Universidade da Beira Interior
Engenharia

Streamlining the Usage of Authorization or Digital Signature in Digital Processes

Rui Miguel Monteiro Raposo

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2.º Ciclo de Estudos)

Orientador: Professor Doutor Pedro Ricardo Morais Inácio

Covilhã, março 2023

Declaração de Integridade

Eu, Rui Miguel Monteiro Raposo, que abaixo assino, estudante com o número de inscrição 10696 de Engenharia Informática da Faculdade de Engenharias, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 31/03/2023

Acknowledgements

The work described herein is also the sum of different personal contributions that enabled me to reach this stage, some of them I mention in this section.

First of all, I would like to acknowledge my supervisor, Professor Pedro Inácio, for always trying to help up to the last minute. His guidance was crucial to reach this stage.

I would like to thank the dearest friends and research team members Joana Costa, Bernardo Sequeiros and Tiago Roxo, for all the amazing discussions and humour. Their work ethics and focused approach to tasks motivated me throughout this last year.

I would like to thank my family, for supporting and encouraging me at all times, either directly in my studies, or indirectly through their wishes and hopes.

The work described in this dissertation was carried out at the Instituto de Telecomunicações, *Network Applications and Services* – Covilhã Laboratory, in Universidade da Beira Interior, at Covilhã, Portugal.

Resumo

Está em curso o chamado processo de transformação digital, alavancado pelo rápido desenvolvimento tecnológico em geral, e até pela recente pandemia COVID-19 em particular. Apesar de, numa abordagem superficial, esta transformação possa parecer apenas a transposição de procedimentos ou documentos para um suporte informático, a verdade é que nem tudo o que o humano faz de forma simples manualmente pode ser transponível facilmente para o digital (e.g., uma eleição), tal como há aspetos que são muito melhor conseguidos no mundo digital que no real (e.g., uma assinatura digital qualificada). O trabalho espelhado nesta dissertação explora a transposição para o digital de tarefas muito importantes em organizações e entidades dos dias de hoje: as de autorizar ou não autorizar, aprovar ou não aprovar ou despachar assuntos em sistemas de gestão documental.

A integração de fortes garantias de segurança em processos digitais está tipicamente associada a custos computacionais e de usabilidade. O tema particular da dinamização e motivação da utilização de autorizações digitais é difícil de abordar hoje em dia porque não existe um mecanismo amplamente adotado ou acordado para as mesmas, embora exista regulamentação e tecnologia amplamente reconhecida para as assinaturas digitais na Europa. Não existe um formato padrão para as autorizações digitais, embora a intuição sugira que estas devem ser formadas por uma mensagem com pertinência temporal, colada a um documento de apoio através de alguns meios digitais fortes, tais como uma assinatura digital qualificada ou um mecanismo de código de autenticação de mensagem.

Este projecto analisou o panorama em termos de legislação e mecanismos recomendados para autenticação, assinatura digital e selos digitais na Europa, para depois avançar para a proposta de uma estrutura de dados para uma autorização digital e mais tarde para a proposta dos algoritmos para construir e verificar as autorizações digitais. O esquema aqui proposto baseia-se apenas em criptografia de chave simétrica, com o objetivo de minimizar o impacto na gestão de chaves e maximizar a sua potencial adopção. Entretanto, são também identificados e discutidos os possíveis regimes (hierárquicos vs. igualdade) para a estruturação de autorizações em grandes organizações ou entidades, uma vez que as suas diferenças têm repercussões em termos das diferentes primitivas criptográficas e tecnologias que podem ser aplicadas. Algumas destas primitivas são também discutidas nesta dissertação.

Os algoritmos propostos para a construção e verificação de autorizações digitais foram validados utilizando a ferramenta *ProVerif*. As conclusões principais são que ainda há muito terreno a percorrer neste contexto, mas que é possível integrar esquemas seguros de autorização digital no curto a médio prazo, e que os esforços se devem concentrar a seguir na definição do formato e dos mecanismos que precisam de ser amplamente adotados e reconhecidos, permitindo passar de intra-entidades para inter-entidades.

Palavras-chave

Autorização Digital, Assinatura Digital, Cibersegurança, Esquema de Lamport, Função de *Hash*, Requisitos de Segurança, Selo Digital, Verificação de Esquemas Criptográficos

Resumo Alargado

Introdução

Este capítulo descreve o âmbito e motivação desta dissertação, apresentando o enquadramento e descrição dos problemas encontrados. A necessidade do desenvolvimento desta dissertação, bem como os objetivos e as principais contribuições, são também discutidos neste primeiro capítulo.

Enquadramento, Descrição do Problema e Objetivos

O processo de transformação digital acentuou-se durante a pandemia COVID-19. Esta transformação visa, fundamentalmente, a transposição de procedimentos ou documentos para um suporte informático que, logicamente, consiste normalmente numa tarefa complexa. O conceito de autorização, bastante utilizado no quotidiano (físico), permite a realização de uma ou mais tarefas. No entanto, não existe qualquer valor legal associado a essa mesma autorização, visto que a maioria das vezes é efetuada verbalmente. Dada a importância das autorizações nas rotinas diárias das entidades, torna-se necessário desenvolver um mecanismo digital que represente tal processo e que apresente confiança para com os utilizadores.

Cada entidade possui os seus próprios regulamento e modo de funcionamento e, consequentemente, as suas próprias rotinas digitais. Estas rotinas são esquematizadas com base na associação da génese da entidade em questão com os regimes hierárquicos e igualdade, distinguindo-se pela possibilidade de indivíduos poderem assinar/autorizar em nome de outros. Com isto, nesta dissertação definem-se como objetivos: 1) a definição de autorização digital; 2) a construção de um mecanismo digital que defina as autorizações digitais; 3) o desenvolvimento de um protótipo onde as autorizações digitais (mecanismo digital) estejam embutidas de forma segura.

Principais Contribuições

Com base nos objetivos identificados previamente, as principais contribuições desta dissertação consistem em: 1) um protótipo alusivo à implementação de autorizações digitais utilizando apenas criptografia de chave simétrica; 2) uma análise extensa ao estado de arte relativo às assinaturas eletrónicas e o enquadramento legal destas; e 3) identificação e análise dos regimes adotados pelas entidades no quotidiano, visando a melhor forma de implementar as assinaturas/autorizações digitais.

Estado da Arte

O capítulo do Estado da Arte aborda quatro tópicos fundamentais. Primeiramente, os diferentes tipos de assinaturas eletrónicas e o seu enquadramento legal são identificados, enaltecendo-se as assinaturas eletrónicas qualificadas, visto que possuem o mesmo valor de uma assinatura manuscrita. Seguidamente, dois tipos de assinaturas digital são descritos, permitindo replicar certas ações não inerentes às assinaturas manuscritas, acrescentando e assegurando que restrições específicas são devidamente seguidas, mas também otimizar a produções dessas mesmas assinaturas digitais (armazenamento e rotação de chaves criptográficas). Em terceiro plano descreve-se o esquema em cuja verificação de autorizações digitais se baseia, evidenciando-se a necessidade de ajustar a sua implementação dada a vulnerabilidade adjacente a este esquema. Por fim, a ferramenta *ProVerif* é introduzida e a sua importância devidamente justificada, dado que é responsável pela validação do protótipo referente às autorizações digitais. Note-se que esta ferramenta apenas permite efetuar a validação do secretismo e autenticação presente no protótipo desenvolvido.

Análise de Autorizações Digitais e Regimes

Neste capítulo, é incluida a definição de autorização digital, fundamentada pela diferença aquando comparada com as assinaturas digitais. Para além disto, o âmbito em que pode ser introduzida, e um mecanismo passível de ser implementado para garantir a produção de autorizações digitais é também apresentado. As autorizações digitais, tal como o próprio nome indica, permitem autorizar ou aprovar uma dada atividade a ser realizada sem necessitar qualquer ação formal (assinatura), sendo também equiparada com uma autorização via oral por parte de um indivíduo. Com isto, o mecanismo apresentado assenta sobre a utilização de mecanismos criptográficos seguros, mais precisamente, Message Authentication Code (MAC) e funções de *Hash*.

Por fim, considerando que múltiplas entidades possuem métodos distintos de abordar as suas rotinas digitais diárias, são descritos os regimes mais utilizados no quotidiano pelas entidades, mais concretamente, os regimes hierárquicos e de igualdade. O regime hierárquico consiste na atribuição de documentos por múltiplos indivíduos, requerendo as suas autorizações/pareceres até atingir o último destinatário do fluxo predefinido, *e.g.,* reitor de uma universidade. Note-se que este último deve fornecer uma assinatura digital ao invés de uma autorização digital (sendo este o único a poder providenciá-la). Quanto ao regime igualdade, não existe o processo de seguir um fluxo predefinido que estabelece quais os indivíduos a autorizar um específico documento, podendo qualquer indivíduo autorizar ou assinar em nome de um outro (necessitando ter os devidos requisitos para efetuar tal operação). Após descritos ambos os regimes, é feita a descrição do caso de estudo cujos protótipos vão ser desenvolvidos: o System for Document Management used in UBI (GDUBI).

Protótipos

Este capítulo consiste nas descrições dos protótipos desenvolvidos no âmbito desta dissertação. Como já referido anteriormente, os protótipos são especificamente esquematizados em prol da aplicação GDUBI, sendo os protótipos da assinatura e selo digital aglomerados num só protótipo. Deste modo, o protótipo das autorizações digitais merece o maior destaque ao longo deste capítulo, sendo apresentados os algoritmos que constituem todo o ciclo das autorizações. Os algoritmos relativos à geração de chaves criptográficas, produção e verificação das autorizações digitais, e o mais importante para garantir a integridade de todo o sistema, renovação das autorizações digitais, são apresentados. Segue-se uma análise de segurança do protótipo das autorizações digitais, evidenciando-se a importância da forma como se gere e produz os dados categorizados como críticos/sensíveis (*i.e.,* chaves criptográficas, palavras-passe, etc.). Demonstra-se ainda o quão dependente o sistema é do bom funcionamento do algoritmo de renovação de autorizações digitais, sendo descartada a vulnerabilidade associada ao processo de verificação de autorizações digitais (*small n attack*).

Verificação Formal do Protótipo de Autorizações Digitais

Para comprovar que o protótipo referente às autorizações digitais estaria devidamente desenvolvido, submeteu-se à verificação formal através da ferramenta *ProVerif*, avaliando rigorosamente o nível de secretismo e autenticação presente. Tratando-se de um protótipo amplo com vários algoritmos identificados, necessita-se numa primeira fase definir qual o ponto crítico que poderá comprometer o sistema, tendo sido diagnosticada a fase de verificação de autorizações digitais como aquela que possui uma maior taxa de exploração por parte de atacantes. Este processo de verificação foi traduzido para a linguagem *ProVerif*, bem como o protocolo *Handshake* que assegura a transferência de dados entre as entidades envolvidas (cliente e servidor).

Após devidamente implementado, foram desenhados testes sob o código produzido para espelhar a veracidade do sistema, ou seja, averiguar se os requisitos do sistema foram corretamente atingidos. Os testes assentam na exposição de vários dados ao longo do processo de verificação, tais como, palavra-passe, *salt*, etc. Os resultados obtidos corresponderam aos expectáveis, levando a que o protótipo corresponda aos requisitos previamente definidos.

Conclusões e Trabalho Futuro

O trabalho desenvolvido ao longo desta dissertação permitiu extrair diversas conclusões secundárias. A definição precisa do conceito de autorização digital ainda não foi formada de uma forma generalizada e na literatura da especialiade, colocando assim um obstáculo num dos objetivos desta dissertação. Como tal, recorreu-se à própria definição deste conceito, tendo desbloqueado a esquematização do seu possível mecanismo. Contudo, desconhece-se também um formato específico da autorização digital, *i.e.,* a forma como é embutida, ou não, nos documentos; quais os mecanismos criptográficos a usar, etc. Desse modo, múltiplos mecanismos foram desenhados, tendo prevalecido aquele que se encontrava em maior concordância com a definição, mas também com a menor necessidade de recursos possível.

Como trabalho futuro são sugeridas algumas melhorias a realizar, nomeadamente a necessidade de formular as provas para abranger todo o leque de requisitos de segurança (podendo ser ainda acrescentados mais), bem como a própria implementação do protótipo descrito ao longo desta dissertação.

Abstract

The so-called digital transformation process is underway, leveraged mostly by the rapid technological development, but also more recently by the COVID-19 pandemic in particular. Although, the untrained eye, this transformation may seem to be only the transposition of procedures or documents to a corresponding digital format, the truth is that not everything that humans do manually and in a simple way can be easily transposed to digital (e.g., an election), just as there are aspects that are much better accomplished in the digital world than in the real one (e.g., a qualified digital signature). The work discussed in this dissertation explores the transposition to the digital world of very important tasks in organizations and entities today: those of authorizing or not authorizing, approving or not approving, or dispatching issues in document management systems.

Integrating strict security assurances to digital processes typically comes with both computational and usability costs. The particular subject of streamlining the usage of digital authorizations is difficult to address nowadays because there is no widely adopted or agreed mechanism for them, thought there is regulation and widely recognized technology for digital signatures in Europe. There is no standard format for digital authorizations, though intuition suggests that they should be formed by a message with temporal pertinence glued to a supporting document via some strong digital means such as a qualified digital signature or a message authentication code mechanism.

This project looked into the landscape in terms of legislation and recommended mechanisms for authentication, digital signature and digital seals in Europe, to then step up to the proposal of a data structure for a digital authorization and later on to the proposal of the algorithms to build and verify digital authorizations. The scheme proposed herein is based on symmetric key cryptography only, aiming to minimize impact on key management and maximizing potential adoption. In the meanwhile, the possible regimes (hierarchical vs. equality) for structuring authorizations in large organizations or entities are also identified and discussed, since their differences resonate into different cryptographic primitives and technologies being later applied. Some of these primitives are also discussed in this dissertation, specially the ones used to build the algorithms of the proposed scheme.

The proposed algorithms for constructing and verifying digital authorizations were validated using the *ProVerif* tool. The main conclusions are that there is still much ground to be covered in this context, but that it is possible to integrate secure digital authorization schemes in the short to medium term, and that efforts should be focused next in the definition of the format and mechanisms that need to be widely adopted and recognized, enabling moving from intra- to inter-entities.

Keywords

Cybersecurity, Digital Authorization, Digital Seal, Digital Signature, Hash Functions, Lamport One-Time Password Scheme, Security Requirements, Verification of Cryptographic Schemes

Contents

List of Figures

List of Tables

Acronyms

ACM Association of Computing Machinery

API Application Programming Interface

AWS Amazon Web Services

CA Certification Authority

CCS Computer Classification System

DBMS Database Management System

eIDAS Electronic Identification and Trust Services

ENISA European Network and Information Security Agency

EU European Union

EC European Commission

GDUBI System for Document Management used in UBI

HTTPS Hypertext Transfer Protocol Secure

ID Identifier

IV Initialization Vector

KDF Key Derivation Function

KMS Key Management Service

MAC Message Authentication Code

PKI Public-Key Infrastructure

QSCD Qualified Signature Creation Device

RSA Rivest Shamir Adleman

SAD Signature Activation Data

QTSP Qualified Trust Service Provider

TSP Trust Service Provider

UBI Universidade da Beira Interior

Chapter 1

Introduction

This chapter describes the scope of the dissertation and motivation in section 1.1, followed by the problem statement and objectives in section 1.2. A discussion on the proposed approach and main contributions is included in section 1.3, and section 1.4 finally presents the structuring of this document.

1.1 Motivation and Scope

Internet has conditioned the everyday life by streamlining tasks and forcing several security areas to be tuned in to guarantee the confidence within the digital world. While the ability for companies to seal contracts in a matter of seconds represents, on one hand, an essential service that the digital era provides to the institutions, on the other hand, there is a need to establish cryptographic mechanisms that ensure the veracity and security of all the concerned steps. Digital signatures ensure a variety of properties which are vital within the business landscape. However, implementing signatures in such an environment requires a solid planning, since there are legislative norms that must be followed in order to produce legally recognizable signatures. This dissertation describes the work performed in the areas of computer security and document management systems. It focuses on how to guarantee the integrity of an information flow in which entities (people) provide consent, comment and authorize a given step of that flow based on previous authorizations and a document (e.g., a memorandum). With this, under the 2012 version of Association of Computing Machinery (ACM) Computer Classification System (CCS) [3] the scope of this dissertation is under the following categories:

- **Security and privacy → Cryptography → Symmetric cryptography and hash functions → Hash functions and message authentication codes**;
- *Security and privacy → Formal methods and theory of security → Logic and verification*;
- Security and privacy → Cryptography → Information-theoretic techniques.

Along with this, every case scenario should be handled in a particular form, since each one has a specific objective. However, the development of pre-defined prototypes may support common case scenarios, speeding up their implementation. In this way, the worst case scenario stands for a completely customizable prototype, requiring the fulfilment of several

issues in order to generate the desired and suitable prototype. The development of these prototypes requires a substantial effort for researching case scenarios, as well as leveraging the most common ones. However, the opportunity to create those leads to possibly upgrading institutions into a new digital age.

1.2 Problem Statement and Objectives

Despite the benefits of digital signatures, there are still major barriers to their implementation within institutions. Individuals are used to the handwritten signature and often emphasise any user friendliness issues as major downsides of digital signatures. However, the sealing process in the context of establishing agreements or contracts with other institutions is rather slow when using handwritten signatures, in which case users can more easily identify advantages of going full digital. Adding to this, common weak digital signatures (or digitalized signatures) cannot provide all the features of handwritten signatures, limiting the range of individuals that are able to use them, leading institutions to prefer waiting a few days for the conclusion of contracts rather than implementing digital signature in such environments, since the adaptation and learning curve for individuals might be costly.
Fortunately, due to cryptographic advances, several features of handwritten signatures can be achieved using specific digital signature schemes, allowing individuals to boost their confidence in this digital environment. With this, only the development of prototypes that supports institutions to apply this mechanism in their daily routines remains. This development require a deep analysis concerning the proposed case scenario, leveraging several concepts related to the regime of the institution, *e.g.*, hierarchical, but also to the recognition level of the generated signatures, since institutions may not desire the generation of recognizable signatures across the European Union (EU).
Along with the problems concerning wide acceptance of new technologies, there is also the issue of correctly implementing them in enterprise processes. One such process is the one of the management of documents and authorisations. Nowadays, many internal processes of an institution or company depend on a chain of authorizations/opinions/decisions that are often associated with documents supporting them. An authorization or dispatch is often a message (e.g., *Payment is allowed.* or *Payment is not allowed.*) that is glued to a document, either digitally or physically. Obviously, the right way to glue two artefacts in the digital realms is through a qualified digital signature, but this poses usability and implementation challenges that may hinder the usefulness of such system. The work described herein focuses precisely on how authorizations in such systems can be cryptographically assured without compromising the usefulness and efficiency.
The main objetives of this work were thus to better define digital authorizations, propose a format for such authorizations and to assess to what extent could security assurances be embedded in a document and authorization system without becoming a key management,

implementation or usability burden. This work focused on assessing if it was possible to achieve the functionality of a user being able to securely verify such authorizations without resorting to public key cryptography.

1.3 Proposed Approach and Main Contributions

To tackle the previously identified problems and achieve the aforementioned objectives, this work was structured into the five main steps:

1. Studying the legal framework and regulation in terms of digital signatures, seals and authorizations in the EU, so as to understand if and how such specifications could be used in the context of this project;

2. Focusing more on the digital authorization topic, aiming to understand how this is perceived or used in the context of contemporary companies or entities;

3. Analyse and reflect on the main security and functional requirements of digital authorizations, and study particular cryptographic primitives, signature and authentication schemes, available in the specialized literature and with potential applicability in the scope of the project at hands;

4. To propose a scheme for digital authorizations only dependent on symmetric key cryptography (to minimize the potential impact of having to deal with public/private key pairs by default);

5. To provide some closure to the entire discussion by formally validating the proposed scheme.

An additional step not described above comprised the writing of this dissertation.
The main contribution of this work was the conceptualization of a scheme for digital authorizations based solely on symmetric key cryptography. This contribution is the subject of chapter 4. Nonetheless, the discussion on details of the legal framework and accepted technologies in terms of electronic signatures and electronic seals, mostly at the European scale, comprise a very interesting outcome of this work also, specially the investigation on the signatures used in digital contracts with experts acting for the European Commission (EC). This contribution is the main subject of chapters 2 and 3. Last, but not less important, is the discussion on what a digital authorization is, how it can be represented, and which regimes (hierarchical or equality) might apply in different management structures. Chapter 3 focuses partially on this contribution.

1.4 Document Organization

This document is organized in six chapters. The contents and overall structuring of the chapters of this document can be summarized as follows:

- Chapter 1 – **Introduction** – presents the dissertation scope and motivation, the addressed problems, the goals and the approach to achieve them and, finally, the organization of this document;
- Chapter 2 – **State of the Art** – describes the current legislation surrounding the digital signatures across the EU, identifies and clarifies different notions of electronic signatures and seals. Lastly, two digital signature schemes are explained, as well as the impact they may have within the scope of application addressed in this dissertation;
- Chapter 3 – **Research of Digital Authorizations and Regimes** – presents the different ways that entities approach digital routines involving digital authorizations, discussing the cons and pros of each of them. Digital authorizations are defined, and compared with digital signatures. Moreover, the developed digital authorization mechanism is also described;
- Chapter 4 – **Prototypes** – describes the developed prototypes regarding digital authorizations, signatures and seals. The digital authorization prototype is further detailed, since it comprises the main innovation point of this work;
- Chapter 5 – **Formal Verification of Prototypes** – after being developed, the way to prove that the prototypes are secure requires the application of formal verification methods. Therefore, the work described in this chapter is the one ensuring that the digital authorization prototype is surrounded with secrecy and authentication;
- Chapter 6 – **Conclusions and Future Work** – identifies the main conclusions retrieved from this project, as well as the future work that must be performed to improve what has been done.

Chapter 2

State of the Art

2.1 Introduction

This chapter describes the current environment of electronic signatures, clarifying and explaining crucial concepts, as well as the legislation surrounding them in section 2.2. Furthermore, two digital signature schemes are analyzed in section 2.3, followed by the description of the Lamport authentication scheme in section 2.4, and the formal verification tool that was used in this work in section 2.5. Lastly, the conclusions drawn in section 2.6.

2.2 Electronic Signatures and Legislative Framework

Nowadays, the concept of digital signatures is often used out of context, since people associate every possible way to identify someone through digital frameworks as a signature. In 2016, when Portugal joined the Electronic Identification and Trust Services (eIDAS) Regulation[1] [4], concepts concerning the types of electronic signatures were explicitly defined, avoiding misunderstandings. Therefore, these types are detailed in sections 2.2.1, 2.2.2 and 2.2.3, presented in ascending order regarding reliability. Section 2.2.4 presents the Qualified Trust Service Providers (QTSPs) [5] importance within such environment and, finally, section 2.2.5 describes the importance of the use of electronic seals, and also its differences with electronic signatures.

2.2.1 Simple Electronic Signatures

The introduction of the name of a person in the last line of a document could correspond to an electronic signature according to the eIDAS Regulation [4]. However, in the digital environment, it is impossible to guarantee the non-repudiation factor and, fundamentally, the integrity and authenticity of the document.
Despite these flaws, the security level[2] can be increased through additional methods (*e.g.,* multi-factor authentication), where the signer requires completing an additional step before signing. Moreover, the audit trail[3] generation is not required. However, several user

[1] Framework that recognizes eSignatures as legally valid in the EU.
[2] Legal and strength level.
[3] Security-relevant chronological record of a transaction.

details (*e.g.,* e-mail) can contribute to building a generic signature, allowing possible traces in critical scenarios.

2.2.2 Advanced Electronic Signatures

Providing a high security level, this advanced type of signature complies with the norms presented in article 26 of the eIDAS Regulation [4]. Specifically, it states that an electronic signature of this type must accomplish four crucial requirements: i) uniquely linked to the signatory; ii) capable of identifying the signatory; iii) created in a way that allows the signatory to retain control; and iv) any subsequent change in the data is detectable.
Based on these requirements, it is possible to verify that each point comprises a feature that the digital signature mechanism contains, since the Public-Key Infrastructure (PKI) is present. Thus, a digital certificate links a signature to a verified Identifier (ID), which is issued by a Trust Service Provider (TSP) or a Certification Authority (CA). Despite the security level of this approach, it is still not possible to claim that it has the same value as a handwritten signature.

2.2.3 Qualified Electronic Signatures

As mentioned in section 2.2.2, the advanced electronic signature provides a truly advanced security level. However, in order to ensure the same weight as the handwritten signature, an additional level of assurance on the identity of the signatory and signature creation must be performed.
Considering these two requirements, firstly, the identity of the signatory is achieved by a TSP, as the advanced electronic signatures, but those must be marked as a QTSP[4], so that it possesses a greater recognition level. Secondly, the signature creation requires it to be performed using a qualified device. A Qualified Signature Creation Device (QSCD) must ensure: i) private keys secrecy; ii) generation of the key-pair (requiring cryptographic techniques); iii) private keys can only be used by the right owner; and iv) compliance to the stringent standards for Qualified Electronic Signature [4].
In order to digitally sign a document, the signatory needs to access an application (*e.g.,* web application) and authenticate themselves. However, the application must ensure as much as possible that:

- the account was not compromised;
- the flow is clear, and the signatory is aware of all the actions that have been performed (avoiding miss clicks or mistakes).

[4]A QTSP stands for a TSP that has been granted a qualified status by a national competent authority.

The following figure 2.1 illustrates a case scenario of a qualified signature process, while using the European Commission Expert Portal web application.

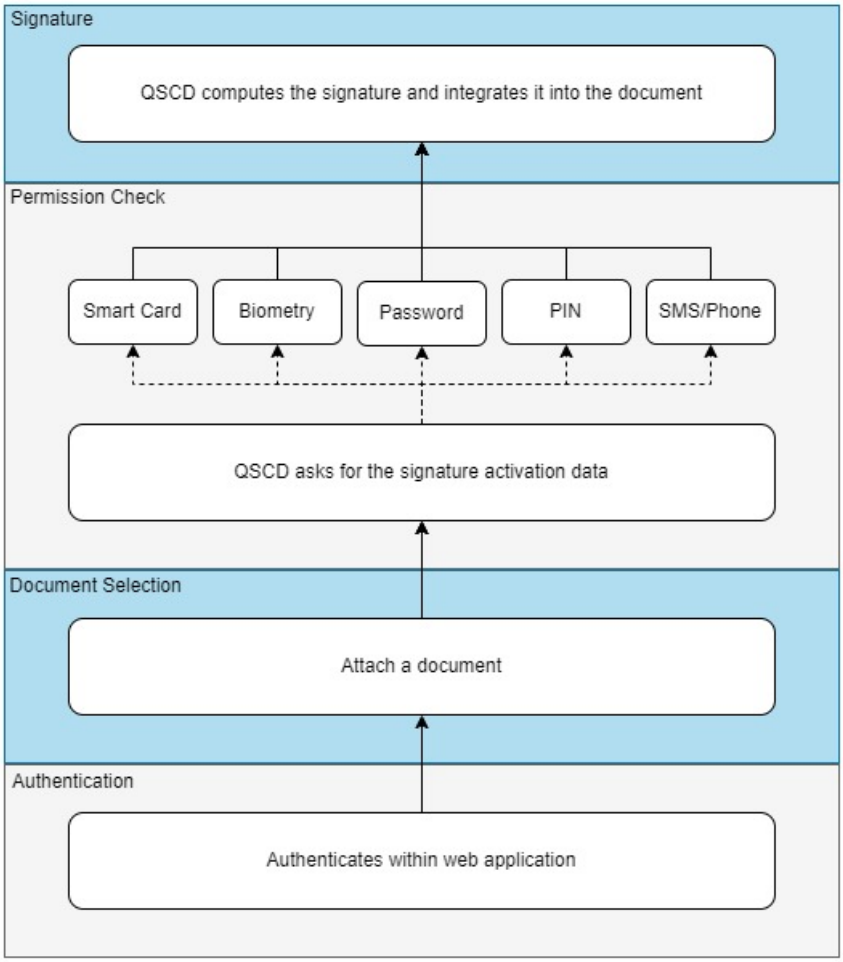

Figure 2.1: Qualified signature creation process.

Based on the previous figure, it is possible to identify how user-friendly the portal is, with the Signature Activation Data (SAD)[5]being the most important factor within the process from the presented four phases:

1. Authentication – authenticate within the web application (*e.g.,* European Commission web application) using the required credentials;
2. Document Selection – attach the desired document;
3. Permissions Check – after the document attachment, the user must introduce the SAD, ensuring that no mistake was made and verifying the signatory identification, unlocking the access to the private key;

[5]Factor that enables the ability of using the private key.

4. Signature – after correctly introducing SAD, the QSCD computes the signature and integrates it into the document.

Although all the agreements and contracts related to European Commission possess such type of signatures, their integration is not explicitly described, requiring a detailed analysis to identify the way it is embedded. In the scope of this work, such signatures were analyzed (figure 2.2). In both types of document there are, at least, two restricted signature fields addressing: i) member of European Commission; and ii) beneficiary/expert.

Given this, in order to try to understand how the signature is embedded within the document, as well as where it is located, the following figure 2.2 presents of the graphical artefact of the produced digital signatures within the contract.

Figure 2.2: Signature fields within a contract.

Based on figure 2.2, it is possible to identify similarities between both fields, more specifically, the value between the last two dashes which, from now on, is defined as γ. The initial idea of what this value would be referred to was based on the serial number of the document. However, after performing the same analysis for a longer agreement involving almost 40 digital signatures, part of this γ value was also used in several signature fields, meaning that the initial idea is easily rejected. In order to avoid misunderstandings, the following table 2.1 presents the most commonly used γ prefixes of both documents[6] as well as the used suffixes.

[6]Note that, documents were submitted in different years, the contract in 2017 and the agreement in 2016.

	γ	
	Preffix	**Suffix**
Contract	Jj71zxYb8yr	8W2ZCuUHyfO
Agreement	PHsIUMVSXYC	I4G08U3ZwBS
	PHsIUMVSXYC	Mc6XrQfmgmu
	Jj71zxYb8yr	yEnwGP06Oly
	Jj71zxYb8yr	XzfzJskzRKva

Table 2.1: γ values presented on both documents.

Even with all these patterns, it is impossible to attest that this γ stands for the serial number of the document, since nothing is formally documented either by European Network and Information Security Agency (ENISA) [6] or the EU, to the best of the knowledge of the author. Adding to this, the value which precedes γ may be the digital signature. However, the size of that value does not belong to any ordinary digital signature result, meaning that it can be the digital signature concatenated with something else, or something else entirely.

2.2.4 QTSPs Across EU

Although multiple entities have the ability and resources to provide an Application Programming Interface (API) featuring access to develop electronic signatures, these entities are required to be approved within the EU as part of a QTSPs list, so that their generated signatures become recognizable. Considering the EU countries, each of them has a different list. However, this does not imply that a specific country may only trust in its own list.
QTSP has a major role within the electronic seals and signatures environment, performing a fundamental task concerning augmenting signatures, providing resilience within processes through resources such as trusted time stamps, enabling a trust evidence of the signing time of a specific signatory and preventing signature rejections based on the expiration of signing certificates.
DocuSign [7] is a QTSP often used across Europe and makes the electronic signature process simple for anyone. Individuals are required only to authenticate within the mobile or web application, attach the document, locate the signature field and, finally, send it through the email to the desired recipients. The way the non-registered recipients sign the document must be performed carefully, since any data is provided to the application. However, DocuSign provides a simple solution to solve this specific aspect, since it generates a cryptographic key-pair to the corresponding email of the recipient. Note that DocuSign also works as a CA. However, the registered users can always select the desired one, from the defined partners, to establish the connection with their own keys.

2.2.5 Electronic Seals

Multiple entities stamp their seals into relevant documents as a way of guaranteeing its integrity and origin. Besides this, similarly to the electronic signatures, the seals can be

typified in Advanced and Qualified Electronic Seals [6].
Despite this, seals and signatures have different purposes in a way that signatures can be performed by any individual, expressing an opinion of agreement or disagreement. Moreover, it can also be used when a legal person does not possess a linked seal, requiring the generation of a signature based on individual key-pair to ensure a document authenticity and origin on behalf of the entity, *e.g.,* top manager of a signing entity. On the other hand, seals are associated to legal persons, since these cannot afford any signature according to eIDAS Regulation [4]. Logically, electronic seals use PKI, since it is necessary to link the electronic seal to a legal person figure through a digital certificate. Table 2.2 compares several properties concerning qualified electronic seals and signatures.

	Qualified Electronic Seals	**Qualified Electronic Signatures**
Authenticity	X	X
Origin	X	X
Non-Repudiation	X	X
Integrity	X	X
Legal Persons	X	
Triggered by anyone	X	
PKI	X	X

Table 2.2: Comparison between qualified electronic seals and signatures.

Based on table 2.2, electronic signatures cover almost all the functionalities of electronic seals. However, stamping an electronic seal is not exclusive to a single person within the signatory entity (*i.e.,* several persons may sign on behalf of an entity). On the other hand, documents must be signed using a per user private key, meaning that only a single person must perform it. Therefore, when a given entity creates a document that establishes a contract, it should stamp the electronic seal within the document to ensure its integrity and origin, and an individual that belongs to the management sector of the concerned entity must digitally sign in the specific field (commitment).

2.3 Digital Signature Schemes

Digital signatures are currently the best digital mechanism to ensure the authenticity and integrity of a message. However, multiple features which are provided in handwritten signatures cannot be accomplished within ordinary digital schemes, requiring the emergence of other schemes for that purposes. The subsections 2.3.1 and 2.3.2, describe the main features of two schemes which provide useful features in daily document signing routines.

2.3.1 Forward-Secure Digital Signature

The responsible entities for signing documents need to ensure a secure solution to counter-attack the private key exposure problem, since all the signatures provided by an exposed

key can be forged [1]. The most simple and widely used solution is based on the distribution of the key across multiple servers, through secret sharing approach, nevertheless, this has a considerable cost and users may not have enough resources to distribute keys over multiple servers. Based on this, *forward security* is suggested as a solution for this scenario, ensuring the protection of the key exposure without the need of multiple resources and, involving the timestamp of the signature as a crucial variable.

Initially, this scheme requires a pre-definition of T time periods where each of these is associated to a private key SK_i as shown in figure 2.3.

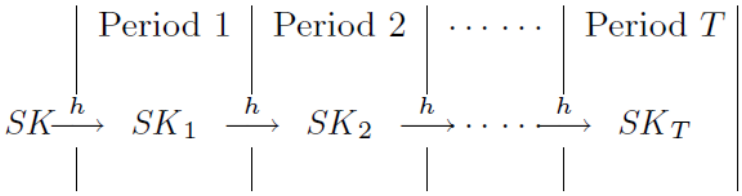

Figure 2.3: Key evolution diagram adapted from [1].

As soon as the first time period expires, the private key SK updates to a new one SK_1, implicitly implying the presence of a secure key evolution function h, *i.e.,* a one-way evolution function. Hence, entities only require to be aware of the current signatures and the succeeding ones, since the previous ones cannot be forged.

Furthermore, the output of the function must form a valid pair with the fixed public key, increasing the need for a mechanism which relates the output key to the input one and, consequently, to the public key. Thus, Bellare and Milner [1] rely on the hardness of factoring Blum-Williams integers [8] to increase the security level, but also to define the composition of the keys and the relation between both; according to this, the following table 2.3 presents the description of the variables which are used in the key-pair generation algorithm demonstrated in figure 2.4.

Variable	**Description**
p, q	Large prime numbers, each of them congruent to $3 \ mod \ 4$.
l	Amount of points in the key.
T	Total of time periods.
SK	Private key.
PK	Public key.
U	Values for the public key (core).

Table 2.3: Description of the variables regarding the key-pair generation algorithm.

```
Algorithm_Key_Generator(l, T)
  N ← p × q
  for i = 1, ..., l do
    S_i ← Z*_N;  U_i ← S_i^{2^(T+1)} mod N
  endfor
  SK_0 ← (N, T, 0, S_{1,0}, ..., S_{l,0}); PK ← (N, T, U_1, ..., U_l)
  return (PK, SK_0)
```

Figure 2.4: Algorithm of the generation function of the key-pair based on [1]

Figure 2.4 illustrates the relation between both keys and how the incorporation of Blum-Williams integers increase the hardness of breaking this scheme as well. Moreover, the private key SK possesses one more attribute than the public one, since the third attribute of the private key stands for the key iteration, which means that, when generating keys, this value starts at zero, and it is incremented after each iteration of the key evolution function. This last comprises two small steps: square and modulo, which guarantee the hardness to discover the private key and, essentially, the connection between keys of distinct time periods. Figure 2.5 presents the key evolution algorithm.

```
Algorithm_Key_Evolution(SK_{t-1}, t)
  for i = 1, ..., l do
    S_{i,t} ← S^2_{i,t-1}
  endfor
  SK_t ← (N, T, t, S_{1,t}, ..., S_{l,t})
  return SK_j
```

Figure 2.5: Algorithm of the key evolution function based on [1].

Due to this function, a system which integrates the forward-secure scheme does not need to store unnecessary private keys, since those that do not correspond to the current time period can be deleted, and it is always possible to verify signatures issued by them with the public key. Therefore, the storage allocated by the cryptographic keys are practically the same as if the system was using an ordinary digital signature scheme. However, the computational cost stands for a significant disadvantage while using this forward-secure scheme. In a large-scale system where a forward-secure scheme is implemented, and thousands of users are registered, with each of those having a specific key-pair, the amount of computations related to cryptographic management reaches large values, being directly proportional to the registered users. Considering this, choosing the right digital signature scheme demands a deep analysis concerning all functional requirements of the system, but

also the non-functional ones, since computational resources may be limited.
Furthermore, key lifetimes and rotations [2] prove to be relevant topics when comparing both schemes, since keys are usually modified if certain scenarios are experienced, specifically the ones presented in the following table 2.4.

Name	Description
Exposed Key	The key has been compromised or someone from an organization has left.
Cryptoperiod	The associated time period has elapsed.
Cryptogram Total Volume	The key has been used to encrypt a specific amount of data.
Algorithm Security	The algorithm security has been compromised.

Table 2.4: Scenarios that cause the modification of cryptographic keys [2].

Regardless of the scenario, each of them will lead to storage of old keys, which means that when comparing to the forward-secure one, unnecessary storage is being used. Beyond this, when a key is compromised, all the data which was associated to it is discarded and a new key-pair must be generated and used to provide new signatures. This forward-secure scheme is only dedicated to digital signatures. A key exposure for encrypted data would, on the contrary, lead to the compromise of the data.

2.3.2 Proxy Digital Signature

There are countless situations where a document has to be signed in a specific place by a person that is unavailable at that moment, postponing possible agreements or contracts between entities for an unspecified date. Given this, the proxy scheme allows, during the process of key-pair generation, the delegation of specific members to sign on behalf of someone without compromising the private key [9]. Therefore, an additional key is defined and necessary for this approach to work, the **proxy key**. The usual key-pair used in ordinary digital signature schemes, will now be composed of three different ones. In order to prevent misunderstandings concerning signers and keys, consider the following definitions regarding the nomenclature presented in Table 2.5.

Term	Definition
Original Signer	Private key owner.
Proxy Member/Signer	Member chosen by the original signer to sign on his behalf.
Proxy Key	Cryptographic key used by a proxy member(s).

Table 2.5: Definition of several concepts regarding proxy signature scheme.

Even though the capability of the proxy members to sign on behalf of the original signer, the private and proxy keys must be different, avoiding possible concerns of repudiation and, fundamentally, guarantees the generation of distinct signatures for the same message m, but also the assurance that the private key remains secret to anyone beyond the original signer.

Based on this, the verification function carries an additional feature which enables the identification of the real signatory, since, in ordinary schemes, the verification function just has one way to verify if a given public key matches the signature of a certain document. On the other hand, in the proxy scheme, the way of verifying a signature depends on the key which was used to generate the signature, due to the mathematical properties involved in the signing process. Logically, there is a straightforward way to verify a signature which is based on verifying for both keys (private and proxy). This last approach is presented in figure 2.6 along with an ordinary verification function, leveraging the main differences between both. Although the proxy verification function seems appropriated, the double verification computation is unnecessary if there are two pre-defined values α and β that may be integrated after the calculus of the signature, significantly increasing the efficiency of the function.

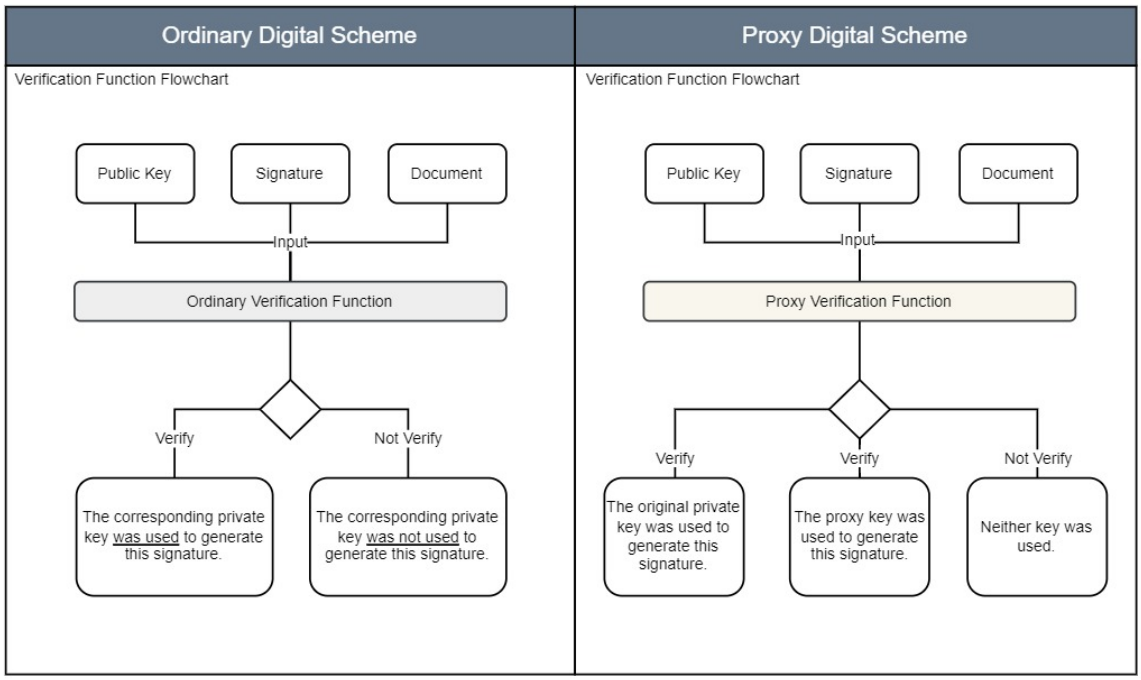

Figure 2.6: Flowcharts of the verification functions.

Based on Figure 2.6, it is noticeable that repudiation may occur within this scheme, since it is impossible to identify which proxy member has signed. In order to clarify this sentence, consider the following case scenario presented in the following enumeration, where there are three proxy members, and one of them digitally signed a document on behalf of the original signer.

1. The original signer chooses three members to constitute the proxy and delivers a unique proxy key to them;

2. The original signer receives a document and informs the proxy members that the document must not be digitally signed by any of them, because it may leverage a few

problems within institutions;

3. A proxy member digitally signs the document.

Note that even though the opportunity to identify if the signature was generated through the proxy key, it is impossible to clarify which proxy member has signed the document, leading to possible concerns between them and, fundamentally, within entities; with this, choosing the members that constitute the proxy is very important, however there must be a way to ensure the identification of the member. Databases can support this issue by storing a few details concerning the signature, *e.g.*, timestamp. Figure 2.7 illustrates the flowchart of a web application which enables digitally signing of documents.

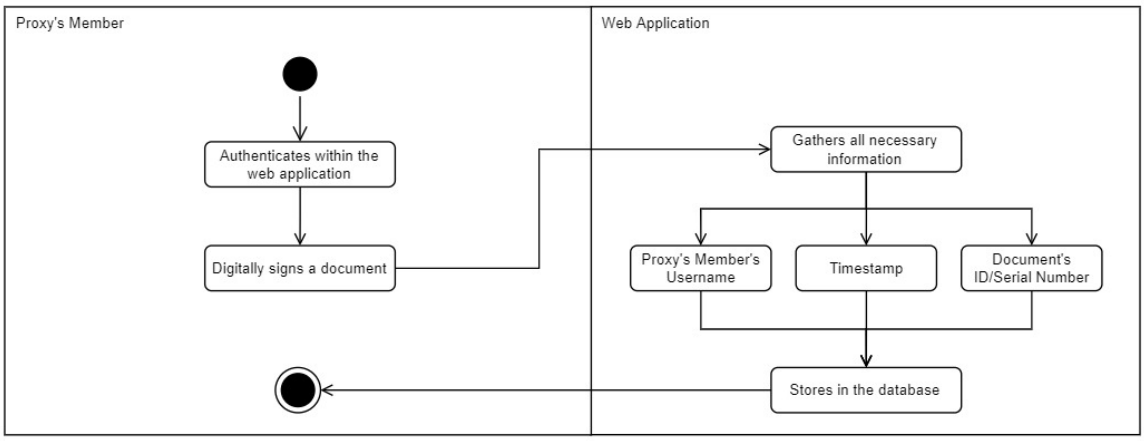

Figure 2.7: Possible activity diagram of the web application.

These three details provide a significant support to users. However, all the software involving the web application must be considered secure, *i.e.,* each piece of the software must be formally verified and, essentially, the timestamp must be afforded through a QTSP, increasing the trust factor associated to the whole system.

Lastly, the way an attacker could exploit this scheme as a third party is practically null given that, considering that this scheme is embedded in the ElGamal scheme, they have to solve the discrete logarithm problem. On the other hand, the attacker may be a proxy member, meaning that they may possess valuable information concerning the private key. Lee, Hwang and Wang [10] stated that a proxy member could cheat in the key delegation protocol and obtain the signature of the original signer on any message; fortunately, Ghodosi and Pierpzyk [10] developed a cheating detection protocol, which is embedded in the main proxy signature scheme, and it involves several tests between the original signer and the proxy member to guarantee that, this last, is not interested in forging the signature of the original signer.

2.4 Lamport Authentication Scheme

When two parties communicate with each other, both of them must ensure that the exchanged data has the correspondent receiver, raising the importance of verifying each side of the correspondent channel, avoiding a possible man in the middle attack, and guaranteeing that both parties are authenticated. There are already several categories for different security levels concerning entity authentication protocols: i) **weak authentication**; ii) **reasonably strong authentication**; iii) **strong authentication**; and iv) **zero knowledge protocols**.

Lamport authentication scheme [11] fits in the second category (**reasonably strong authentication**), where two entities A and B share an initial secret S, and apply cryptographic hash function t times, *i.e.*, $h_0 = H_t(S),\ h_1 = H_{t-1}(S),\ ...,\ h_t = H(S)$. The requester A stores the entire sequence, and the verifier B just stores h_0, and the initial secret S is discarded by both entities.

Authentication is performed through an initial request that is answered as an index i provided by B. This last value consists of the amount of times which the initial secret S is being successively submitted into a hash function $H(.)$, ***i.e.,*** $h_t = H(H(...(H(s)))$, t times. After B calculates that previous value, it will be sent to A, followed by the comparison of the stored value from B with the received one; if both values match, the initial index i is incremented. Figure 2.8 represents this authentication scheme.

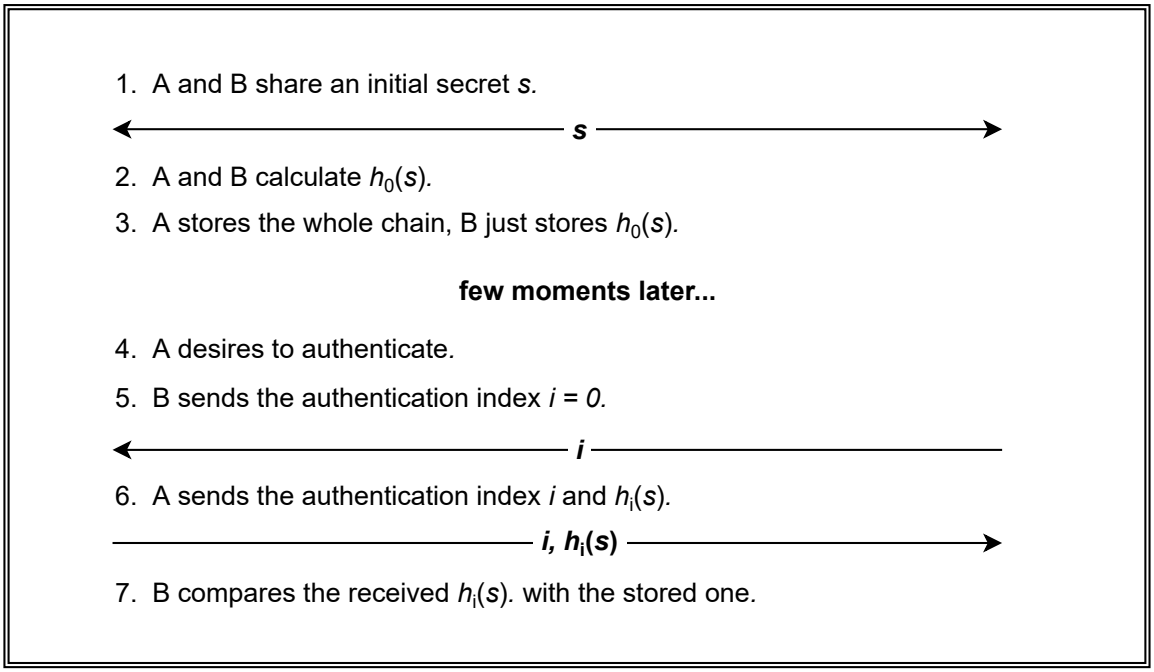

Figure 2.8: Lamport authentication scheme mechanism.

Although it all seems secure and effective, this scheme carries a vulnerability regarding the

hash value which is transmitted through the channel, more specifically, the small n attack, where an attacker may operate as the verifier and asks for an advanced authentication, possessing all the required data to deduce all the others following the scheme.
In the scope of this work, the rationale behind the Lamport authentication scheme was used to implement the mechanism of digital authorizations.

2.5 *ProVerif*

ProVerif [12] is a formal verification tool that analyzes protocols written in the applied pi-calculus using an extension that can describe cryptographic protocols. Internally, this tool transforms protocols into Horn clauses and obtains validity of them, guaranteeing the same for the proposed cryptographic protocol.
This tool handles many different cryptographic primitives, including shared- and public-key cryptography (encryption and signatures), hash functions, and Diffie-Hellman key agreements [13], specified both as rewrite rules or as equations. The verifier produces false attacks, but when it says that the protocol fulfil a property, the property is indeed fulfilled. A wide range of protocols are terminated by the resolution algorithm under consideration (the so called "tagge" protocols). With this tool, it is possible to prove secrecy, authentication, strong secrecy and equivalences between processes that differ only by terms.
ProVerif was used in the scope of this work to prove that the protocols developed inside the prototypes ensure secrecy and authentication, raising the trust factor on the results of this project.

2.6 Conclusions

Electronic signatures do not necessarily imply the usage of public-key based digital signatures. In fact, only advanced and qualified electronic signatures incorporate PKI, meaning that the simple ones have null credibility in most cases. The same often occurs while using advanced electronic signatures, since associated TSPs do not contain the qualified mark, leading to the use of the qualified ones, ensuring that the generated signatures are equivalent to the handwritten ones according to article 26 of the eIDAS Regulation [4].
Furthermore, all existing QTSPs provide signatures using ordinary digital signature schemes, *e.g.,* Rivest Shamir Adleman (RSA) [14], limiting the individuals within institutions to provide signatures, since those require to authenticate themselves within the API of the QTSP, providing a possible manner to postpone an entire process if the signatory absents. With this, the importance of research of other digital signature schemes constitutes an important factor to boost such digital environment, leveraging needs in daily digital routines of institutions, *e.g.,* sign on behalf of someone, to counter-attack the key exposure problem.

Finally, the presented authentication scheme illustrates a clever process to guarantee possible verification processes within client and server, never discarding the mentioned vulnerability. However, the author believes that this specific scheme can be implemented within verification processes securely. In order to prove that, ProVerif is used later on to guarantee that secrets and authentication are not compromised between the involved entities.

Chapter 3

Research of Digital Authorizations and Regimes

3.1 Introduction

This chapter reflects part of the research and surveying of the state-of-the-art activities on the subject of digital authorizations, and aims to highlight their importance in section 3.2, but also of regimes that are adopted by entities while facing their daily routines, identifying work ethics, etc. in section 3.3. Lastly, the conclusions from this chapter are presented in 3.4.

3.2 Digital Authorizations

Entities constantly generate and receive documents that require an analysis, and subsequent deliberation (herein referred to as an authorization) by a group of individuals for further processing. These authorizations can be as simple as clicking a simple checkbox or as more elaborate as digitally signing a document. However, digitally signing a document this way may raise issues, since both concepts (authorization and signing) have different purposes and weights. Digital authorizations and signatures are both associated with the triggering of an action. However, digital signatures are the only mechanism duly recognized and legislated, having a greater weight when compared with digital authorizations within the digital environment. Table 3.1 presents the definition of both concepts.

Concept	Description
Qualified Digital Signature	Mathematical algorithm routinely used to validate the authenticity and integrity of a message. This concept is the digital equivalent of a handwritten signature or stamped seal (vastly more powerful than handwritten signature).
Digital Authorization	Digital content, linked to an individual, that manifests the authorization, or not, of any content. This concept is typically not recognizable, and it is often the equivalent of an oral consent without any sort of record.

Table 3.1: Definition of digital authorizations and signatures.

The unique method that follows the regulation is based on writing authorizations on a document and then digitally signing them, meaning that despite the previous distinct definitions, signatures are always included on the authorizations process to ensure its validation and reliability across the world. Thus, the idea of creating a mechanism representing digital authorizations solely (without digital signatures) will result in a non widely recognizable

product. However, it could be used to streamline digital processes inside entities. Hence, the supposed mechanism must associate reliable digital data and an individual. Logically, several concerns related with policies or data storage will be raised by customers, demanding a well-structured mechanism proposal to clarify them. Table 3.2 discusses main issues and topics that need to be addressed in the context of digital authorizations.

ID	Topic	Description
1	Recognition	Digital authorizations are not recognizable across the world. However, their presence within entities can have impact on daily digital routines, requiring the introduction of its practice in the entities internal regulation.
2	Reliable Data	Despite the fact that digital authorizations are not recognizable, it involves the usage of secure cryptographic functions, resulting in secure produced data. In this manner, data storage has an important and difficult role within this topic. However, this depends on the architecture of the developed mechanism.
3	Usability	It is assumed that digital authorizations are integrated within web or mobile applications, their production can be achieved through a simple click; this last process can be complemented with the introduction of SAD.

Table 3.2: Issues and topics that need to be addressed in the context of digital authorizations.

Creating such a digital mechanism that guarantees reliability and data integrity without using asymmetric cryptography reveals itself as a difficult task. However, the author of this dissertation explored an approach based on hash functions and MAC algorithms [15], ensuring that the produced artefacts are secure, and complying with the aforementioned issues. The author discarded asymmetric cryptography integration in order to provide a mechanism independent of digital signatures, for scenarios requiring to formally glue that certain data belongs to someone, as it can be seen in Subsection 3.2.1, where the idea behind the proposed mechanism is described.

3.2.1 MAC + Hash Approach

Digital authorizations need to be seen as reliable as possible, thus possessing the minimally mandatory security requirements of integrity and authenticity. Searching for cryptographic primitives that correspond to those requirements becomes challenging, since asymmetric cryptography is not allowed. Thus, the MAC mechanism is the chosen primitive to support this approach. Similarly to digital signatures, the authorizations are data structures that need to be attached to the respective document so that any individual can verify it. However, individuals cannot forge authorizations others then their own, requiring support from a specific entity due to the confidentiality surrounding the key (used in MAC). Furthermore, such mechanism is supposed to be implemented within an application that provides the ability to digitally authorize documents. As such, to produce digital authorizations, the data which comprise the input of MAC demands being defined. Beyond the key, the concerning document (*doc*) cannot constitute the message m by itself, requiring the presence

of randomness (*nonce*), the *timestamp* and also the text regarding the authorization (*text*). Equation 3.1 presents this initial MAC calculation:

$$\text{MAC}((doc,\ text,\ nonce,\ timestamp),\ k) = t. \quad (3.1)$$

In the proposed scheme, the resulting token t is not the digital authorization. Instead, it is herein proposed that the token is recursively submitted into i cryptographic hash functions H, where $i \in \{1024, 2048\}$, as illustrated by expression 3.3, whose final result is the digital authorization:

$$dauth = H_i(t) = \quad (3.2)$$

$$= H(H(H(...H(\text{MAC}((doc,\ text,\ nonce,\ timestamp),\ k))))). \quad (3.3)$$

Now that the digital authorization calculation algorithm has been presented, the remaining part of this section can be devoted to discussing the verification procedure, as well as minor other practical details of the proposed scheme.

Up to this point, it might not be clear why the recursive application of a cryptographic hash function is needed in the calculation of the digital authorization. The rationale behind that specific part of the approach is to provide means for the client (e.g., a browser) to verify the authorization without exposing the key. The obvious approach would be for the server to send the (symmetric) key to the client for the recalculation and comparison of the MAC, but this would indeed expose the keys to any user wishing to perform a verification. This is why the digital authorization is herein an hash value calculated recursively. This strategy allows the verification of the authorization to be performed using a challenge: (i) the client signals the intention to verify; (ii) the server performs part of the calculations, and sends an intermediate hash value to the client; (iii) the client performs the remaining calculations until the hash values coincide. This will be explained afterwards.

Furthermore, although this approach is not broadly recognizable, such digital authorizations can be integrated into documents which are can be additionally digitally signed and sealed. This idea can be applied when users are handling critical documents. Figure 3.1 presents the proposed idea to handle critical documents.

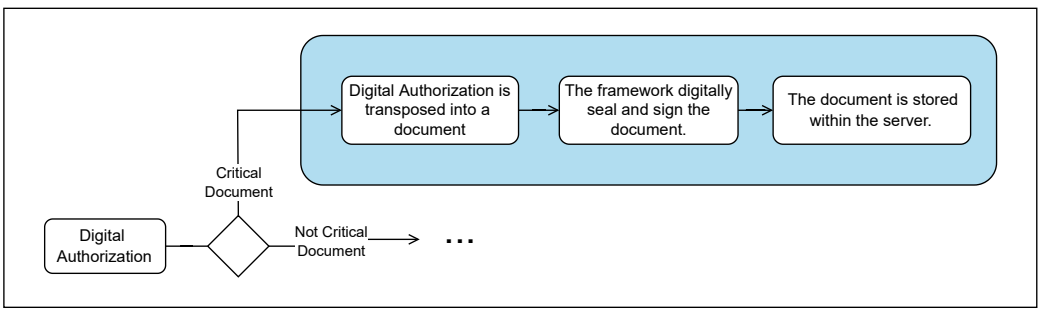

Figure 3.1: Procedure to handle critical documents (adding digital signature to a digital authorization).

Notice that no specific MAC and hash algorithms were suggested along the previous explanation. Additionally, no details were provided regarding the generation of keys and data storage, even though both are essential to ensure the security of the approach. To provide closure to this subject, Table 3.3 presents a list of recommended cryptographic algorithms that developers may take into account when implementing the described digital authorization, along with suggested key sizes and sources for randomness.

MAC	**MAC Key Size**	**Hash**	**Nonce**
HMAC [16]	512 bits	SHA256	Windows/Linux random source
HMAC	512 bits	SHA512	Windows/Linux random source
AES-CMAC [17]	128, 192 or 256 bits	SHA256	Windows/Linux random source
KMAC [18]	128 or 256 bits	SHA256	Windows/Linux random source

Table 3.3: Possible cryptographic primitives combinations for the proposed .

Consider that the MAC keys are often generated through a Key Derivation Function (KDF) [19], where the user password is part of its input, as well as *OtherInputs* typical of these algorithms. Specifically, the second input consists of additional information that may be required by the KDF, *e.g., salts* or *Initialization Vectors (IVs)*. Furthermore, there are MAC algorithms that apply randomness by construction. Independently of that, the usage of values sourcing from the resources provided the the operating system (e.g., Windows and Linux provide such resources) are also a secure option. To conclude this section, Table 3.4 presents the pros and cons of the proposed approach.

ID	**Description**
P1	Streamline digital routines using secure cryptography algorithms, never compromising the user in potential conflicts outside the entity.
P2	The integration of the mechanism into frameworks becomes fairly straightforward, leveraging the importance of critical concepts such as: data storage and cryptographic keys management.
P3	Possibility of associating keys to users in a recognizable manner through the use of electronic signatures (qualified).
C1	Not recognizable across the world.
C2	This mechanism requires several tests over its security, more specifically, after being implemented within a framework, since there are several stages embedded within the approach that must be designed based on the case scenario.

Table 3.4: Pros and cons of using the proposed MAC + Hash approach.

3.3 Regimes

Digital signatures can be triggered through different methods, always depending on the entity purpose, more specifically, the adopted regime. It is possible to identify numerous regimes within the business landscape. However, this section focuses on two different approaches: hierarchical (subsection 3.3.1) and equality regime (subsection 3.3.2). Furthermore, subsection 3.3.3 describes all the details regarding the chosen case scenario that will

be the basis for developing the prototypes.

3.3.1 Hierarchical Regime

This regime stands for the most commonly used across entities, where most documents require a review from the owner in order to seal possible agreements or contracts. In a large entity, comprised of several departments and where dozens of documents are received daily, each department has specific assigned tasks (*e.g.*, risk analysis). Nevertheless, the owner involvement is necessary, since digital signature (commitment) is often required. With this, the introduction of a digital signature scheme with forward-security may streamline these digital concerns. On the other hand, a proxy scheme will modify the idea of the entity, since it requires a **complete confidence towards the members of the proxy**. Furthermore, as mentioned at the beginning of this section, although multiple tasks can be triggered through digital authorizations, this methodology is not covered in any regulation, leading entities to develop or adapt an interface to manage these authorizations following the internal entity regulation. The next enumeration presents an example path that a document performs when it arrives in an entity inbox:

1. A document D arrives at the entity inbox and its analysis is assigned to the marketing department;
2. Each member of the marketing department analyses D and agrees/disagree with it through digital authorization (and this may be performed in a hierarchical manner also);
3. If the majority of members authorize it, the person in charge of the entity receives it to take a final decision.

This new digital environment must be implemented in such a way that increases the autonomy and flexibility of each employee, and **never providing capacities beyond those pre-determined in regulation or by the main responsible**. On the other hand, the digital seal environment allows greater flexibility, since the responsible must select a number of people who can perform that action, providing authenticity to the documents which the entity provides to other institutions.

3.3.2 Equality Regime

Contrary to the hierarchical regime, the equality regime is based on the ability of several members digitally signing or authorizing without sense of hierarchy. In some cases, it may include members signing or authorize on behalf of others. Thus, **both digital signature schemes can be easily integrated** within entities that follow this regime, since several members rely on others to sign on behalf of them. Despite this, before proceeding with the

implementation of the proxy scheme, the responsible should formulate several questions concerning the pros and cons of such implementation. Table 3.5 presents several of those questions (the listed requirements are only the most general ones, and thus the list is not exhaustive).

ID	Question
1	Do I trust someone to sign on my behalf?
2	Is it worth exposing the entity to any risk?
3	Is it possible to identify who signed on my behalf?

Table 3.5: Several questions that user may leverage on facing proxy scheme features.

The questions presented in Table 3.5 can be easily answered, supporting decisions related to the way the scheme is implemented, given that such frameworks require to be trimmed in order to follow the entity regulation, *i.e.*, **flexible framework**. Given all this, building a prototype that covers this regime becomes the most challenging amongst both, since it has to be as general as possible, with the possibility to customize it as far as the entity needs.

3.3.3 GDUBI

Universidade da Beira Interior (UBI) has developed its own web application for managing documents and authorizations, which provides several employees with the possibility to attach authorizations to documents associated to the university landscape, streamlining agreements, contracts and decisions within the institution. The workflow of GDUBI is based on three activities, which are presented in Table 3.6.

ID	Description
1	A predefined user must associate documents to other users.
2	Users authorize or not the assigned document.
3	Users give/write their opinion concerning the document along with the authorization.

Table 3.6: Main features of GDUBI.

GDUBI works perfectly on daily routines of the employees. However, the presented activities lack formal assurances, specifically, the second one, since the authorization provided by users corresponds solely to a database fields adjustment and update. In order to illustrate the workflow of GDUBI, as well as how the defined activities are connected, consider Figure 3.2.

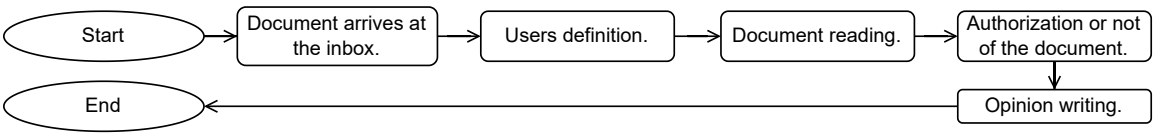

Figure 3.2: Flowchart of GDUBI.

After all the involved users have participated, the document and authorizations are archived. The user definition step follows the hierarchy of the institution (*e.g.,* the rector is many times the last user of the flow in terms of authorizations). Furthermore, as mentioned before, the ability to authorize a given request is provided to the users. However, this application cannot provide any other significant functionality. The introduction of MAC + Hash approach while producing digital authorizations, as well as digital signature and seal would clearly streamline digital routines of the university, since those three concepts could be performed in the same application.

Embedding such digital mechanisms require deep knowledge over the GDUBI platform, given that the development of prototypes needs to be seen as an improvement regarding digital security, leading to a consequent streamlining of the whole university concerns. Therefore, the presence of such mechanism needs to be as unnoticeable as possible to the user, increasing the reliability of GDUBI without hindering operations. Figure 3.3 presents the activity diagram concerning the authorization process provided by a user within GDUBI, highlighting several details that need to be considered when developing the prototypes.

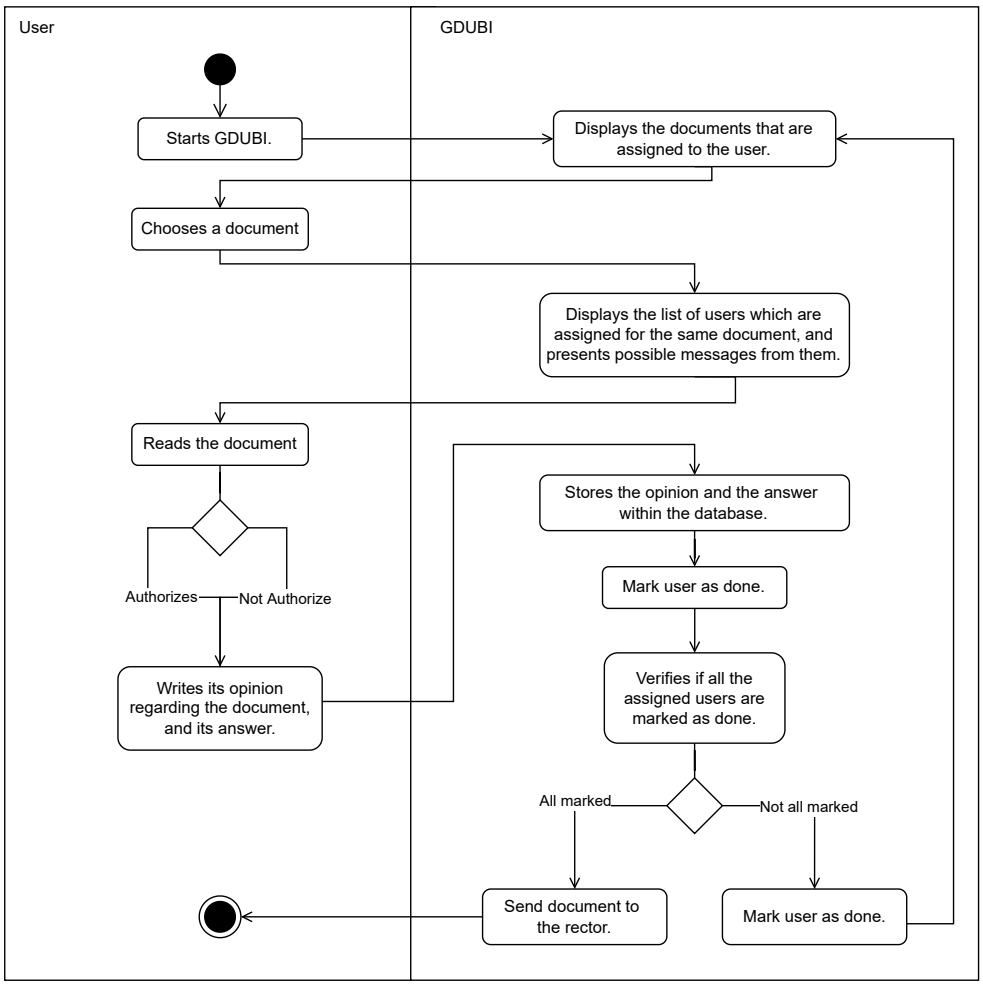

Figure 3.3: Activity diagram of GDUBI when a user digitally authorizes a document.

3.4 Conclusions

This chapter describes two regimes that can be applied to entities regarding document management and associated authorizations. Hierarchical is the most used regime across the world, being all the decisions authorized, or not, by predefined employees (organized according to their functions within the entity), and the final decision taken by the main responsible. With this, providing digital authorizations requires developing from scratch a secure cryptographic mechanism as mentioned in Subsection 3.2.1, where hash and MAC functions were integrated in a manner that ensures the reliable production of digital content entitled to digital authorization.
GDUBI, which is the document management system used in UBI, was fairly well-described in this chapter, and it consists of the specific case scenario which the prototypes are based on, demanding an association with the presented regimes in order to proceed to the development stage. This web application is in continuous operation in UBI, and its access is restricted to predefined individuals, leading to low traffic of daily active users. Furthermore, it comprises an interesting application to research and implement prototypes in the context of this dissertation, since it is a potential use case for all the three studied mechanisms: i) digital authorizations; ii) digital signatures; and iii) digital seals. Beyond this, as mentioned before, GDUBI access is restricted and, consequently, the author did not have direct access to it, being the presented flowcharts and activity diagrams grounded on questions performed to its users.

Chapter 4

Prototypes

4.1 Introduction

This chapter describes the developed prototypes in this dissertation, taking the GDUBI platform as the case scenario. The description and analysis of the digital authorization prototype are respectively presented in Sections 4.2 and 4.3, followed by the description of the digital signatures prototype in Section 4.4. Finally, the main conclusions of the chapter are included in Section 4.5.

4.2 Digital Authorization Prototype

This prototype of a cryptographic scheme for digital authorizations materializes one of the main innovation of this project, since producing and verifying digital authorizations without the use of asymmetric cryptography is comes with several limitations related with the security involved on such prototype. The idea of this prototype consists of implementing the previously mentioned MAC + Hash approach, hypothetically in tune with the existent GDUBI platform (the prototype was not integrated with GDUBI in the timeframe of the project). The scheme presumes the existence of several algorithms that tackle the different operations surrounding the concept at hands, which in this case are the following: *generation of keys* (Subsection 4.2.1), *production of digital authorizations* (Subsection 4.2.2), *verification of digital authorizations* (Subsection 4.2.3) and *renewal of digital authorizations* (Subsection 4.2.4).

4.2.1 Generation of Keys

Generation of keying material or cryptographic secrets is typically performed at an early stage of interaction, typically in a phase called *Enrollment*. In this work, it was assumed that users may already be registered in the application, and it was assumed that their passwords were securely stored through the use of a slow KDF. However, the conceptualized prototype requires the additional initial generation of three different keys for each user. This process is guaranteed through the digesting of the user password by a chosen secure KDF along with randomizing material. Equation 4.1 specifically presents the algorithm used to generate those three keys (where `pwd`, `cost` and `salt` denote the password, the number of iterations

of the slow KDF and a 64 or 128 bits random value, respectively, and where k is the resulting key outputted by the KDF):

$$\text{KDF}(pwd, cost, salt) = k. \quad (4.1)$$

The three aforementioned keys are separately used for *authentication*, *production* and *verification of digital authorizations* purposes. Specifically, k_1 can be understood as a secure representation of the password of the user, used for authentication purposes. k_2 is a key used to construct and verify digital authorizations of the server, while k_3 is the key used to contruct and verify digital authorazations directly by the user. k_2 and k_3 are further used as inputs in the *MAC + Hash* approach. While k_1 and k_2 should be stored in the (secure) server, k_3 should not be stored anywhere by design, and instead be generated each time the key is necessary. This will add an additional assurance that the user needs to be online during the verification of a digital authorization.

4.2.2 Production of Digital Authorizations

This second proposed algorithm concerns the calculation of two distinct digital authorizations (herein referred to as $h_{r_1}(t_2)$ and $h_{r_2}(t_3)$) for the same message M using two different keys (k_2 and k_3). Both digital authorizations are produced according to the MAC + Hash approach. After digital authorizations have been successfully produced, the surrounding data which was used to produce them needs to be securely stored. However, only the resultant digital authorizations and the corresponding r_1 and r_2 are stored within the database. Moreover, the *salt* used for the $h_{r_2}(t_3)$ generation is also stored within the database, so as to allow its recalculation through the insertion of the user password. Furthermore, this process is triggered through the correct insertion of the author password (matched with k_1). Figure 4.1 presents the described digital authorization generation process based on k_3.

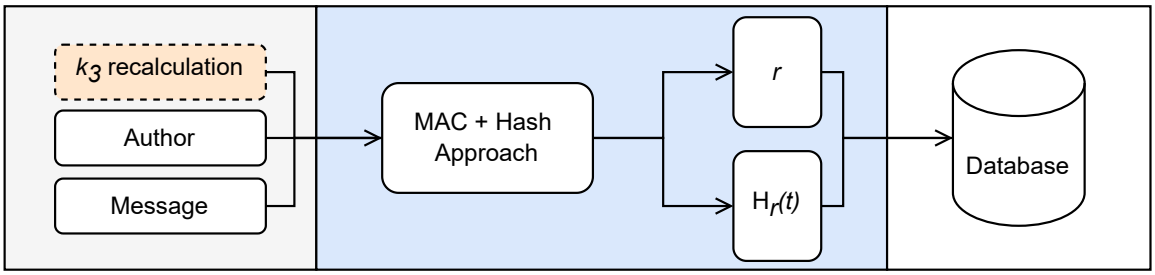

Figure 4.1: Construction of digital authorizations prototype scheme.

Digital authorizations are associated with the users and should be generated following a predefined order defined by a specific user (typically, the user that submits the document

or an administrator of the platform). Ensuring the correct order of the flow can be achieved using a blockchain [20], which could be added to the system for this specific purpose, guaranteeing also data integrity. In this case, each block added to the blockchain would have to possess information that reveals the concerned document authorization details (metadata). This optional blockchain integration is illustrated in Figure 4.2, but it was not further developed in the scope of this dissertation.

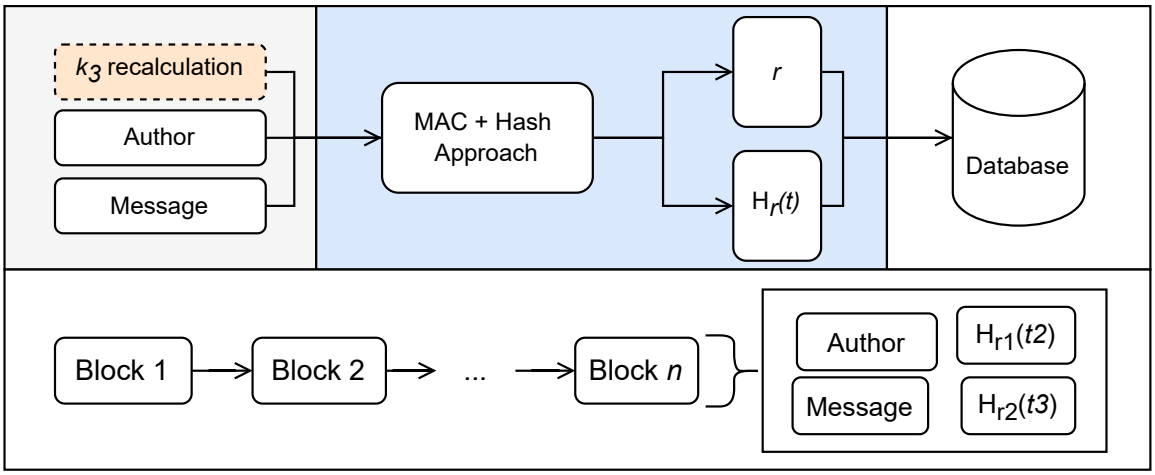

Figure 4.2: Production of digital authorizations prototype scheme with blockchain integration.

4.2.3 Verification of Digital Authorizations

The third proposed algorithm is that of verification of digital authorizations, which is inspired on the usage of Lamport authentication scheme. As mentioned in Subsection 4.2.2, there are two ways of verifying digital authorizations based on previously defined keys k_2 and k_3: ***user offline*** and ***user online***.

***User offline* verification** is inspired on the Lamport authentication scheme and it works as follows. When client desires to verify a digital authorization $h_{r_1}(t_2)$, it sends (to the server) the concerned *message*, its *author* and $r_{1_{min}}$, where $r_{1_{min}}$ is a random positive integer number such that $r_{1_{min}} < r_1$. When the server gets the aforementioned values r_1 and k_2, it replicated the *MAC + Hash* approach using $r_{1_{min}}$, obtaining $h_{r_{1_{min}}} = H_{r_{1_{min}}}(t_2)$. The server then sends $h_{r_{1_{min}}}(t_2)$ and $r_1 - r_{1_{min}}$ back to the client, where this last value corresponds to the necessary hash computations that need to be applied to $h_{r_{1_{min}}}(t_2)$ before comparison with the initial digital authorization $H_{r_1}(t_2)$. The remaining hash computations and the final comparison of digital authorizations are performed on the *client side* (e.g., using technologies such as JavaScript [21]). Figure 4.3 presents the diagram of the presented verification algorithm through *offline user* approach.

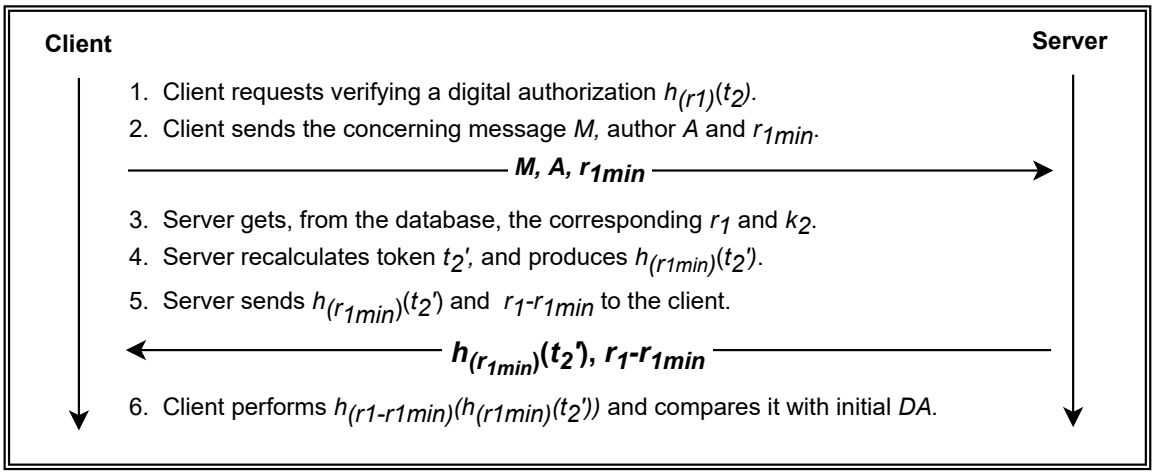

Figure 4.3: Verification algorithm (user offline branch).

The ***User online*** **verification** option is more reliable than the *offline user* approach, since the necessary key to recalculate the digital authorization needs to be generated from the user password, which should not be stored anywhere and thus requiring the online presence of the supposed author. Moreover, before generating k_3, the server verifies if the password provided by the author degenerates into the one stored for authentication purposes (k_1) and, from then on, the same previously presented procedure for verifying the digital authorization (the *user offline* approach) is applied, but using r_2 and k_3. Table 4.1 presents the main features of both verification approaches. The k_3 recalculation can be seen in Figure 4.1.

Approach	Description
User Offline	The proposed verification algorithm is performed using r_1 and k_2. Both values are stored in the database, and the concerned author does not need to do any action.
User Online	Performs the proposed verification algorithm using r_2 and k_3. None of the values are stored in the database, requiring generation of k_3. Thus, the author must introduce his password, following the k_3 recalculation scheme. This approach presents greater reliability.

Table 4.1: Description of both verification branches (online and offline).

Considering the variety of documents that GDUBI manages, the platform should automatically restrict several documents to be verified through the *user offline* approach, *i.e.,* digital authorizations of critical documents must be verified through the *user online* approach (Figure 4.4).

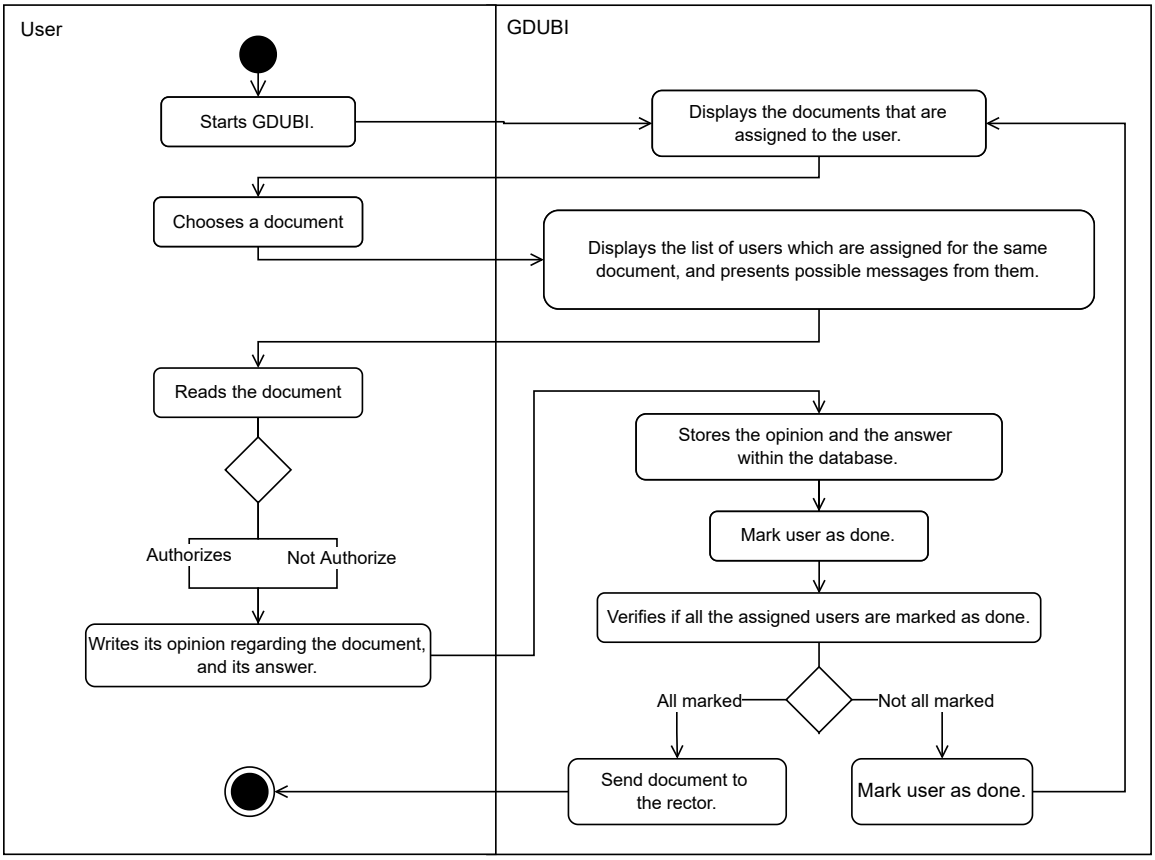

Figure 4.4: Activity diagram of the verification algorithm according to GDUBI.

4.2.4 Renewable Digital Authorization

The fourth proposed algorithm addresses the renewal of digital authorizations. The verification algorithm is inspired on the Lamport authentication scheme, raising the opportunity of exploiting the well known vulnerability that affects this scheme: the small *n* attack. This attack consists of acquiring advanced hash values in such way that the forging of the entire verification chain is possible, allowing an attacker to impersonate as the server and sending hash values that lead to valid (forged) authorizations. Therefore, the author searched for a simple solution to solve this issue.

Specifically, the small *n* attack can be exploited if the concerned digital authorization *remains the same* after someone tries to verify it. Thus, *renewing digital authorizations* presents a solution that would deny anyone from exploiting such an attack. This *renewal process* consists of recalculating both digital authorizations again, requiring the modification of the *nonce* value (added to the message M). Cryptographic keys k_2 and k_3 can still be used on production and verification of digital authorizations. Figure 4.5 contains the flowchart of the actions that GDUBI must integrate after a verification has been concluded.

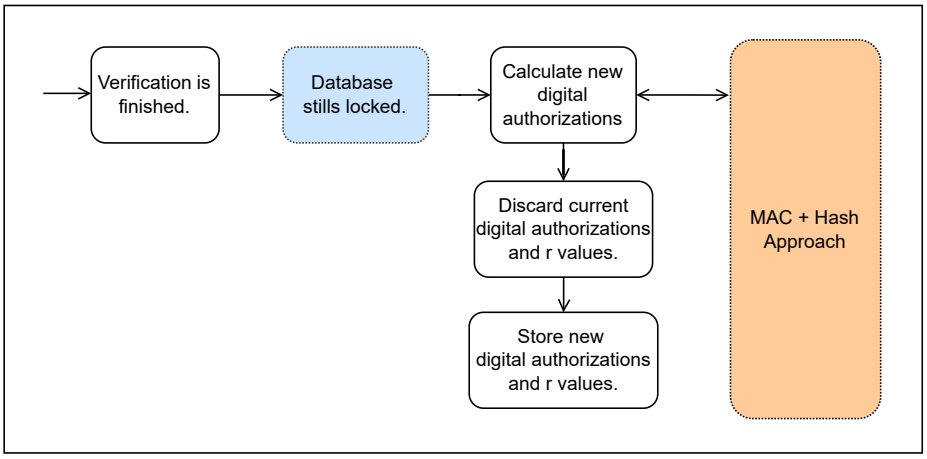

Figure 4.5: Renewal of digital authorizations process.

The proposed solution has computational (processing and memory) costs. Fortunately, this approach can be implemented within GDUBI, since document management is mostly for administrative staff within organizations. Every time a user tries to verify a digital authorization, a report must be produced, avoiding a flood of verification requests. Furthermore, database transactions must be carefully implemented, since authorizations cannot be changed while someone is trying to verify them. This prototype does not define the specific locks that should be used within the renewing process. However, the author advises the developers for the way that locks are implemented (type and location).

4.3 Digital Authorization Analysis

The final portions of the prototype have undergone several modifications due to security concerns. Data storage and data transmission are the two aspects that require more attention, assuming that the remaining primitives and algorithms are implemented correctly. Furthermore, despite the description of the four algorithms, part of the reasoning for the way they were developed was not previously presented. Therefore, this section aims to discuss and deepen the mentioned algorithms and critical concepts by analyzing example attacking case scenarios to the system, *i.e.,* keys storage and generation (Subsection 4.3.1), and the lifecycle of digital authorizations (Subsection 4.3.2). Note that the presented scenarios were those which the author identified as being the most critical, given the design and functioning of the proposed digital authorization prototype.

4.3.1 Sensitive Data Generation and Storage

Generation of keys is the algorithm on which the whole process relies primarily, and it is where three different keys are generated. Such sensitive data must be carefully handled, raising the importance of introducing encryption before its storage within or outside the

system. As mentioned in previous sections, keys are generated through secure slow KDFs using the registration password as an input, preventing personal user data from being compromised. However, such secure data generation process becomes insignificant if its storage is not equally secure. There are various ways of storing data without compromising the entire platform workflow or user data, *e.g.*, internal storage (GDUBI) or external (cloud). Ensuring that such process is secure requires to exploit features which may compromise the workflow of GDUBI, fixing possible vulnerabilities and prepare the system in case this happens. Consider Table 4.2, where an attack case scenario is presented, describing the main features regarding keys storage and generation (S_n), and the main vulnerability (V) based on database access.

ID	Description
S1	Document and keys, which are generated through a secure KDF, are stored within the database.
S3	Root user is the single registered database user within the Database Management System (DBMS).
S4	The database is inside the system (GDUBI).
S5	The stored data is not encrypted.
V	The attacker managed to access the database.

Table 4.2: Description of the attack case scenario properties over the digital authorization prototype (keys storage and generation).

If an attacker manages to access the database through its single user (root), several issues can be exploited and, consequently, the veracity and integrity of all produced documents and digital authorizations are compromised. Moreover, a skilful attacker could even camouflage their entry into the system and discover patterns within user data. Table 4.3 presents possible issues that may occur given the presented scenario.

ID	Issue	Description
1	Compromised authorizations	Authorizations are required to be verified through keys k_2 or k_3, meaning that their modification leads to the failure of carrying out verification. With this, the client will assume that the concerned author is not the one which produced the authorization, generating an uncertainty regarding the veracity of the application.
2	Compromised user data	The used KDF must possess a significant security factor, so that it cannot be possible to find the original password through the produced value (Rainbow Table attack), exposing sensitive user data beyond this application.
3	Compromised documents	Documents are the main focus of this application, and since their storage is done in the database, the attacker may read their content, or even worse, orchestrate the authorization of a given document by any given user.
4	Data loss	Given that the attacker has the ability to modify and delete data from the database, data may potentially be erased and lost.
5	End of the application	Given the previous four issues, allowing an attacker to directly manage the database will certainly lead to legal issues for the institution.

Table 4.3: Potential issues raised by the proposed vulnerability (database access).

The system must adopt several measures to avoid the several enumerated problems, since the first four listed issues entirely compromise the workflow of GDUBI and, consequently, may lead to the complete distrust of the application. Table 4.4 presents several technologies, controls or measures that must be taken into account while implementing the prototype.

ID	Issue	Approach	Description
1	1,2,3	Data Encryption	Symmetrically encrypting all the stored data within the database may partially solve the problems regarding compromise of information. However, this encryption requires the storage of the used key (master), leading to another internal problem. Normally, this approach is supported by the cloud, since that master key may be stored there.
2	1,2,3	Deploy database in the cloud	Deploying the database in the cloud may streamline the entire data storage process, as there are specialized packages which provide data confidentiality embedded in the cloud. The system manager does not need to worry about the way the data is stored. However, this becomes contradictory, since there is no control over the data.
3	2	Secure KDF	The use of a secure KDF while generating or renewing keys is essential to ensure the veracity and credibility of the system. Moreover, if the produced data is not reliable, there is no sense in storing it securely.

Table 4.4: Measures to prevent the proposed vulnerability (database access).

The three presented measures or technologies presented in Table 4.4 can be implemented in a system at the same time. However, it should be considered that data encryption can be guaranteed through cloud infrastructure. With this, the portion which involves these two concepts (keys generation and storage) is based on the presence of a cloud broker, embedded in a secure enclave, within the system. Figure 4.6 represents the potential architecture of the system integrating all of the proposed suggestions.

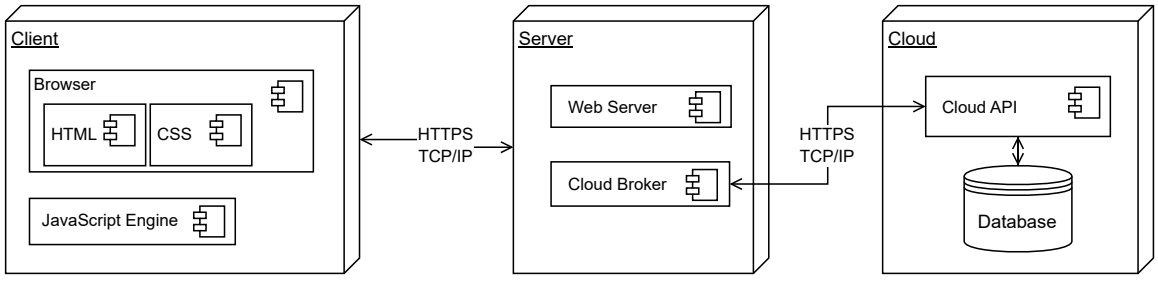

Figure 4.6: Architecture of GDUBI after implementing the digital authorization prototype.

Despite all the previously described suggestions to avoid attacks, the possibility that new vulnerabilities are discovered or injected into updates of the system and are effectively exploited by attackers should be always considered. Therefore, a mitigation plan must be devised in order to support developers if such attack scenarios arise. A draft of a general mitigation and recovery plan is presented in Table 4.5.

Task	Description
A	Shut down the entire system.
B	Understanding how the database was accessed, and which content was affected.
C	Warn users that the system has been compromised, and it will be unavailable for a while.
D	Verify the system firewall status.
E	Detailed analysis of the stored digital authorizations and keys. Recreating every single digital authorization based on the corresponding stored key and salt value. Blockchain is fundamental to accomplish such task.
F	Discredit all the documents which were produced after the date of the attack.

Table 4.5: General mitigation and recovery plan.

4.3.2 Authorizations Lifecycle

The proposed prototype suggests performing two authorizations that are stored within the blockchain and the database. On one hand, blockchain storage consists of creating a new block and introduce it within the chain through storage. On the other hand, database storage involves *renewing* specific values or states without adding any others, revealing itself a fundamental stage that attackers may exploit. Specifically, the authorizations become vulnerable when the *verification algorithm* is being performed, since the attacker has access to valuable information that may lead to the orchestration of the *small n attack*. That same access can also be guaranteed through *root access*. However, such possibility is linked to numerous, yet more remote, techniques that the attacker may use, *i.e.,* SQL injection, phishing, spoofing, etc.
Fortunately, *renewing* valuable data after verifying an authorization solves part of the aforementioned attack possibilities. Thus, in order to demonstrate the crucial importance of the renewal of data within the authorizations lifecycle, consider Table 4.6 where system features (S_n) related to the authorizations lifecycle, and also the vulnerability that the attacker can exploit (V), are presented. Notice that, to show the importance of the *renewing algorithm*, it is forcefully assumed that the system is not implementing that procedure.

ID	Description
S1	Authorizations are securely stored on blockchain.
S2	Each hash authorization value is securely stored within the database (e.g., cloud).
S3	The broker is embedded in a secure enclave.
S4	Authorizations are not renewed when someone tries to perform a verification.
V	Given that authorizations are not renewed, the attacker exploits the Lamport vulnerability.

Table 4.6: Description of the attack case scenario properties over digital authorization prototype (authorizations lifecycle).

Considering that the *verification algorithm* is partially based on the Lamport authentication scheme, and the *renewal algorithm* is not implemented, the *small n attack* may occur, which provides the attacker with the possibility of manipulating any user *offline digital authorization verification* procedure. Therefore, it is possible to identify several issues that

may arise after this attack has been successfully performed. Table 4.7 presents them.

ID	Issue	Description
1	Compromised authorizations	The verification process is compromised, since the small *n* attack can be potentially explored. Although the attacker cannot manage digital authorizations, they can easily raise several issues. Moreover, an attacker, through this attack, can easily manipulate a specific digital authorization to be associated to someone that has not produced it.
2	Verification offline branch worthless	The presented user offline verification branch has no impact considering the previous row. Despite the development of two different ways to verify digital authorizations, only the online one ensures, at least, that the inserted password is valid or not, emphasizing the importance of SAD and, consequently, providing a source of trust for this branch.
3	Compromised verifications	Given the above, the content of all documents associated with this application will be regarded as dubious or untruthful, since possible previous documents may have been verified through the offline verification branch.

Table 4.7: Potential issues raised by the proposed vulnerability (small *n* attack).

GDUBI must implement the *renewal algorithm* each time someone tries to verify a digital authorization, effectively solving the discussed issues. The presented *renewal algorithm* requires being analyzed by GDUBI administrators and developers, since this algorithm ensures the platform reliability and integration specific details depend on how the database and information flows are implemented. Table 4.8 presents all procedures that may contribute to the trustworthiness of the authorizations lifecycle.

ID	Issue	Approach	Description
1	2	Renew digital authorizations	The renewal of digital authorizations makes it unfeasible for an attacker to explore the small *n* attack, since the hash value is changing every time someone tries to verify any digital authorization. This approach requires some computational resources, creating the need for developing an optimal solution which defines a specific amount of verifications per authorization before renewing the current one.
2	1, 3	Renew cryptographic keys	Throughout the sections, it has never been specified how the keys should be updated. However, renewing cryptographic keys is a cornerstone for anything involving cryptography. Obviously, previous keys must be securely stored in the cloud (database) after renewal, since they are necessary for verification purposes, unless authorizations are recalculated with the new keys.
3	1, 3	Blockchain as a tracer	Blockchain guarantees data integrity, but can also work as a guide to understand who produced a specific digital authorization, avoiding possible repudiation from users. This approach supports GDUBI in case of it being attacked.

Table 4.8: Measures to prevent the proposed vulnerability (small *n* attack).

4.4 Digital Signature Prototype

The prototype regarding digital signatures covers a recognizable digital mechanism, not currently provided by GDUBI, since the previous one is grounded on the author proposal using secure cryptographic functions. Signatures are commonly used on GDUBI, by the rector, when a certain document is authorized by every involved member. However, the application lacks the ability to digitally sign, requiring the physical printing of the document and subsequent handwritten signature, or using other frameworks, such as Chave Móvel Digital, introducing qualified digital signatures to the flow. The proposed prototype aims to provide GDUBI with a secure way to introduce digital signatures within its feature landscape. In order to produce digital signatures, the user registration process requires the generation of an asymmetric cryptographic key-pair, where the rector is the unique user who is able to use them. The other two algorithms, concerning the environment of digital signatures, are automatically involved: production and verification of digital signatures. These three algorithms cannot be performed through a qualified device, since the institution avoids incurring costs, strengthening the idea that each digital signature produced through GDUBI cannot be legally recognized. Subsection 4.4.1 describes a crucial concept regarding the concerned prototype, more specifically, how the system must securely store the cryptographic asymmetric keys.

4.4.1 Keys Storage

The production and verification of digital signatures are straight mapped processes. Considering the chosen algorithm for generating the keys, it is impossible to bypass the predefined paths. In this manner, similarly to the previous prototype, keys storage deserves a special attention while developing the whole prototype.

Despite the generation of such keys not being recognizable across the EU, GDUBI must guarantee that all data associated with it is secure. Therefore, keys must be securely stored, since documents cannot be digitally signed on behalf of other, pointing to the final idea that nobody can sign on behalf of the rector. Cloud was the chosen infrastructure to ensure the storage of keys, requiring the presence of a cloud broker within the system in a secure enclave, hindering the capabilities of a possible attacker. Table 4.9 presents several approaches/concepts that should be discussed to provide a secure keys storage.

ID	Name	Description
1	Keys Generation	Considering the chosen public-key encryption algorithm, the keys should be secure, *i.e.*, in the case of choosing RSA, the key sizes must be at least 2048 or 4096 bits.
2	Keys Lifetime and Rotation	Keys must be recalculated after a given period of time (cryptoperiod). However, this may take place if other scenarios emerge, *e.g.*, private key has been exposed, encryption of certain amount of data, security updates within the concerned algorithm, etc.

3	Encrypting stored keys	Several companies follow this approach due to the clarity linked to the process. When trying to store cryptographic asymmetric keys, the use of symmetric encryption could be a solid solution, therefore, the generated public key encrypts the concerned user data, and the associated symmetric key encrypts the private key, formulating the main concepts of this approach: Key Encryption Key and Data Encryption Key.
4	Cloud Services	As mentioned before, this approach stands for the most common on daily routines. Beyond the ability of storing sensitive data within the deployed database, there are services, provided by the cloud, which focus on key management, *e.g.*, Amazon Web Services (AWS) Key Management Service (KMS), streamlining the generation and management of cryptographic keys.

Table 4.9: Measures to ensure the integrity of keys storage.

Integration of cloud services eases the whole storage process. On Table 4.9, the last row presents the most secure option given the current situation of GDUBI, covering the first three rows without any concerns from the developers. Beyond this, such cloud services offer the possibility to split application data, *i.e.,* document, etc., from the cryptographic one, hindering the work of the attacker if the application is compromised.

4.5 Conclusions

The proposed prototypes (digital authorizations and signatures/seal) were described in the previous sections, being the authorizations one analyzed regarding possible attack case scenarios which could happen after being implemented in GDUBI. Moreover, it was said that the verification and renewing algorithms are the most (security) critical procedures within the prototype, with the latter being key to ensure that attackers cannot exploit vulnerabilities such as the small n attack. The presented attack case scenarios were built in a manner that leverages the importance of the correct implementation of certain aspects/mechanisms in GDUBI, more specifically, the storage and generation of keys, and the authorizations lifecycle. Therefore, previous decisions raising several questions were addressed and clarified in the section devoted to the analysis of the proposed digital authorization. Although the structure, the discussion and the analysis of the proposed prototypes all suggest that a system integrating them will be secure (for the specific intended purpose of digital authorizations), that assurance requires a more formal analysis, which is the subject of the next chapter.

Chapter 5

Formal Verification of the Digital Authorization Prototype

5.1 Introduction

This chapter describes the work performed to prove that the developed digital authorization prototype provides strong secrecy and secure authentication, using a formal approach and resorting, in this case, to *ProVerif*. To accomplish such proof, the importance of the handshake protocol (which is crucial for establishing trust in the secrecy of exchanged information) is discussed in Section 5.2, followed by the description of the digital authorization prototype integration in Section 5.3. Finally, Section 5.4 describes the tests performed on the final *ProVerif* setup, and Section 5.5 draws the main conclusions of this chapter.

5.2 Handshake Protocol

The digital authorization prototype proposed herein is to be embedded in GDUBI, and its algorithms require being adapted to the (already existing) application functioning. Specifically, the data flows between the client and server through the handshake protocol, guaranteeing the secrecy of the data and the authentication of both involved parties, should be implemented in the system. Taking that as granted, proving the security of the entire digital authorization apparatus requires defining first the handshake protocol in the *ProVerif* language. Fortunately, such protocol implementation is already provided in [**?**], easing the author work. Nevertheless, a detailed description of it is presented in Subsections 5.2.1, 5.2.2 and 5.2.3.

5.2.1 Types

ProVerif works as a strongly typed language where any variable has a specific type, such as `bitstring` or `nat`. Users can also create types and assign them to variables. Considering that the handshake protocol uses both asymmetric and symmetric encryption, the involved keys must have a customized type. Code sample 5.1 illustrates such types.

```
1 (* Symmetric Encryption key *)
2 type key.
3 (* Asymmetric Encryption key-pair *)
4 type pKey.
```

```
type sKey.
(* Asymmetric Encryption key-pair targeting digital signatures *)
type ssKey.
type spKey.
```

Listing 5.1: Types definiton.

Four of the five types defined above are focused on asymmetric encryption, following good practices on using different keys for digital signatures and for encryption. This contributes to a clearer understanding when writing *destructors* and *constructors* (see below), and more specifically, when defining functions regarding the client and server.

5.2.2 Constructors and Destructors

The *constructors* and *destructors* describe the asymmetric and symmetric encryption algorithms. Specifically, *constructors* refer to the encryption/signing algorithms, and the *destructors* to the decryption/verification algorithms. Code sample 5.2 shows how the proposed *constructors* and *destructors* were implemented in *ProVerif* for our case.

```
(* Symmetric Encryption Constructor & Destructor *)
fun senc(bitstring, key): bitstring.
reduc forall m:bitstring, k:key; sdec(senc(m, k), k) = m.

(* Asymmetric Encryption Constructors & Destructor *)
fun getPkey(sKey): pKey.
fun aenc(bitstring, pKey): bitstring.
reduc forall m:bitstring, k:sKey; adec(aenc(m, getPkey(k)), k) = m.

(* Asymmetric Encryption targeting Digital Signatures Constructors &
   Destructor *)
fun getSPkey(ssKey): spKey.
fun sign(bitstring, ssKey): bitstring.
reduc forall m:bitstring, k:ssKey;
      verifySignature(sign(m, k), getSPkey(k)) = m.
```

Listing 5.2: *Constructors* and *destructors* implemented in *ProVerif* for the formal validation.

The listings show that *destructors* are based on an equality involving the *constructors*. Moreover, the cryptographic algorithms must be strictly followed, *i.e.,* the same amount of inputs and outputs, as well as their types, are specifics that need to be respected. Therefore, the *destructors* `getPkey` and `getSPkey` implementation is fundamental, since both illustrate the relation between the private and public keys and, consequently, ensure that the input/output properties are accomplished.

5.2.3 Client and Server

Since cryptographic primitives are already defined, all the necessary components are in place to begin the data flow between the client and server. Firstly, the communication channel needs to be correctly defined through the use of the type `channel`. Furthermore, *ProVerif* also provides two functions that enable data to be received/sent from/into the channel, *i.e.,* `in(...)` and `out(...)`. Code sample 5.3 illustrates both functions.

```
let client(pkA:pKey, skA:sKey, pkB:spKey) =
  out(c, pkA);
  in(c, x:bitstring);
  let y = adec(x, skA) in
  let (=pkA, =pkB, k:key) = verifySignature(y ,pkB) in
  event acceptsClient(k);
  out(c, senc(s, k));
  event termClient(k, pkA).

let server(pkB:spKey, skB:ssKey, pkA:pKey, reqMAC:bitstring) =
  in(c ,pkX: pKey);
  new k:key;
  event acceptsServer(k, pkX);
  out(c, aenc(sign((pkX, pkB, k), skB), pkX));
  in(c, x:bitstring);
  let z = sdec(x, k) in
  if pkX = pkA then event termServer(k).
```

Listing 5.3: Client and Server functions.

From here on, the client and server know that both are duly identified, and the data that is sent/received is properly encrypted through the use of symmetric encryption. From this step on, the digital authorization prototype is ready to be embedded while taking the presented handshake protocol as granted.

5.3 Prototype Integration

The proposed prototype for digital authorization included the definition of four algorithms. The algorithm concerning the verification of digital authorizations requires multiple data transactions between the client and server. Specifically, the necessary calculations are performed by both sides, with the ultimate verification being performed by the client (becoming one of the most targeted stages by the attackers).
The verification of a digital authorization can be achieved through two different branches: online and offline, where the online one involves the presence of the suggested author, more specifically, its account password which is used in a prior authentication. Therefore, this

section is divided in two different parts: authentication and verification. Subsections 5.3.1 and 5.3.2 thus present the authentication and verification processes in the *ProVerif* language separately.

5.3.1 Authentication

The supposed author of a digital authorization needs to input his password in the web application to unlock such verification branch. The password is sent to the server through a secure channel, *e.g.,* using Hypertext Transfer Protocol Secure (HTTPS) (handshake protocol), and its authenticity is evaluated there. Specifically, the password is processed through the KDF and the result compared with the stored k_1. Therefore, this procedure must be established in the *ProVerif* language, requiring the prior definition of a solely KDF (*destructor*), as well as two new types regarding the generated key, and the *salt* value (Code sample 5.4).

```
type key.
type int.

fun KDF(bitstring, int) : key.
```

Listing 5.4: Definition of the digital authorization prototype types in *ProVerif*.

k_1 is acquired through a query to the database, requiring also the database table definition. Consider Code sample 5.5 where the database has a table `Users` with four attributes: an *ID* (`int`), *username* (`bitstring`), k_1 (`key`), and a *salt* value (`int`).

```
table Users(int, bitstring, key, int).
```

Listing 5.5: Database table definition in *ProVerif*.

The comparison between the calculated *password* representation and the stored k_1 can be achieved without defining any *destructor*. However, another *constructor* needs to be implemented to guarantee the *salt* extraction from a given key k_i (Code sample 5.6).

```
fun getSalt(key) : int.
```

Listing 5.6: Constructor which extracts the salt from a given key.

With this, this initial authentication process is performed through: (i) *salt* extraction; (ii) calculation of k_1' given the inserted *password* and *salt*; and (iii) comparison between k_1' and stored k_1 (Code sample 5.7).

```
let (k1_new:bitstring) = KDF(ins_pwd, getSalt(stored_k1)) in
if k1_new = stored_k1 then ...
```

Listing 5.7: Initial authorization process.

5.3.2 Verification

After ensuring that the supposed author correctly introduced his/her password into the system, the *digital authorization verification algorithm* begins. Initially, the server receives a message m, the supposed *author*, and an integer r_2min, $0 < r_2min < r_2$. Moreover, the k_3 *recalculation* is performed, followed by the MAC + Hash mechanism using k_3 and r_2min instead of generating a r_2. On one hand, the k_3 *recalculation* is similar to the authentication stage, since it extracts the stored $salt$ value, and inserts it together with the *password* into the `KDF` *destructor*, and generating k_3'. On the other hand, the cryptographic primitives that comprise such MAC + Hash mechanism need to be defined through *constructors*. Code sample 5.8 illustrates how the `MAC` was defined in *ProVerif*. This function receives a `bistring` (message m) and a `key` (k_3) as input variables.

```
fun MAC(bitstring, key): bitstring.
```

Listing 5.8: MAC constructor in *ProVerif*.

Furthermore, the resulting value from the previous MAC algorithm is recursively submitted r_2 times to a hash function H. *ProVerif* cannot operate *loops* while defining *constructors* or *destructors* (Code sample 5.9).

```
fun Hash(bitstring): bitstring.
```

Listing 5.9: Hash function constructor in *ProVerif*.

Finally, the previously defined *constructors* require being implemented in the client and server side following the proposed digital authorization prototype flow. Code samples 5.10 and 5.11 represent such integration.

```
let server(...) =
 ...
 in(c, data:bitstring);
 let (msg:bitstring, suppAuthor:bitstring, r2min:int, =pwd) = sdec(data,
    k) in
 out(c, senc((Hash(MAC(msg, (KDF(pwd, salt))))), k));
 ...
```

Listing 5.10: Digital authorization prototype server side.

```
let client(...) =
 ...
 out(c, senc((m, author, r2min, password), k));
 in(c, data:bitstring);
 let (predigitAuth:bitstring, rdiff:int) = sdec(data, k) in
 ...
```

Listing 5.11: Digital authorization prototype client side.

5.4 Tests

In order to evaluate the produced *ProVerif* script, the author developed several test scenarios. These tests scenarios are presented below according to the following structure: (i) presenting the issue surrounding the test; (ii) presenting the expected results; and (iii) analysing the obtained results. In these tests, the attacker focuses on discovering specific variables by applying the man-in-the-middle attack (passive). Figure 5.1 depicts the general attack model and placement of the attacker.

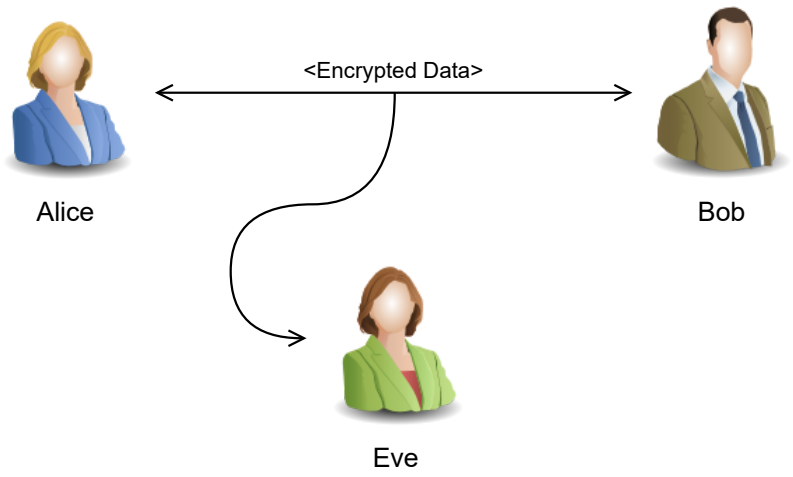

Figure 5.1: Placement of the attacker in the consider tests scenario (passive man-in-the-middle attack).

Subsections 5.4.1, 5.4.2 and 5.4.3 describe the proposed scenarios, with the last one consisting of the most critical one, since both client and server are considered as being compromised. Beyond this, after the completion of a test, each evaluated variable is associated to a `query` that identifies if it was compromised (`false`) or not (`true`). Finally, these tests are considered as successful if the attacker cannot acquire other variables beyond the exposed ones, and unsuccessful otherwise. The target variables are the following:

- `digitAuth` - digital authorization to be verified;
- `author` - supposed author;
- `m` - message;
- `password` - author password;
- `r2` - random stored integer.

Note that the listed variables are the ones which are transmitted through the communication channel `c`.

5.4.1 Compromised Password

Issue

This test addresses the consequences that may arise from the exposure of a user password on the online verification process. In order to reach such objective using *ProVerif*, the `password` cannot be defined as `private`, providing it to the attacker when the victim is verifying a specific digital authorization.

```
free password:bitstring.
```

Listing 5.12: Password secrecy definition in *ProVerif*.

Expected Results

The exposure of such sensitive content can compromise the entire workflow of the original author of the digital authorization, granting the attacker the ability of forging/verifying digital authorizations. Regardless of the way the attacker acquired such secret material, while verifying a digital authorization, the data which is sent by the server is also compromised. However, the *renewing of digital authorizations* avoid any issue, since data is *renewed* and the secret *salt* is never transmitted to the client. Note that this specific case scenario only takes place when the attacker performs the verification, being impossible to reach data due to the usage of the handshake protocol.

Given this, the mitigation plan consists of changing, as soon as possible, the account password, and warn the other application users that the digital authorizations associated with the victim are not reliable at all. Moreover, the victim should minimize this problem by assuming which of those were actually produced by him/her.

Analysis of the Obtained Results

After executing the script for the described scenario, the output of *ProVerif* confirms that the attacker cannot access anything beyond the `password`. Code sample 5.13 presents such results.

```
Verification summary:

Query not attacker(password[]) is false.
Query not attacker(digitAuth[]) is true.
Query not attacker(m[]) is true.
Query not attacker(author[]) is true.
```

Listing 5.13: Obtained results from the test (compromised password).

Moreover, although the attacker had managed to access the transmitted data that is sent from the client (*i.e.,* message, author and r_2min), the private contents remain secure. This test was successfully completed, since the attacker could not reach any other variable that

would compromise private content beyond the *password*, *i.e.,* the *salt* value that is stored within the database.

5.4.2 Compromised Handshake Protocol

Issue

This test targets the consequences resulting from the exposure of the session key k. To reach such goal using *ProVerif*, the variable `k` cannot be defined as `private`, since the attacker managed to discover it.

```
free k:symKey.
```

Listing 5.14: Key secrecy definition in *ProVerif* for one of the test scenarios.

Expected Results

The digital authorization prototype is highly dependent of the handshake protocol, and the exposure of such crucial content compromises the entire communication channel. Therefore, it is expected that the variables which are exchanged between the client and server are also compromised. Specifically, the main issue consists of the exposure of the author *password*, since it is the unique private content within the digital authorization prototype that is transmitted. Thus, such exposure also accumulates the problems of the previous test scenario.

Furthermore, the session key k is generated when a client desires to communicate with the server, working as a one time session password that may raise several issues. Considering that the attacker has the ability to eavesdrop the data, this test consists of the *passive Man-in-the-Middle* (MitM).

Given this, the mitigation plan involves the modification of the author password, the expiration of the client session, and the review of the key generation algorithm.

Analysis of the Obtained Results

After executing the script for the described scenario, the output of *ProVerif* confirms the expected results, since the attacker managed to access the exchanged data. Code sample 5.15 presents a summary of such results.

```
Verification summary:

Query not attacker(password[]) is false.
Query not attacker(digitAuth[]) is false.
Query not attacker(m[]) is false.
Query not attacker(author[]) is false.
```

Listing 5.15: Obtained results from the test (compromised handshake protocol).

The way the attacker managed to discover the symmetric key should be investigated. The server must ensure the secure generation and exchange of the symmetric key, respecting the associated requirements of the chosen symmetric encryption algorithm. Thus, a potential failure within this generation process can compromise the generated key and, consequently, the entire flow.

5.4.3 Compromised Password + Salt

Issue

This test addresses the most critical case scenario that the document management system could face in terms of the proposed digital authorization scheme, since both client and server sides are compromised. More specifically, this scenario assumes that the *password* belonging to the client and the *salt* value stored in the server are both exfiltrated. Considering that the *salt* value is not transmitted through the channel, such test cannot be be reproduced in *ProVerif* language. Regardless of this, the expected results can still be drawn.

Expected Results

The data exposure considered in this last test represents the most alarming scenario. In the first presented test, it was proved that the *password* exposure allows the attacker to perform both production and verification of digital authorizations, being impossible to recreate such processes due to the lack of the *salt* value. Obviously, to acquire the *salt* value the server require being compromised, more specifically, the attacker has discovered a manner to extract such value from the database. Thus, such data exposure compromise the entire application, since the server has been compromised, meaning that any user can be targeted.

5.5 Conclusions

Formal verification was applied in the proposed digital authorization prototype aiming to provide minimal assurances of the security of the adopted approach and reasoning. Starting from the assumption that data exchanges are performed in a confidential and authenticated manner (via the establishment of cryptographic secrets at the beginning of the communications via a secure the handshake protocol), several tests were performed using *ProVerif*, varying the data that the attacker has access to, and attempting to exploit possible flaws concerning the defined *destructors* or *constructors*. The results obtained from the tool confirm that, per se, the security of the entire scheme depends of secrecy of the password and salt. No flaws in the construction of the digital authorization scheme allows the attacker to compromise that secrets. Nonetheless, even though secrecy and authentication are proved, ensuring that a potential integration of the proposed scheme algorithms is fully

secure involves reaching the common criteria levels, since there are many other variables that affect that security, namely: data storage, choice of cryptographic algorithms, cloud services, network protocols, etc.

Chapter 6

Conclusions and Future Work

6.1 Main Conclusions

The work discussed herein was focused on streamlining the usage of digital authorizations in digital processes, in the sense of easing their correct adoption in the context of contemporary organizations or enterprises. This subject is difficult to address nowadays because there is no widely adopted or agreed mechanism for digital authorizations, thought those aspects are mostly covered for digital signatures in Europe. There is no standard format for digital authorizations, though intuition suggests that they should be formed by a message with temporal pertinence glued to a supporting document via some strong digital means such as a qualified digital signature or a message authentication code mechanism. Obviously, adding qualified digital signatures to internal processes where several authorizations are issued per day and flow would come with the burden of managing a PKI.
The objectives set for this work were achieved by the end of the project. The discussion and proposal of a potential format for digital authorizations is perhaps one of the best contributions of this work, followed by the proposal of the scheme for building and verifying those digital authorizations. These contributions were respectively preceded and succeeded by an analysis of the regulations and recognized related mechanisms available in Europe, and by the formal analysis of the proposed scheme using the well known specialized tool *ProVerif*.
The proposed format captures the temporal pertinence by including the timestamp, and glues everything up via hashes to the associated document and to the approval/opinion/dispatch text. Per se, the digital authorization is then a MAC calculated with keys that are derived from the password of users in a secure manner. In order to deal with many functional (online and offline users) and security requirements (specially adding the possibility for a user to verify an authorization client side), the proposed scheme adapts the intuition behind the Lamport one-time password scheme, and the actual MAC codes are replaced with hash values resulting from the recursive application of a secure hash function to the initial authorization code. This strategy allows a user to prove another user that (s)he issued a given authorization without needing to disclose the (symmetric) key used to calculate the MAC.
Since state wide PKIs are becoming commonplace (as in Portugal), it is possible that the future of digital authorizations will include digital signatures or seals instead of MACs. Nonetheless, the standardization of the format and schemes are of utmost importance for the adoption and recognition of these mechanisms, even between entities or organizations.

One main and final conclusion is that this work shows that there is still a lot of ground to cover in this particular subject, but also that it is possible to streamline the usage of correct and security digital authorizations in the short to medium term. It should be possible to assure many security requirements of digital authorizations resorting only to symmetric key encryption, although further tweaking to the authorization format or algorithms may be necessary.

6.2 Future Work

This project already covered a considerable amount of terrain in the context of embedding digital authorizations in processes of organizations or enterprises. However, since the work was focused on exploring and setting up the ground for the integration of those mechanisms, it did not go as far as implementing them in a real world document management system. This would perhaps be one of the most useful next steps, since that implementation would surely bring up new details and unforeseen challenges that need to be addressed by the construction.

The previous line of future work already hints at that the proposed construction will surely need to be iterated to cope with new challenges or security requirements. Actually, the requirement of non-repudiation is not completely covered in the proposed scheme, since a user might refuse to provide his or her password (or purposely type it wrongly) during a verification of an authorization, simultaneously claiming that (s)he did not issue that digital authorization. This possibility is partially mitigated because the system also possesses a second means of verification (that depends on the password of the user), but this needs to be nonetheless addressed differently in scenarios where repudiation is more probable.

Another line of work that deserves attention is the assessment of the impact of embedding secure digital authorizations into existing systems, namely in terms of human and computational overhead. In terms of computational overhead, it should be interesting to assess the processing cost for building and verifying authorizations, and the storage requirements for keys and digital authorizations. Regarding human impact, it would be important to somehow measure the impact on the workflow of the users, namely in terms of usability and perception of security, but also in terms of burden imposed to the helpdesk when dealing with questions concerning the new technology. The comparison with systems using digital signatures instead of MACs and hashes would ultimately prove if the approach followed here is really advantageous or not.

Bibliography

[1] M. Bellare and S. K. Miner, “A Forward-Secure Digital Signature Scheme,” 1999. [Online]. Available: https://cseweb.ucsd.edu/~mihir/papers/fsig.pdf xvii, 11, 12

[2] OWASP, “Cryptographic Storage Cheat Sheet,” January 2021. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html#key-lifetimes-and-rotation xix, 13

[3] ACM, “The 2012 ACM Computing Classification System,” February 2022. [Online]. Available: https://www.acm.org/publications/class-2012 1

[4] T. E. Parliament and the Council of the European Union, “Electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC,” July 2014. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014R0910 5, 6, 10, 17

[5] E. Comission, “What does QTSP or QTS mean,” December 2019, last accessed: 2022. [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/ESIGKB/What+does+QTSP+or+QTS+mean 5

[6] ENISA, “Security guidelines on the appropriate use of qualified electronic signatures,” June 2017. [Online]. Available: https://www.enisa.europa.eu/publications/security-guidelines-on-the-appropriate-use-of-qualified-electronic-signatures 9, 10

[7] DocuSign, “Every day someone signs something that means everything.” February 2022. [Online]. Available: https://www.docusign.com/ 9

[8] asecuritysite, “Blum Integers,” January 2022. [Online]. Available: https://asecuritysite.com/encryption/blum02?pval=32 11

[9] F. E. S. Dunbar, “Digital Signature Schemes Variations,” 2002. [Online]. Available: https://uwspace.uwaterloo.ca/bitstream/handle/10012/1076/fesdunbar2002.pdf?sequence=1&isAllowed=y 13

[10] H. Ghodosi and J. Pieprzyk, “Repudiation of cheating and non-repudiation of zhang’s proxy signature schemes,” in *Information Security and Privacy*, J. Pieprzyk, R. Safavi-Naini, and J. Seberry, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 129–134. 15

[11] L. Lamport, “Password authentication with insecure communication,” *Commun. ACM*, vol. 24, no. 11, p. 770–772, nov 1981. [Online]. Available: https://doi.org/10.1145/358790.358797 16

[12] B. Blanchet, “Modeling and verifying security protocols with the applied pi calculus and proverif,” *Foundations and Trends® in Privacy and Security*, vol. 1, no. 1-2, pp. 1–135, 2016. [Online]. Available: http://dx.doi.org/10.1561/3300000004 17

[13] M. Just, *Diffie–Hellman Key Agreement*. Boston, MA: Springer US, 2011, pp. 341–342. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_75 17

[14] S. Nisha and M. Farik, “Rsa public key cryptography algorithm – a review,” *International Journal of Scientific Technology Research*, vol. 6, pp. 187–191, 07 2017. 17

[15] C. Paar and J. Pelzl, *Message Authentication Codes (MACs)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 319–330. [Online]. Available: https://doi.org/10.1007/978-3-642-04101-3_12 20

[16] B. Preneel, *HMAC*. Boston, MA: Springer US, 2011, pp. 559–560. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_581 22

[17] M. J. Dworkin, “Sp 800-38b. recommendation for block cipher modes of operation: The cmac mode for authentication,” Gaithersburg, MD, USA, Tech. Rep., 2005. 22

[18] J. Kelsey, S. jen Chang, and R. Perlner, “Sha-3 derived functions: cshake, kmac, tuplehash and parallelhash,” 2016-12-22 00:12:00 2016. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=922422 22

[19] C. Adams, G. Kramer, S. Mister, and R. Zuccherato, “On the security of key derivation functions,” in *Information Security*, K. Zhang and Y. Zheng, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 134–145. 22

[20] G. Chen, B. Xu, M. Lu, and N.-S. Chen, “Exploring blockchain technology and its potential applications for education,” *Smart Learning Environments*, vol. 5, 01 2018. 29

[21] Mozilla, “What is javascript?” September 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript 29