Kabir, J. Han, and P. Sarda, "Blockchain as a notarization service for data sharing with personal data store," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE In-*

ternational Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 1330–1335, August 2018.

[14] B. Faber, G. Michelet, N. Weidmann, R. R. Mukkamala, and R. Vatrapu, "Bpdims:a blockchain-based personal data and identity management system," in *Conference: Hawaii International Conference on System Sciences*, January 2019.

[15] J. Liang, W. Han, Z. Guo, Y. Chen, C. Cao, X. Wang, and F. Li, "Desc: enabling secure data exchange based on smart contracts," *Science China Information Sciences*, vol. 61, 04 2018.

[16] M. U. Rahman, F. Baiardi, B. Guidi, and L. Ricci, "Protecting Personal Data using Smart Contracts," *arXiv e-prints*, p. arXiv:1910.12298, Oct. 2019.

[17] G. Zhao, H. He, and B. Di, "Design and implementation of the digital education resources authentication system based on blockchain," in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, ICCSP 2020, (New York, NY, USA), p. 100–104, Association for Computing Machinery, 2020.

[18] H. Wang, Y. Yuan, and F. Yang, "A personal data determination method based on blockchain technology and smart contract," in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, ICCSP 2020, (New York, NY, USA), p. 89–94, Association for Computing Machinery, 2020.

[19] G. Zyskind, O. Nathan, and A. . Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*, pp. 180–184, May 2015.

[20] S. Kirkman, "A data movement policy framework for improving trust in the cloud using smart contracts and blockchains," in *2018 IEEE*

International Conference on Cloud Engineering (IC2E), pp. 270–273, April 2018.

[21] O. Choudhury, H. Sarker, N. Rudolph, M. Foreman, N. Fay, M. Dhuliawala, I. Sylla, N. Fairoza, and A. K. Das, "Enforcing human subject regulations using blockchain and smart contracts," *Blockchain in Healthcare Today*, vol. 1, Mar. 2018.

[22] N. Duong-Trung, H. X. Son, H. T. Le, and T. T. Phan, "On components of a patient-centered healthcare system using smart contract," in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, ICCSP 2020, (New York, NY, USA), p. 31–35, Association for Computing Machinery, 2020.

[23] N. Duong-Trung, H. X. Son, H. T. Le, and T. T. Phan, "Smart care: Integrating blockchain technology into the design of patient-centered healthcare systems," in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, ICCSP 2020, (New York, NY, USA), p. 105–109, Association for Computing Machinery, 2020.

[24] M. Zichichi, S. Ferretti, G. D'Angelo, and V. Rodríguez-Doncel, "Personal data access control through distributed authorization," in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pp. 1–4, Nov 2020.

[25] M. Davari and E. Bertino, "Access control model extensions to support data privacy protection based on gdpr," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 4017–4024, Dec 2019.

[26] M. Barati and O. Rana, "Tracking gdpr compliance in cloud-based service delivery," *IEEE Transactions on Services Computing*, pp. 1–1, 2020.

[27] M. M. Merlec, Y. K. Lee, S.-P. Hong, and H. P. In, "A smart contract-based dynamic consent management system for personal data usage under gdpr," *Sensors*, vol. 21, no. 23, 2021.

[28] S.-S. Jung, S.-J. Lee, and I.-C. Euom, "Delegation-based personal data processing request notarization framework for gdpr based on private blockchain," *Applied Sciences*, vol. 11, no. 22, 2021.

[29] A. M. McDonald, R. W. Reeder, P. G. Kelley, and L. F. Cranor, "A comparative study of online privacy policies and formats," in *Privacy Enhancing Technologies: 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings 9*, pp. 37–55, Springer, 2009.

[30] N. Vyas, A. C. Squicciarini, C.-C. Chang, and D. Yao, "Towards automatic privacy management in web 2.0 with semantic analysis on annotations," in *2009 5th international conference on collaborative computing: networking, applications and worksharing*, pp. 1–10, IEEE, 2009.

[31] C. Yue, M. Xie, and H. Wang, "Automatic cookie usage setting with cookiepicker," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pp. 460–470, IEEE, 2007.

[32] C. Yue, M. Xie, and H. Wang, "An automatic http cookie management system," *Computer Networks*, vol. 54, no. 13, pp. 2182–2198, 2010.

[33] O. Kulyk, P. Mayer, M. Volkamer, and O. Käfer, "A concept and evaluation of usable and fine-grained privacy-friendly cookie settings interface," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE In-*

ternational Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 1058–1063, IEEE, 2018.

[34] V. Ha, K. Inkpen, F. Al Shaar, and L. Hdeib, "An examination of user perception and misconception of internet cookies," in *CHI'06 extended abstracts on Human factors in computing systems*, pp. 833–838, 2006.

[35] X. Wu, J. Yan, N. Liu, S. Yan, Y. Chen, and Z. Chen, "Probabilistic latent semantic user segmentation for behavioral targeted advertising," in *Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising*, ADKDD '09, (New York, NY, USA), p. 10–17, Association for Computing Machinery, 2009.

[36] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen, "How much can behavioral targeting help online advertising?," in *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, (New York, NY, USA), p. 261–270, Association for Computing Machinery, 2009.

[37] S. Yuan, J. Wang, and X. Zhao, "Real-time bidding for online advertising: measurement and analysis," in *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ADKDD '13, (New York, NY, USA), Association for Computing Machinery, 2013.

[38] E. C. Malthouse, Y. K. Hessary, K. A. Vakeel, R. Burke, and M. Fudurić, "An algorithm for allocating sponsored recommendations and content: Unifying programmatic advertising and recommender systems," *Journal of Advertising*, vol. 48, no. 4, pp. 366–379, 2019.

[39] M. Liu, W. Yue, L. Qiu, and J. Li, "An effective budget management framework for real-time bidding in online advertising," *IEEE Access*, vol. 8, pp. 131107–131118, 2020.

[40] E. Shehu, N. Abou Nabout, and M. Clement, "The risk of programmatic advertising: Effects of website quality on advertising effectiveness," *International Journal of Research in Marketing*, vol. 38, no. 3, pp. 663–677, 2021.

[41] R. Gomer, E. M. Rodrigues, N. Milic-Frayling, and M. Schraefel, "Network analysis of third party tracking: User exposure to tracking cookies through search," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 1, pp. 549–556, 2013.

[42] L. Olejnik, T. Minh-Dung, and C. Castelluccia, "Selling off privacy at auction," in *Network and Distributed System Security Symposium (NDSS)*, 2014.

[43] M. Veale and F. Zuiderveen Borgesius, "Adtech and real-time bidding under european data protection law," *German Law Journal*, vol. 23, no. 2, p. 226–256, 2022.

[44] M. Veale, M. Nouwens, and C. Santos, "Impossible asks: can the transparency and consent framework ever authorise real-time bidding after the belgian dpa decision?," *Technology and Regulation*, pp. 12–22, 2022.

[45] D. Chakrabarti, D. Agarwal, and V. Josifovski, "Contextual advertising by combining relevance with click feedback," in *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, (New York, NY, USA), p. 417–426, Association for Computing Machinery, 2008.

[46] F. Provost, B. Dalessandro, R. Hook, X. Zhang, and A. Murray, "Audience selection for on-line brand advertising: privacy-friendly social network targeting," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, (New York, NY, USA), p. 707–716, Association for Computing Machinery, 2009.

[47] J. Ryan and C. Santos, "An unending data breach immune to audit? can the tcf and rtb be reconciled with the gdpr?," *SSRN*, March 23 2022.

[48] J. Camenisch, A. Lysyanskaya, and G. Neven, "Practical yet universally composable two-server password-authenticated secret sharing," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), p. 525–536, Association for Computing Machinery, 2012.

[49] N. Lehto, K. Halunen, O.-M. Latvala, A. Karinsalo, and J. Salonen, "Cryptovault - a secure hardware wallet for decentralized key management," in *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pp. 1–4, 2021.

[50] Q. Wei, S. Li, W. Li, H. Li, and M. Wang, "Decentralized hierarchical authorized payment with online wallet for blockchain," in *Wireless Algorithms, Systems, and Applications: 14th International Conference, WASA 2019, Honolulu, HI, USA, June 24–26, 2019, Proceedings 14*, pp. 358–369, Springer, 2019.

[51] S. He, Q. Wu, X. Luo, Z. Liang, D. Li, H. Feng, H. Zheng, and Y. Li, "A social-network-based cryptocurrency wallet-management scheme," *IEEE Access*, vol. 6, pp. 7654–7663, 2018.

[52] R. Soltani, U. T. Nguyen, and A. An, "Practical key recovery model for self-sovereign identity based digital wallets," in *2019 IEEE Intl*

Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 320–325, IEEE, 2019.

[53] X. He, J. Lin, K. Li, and X. Chen, "A novel cryptocurrency wallet management scheme based on decentralized multi-constrained derangement," *IEEE Access*, vol. 7, pp. 185250–185263, 2019.

[54] Y. Zhang, M. Wang, Y. Guo, and F. Guo, "Towards dynamic and reliable private key management for hierarchical access structure in decentralized storage," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 3371–3380, 2023.

[55] M. Aydar, S. C. Cetin, S. Ayvaz, and B. Aygun, "Private key encryption and recovery in blockchain," *arXiv preprint arXiv:1907.04156*, 2019.

[56] G. Li and L. You, "A consortium blockchain wallet scheme based on dual-threshold key sharing," *Symmetry*, vol. 13, no. 8, p. 1444, 2021.

[57] F. Zhu, W. Chen, Y. Wang, P. Lin, T. Li, X. Cao, and L. Yuan, "Trust your wallet: A new online wallet architecture for bitcoin," in *2017 International Conference on Progress in Informatics and Computing (PIC)*, pp. 307–311, IEEE, 2017.

[58] R. Gennaro, S. Goldfeder, and A. Narayanan, "Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security," in *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings 14*, pp. 156–174, Springer, 2016.

[59] D. Boneh, R. Gennaro, and S. Goldfeder, "Using level-1 homomorphic encryption to improve threshold dsa signatures for bitcoin wallet security," in *Progress in Cryptology–LATINCRYPT 2017: 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20–22, 2017, Revised Selected Papers*, pp. 352–377, Springer, 2019.

[60] Z. Jian, Q. Ran, and S. Liyan, "Securing blockchain wallets efficiently based on threshold ecdsa scheme without trusted center," in *2021 Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*, pp. 47–51, IEEE, 2021.

[61] P. Dikshit and K. Singh, "Efficient weighted threshold ecdsa for securing bitcoin wallet," in *2017 ISEA Asia Security and Privacy (ISEASP)*, pp. 1–9, IEEE, 2017.

[62] H. Zhao, Y. Zhang, Y. Peng, and R. Xu, "Lightweight backup and efficient recovery scheme for health blockchain keys," in *2017 IEEE 13th international symposium on autonomous decentralized system (ISADS)*, pp. 229–234, IEEE, 2017.

[63] H. P. Singh, K. Stefanidis, and F. Kirstein, "A private key recovery scheme using partial knowledge," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, IEEE, 2021.

[64] J. Seo, D. Ko, S. Kim, V. Sugumaran, and S. Park, "Reminisce: Blockchain private key generation and recovery using distinctive pictures-based personal memory," *Mathematics*, vol. 10, no. 12, p. 2047, 2022.

[65] H. Rezaeighaleh and C. C. Zou, "New secure approach to backup cryptocurrency wallets," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[66] C. Daudén-Esmel, J. Castellà-Roca, A. Viejo, and J. Domingo-Ferrer, "Lightweight blockchain-based platform for gdpr-compliant personal data management," in *5th IEEE International Conference on Cryptography, Security and Privacy, CSP 2021, Zhuhai, China, January 8-10, 2021*, pp. 68–73, 2021.

[67] C. Daudén-Esmel, J. Castellà-Roca, and A. Viejo, "Blockchain-based access control system for efficient and gdpr-compliant personal data management," *Computer Communications*, vol. 214, pp. 67–87, 2024.

[68] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Advances in Cryptology-CRYPTO 90* (A. J. Menezes and S. A. Vanstone, eds.), (Berlin, Heidelberg), pp. 437–455, Springer Berlin Heidelberg, 1991.

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

[70] R. C. Merkle, *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford University, Stanford, CA, USA, 1979. AAI8001972.

[71] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," *2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 557–564, 2017.

[72] R. Kaur and A. Kaur, "Digital signature," in *Proceedings of the 2012 International Conference on Computing Sciences*, ICCS '12, (USA), p. 295–301, IEEE Computer Society, 2012.

[73] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014.

[74] B. K. Mohanta, S. S. Panda, and D. Jena, "An overview of smart contract and use cases in blockchain technology," in *2018 9th Inter-*

national Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–4, 2018.

[75] OASIS, "extensible access control markup language (xacml) version 3.0 core specification," *OASIS Standard Specification*, vol. 3.0, pp. 1–154, January 2013.

[76] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2020.

[77] G. Gutoski and D. Stebila, "Hierarchical deterministic bitcoin wallets that tolerate key leakage," in *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pp. 497–504, Springer, 2015.

[78] C. Daudén-Esmel, J. Castellà-Roca, A. Viejo, and E. J. Bel-Ribes, "Privacysmart: Automatic and transparent management of privacy policies," in *Computer Security. ESORICS 2023 International Workshops* (S. Katsikas, F. Cuppens, N. Cuppens-Boulahia, C. Lambrinoudakis, J. Garcia-Alfaro, G. Navarro-Arribas, P. Nespoli, C. Kalloniatis, J. Mylopoulos, A. Antón, and S. Gritzalis, eds.), (Cham), pp. 216–231, Springer Nature Switzerland, 2024.

[79] A. Squicciarini, A. Bhargav-Spantzel, A. Czeskis, and E. Bertino, "Traceable and automatic compliance of privacy policies in federated digital identity management," in *Privacy Enhancing Technologies: 6th International Workshop, PET 2006, Cambridge, UK, June 28-30, 2006, Revised Selected Papers 6*, pp. 78–98, Springer, 2006.

[80] M. Palatinus, P. Rusnak, A. Voisine, and S. Bowe, "Bip-39: Mnemonic code for generating deterministic keys," *Bitcoin Improvement Proposals*, vol. 39, 2013.

[81] P. Wuille, "Bip-32: Hierarchical deterministic wallets," *Bitcoin Improvement Proposals*, vol. 32, 2012.

[82] C. Daudén-Esmel, J. Castellà-Roca, A. Viejo, and V. Torra, "Blockchain-enhanced user consent for gdpr-compliant real-time bidding," in *Data and Applications Security and Privacy XXXIX* (S. Katsikas and B. Shafiq, eds.), (Cham), pp. 137–155, Springer Nature Switzerland, 2025.

[83] O. Busch, "The programmatic advertising principle," in *Programmatic advertising: The successful transformation to automated, data-driven marketing in real-time*, pp. 3–15, Springer, 2015.

[84] P. S. C. W. Group, "Openrtb version 2.6," *IAB Tech Lab Standards*, vol. 2.6, 2022.

[85] D. Sanchez, M. Batet, and A. Viejo, "Automatic general-purpose sanitization of textual documents," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, p. 853–862, 2013.

[86] C. Daudén-Esmel, J. Castellà-Roca, and A. Viejo, "Sistema para la gestión automática de las políticas de privacidad y uso de las cookies," in *XVII Reunión Española sobre Criptología y Seguridad de la Información - RECSI 2022*, pp. 83–87, Santander, Spain, 2022.

[87] C. Daudén-Esmel, J. Castellà-Roca, A. Viejo, and I. Miguel-Rodríguez, "Multi-platform wallet for privacy protection and key recovery in decentralized applications," *Blockchain: Research and Applications*, vol. 6, no. 1, p. 100243, 2025.

[88] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, pp. 36–63, January 2014.

[89] S. Suratkar, M. Shirole, and S. Bhirud, "Cryptocurrency wallet: A review," in *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pp. 1–7, 2020.

[90] O. Pal, B. Alam, V. Thakur, and S. Singh, "Key management for blockchain technology," *ICT Express*, vol. 7, 08 2019.

[91] S. Eskandari, D. Barrera, E. Stobert, and J. Clark, "A first look at the usability of bitcoin key management," in *Proceedings 2015 Workshop on Usable Security*, Internet Society, 2015.

[92] D. R. Brown, "Sec 2: Recommended elliptic curve domain parameters," *Standars for Efficient Cryptography*, 2010.

[93] M. Nystrom, "Identifiers and test vectors for hmac-sha-224, hmac-sha-256, hmac-sha-384, and hmac-sha-512." RFC 4231, Dec. 2005.

[94] J. Benet, "Ipfs - content addressed, versioned, p2p file system," 2014.

[95] R. Kumar and R. Tripathi, "Implementation of distributed file storage and access framework using ipfs and blockchain," in *2019 Fifth International Conference on Image Information Processing (ICIIP)*, 2019.

[96] L.-Y. Yeh, C.-Y. Shen, W.-C. Huang, W.-H. Hsu, and H.-C. Wu, "Gdpr-aware revocable p2p file-sharing system over consortium blockchain," *IEEE Systems Journal*, vol. 16, no. 4, pp. 5234–5245, 2022.

[97] M. J. Dworkin, E. B. Barker, J. R. Nechvatal, J. Foti, L. E. Bassham, E. Roback, and J. F. Dray Jr, "Advanced encryption standard (AES)," *Federal Inf. Process. Stds. (NIST FIPS)*, vol. 197, 2001.

[98] P. A. Grassi, J. L. Fenton, E. M. Newton, R. Perlner, A. Regenscheid, W. E. Burr, J. P. Richer, N. Lefkovitz, J. M. Danker, Y.-Y. Choong,

et al., "Digital identity guidelines: Authentication and lifecycle management [includes updates as of 03-02-2020]," *National Institute of Standards and Technology Special Publication 800-63B*, 2020.

[99] V. Gayoso Martínez, L. Hernández Encinas, and A. Queiruga Dios, "Security and practical considerations when implementing the elliptic curve integrated encryption scheme," *Cryptologia*, vol. 39, no. 3, pp. 244–269, 2015.



UNIVERSITAT
ROVIRA i VIRGILI