




Article

A Comprehensive and Privacy-Aware Approach for Remote Qualified Electronic Signatures

Iulian Acioabăniței ^{1,*}, Ștefan-Ciprian Arseni ^{1,2,†}, Emil Bureacă ^{1,†} and Mihai Togan ^{1,3,†}

¹ Faculty of Information Systems and Cyber Security, Military Technical Academy “Ferdinand I”, 050141 Bucharest, Romania; stefan.arseni@mta.ro (Ș.-C.A.); emil.bureaca@mta.ro (E.B.); mihai.togan@mta.ro (M.T.)

² Faculty of Electronics, Telecommunications and Information Technology, National University of Science and Technology POLITEHNICA Bucharest, 060042 Bucharest, Romania

³ Research and Innovation Department, CertSIGN S.A., 050881 Bucharest, Romania

* Correspondence: iulian.acioabantei@mta.ro

† These authors contributed equally to this work.

Abstract: The current shift towards digital transactions emphasizes the need for robust Qualified Electronic Signature (QES) frameworks that safeguard integrity and privacy. Having the potential to become the leading type of adopted QES, the main challenge that Remote QESs present to end users is choosing between transmitting the entire document or only its digest to the Trust Service Provider (TSP). The first option compromises the document’s confidentiality, while the second one requires the development of signature applications compliant with advanced signature formats, a task that often needs additional time and resources. In this paper, we introduce a comprehensive strategy for remote QESs, designed for seamless integration with current client applications, while simultaneously maintaining user privacy. The main topics approached in this paper are the following: a comprehensive architecture for privacy-aware remote QES systems, relevant standards and legislation, integration scenarios for clients, and remote QES standard protocols to assure communication between client and TSP environments. Furthermore, we also explore the integration of our proposed solution with an enhanced long-term preservation service that uses Ethereum smart contracts and methodologies to implement signature applications with advanced electronic signatures via open-source libraries while ensuring document privacy. The main result of this work is a flexible on-premise module that provides the ability to sign, validate, and preserve documents, with a minimal integration effort.

Keywords: remote qualified electronic signatures; long-term preservation; privacy-aware



Citation: Acioabăniței, I.; Arseni, Ș.-C.; Bureacă, E.; Togan, M. A Comprehensive and Privacy-Aware Approach for Remote Qualified Electronic Signatures. *Electronics* **2024**, *13*, 757. <https://doi.org/10.3390/electronics13040757>

Academic Editors: Aryya Gangopadhyay, Vasile-Daniel Pavaloaia, Rodrigo Martin-Rojas and Piotr Sulikowski

Received: 16 December 2023

Revised: 15 January 2024

Accepted: 6 February 2024

Published: 14 February 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Since the eIDAS Regulation [1] came into force, electronic signatures have become more appealing to businesses that intend to digitize their processes and flows. Also, the COVID-19 pandemic accelerated this digitization process. Therefore, RQESs (Remote Qualified Electronic Signatures) became more popular and adopted on a larger scale. This paper delves into the realm of Qualified Electronic Signatures (QESs), focusing on Remote QES solutions, which are pivotal in validating digital transactions and documents while ensuring their legal standing.

Despite their growing importance, the implementation of QESs, particularly in remote environments, poses significant challenges. These challenges stem from the need to balance security, privacy, interoperability, and the ease of integration into existing systems. Furthermore, with the rapid pace of technological advancements and evolving legislative landscapes, there is a pressing need for solutions that are not only compliant with current standards but also adaptable to future changes. Another factor to consider is the ease of integration with current client applications, since companies also have to evaluate the development time and user experience changes for such integrations. This paper presents

a comprehensive architecture for an RQES system that addresses the abovementioned challenges. The main objectives of this study are to achieve the following:

- Develop a privacy-centric RQES architecture: We design a system that ensures the user's document privacy, while keeping the advantages of RQESs over local QESs.
- Ensure compliance: The proposed solution needs to be aligned with security requirements from relevant standards and legislation, mainly the eIDAS Regulation and CEN and ETSI standards.
- Ensure interoperability: The design of the proposed solution must be modular and consider various protocols proposed in the standards. Also, the system needs to ensure interoperability by respecting the advanced electronic signature formats (PAdES, CAdES, XAdES, and ASiC).
- Ease of integration: We propose a flexible solution with different integration options for different client requirements.
- Long-term signature preservation: We utilize, in a transparent manner, a long-term preservation service for enhancing the durability and validity of the electronic signatures.

Using RQESs brings the need for the client to send the entire document or a digest of the document to be signed to the TSP environment. In our proposal, only digests are sent by the client to the TSP, such that obtaining the QES does not involve the disclosure of document content. Still, this approach puts a burden on the client, who must implement digest extraction and then integrate the signed digest into different signature formats, such as PAdES, CAdES, XAdES, and ASiC. This is not a trivial task, and here we implement a module that would increase the ease of integration with the TSP for clients to provide remote QES services.

The overall scope of this paper is to contribute significantly to the domain of Qualified Electronic Signatures, specifically in remote environments, offering insights and solutions that resonate with current and future digital transaction needs.

The remainder of this paper is structured as follows. Section 2 presents our previous work, published in a series of research papers over a few years, which made it possible to understand the challenges and requirements of such a system. Section 3 provides an overview of two topics: the current technological status and relevant standards and legislation. Section 4 offers a detailed view of the proposed system architecture, including implementation details, relevant flows, and proprietary module description. Section 5 discusses various deployment scenarios, showcasing the system's adaptability and ease of implementation for clients. Section 6 presents a discussion on the key aspects of the solution, such as security, ease of integration, and interoperability. Finally, Section 7 concludes the paper with final remarks and potential directions for future research.

2. Previous Work

In the past few years, we have worked on different problems regarding RQESs. Some of the most important research directions were the following:

- Harmonizing local and remote signatures, from the signature application's perspective [2,3].
- Using blockchain to improve the security of RQES infrastructure [4] via GSuite and Android.
- Surveying methods for implementing RQESs using open-source cryptographic libraries [5].
- Implementing a Long-Term Signature Preservation Service [6].
- Different integrations for RQESs with a set of platforms, like Android and Google GSuite, and a set of security protocols, like QRP and SQRL.

This section summarizes the findings and provides a brief overview of these different research directions.

2.1. Standards Harmonization

During the 1999–2016 period, users could obtain a QES only by physically owning a hardware cryptographic token. To work with such a device, most of the signing applications implemented PKCS#11 integrations. Furthermore, CNG-API was also used by Windows signature applications as a high-level cryptographic layer.

After the introduction of RQESs with legal value, users were reluctant to adopt this new paradigm, mainly because it meant using new signature applications. To combine the advantage of using a known application, without the need for its developers to make additional implementations, with the ease of not having a physical device, two technical solutions were proposed. The implementation details, the signing flows, and the main technical difficulties were presented for two such modules: PKCS#11 and CNG KSP. Most popular electronic signature applications (e.g., Adobe Acrobat Reader and browsers) were covered by the two proposals.

In short, the PKCS#11 module [2] was implemented as an offshoot of the open-source SoftHSMv2 project. It also contains a CSC client. The most challenging problem to solve was that the PKCS#11 protocol was not compatible with the CSC protocol. For example, PKCS#11 did not support inserting an OTP as a second authentication factor. To solve this, our PKCS#11 module opened a GUI for the user to insert the necessary credentials and then sent them to the CSC server.

For the implementation of the proposed CNG module [3], we opted to skip over the CryptoAPI CSP and use the newer KSP (Key Storage Provider) of the CNG. Some of the challenges faced were the following:

- Finding a way to install the KSP in a user-friendly manner.
- Installing the user certificates in Windows Certificate Store in such a manner that they appear to have a corresponding private key.
- Overcoming some of the KSP interface limitations.

Similar to the PKCS#11 module, the KSP module uses a GUI to collect user credentials. Implementation was based on the Cryptographic Provider Development Kit 8 project, distributed by Microsoft.

2.2. More Secure RQESs

The main requirement for obtaining an RQES with legal value is that the signature is created with the sole control of the user over the private key. Since it is stored in the TSP environment, users might suspect that the TSP could maliciously use their private key to sign documents. In this regard, SABRES [4], the proposed solution, uses blockchain to improve the transparency of RQES solutions.

The proposed system leaves a minimal imprint on the infrastructure of the TSP and complies with the standards in force: including ensuring that the signing protocol set out by the TSP is the same. On the server side, we implemented a modular CSC proxy, so that there is no additional implementation needed from the TSP side.

In short, by using this solution, each signing request is added to the blockchain by the client application, under the control of the user. Successful calls for the signature/signHash CSC endpoint trigger an addition in the blockchain by the server-level signing application.

Finally, the key point of the solution is the means of validating the signatures: in addition to classic validations, the validation application must verify if, for that specific signature, there is a transaction stored in the blockchain, indicating the user's intention to sign. Otherwise, the signature is declared invalid.

Using SABRES, TSPs are not restricted as to their implementation, and users benefit from a boost in the transparency of private key access. As blockchain is used, one obtains a distributed and immutable source of trust that certifies the user's intention to sign. The source code is publicly available on GitHub.

2.3. Open-Source Cryptographic Libraries for RQESs

TSPs that offer RQES services need to demonstrate a standard signing protocol. One of the most popular RQES protocols is the CSC Protocol, which is privacy-aware by design since it only supports sending the hash to the TSP environment. This approach has been well received from a security standpoint but introduces some difficulties regarding signature formatting. For example, many cryptographic libraries allow for signing a PDF file using a .p12 file via PKCS#11 or CNG API. Signing using a remote key was not taken into account when these libraries were designed and implemented; therefore, this task is not always as straightforward as one would expect. Our survey paper [5] presented how one might implement an RQES using various cryptographic libraries and cryptographic modules: PKCS#11, CNG-API, CSP CryptoAPI, OpenSSL, BouncyCastle, Apache PDFBox, SecureBlackbox from nsoftware, and JCE.

2.4. Long-Term Signature Preservation Service

An important problem in the digital signature environment is the volatile nature of digital signatures, which are only valid for a limited time due to the expiration of the asymmetric private key used in their creation. This poses a significant challenge in validating signed documents after several years. To address this, legislative and standardization [7,8] efforts have led to the creation of legal and technical frameworks for long-term preservation services. Ref. [6] presented an implementation of a long-term preservation service compliant with ETSI standards that uses existing PKI infrastructure and incorporates blockchain technology for added trust, resilience, and transparency.

Some of the most important challenges of long-term signature status preservation services are as follows:

- The weakening of cryptographic algorithms caused by technological breakthroughs.
- Digital signature expiration, and the evolution of data formats.
- Maintaining a standard-compliant application while adding blockchain integration.

The main strategy of the proposed service to achieve its goal was signature augmentation using certificate revocation information (CRL, OCSP); certificate chain; and timestamps so that it could be validated over a longer period. For example, by using this approach, a PAdES B-B or B-T signature is elevated to the PAdES LTA level.

Usually, this type of service is also integrated with an Electronic Archival System. Note that this integration is not standardized in any way but is dependent on each EU member country's legislation, because of the way electronic archival legislation evolved.

We consider this service to be foundational for a comprehensive approach to using QESs, for both remote and local signatures. Most industries have requirements regarding the time to archive legally binding documents. Just storing the signed document for that period (tens of years) would result in an indeterminate signature status. Therefore, we consider that long-term preservation services will become more and more popular in the coming years.

3. Current Technological Context

3.1. Working Standards and Legislation

There are two broad groups of laws that apply to electronic signatures globally. For instance, the USA embraced the open category, where any kind of computerized sound, figure, or method of communication to convey the aim of signing a document is recognized as an electronic signature [9]. Briefly, proving the intention to sign is the main objective of this class. On the other hand, mostly in nations that are members of the EU, guaranteeing that the users have sole authority over their private keys defines the core principle of the closed legislation category. The private and public sectors are clearly distinguished by American law, which allows the former to set up its own methods of signature creation and verification. These legislative systems are comparable in that they allow for the use of various signature assurance levels, even though they are not expressly stated. Thus, several technical standards have been developed over the years to provide concrete specifications.

Marking the birth of the electronic signature legislative framework in Europe, the Directive 1999/93/EC [10] laid out fundamental context by defining core terms like electronic signature, advanced electronic signature, and qualified certificate. Moreover, it drew a line of equivalence between advanced electronic signatures and physical ones. On the other hand, the electronic category stands out due to its recognition across European states by the same ruling, while the holographic type depends on clauses specific to each country. During 2014, a major upgrade came into place with the publication of the eIDAS Regulation, introducing the specific notion of a qualified electronic signature and timestamp [1]. Furthermore, applying qualified electronic signatures is possible even for users who do not possess a cryptographic hardware device. This responsibility is delegated to a TSP owning a cloud signature services that manages all the cryptographic material with a high protection level while assuring the users' sole control over their private keys. Subsequently, several other legislative acts supplemented the eIDAS Regulation:

- Decision (EU) 2015/1502 set out the methods and minimal technical requirements for electronic identification assurance levels.
- Decision (EU) 2015/1505 established the formats and technological requirements for trusted lists.
- Decision (EU) 2015/1506 specified the requirements and formats for advanced electronic signatures and seals that public sector organizations must accept. Moreover, it referenced several ETSI standards that define baseline signature formats for CAdES [11], XAdES [12], PAdES [13], and ASiC [14].

Considering the complexity and large number of existing standards at the European Union level, as well as the numerous legislative acts, Figure 1 depicts an architecture that identifies the key components required for the development of an interoperable infrastructure providing remote signature services.

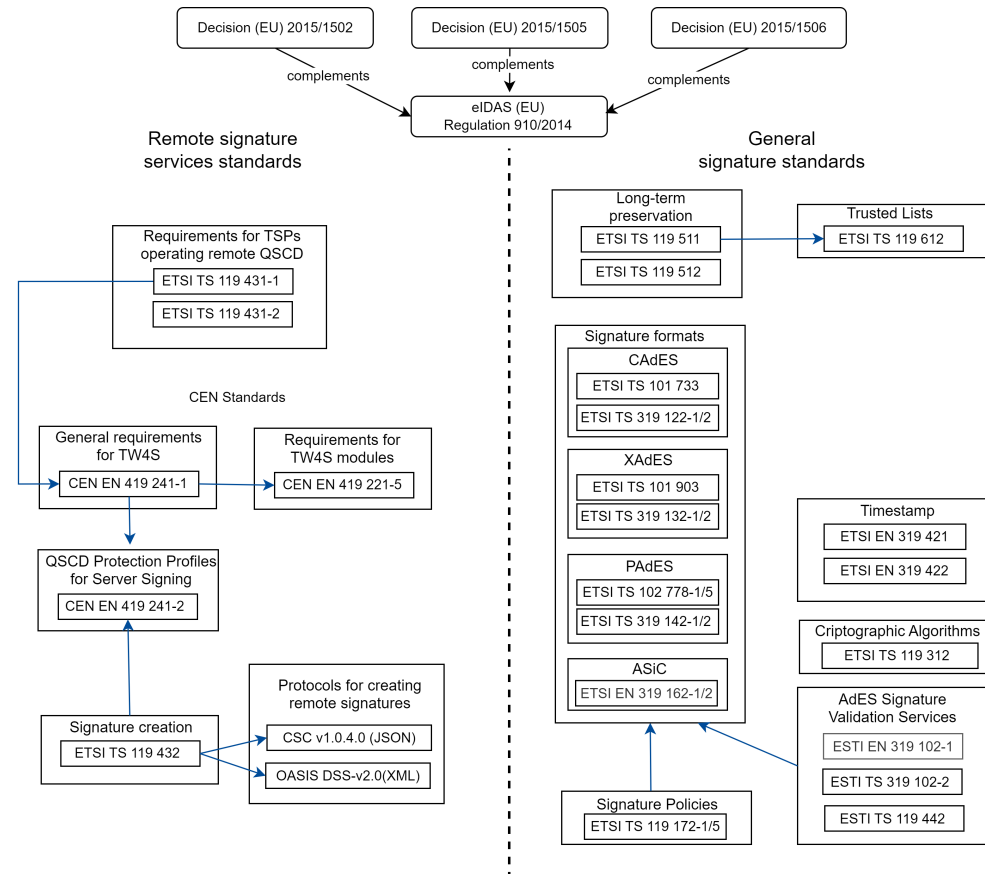


Figure 1. General remote signature standards architecture.

3.1.1. CEN EN 419 241

To address the fundamental concern of the user's exclusive control over their confidential key(s) when targeting a system that supports remote electronic signature operations, the CEN 419 241-1 standard describes the following two Sole Control Assurance Levels [15]:

- Low (SCAL1)—Signing keys are used with a low level of trust, under the sole control of the signer, and the activation of the signing key can remain for a certain period, or for several signatures. The use of the key by the authorized signer is enforced by the SSA that handles user authentication;
- High (SCAL2)—Signing keys are used with a high level of trust, under the sole control of the signer. The use of the key by the authorized signer is enforced by the SAM module that provides signature activation data to the signer using an activation protocol.

Additionally, several elements are involved in the remote design:

- SCA (Signature Creation Application)—the component whose principal responsibility is handling the document that needs to be signed, thus generating an input for the signing server and administering the resulting signature.
- SSA (Server Signing Application)—the remote infrastructure unit obtaining from the SCA a representation of the document to be signed and performing the signature creation under the sole control of the calling user.
- SCDev (Signature Creation Device)—usually defined through an HSM, its main purpose is to create a digital signature value using the user's key.
- SIC (Signer's Interaction Component)—user component that provides two-factor authentication to establish a relation between the signer and the signature as part of the signature activation data.
- SAM (Signature Activation Module)—located in a tamper-protected environment, it uses the signature activation data (SAD) to guarantee, with a high level of confidence, SCAL2 assurance.

Participating in two protocols, the SAP (Signature Activation Protocol) and Signature Creation Protocol, provides the synergy of the aforementioned components. While the former generates the SAD for the remote-controlled SAM, ensuring exclusive control over the signing keys, the other acts as a bridge between the SCA and SSA, adding a layer of interoperability between cloud signature services and client apps.

3.1.2. ETSI TS 119 432

Targeting an environment characterized by interoperability and usability, the ETSI group developed a series of standards and specifications that came as technical support for the eIDAS 910/2014 [1] Regulation regarding electronic identification and trust services for electronic transactions. Thus, one primary document modeling the protocols for the remote creation of Advanced (AdES) and Qualified (QES) Electronic Signatures based on public key infrastructure is represented by ETSI Standard [16].

The remote signature creation process is composed of a concise series of steps and components. For example, a signature creation scenario refers to the case where the signature key is stored within a cryptographic module, SCDev, operated by a trusted provider, the SCSP (Signature Creation Service Provider).

The process contains two main components:

- **SSASC (Server Signing Application Service Component)**—This is the interface that provides support for creating the digital signature value. It interacts with the cryptographic signature creation module, SCDev, through which the signer controls the key with a certain level of trust.
- **SCASC (Signature Creation Application Service Component)**—This component receives the data to be signed together with other parameters and transmits the request to create the signature value to the SSASC.

Additionally, two operating modes exist:

- **Synchronous**—This refers to the fact that after sending the request to the SCS (Signature Creation System), the client waits for a final response before continuing its processing.
- **Asynchronous**—This is defined by the idea that the client application, after sending the request to the SCS, can continue to process other tasks without waiting for the completion of the operation, with the possibility of being notified by the server when the request processing is finished. The client will then send a request to receive the result.

3.1.3. Cloud Signature Consortium

With Adobe as its primary founding member, the Cloud Signature Consortium (CSC) is a private organization that advocates for a second standard in the context of Remote Electronic Signatures and keeps tight ties with the ESI working group of ETSI.

The CSC leverages HTTP requests to define an API for signing in the cloud [17]. The information sent to the server and the received responses are in JSON format and use BASE64 encoding. To access the API, the client must use a base URI of the remote service. While it is also possible that the remote service can make use of security measures, such as a VPN, TLS must be implemented by the remote service in order to guarantee the confidentiality and integrity of the communication channel between the client and server.

The protocol makes use of authorization and authentication to determine whether access to resources is permitted, whilst defining two levels of access: to the API and to the private key. Aside from the `oauth/authorize`, `auth/login`, and `info` methods, all require the use of an access token.

There are 14 methods in the present version (v2.0) of the CSC protocol. Three of them, starting with `oauth2`, are particular to the OAuth service and are implemented by the authorization server; the remaining ones are protocol-specific and are implemented by the TSP. Of these, we list only those necessary for the execution of a flow resulting in a remote signature value obtained by sending a representation of the document to be signed (i.e., a hash):

- **info**—This provides details on the remote signing service, including the CSC version that is in use, the supported methods, and the available authentication options.
- **oauth2/authorize**—The behavior varies based on the transmitted parameters, as follows:
 - The method generates an access code if the option `scope = service` is present. This code is then utilized by `oauth2/token` to retrieve an API access token.
 - For `scope = credential`, the generated code will actually be the SAD. Naturally, signing the hash value or values is also necessary for this mode of use.
- **oauth2/token**—Based on a code obtained via the `oauth2/authorize` method with `scope = service`, this is utilized to obtain an API access token.
 - Additionally, based on the previous, still-valid token, this approach can be used to renew an access token.
- **signatures/signHash**—This carries out the cryptographic computation needed to produce the transmitted hash's signature. The SAD and API access token are required for this procedure to be approved.
- **credentials/list**—This endpoint provides a list of the user's credentials, identified by using the previously acquired token.
- **credentials/info**—This method returns a structure containing the user certificate's data. The information contained includes elements such as the base64 encoded certificate, the length of the key, accepted algorithms mentioned using OIDs, and additional certificate fields.

Depending on the authentication and authorization method that has been implemented, the CSC protocol can be used for obtaining signatures. Web applications are appropriate for this method since using an OAuth 2.0 authorization server necessitates a series of redirects between the signing application and the authorization server.

3.1.4. Digital Signature Service Core Protocols

A high-level remote signature creation and verification protocol, known as DSS, and developed by the OASIS committee, is currently at a fairly mature stage. It is defined as a request–response protocol that offers transport bindings for HTTP and SOAP, while its specification focuses on the main data structures used for the information transfer between the client and the server, which encompass XML and JSON formats. Furthermore, several optional elements are described to support protocol customization to fit with the specific signature formats, attributes, and cryptographic keys. Thus, the client has a rich variety of selections when sending the signature request containing the target document, or a representation of the document to be signed (e.g., a digest of the document), under the specified options and signature format (e.g., CAdES, PAdES, XAdES, JAdES, and ASiC).

The core notions of the DSS signature creation procedure revolve around its signing design model. Starting from receiving one of its main defined data structures, SignRequest, the server decides on the signature type to be built, belonging to either the XML or CMS category. There are clear distinctions between the aforementioned classes of signature formats. However, before the last step of the flow, corresponding to the SignResponse composition, an additional option arises, representing the addition of a timestamp to the newly created signature [18].

3.2. Similar Integrations

As mentioned in the previous sections of this paper, remote document signing services have seen an increase in popularity in the past few years, mainly because of the favorable environment created by the adoption of various standards and legislation that have consolidated users' trust in the signing process and enforce their data privacy. Several solutions have been on the market for a few years and have shown that remote signing can be a trustworthy, reliable, and easy-to-use process for the final user. In this section, we analyze several solutions that are available today.

3.2.1. Cryptomathic Signer

Cryptomathic Signer [19,20] is an eIDAS-certified remote signing server that can create Advanced Electronic Signatures (AdESs) or Qualified Electronic Signatures (QESs). This application also integrates WYSIWYS (What You See Is What You Sign) technology, patented by Cryptomathic, which opens a secure browsing session in which the user can see their original document, thus assuring them that their signature will be on that specific form of the document. Given the fact that Cryptomathic Signer is offered as a signing server, there are three possible scenarios for its deployment:

- On-Premise—The solution is installed in the customer infrastructure and is integrated with other existing services. This option is mainly directed towards TSPs (Trusted Service Providers) that want to expand their offered services palette.
- Hybrid Signing Service—The solution is offered as a service and is integrated into the organization's infrastructure as a remote TSP.
- Fully Managed TSP—The solution is offered completely remotely, with users accessing the remote signing services through specific applications provided by Cryptomathic.

Regardless of the chosen type of deployment, the solution is presented to ensure the same level of compliance with current standards and regulations. The supported signature profiles include the ETSI-defined PAdES, XAdES, and CAdES profiles.

3.2.2. Bit4id SignCloud

Bit4id SingCloud [21] is another remote signing solution that enables users to remotely sign documents, no matter what type of device is in use. The solution is provided as a collection of hardware and software resources with specific interconnections. This solution offers compliance with the same signature profiles defined by ETSI, namely PAdES, XAdES, and CAdES.

3.2.3. DigitalSign SigningDesk

DigitalSign is a QTSP from Portugal that offers several solutions that could be integrated with targeted infrastructure to provide remote signing services. One of these solutions is SigningDesk [22], enabling users to remotely sign documents, regardless of the two types of deployment it offers:

- Private Cloud platform—A single platform that integrates into the client infrastructure and offers the possibility of customization, depending on the client's needs. It can be integrated with a Single Sign-On (SSO) authentication service;
- Shared Cloud platform—Services are offered remotely, eliminating the need for local integration within the client infrastructure.

3.2.4. Nextsense Signing Suite

The Signing Suite [23] is a collection of services offered by Nextsense, a company offering services for the digital transformation of organizations. The Signing Suite collection comprises services that enable users to digitally sign, timestamp, seal, or verify documents without limitations as to the device they use. Their provided signature profiles are compliant with ETSI standards, targeting PAdES, XAdES, and CAdES. One key service from this collection is Nextsense Remote Signing, which allows users to remotely sign documents by either directly accessing the services through a web browser or integrating a Remote Signing Bridge into the user infrastructure, enabling local signatures with cloud-stored digital certificates.

3.2.5. SigningHub

SigningHub [24] is a collection of services developed by Ascertia, enabling digital signature creation in different scenarios:

- Remote Signing, by using an HSM or encrypted DB to store signing keys;
- Local Signing, by using signing keys stored on smart cards, USB tokens, or other types of secure hardware or software containers that the end user will manage;
- Mobile Signing, by using signing keys stored in the secure hardware of the end user's mobile device.

SigningHub provides APIs that facilitate seamless integration into specific infrastructure or Connector Apps for SharePoint, Salesforce, and Microsoft Word, enabling users to sign documents from these applications.

3.2.6. Methics

Methics [25] provides a suite of services from which a user can choose to create a Remote Digital Signature system. For mobile-powered remote signing infrastructure, Methics provides three key services that can be interconnected:

- Kiuru Signature Activation Module (Kiuru SAM)—This enables the creation of digital signatures while enforcing the use of private keys only when the user requests such an action.
- Kiuru Mobile Signature Service Provider (Kiuru MSSP)—This service orchestrates the interconnection of several services in the remote signature environment, enabling the creation of AdESs and QESs.
- Alauda PBX app—This mobile application integrates several secure technologies developed by Methics:
 - The Alauda PBX protocol, which creates an end-to-end encrypted connection between the mobile device and the SAM server;
 - Mobile ID, which facilitates the registration of users in the system;
 - Zero-knowledge proofs for user authentication, combined with their proprietary SRP6 protocol for authentication and key exchange;

- The Split Key mechanism for storing the encrypted private key of a user by sending a part to be stored in the SAM database and the other part to be stored on the user's mobile device.

3.3. Assuring Sole Control over the Remote Private Keys

As stated in Article 26 of the eIDAS Regulation [1], for an electronic signature to be recognized as qualified, it needs to be obtained under the sole control of the user. This requirement is fulfilled for local signatures by using a QSCD physically owned by the user. To ensure sole control with a high degree of confidence, a second factor of authentication is needed: the most used solution is a PIN or password known only by the user themselves. In this scenario, sole control over the private key is relatively easy to ensure.

For a remote QES, where the private key is stored server-side, in the TSP environment, ensuring the user's sole control over the private key demands more sophisticated mechanisms. As mentioned in Section 3.1, the most comprehensive standards targeting remote QESs are the CEN standards [15,26,27].

A common requirement between local and remote QESs is the usage of a QSCD for private key generation, storage, and usage. Since the QSCD is not physically owned by the user, ensuring sole control requires additional authorization mechanisms. For example, not even the TSP, which actually stores the private key, can obtain a signature via the user's private key. As it is applied for serving multiple users, the QSCD in a remote QES environment is actually an HSM.

To the best of our knowledge, the only method for ensuring sole control over the private key published in the research literature, compliant with eIDAS Regulation, is that adopted by the Austrian government [28–31]. In summary, the main techniques used to ensure sole control are the following:

- The key pair is generated inside the HSM; therefore, the private key is protected by the QSCD.
- Actual key pair and certificate creation is conducted by the actual user, and not by a TSP employee.
- When creating the key pair and certificate, the user sets a PIN that must remain secret.
- The private key is encrypted using the HSM's master key and the user's PIN. The protected form of the key is then exported from the HSM and stored on a protected server, outside the HSM.
- Every time a signature is requested, the system tries to decrypt the private key using the PIN code inserted by the user for that specific signature request.

Of course, the process is more complex and involves more steps, especially regarding user identification, user enrollment, CSR generation, and certificate generation by the Certification Authority.

Storing encrypted private key material outside the HSM is a well-established practice for most HSMs available for security and cost reasons. Since the export of the key material is carried out only in a wrapped form, it is encrypted with a master key generated inside the HSM. This is considered to have the same level of security as storing it inside the HSM. Also, by activating FIPS 140-2 level 3 or Common Criteria (CC) EAL 4+, private keys cannot be exported as clear text from the HSM.

In the following paragraphs, we will present the Austrian eID solution [28–32] as a case study. First of all, a notable fact is that to ensure the user's sole control over the private keys, the enrollment step should be carefully designed. Therefore, an overview of the user's journey from the beginning to actually obtaining a signature is presented as follows:

1. User identification in person or using a remote system.
2. User registration into the system by a trusted agent (the registration is signed by the agent).
3. Certificate activation proceeds as follows:
 - (a) The user authenticates with two factors.

- (b) The user chooses a secret PIN.
 - (c) The user triggers key pair generation.
 - (d) The CSR is generated.
 - (e) The CSR is signed by the CA.
 - (f) The user receives the qualified certificate.
4. Document signing proceeds as follows:
- (a) The user triggers document signing.
 - (b) User fulfills 2FA (via SMS or mobile app).
 - (c) Employing the user's PIN and the MasterKey, the SSA can unwrap the private key inside the HSM. Then, the hash of the document is signed.
 - (d) The signed hash is embedded in the document by the client application.

Another proposal for a sole control protocol can be found in [33]. In short, the author proposed encrypting the private key inside the HSM, employing a user PIN known only by the user, similar to what the Austrian government employs. Additionally, in this paper, the author proposed encrypting the content to be signed using the public key, so that only inside the HSM, using the associated private key, could the content to be signed be decrypted. The main drawback of this proposal is that it does not comply with working standards and legislation.

3.4. Related Work

This section summarizes the main research published in relation to Remote QESs and eID. Over the last two decades, researchers in this field have had two main interests: (1) technical systems to fulfill the requirements and (2) standards and legislation. Table 1 shows the main literature on server-side signatures. We filtered it for eIDAS-compliant solutions or studies.

Table 1. Related work overview.

Authors	Year	eIDAS-Aware	Main Scope	Comments
Ş. Arseni et al. [6]	2022	Yes	Solution for long-term signature preservation.	Uses blockchain technology for transparency; eIDAS-compliant.
Eric Verheul [34]	2021	Yes	Protocol for sole control for EU ID Wallet.	Very good eIDAS overview; uses a variation of ECDSA.
Ozgun Erdogan et al. [35]	2021	Yes	Analysis on harmonizing eIDAS with Turkey's eID systems.	Survey on eIDAS-compliant server signing solutions.
K. Theuermann et al. [32]	2021	Yes	Improves usability and keeps the same level of security.	Introduces support for biometric authentication.
A. Goransson [36]	2019	Yes	Study on public's perception of eID in Sweden.	Socio-technical approach.
I. Aciobanitei et al. [4]	2019	Yes	Plug-in proposed module for improved transparency of RQESs.	Uses Ethereum blockchain; compliant with ETSI and CEN standards.
I. Aciobanitei et al. [2]	2018	Yes	Proposal for easier adoption of RQESs.	PKCS#11 module for signature in the cloud.
I. Aciobanitei et al. [3]	2018	Yes	Proposal for easier adoption of RQESs.	CNG module for signature in the cloud.
V. Roşca [37]	2017	Yes	Study on the Republic of Moldova's mobile eID.	Identifies the main barriers for a successful adoption.
T. Lenz et al. [38]	2016	Yes	Architectural design for cross-border authorization in the EU.	Uses STORK interoperability framework.

Table 1. Cont.

Authors	Year	eIDAS-Aware	Main Scope	Comments
C. Rath et al. [29]	2015	Yes	2FA using mobile app; improved usability using QR codes.	Describes the Austrian solution.
M. Kubach et al. [39]	2015	Yes	Review of mobile eID deployment in the EU.	Concludes that eID integrations are still underdeveloped.
C. Rath et al. [30]	2014	Yes	An eIDAS-compliant solution for server-side signing.	Not yet production-ready; deployment not flexible.
T. Zefferer [31]	2014	Yes	Challenge–response for 2FA.	Suitable for mobile end user devices; describes the Austrian solution.
W. Kinastowski [33]	2013	No	Simple protocol for sole control.	Encryption of data to be signed via users’ public key.
C. Orthacker et al. [28]	2010	No (too early)	Server signing solution compliant with Austrian signature law.	Describes the Austrian solution; excellent legislative presentation.

In the European space, many countries have adopted eID solutions during the last two and a half decades. Some of the highlights are the following:

- In 1999, Finland was the first to adopt a national eID card [35].
- In 2002, Estonia used the first national eID with digital signatures [35].
- Solutions implemented by Austria and Estonia are appreciated for their ability to handle user mobility [39].
- Austria and Estonia implemented server-based identity and signature solutions [28,31].

Most proposals presented in Table 1 address problems derived from RQES systems. For papers that have approached an entire server-side signature system, we notice a lack of deployment flexibility in the proposed solutions [29]. Two other points not addressed are the ease of client integration and the lack of new deployment technologies, like Docker and Kubernetes.

4. Proposed Architecture

The architecture of a remote signature system is essential for its security, efficiency, and compliance with standards. This chapter presents an in-depth look at a comprehensive and privacy-aware remote signature architecture designed for qualified electronic signatures.

4.1. Overall Architecture

The system architecture is presented in Figure 2. In the lower part of the diagram, we represent modules specific to a remote signature system, compliant with the ETSI and CEN standards, as presented in Section 3.1. In the upper part of the diagram, we present the modules needed for a well-rounded solution for managing documents in electronic form.

In this architecture, an important point to note is that all interaction between the TSP and client environments is carried out using standard protocols.

An HSM is used to fulfill the role of the QSCD. In this particular case, we use a nShield HSM, but this vendor is not mandatory for the scenario. The requirements it should fulfill are mentioned in the CEN standard [26], and it is run using a FIPS 140 II, level 3 compliance. For integration with the SAM, a PKCS#11 interface must be provided. However, most vendors offer such an interface for their SCDs; therefore, replacing the HSM in this architecture should be an easy task.

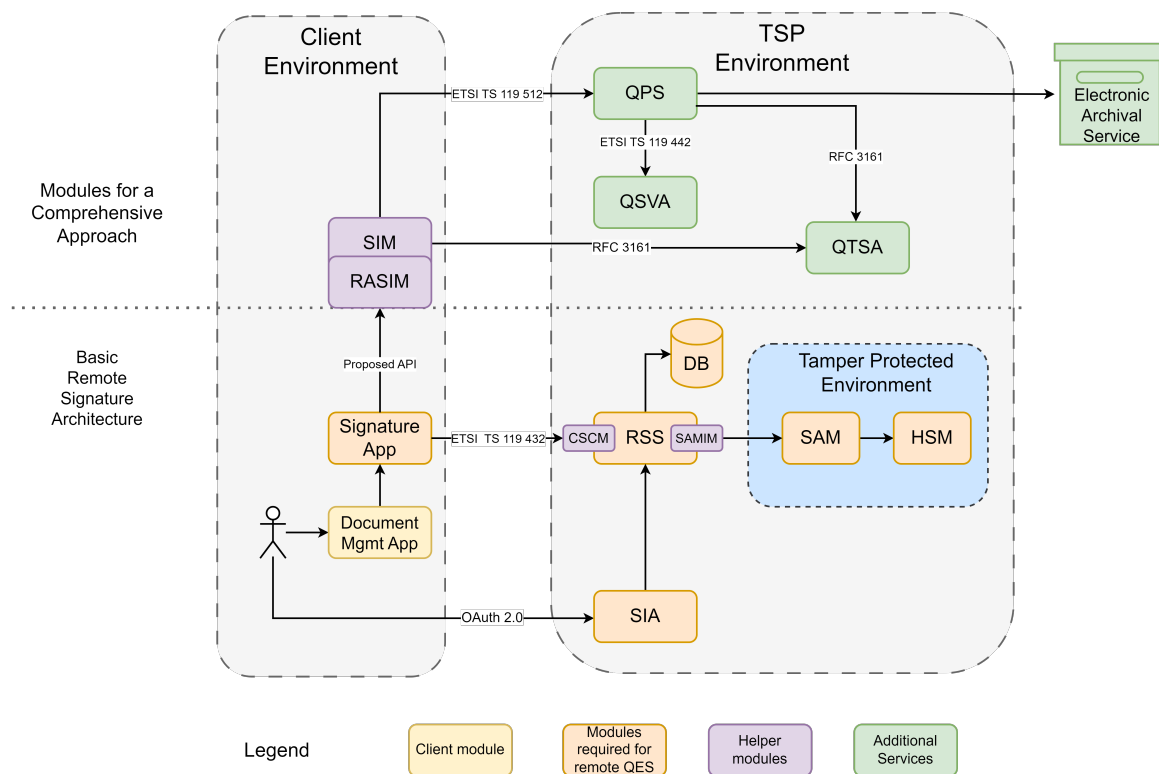


Figure 2. Solution Architecture.

The **SAM** is a module specific to Remote QES infrastructure. Since the user does not physically own a token to protect their private key, there is a need to ensure the user's sole control over the private key stored in the TSP environment. The SAM is responsible for this job. The SAM must be protected by the same means as the QSCD is protected [27], and connection to the HSM must be mutually authenticated. The SAM needs to be compliant with SCAL2; therefore, it is mandatory for QESs.

The **RSS** is the main service of the remote signature solution. This service needs to assure SCAL2; therefore, it needs to be integrated with the SAM via the SAMIM. User authentication information is stored in a database for persistence. The RSS exposes a custom protocol, but in order to comply with the CSC protocol, we developed the CSCM, which provides the CSC Protocol.

The **SIA** is the service that interacts with the user to receive access credentials and create the SAD authorization token that will be checked in the SAM module for signature authorization. Service functionality is achieved through the OAuth 2.0 authorization protocol. The Signature App is a client prototype application tasked with bridging the CASD, RSS, and SIA for the user. This application is implemented through a client-side web application.

The **SIM** (Signature Integration Module) is a component installed in the infrastructure (on-premise) of those who consume Remote Electronic Signature services. The component needs as input the signed hashes of the documents, the documents, and the certificate used to obtain the signature. Using these pieces of information, the SIM will apply the signature in various formats, according to the standards in force: CADES, PAdES, XAdES, and ASiC. This module is implemented in Java; therefore, it presents constraints regarding technologies for client integrations.

The **RASIM** (REST API SIM) uses a REST API to handle advanced signature formats. Thus, by integrating this module, acquirers have the flexibility to implement their signature applications using any technology and without having to carry out the difficult and rigid management of advanced signature formats. The RASIM provides a custom-designed, yet simple protocol.

The **QTSA** is not a mandatory component within a basic RQES system, but it is necessary to be able to obtain signatures on time profiles (T, LT, LTA). Qualified timestamps transparently applied by signature applications represent a popular business request by clients.

The **QPS** is used to preserve the validation status of electronic signatures in the long term. This service will periodically augment the signed document/signatures so that the status of the signatures is extended for long periods of time (e.g., 50 years). The QPS service supports integration with an Electronic Archival Service. For the initial validation of electronic signatures, the QPS delegates this task to a validation authority.

The **Electronic Archival Service** supports customers in the electronic archiving of documents. This must be carried out for electronic documents, according to the legislation in force in each member state. Given the proposed integration method, customers can access this facility in a completely transparent way: the user only signs the document, but it is electronically archived, and the signature status is preserved in the long term automatically.

The **Blockstamp** is a component implemented using Ethereum smart contracts. In this manner, we can store hashes of the preserved objects directly on a blockchain, which ensures an immutable collection. This component is non-standard since it does not appear in a classical PKI. It is optional for each client to activate Blockstamp integration. This module offers a second pseudo-timestamp, besides the one the QPS obtains from the QTSA.

The **Document Mgmt App** client application is employed by the user to manage data or documents. Thus, it is desired that in order to sign a document, the user continues to use the application with which they are already accustomed and does not need to perform intermediate steps in the flow of signing documents.

4.2. Sim Description

To achieve a privacy-aware signature system, documents do not leave the client infrastructure. The proposed CSC protocol takes this into account and only provides a method for signing the hash of the document (PKCS#1 v1.5), not the document itself.

In short, the SIM contains a set of five functions for each of the following advanced signature formats: PAdES, CAdES, XAdES, and ASiC. For example, in the context of PAdES, the SIM provides the following methods:

- **ComputeMessageDigest**—This applies a hash function to the PDF document and the signed attributes. It is used to obtain the digest of the document that would be sent to the RSS. This method is necessary since obtaining the hash to be signed is not a trivial task, as it must comply with the ETSI PAdES formatting standards [40,41]. In short, the hash function is computed on a byte range of the document, together with the signing certificate and other signed attributes.
- **Sign_PAdES_B**—This uses the PDF document, the client certificate, and the signed hash received from the RSS to obtain the PAdES B-B signature.
- **Sign_PAdES_T**—This uses the PDF document, the client certificate, and the signed hash received from the RSS to obtain the PAdES B-T signature. This endpoint also calls the QTSA.
- **Sign_PAdES_LT**—This uses the PDF document, the client certificate, and the signed hash received from the RSS to obtain the PAdES B-LT signature. To achieve the LT level, this endpoint must obtain the revocation status of the signing certificate and the certification chain.
- **Sign_PAdES_LTA**—Thus uses the PDF document, the client certificate, and the signed hash received from the RSS to obtain the PAdES B-LT signature. To achieve the LTA level, this endpoint must obtain the revocation status of the signing certificate and the certification chain. Then, it needs to apply a document timestamp, as described in Section 5.4.3 from ETSI EN 319 142-1 [40].

Our implementation of the SIM was realized starting from the DSS open-source library <https://github.com/esig/dss> (accessed on 1 February 2024). Section 4.5 describes in detail how one may implement PAdES signatures with an external signing device.

4.3. Rasim Description

The RASIM was developed using the SIM as a base and provides a REST API so integrations can be achieved from any client application. Since the SIM is implemented in Java, the only language for the client RQES application is Java. To fulfill its requirements, the RASIM provides two methods for each format:

- **pades/computeHash** obtains the hash of a PDF document and additional information. This hash is to be transmitted to the CSC server.
- **pades/generateSignature**, based on the PDF document and the signed hash, generates the PAdES signature.
- **cadef/computeHash** obtains the hash of a non-formatted document and additional information. This hash is to be transmitted to the CSC server.
- **cadef/generateSignature**, based on a non-formatted document and signed hash, generates the CAdES signature.
- **xades/computeHash** obtains the hash of an XML document and additional information. This hash is to be transmitted to the CSC server.
- **xades/generateSignature**, based on the XML document and the signed hash, generates the XAdES signature.
- **asic/computeHash** obtains the hash of a set of non-formatted documents and additional information. This hash is to be transmitted to the CSC server.
- **asic/generateSignature**, based on the set of documents and the signed hash, generates the ASiC signature.

Any of the generateSignature endpoints also has a parameter to describe the signature level (B, BT, LT, or LTA). In this way, each generateSignature endpoint defers to the four methods of the SIM. All methods are HTTPS POST requests with a JSON body. For a better understanding of the RASIM, two of the eight endpoints are described below.

pades/computeHash Request has the following body:

```
{
  "document": "file content base64", //mandatory
  "cert": "certificate base64", //mandatory
  "pades_signature_details": //optional
  {
    "signature_level": "lta", //options: b, bt, lt, lta, default=b
    "visible": true, //optional, default = false
    "page_nr": 1, //optional, default = 1
    "position_x": 100, //optional, default = 100
    "position_y": 100, //optional, default = 100
    "width": 200, //optional, default = 200
    "height": 200, //optional, default = 100
    "background_image": "base64 png or jpg image" //optional
  }
}
```

pades/computeHash Response returns the hash to be signed, which is base64-encoded:

```
{ "hash": "80IDYnSyMfma5u93zNWK1XKEuiVcl47KzGWSvgTJRJI=" }
```

pades/generateSignature Request has the following body:

```
{
  "document": "file content base64", //mandatory
  "cert": "certificate base64", //mandatory
  "signed_hash": "base64 PKCS#1 hash", //mandatory
  "pades_signature_details": //optional
  {
    "signature_level": "lta", //options: b, bt, lt, lta, default=b
    "visible": true, //optional, default = false
  }
}
```

```

    'page_nr':1,//optional, default = 1
    'position_x':100,//optional, default = 100
    'position_y':100,//optional, default = 100
    'width':200,//optional, default = 200
    'height':200,//optional, default = 100
    'background_image':'base64 png or jpg image'//optional
  }
}

```

pades/generateSignature Response returns the PAdES document, which is base64-encoded:

```

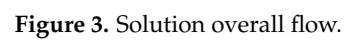
{
  'signed_document': 'base64 pades document'
}

```

4.4. Relevant Flows

For a better understanding of our proposed flow, Figure 3 displays a general view of the application flow with a detailed overview of the credential authorization step. For obtaining a signed document, our system uses the following steps:

- **Initial Interaction**—The user uploads the file to be signed in the browser.
- **Service Authorization**—The user needs to be authorized to access the CSC Service. For this, we implement an authorization flow specific to the OAuth 2.0 protocol. The user sends the username and password directly to the SIA module, so that it cannot be found by any third-party applications. After this step, the Signature App obtains a bearer token used for service authorization. With this token, the Signature App can now call CSC methods like credentials/list and credentials/info.
- **Select Signing Certificate**—The user chooses the certificate and private key to be used for signing. Since a user with an account might have multiple certificates issued on the platform, credential/list might return multiple credentialIDs. For each credentialID, the Signature App calls credential/info. Then, the user is prompted with the certificate information so that they can choose one of them.
- **Obtain Hash to be signed**—The Signature App uses the RASIM to obtain the hash of the document, with respect to the signature format. This step is realized in one simple REST API call.
- **Credential Authorization**—The Signature App needs to obtain the SAD before calling the signatures/signHash CSC endpoint. This step is realized by a set of redirects specific to the OAuth 2.0 protocol. This step also needs the hash to be signed, since the SAM authorizes the actual hash to be signed. The exact same hash needs to be sent for the signatures/signHash endpoint. The user needs to insert the second factor of authentication. The SAM verifies it and then issues the access_code. Using this code, the Signature App obtains the SAM from the SIA using the oauth2/token method.
- **Actual Document Signing**—In this step, the document is finally signed. Still, this process requires two main steps: calling the signature/signHash method and calling generateSignature from the RASIM. After this step, the signed document is ready and can be downloaded by the user.



4.5. PAdES Implementation Details

To save space in this article, we detail only how an external signature might be implemented for PAdES using the open-source library DSS. An external signature means that the private key is not available from the client application; therefore, three main steps need to be carried out:

- Obtain the hash to be signed.
- Sign the hash with the external source.
- Incorporate the signed hash into the file.

Of all the formats implemented in this project (PAdES, CAdES, XAdES, and ASiC), a PAdES external signature has the most complete support in DSS. Another PAdES-specific detail is that for this format, options regarding the visual appearance of the signature need to be provided and processed accordingly.

The most relevant modules used by DSS for external PAdES signatures are as follows:

- *eu.europa.esig.dss.pades.signature.ExternalCMSService*—This component offers the functionality of composing a CMS signature based on a PKCS#1 signature.
- *eu.europa.esig.dss.pades.signature.PAdESWithExternalCMSService*—This service offers support for creating a PAdES signature based on a CMS signature.
- *eu.europa.esig.dss.service*—This implements communication methods for various services provided on the Internet (TSP, CRL, OCSP, HTTP).
- *eu.europa.esig.dss.model*—This includes models and data structures used for managing fundamental entities for signatures (x509 certificates, PKCS#1 signature format, etc.)
- *eu.europa.esig.dss.enumerations*—This dictionary component stores static information, like the names of cryptographic algorithms and baseline profile types.

5. Deployment Scenarios

This section describes the main modalities for deploying our proposed solution. Those four deployment scenarios were developed after business research.

5.1. Signature App

As shown in Figure 4, in this scenario, the client has nothing to implement, since signatures can be applied solely using the Signature App, provided by the TSP.

Pros:

- No development on client side.

Client Responsibilities:

- Install the SIM and RASIM on-premise.
- Install the Signature App on-premise.

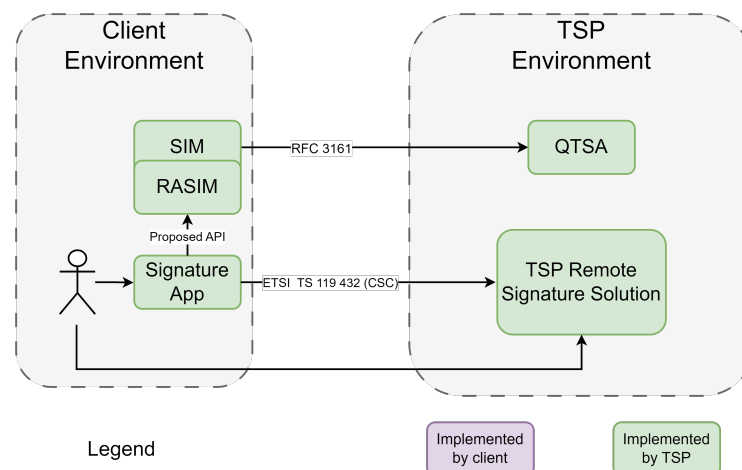


Figure 4. Deployment with Signature App.

5.2. RASIM Integration

In this scenario, as shown in Figure 5, the client is able to use their document management application as usual. The signature can be applied after integration with the signing server via the CSC protocol and with the RASIM via a custom API.

Pros:

- A customized Document Mgmt App is employed, corresponding to the design and user experience requirements of the client.
- The client does not need to implement advanced signature formatting (PAdES, CAdES, XAdES, ASiC).

Client Responsibilities:

- Installation of SIM & RASIM on-premise. Usually, these modules are deployed using Docker technology.
- Implement RASIM integration to obtain advanced signature formats.
- Implement the OAuth 2.0 flow with the SIA to obtain access tokens and the SAD.
- Implement the CSC protocol on the client side to obtain the certificate and signed hash.

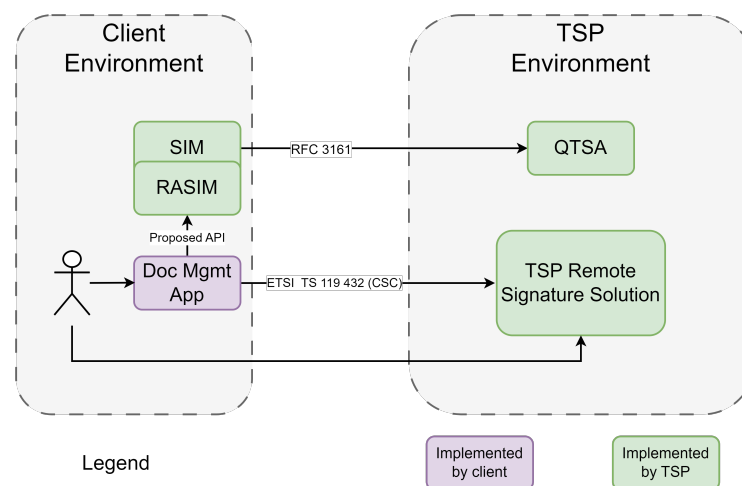


Figure 5. Deployment for RASIM integration.

5.3. SIM Integration

In this scenario, as shown in Figure 6, the client is able to use their document management application as usual. The signature can be applied after integration with the signing server via the CSC protocol. For this deployment scenario, the Doc Mgmt App needs to integrate directly with the SIM; therefore, it needs to be implemented in Java.

Pros:

- A customized Document Mgmt App is employed, corresponding to the design and user experience requirements of the client.
- The client does not need to implement advanced signature formatting (PAdES, CAdES, XAdES, ASiC).

Client Responsibilities:

- Call SIM methods to obtain advanced signature formats.
- Implement the OAuth 2.0 flow with the SIA to obtain access tokens and the SAD.
- Implement the CSC protocol on the client side to obtain the certificate and signed hash.

Constraints:

- The Doc Mgmt App needs to be a Java application.

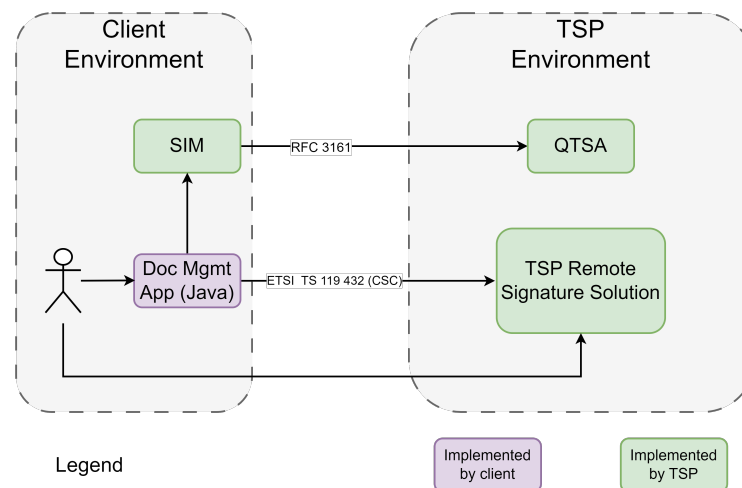


Figure 6. Deployment for SIM integration.

5.4. Direct Integration

For a direct integration, as shown in Figure 7, the client does not use any helper modules. They have control over the implementation, but need to take care of functionalities for managing the following: CSC communication, OAuth 2.0 flow, Advanced Electronic Signature formatting, and timestamp client protocol.

Pros:

- A customized Document Mgmt App is employed, corresponding to the design and user experience requirements of the client.
- Control over the implementation of client-side components.

Client Responsibilities:

- Implement the OAuth 2.0 flow with the SIA to obtain access tokens and the SAD.
- Implement the CSC protocol on the client side to obtain the certificate and signed hash.
- Implement the Timestamp client.
- Implement Advanced Electronic Signature formatting, as stated in the ETSI standards [11–14].

Constraints:

- The Doc Mgmt App should be a web application so that it implements a proper authentication and authorization flow.

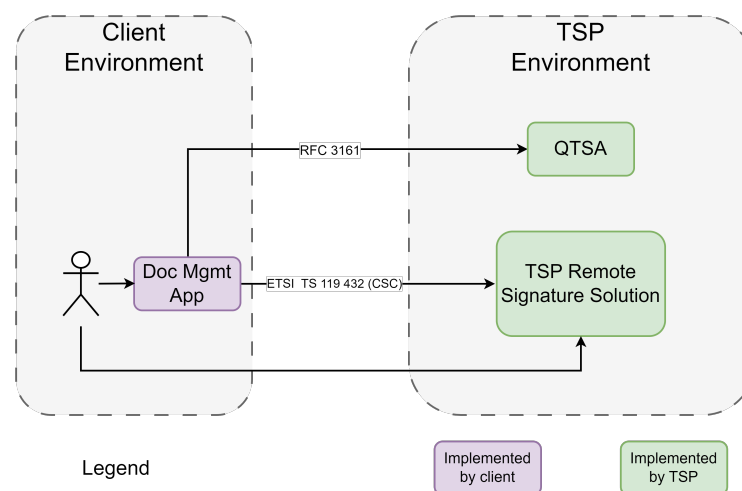


Figure 7. Deployment with direct integrations.

6. Discussion

The research presented in this article offers a comprehensive exploration of remote Qualified Electronic Signature (QES) systems, focusing on their security, ease of integration, and interoperability. This discussion chapter delves deeper into these aspects, examining the challenges, advancements, and potential solutions in the context of remote document signing services.

6.1. System Testing

To validate our proposed RQES system after its development and integration phases, we designed a set of tests that verified the functionalities of the system. Table 2 contains a subset of these tests that focused on the main functionalities of the system: the remote signing of documents, authorization for a signing flow, and the validation of an electronic signature. As it can be observed in the table, the majority of tests covered the signing flow, since it is the main task of the system, and the system is required to always produce compliant electronic signatures no matter how the system is deployed in the targeted infrastructure. More information regarding these deployment scenarios is presented in Section 5.

Table 2. Selection of applied tests.

Test nr.	Input	Receiving Component	Signature Level	Preserve Long Term?	Result
1	PDF not signed	Signature App (PAdES)	B	No	Signed PDF, level B-B (verified in Adobe)
2	PDF not signed	Signature App (PAdES)	B-T	No	Signed PDF, level B-T (verified in Adobe)
3	PDF not signed	Signature App (PAdES)	B-LT	No	Signed PDF, level B-LT (verified in Adobe)
4	PDF not signed	Signature App (PAdES)	B-LTA	No	Signed PDF, level B-LTA (verified in Adobe)
5	Signed PDF	Signature App (PAdES)	B	No	PDF with 2 signatures (verified in Adobe)
6	Multiple Files	Signature App (ASiC)	B	No	ASiC-S level B (verified on EC demo website)
7	Word file	Signature App (CAdES)	B	No	CAdES level B (verified on EC demo website)
8	XML file	Signature App (XAdES)	B	No	XAdES level B (verified on EC demo website)
9	Base64 PDF	RASIM (computeHash)	B	No	Hash of the PDF file (base64-encoded)
10	Base64 PDF + signed hash	RASIM (generateSignature)	B	No	Signed PDF, level B-LT (base64-encoded)
11	PDF content	SIM (ComputeMessageDigest)	-	-	Hash of the PDF file (byte array)
12	PDF content + signedHash	SIM (Sign_PAdES_B)	-	-	Content of a signed PDF, level B-B
13	Hash to be signed	Auth. Server (authorize+token)	-	-	A JWT Token - the SAD (Signature Authorization Data)
14	Hash to be signed + SAD	RSS (signatures/signHash)	-	-	Signed Hash of the document (PKCS#1 v1.5)

Table 2. Cont.

Test nr.	Input	Receiving Component	Signature Level	Preserve Long Term?	Result
15	Signed PDF, Base64	QSPA	-	-	Base64-encoded XML validation report (MainIndication: PASSED)
16	PDF not signed	Signature App	B	Yes	Signed PDF, level B PDF with LTA in Preservation portal

6.2. Security

Security is a paramount concern in the realm of electronic signatures, especially considering the legal and personal implications of document signing. The security of an RQES system is mainly concentrated on the TSP side, involving advanced cryptographic techniques and stringent identity verification processes. Essential details addressed by the relevant standards are the management user's private key; the signing protocol; and, in some instances, document privacy.

Since such a system for RQESs becomes more and more complex, with services relying on each other, one may infer that this complexity could be a source of vulnerabilities at various points. Besides improving the deployment process and reducing component coupling intricacy, the usage of containerization technology offers the whole infrastructure an additional layer of virtual protection. At the same time, it makes room for vulnerabilities and exploitation [42].

While public institutions and private companies continue to migrate a huge majority of their processes into the virtual space, this will increase the rate of digital transactions, but also the motivation of bad actors. A multitude of reported cyber-attacks have had disruptive effects on organizations all over the world [43]. To the best of our knowledge, there is no publicly available detailed research on the security of RQES infrastructure. With papers like this, we aim to raise the research interest in the security of QES infrastructure. We also aim to conduct in-depth research regarding different attack scenarios, specific to RQES architectures.

For local QESs, the two authentication factors used are a physical cryptographic token and, usually, a PIN to access the private key. In comparison, for an RQES, the second factor of authentication is enforced by the SAM. A similarity would be that for both paradigms, private keys are stored and managed using a QSCD. Still, it is worth noting that the QSCD for RQESs involves HSMs, which are considered more robust and have more complex key management capabilities.

Another topic related to the security of an RQES system is the signing protocol, which may or may not support document privacy. As presented in previous sections, we opted for the proposed CSC protocol, which only supports sending the hash to the TSP, and not the entire document. Still, this feature comes with some challenges, mainly regarding the implementation of the signature applications, since usually open-source cryptographic libraries do not support remote (external) signing.

Although traditional cyber threads focus on exploiting known information about systems, both in terms of software and hardware, a different source of attack vectors would be that of side-channel attacks. This approach involves the usage of observable information (e.g., electricity usage, electromagnetic radiation, and the duration of certain operations) to infer signature creation data (private key material) [44].

Moreover, with the advent of quantum computers [45], classical cryptographic systems relying on the mathematical complexity of algorithms like RSA and ECC are prone to becoming insecure [46]. Thus, we think that further research should be encouraged regarding the future need for upgrading RQESs with quantum key distribution technology [47,48] and post-quantum cryptography [49].

6.3. Interoperability

In this paper, we approached interoperability from two main perspectives: signature formats and standard communication protocols between various RQES components.

The main open problem regarding interoperability is the Electronic Archival Service. In the EU, member states have separately adopted legislation regarding electronic archives; therefore, at present, we do not have a unified framework or regulation for electronic archives. As a counterexample, there was no legislation for signature preservation services; therefore, the eIDAS Regulation and ETSI standards could be imposed.

All the interfaces provided by the TSP with the outside architecture are standard. Modules offering standard protocols are RSS, QPS, and QSVA. In this manner, we manage to fulfill various client requirements regarding which modules they need. For example, our proposed architecture could support a scenario where clients sign documents using the RSS and use another TSP's service for long-term preservation or signature validation.

Advanced Electronic Signature formats (PADES, CAdES, XAdES, and ASiC) are used to enforce interoperability between various signature creation and validation applications. These formats are the most important standards to ensure interoperability for electronic commerce in the EU space. Of course, these signature formats are more than just formats and are focused on security, ensuring integrity and authenticity.

In our proposed architecture, we have two implemented modules (SIM and RASIM) that do not provide a standard interface, because there are no standards for this use case. Note that these two helper modules are not mandatory for a secure RQES infrastructure, but they come with easier and faster integration. Also, the RASIM is designed as a stateless REST API with simple requests and responses, which should be quite simple to use.

6.4. Ease of Integration

As we noted from experience and multiple client integrations, this process might take a long time (many months), since it involves development from the client, as well as user experience changes. To support multiple integration scenarios, we designed a modular architecture with client helper modules.

In Table 3, we emphasize the integration specifics of each deployment scenario presented in Section 5, enabling final users to choose the integration type that is in accordance with their needs.

Table 3. Selection of applied tests.

Deployment Scenario	Authentication and Authorization	Signature Flow		
		Component Integration	Integration Level	Required Standard Compliance
1. Signature App	OAuth 2.0	None	None (ready-to-use WebUI)	None
2. RASIM		RASIM	API-level	Proprietary API (Section 4.3)
3. SIM		SIM	Code-level	Proprietary classes in Java
4. Direct		CSCM	API-level	ETSI TS 119 432 [16] and RFC 3161 [50]

This paper also presented a state-of-the-art method regarding current RQES integrations in various digital platforms. Using the standard signing protocols helps the ease of integration. Another important factor is that by using RQESs, the possibility to sign from a web application is now straightforward. As before, with local signatures, signatures from browsers are more difficult to obtain and are susceptible to security breaches.

7. Conclusions

The proposal of this paper comprises a comprehensive approach to potential applications for document signing, specifically with remote private keys. The exploration of various standards, technologies, and deployment scenarios highlighted the robust nature of the proposed architecture and its adaptability to diverse user requirements and technological environments. More specifically, we addressed the following:

- An analysis of relevant standards and legislation, ensuring that the proposed solution is adoptable, secure, and interoperable with other electronic signature solutions.
- A comprehensive approach to electronic signature systems, taking into account various problems such as digital signature validation, digital signature status preservation, and electronic archival.
- User-centric design through the various deployment scenarios discussed. These offer flexibility for user requirements and ease of use, which is critical for widespread adoption.

The main result of this work was a flexible on-premise module that provides the ability to sign, validate, and preserve documents, with minimal integration effort. The ability to remotely sign documents in a secure manner and with legal compliance is crucial for digital transaction adoption. This enhances a company's efficiency while reducing the need for physical documentation.

This study paves the way for a more connected and efficient digital future, where remote document signing plays a pivotal role in various domains, further bridging the gap between companies and countries.

Future Work

A primary direction for subsequent research involves the formal security validation of various modules and protocols within the RQES system. This includes using formal analysis tools such as AVISPA or AVANTSSAR to formally check the security of the used protocols. To the best of our knowledge, protocols like CSC API have not been formally checked.

Another direction of study for enhanced security required detailed research on how the RQES system could be compromised at different points. This involves identifying potential weak spots in the system, implementing various attack scenarios, and developing countermeasures to prevent such breaches.

Author Contributions: Conceptualization, M.T. and I.A.; methodology, M.T. and I.A.; software, E.B.; validation, Ș.-C.A.; formal analysis, Ș.-C.A. and I.A.; investigation, E.B. and I.A.; writing—original draft preparation, I.A. and E.B.; writing—review and editing, I.A., Ș.-C.A., E.B. and M.T.; visualization, Ș.-C.A. and I.A.; supervision, M.T. and I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Romanian National Authority for Scientific Research and Innovation (ANCSI-UEFISCDI) under the project PN-III-P2-2.1-PTE-2021-0655 (ctr. no. 109PTE/2022), Interoperable system based on qualified components for remote electronic signature creation services (CISRES), within PN III.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The results published in this paper were produced during the research project: PN-III-P2-2.1-PTE-2021-0655. In this project, Military Technical Academy and CertSIGN S.A. were partners. In the collaboration contract, Intellectual property rights are addressed and well-defined for the two parties. Author Mihai Togan was employed by the company CertSIGN S.A. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

QES	Qualified Electronic Signature
RQES	Remote Qualified Electronic Signature
SAM	Signature Activation Module
SSA	Server Signing Application
QPS	Qualified Preservation Service
QSVA	Qualified Signature Validation Authority
QTSA	Qualified TimeStamp Authority
CSC	Cloud Signature Consortium
RSS	Remote Signature Service
HSM	Hardware Security Module
QSCD	Qualified Signature Creation Device
SIM	Signature Incorporation Module
RASIM	REST API Signature Incorporation Module
TSP	Trust Service Provider
SCAL	Sole Control Assurance Level
SIA	Signer Identification and Authorization
CSR	Certificate Signing Request

References

1. European Commission. *Regulation (EU) No 910/2014 of the European Parliament and of the Council on Electronic Identification and Trust Services for Electronic Transactions in the Internal Market and Repealing Directive 1999/93/EC*; Official Journal of the European Union: Luxembourg, 2014.
2. Aciobanitei, I.; Leahu, L.; Pura, M. A PKCS#11 Driver for Cryptography in the Cloud. In Proceedings of the 2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania, 28–30 June 2018. [[CrossRef](#)]
3. Aciobanitei, I.; Urian, P.D.; Pura, M. A Cryptography API: Next Generation Key Storage Provider for Cryptography in the Cloud. In Proceedings of the 2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania, 28–30 June 2018. [[CrossRef](#)]
4. Aciobanitei, I.; Dedita, V.; Pura, M.-L.; Patriciu, V.-V. SABRES—A Proof of Concept for Enhanced Cloud Qualified Electronic Signatures. In Proceedings of the 2020 13th International Conference on Communications (COMM), Bucharest, Romania, 18–20 June 2020. [[CrossRef](#)]
5. Ruica, E.C.; Pura, M.L.; Aciobanitei, I. Implementing cloud qualified electronic signatures for documents using available cryptographic libraries: A survey. In Proceedings of the 2020 13th International Conference on Communications (COMM), Bucharest, Romania, 18–20 June 2020; IEEE: New York, NY, USA, 2020.
6. Arseni, Ș.-C.; Togan, M.; Aciobăniței, I.; Bureacă, E.; Coca, M. LTPS—Service for long-term preservation of digital signatures. In Proceedings of the 2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Ploiesti, Romania, 30 June–1 July 2022; pp. 1–6. [[CrossRef](#)]
7. ETSI TS 119 511, v1.1.1; Electronic Signatures and Infrastructures (ESI); Policy and Security Requirements for Trust Service Providers Providing Long-Term Preservation of Digital Signatures or General Data Using Digital Signature Techniques. ETSI: Sophia Antipolis, France, 2019.
8. ETSI TS 119 512, v1.1.1; Electronic Signatures and Infrastructures (ESI); PAdES Digital Signatures; Protocols for Trust Service Providers Providing Long-Term Data Preservation Services. ETSI: Sophia Antipolis, France, 2019.
9. Congress of the United States of America. *Electronic Signatures in Global and National Commerce Act*; Public Law 106–229. June 2000; Congress of the United States of America: Washington, DC, USA, 2000.
10. The European Parliament and the Council of the European Union. *Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for Electronic Signatures*; Official Journal of the European Communities: Brussels, Belgium, 1999; pp. 12–20. [[Google Scholar](#)]
11. ETSI EN 319 122 v1.2.1; Electronic Signatures and Infrastructures (ESI); CAdES Digital Signatures; Part 1: Building Blocks and CAdES Baseline Signatures. ETSI: Sophia Antipolis, France, 2021.
12. ETSI EN 319 132 v1.1.1; Electronic Signatures and Infrastructures (ESI); XAdES Digital Signatures; Part 1: Building Blocks and XAdES Baseline Signatures. ETSI: Sophia Antipolis, France, 2016.
13. ETSI EN 319 142 v1.1.1; Electronic Signatures and Infrastructures (ESI); XAdES Digital Signatures; Part 1: Building Blocks and CAdES Baseline Signatures. ETSI: Sophia Antipolis, France, 2016.
14. ETSI EN 319 162 v1.1.1; Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC); Part 1: Building Blocks and ASiC Baseline Containers. ETSI: Sophia Antipolis, France, 2016.
15. CSN EN 419 241-1; Trustworthy Systems Supporting Server Signing—Part 1: General System Security Requirements. European Standards Organizations: Brussels, Belgium, 2018.

16. ETSI TS 119 432, v1.1.1; Electronic Signatures and Infrastructures (ESI); Protocols for Remote Digital Signature Creation. ETSI: Sophia Antipolis, France, 2019.
17. Cloud Signature Consortium. *Architectures and Protocols for Remote Signature Applications*; Cloud Signature Consortium: Brussels, Belgium, 2023.
18. Digital Signature Service Core Protocols, Elements, and Bindings Version 2.0. Available online: <https://docs.oasis-open.org/dss-x/dss-core/v2.0/dss-core-v2.0.html>, (accessed on 10 December 2023).
19. Cryptomathic Signer. Product Sheet. Available online: https://www.cryptomathic.com/hubfs/Documents/Product_Sheets/Cryptomathic_Signer_-_Product_Sheet.pdf (accessed on 14 December 2023).
20. Cryptomathic White Paper. Guidance on Achieving Qualified Remote eSigning. Available online: <https://www.cryptomathic.com/whitepapers/eidas-compliant-remote-esigning> (accessed on 14 December 2023).
21. bit4id SignCloud. Datasheet. Available online: https://www.bit4id.com/wp-content/uploads/2021/12/signcloud_DS_4.0_EN_LQ.pdf (accessed on 10 December 2023).
22. DigitalSign SigningDesk solution. Available online: <https://www.digitalsign.pt/en/pt/signingdesk/> (accessed on 11 December 2023).
23. NextSense Signing Suite. Available online: <https://nextsense.com/signing-suite.nspix> (accessed on 11 December 2023).
24. Ascertia SigningHub. Architecture and Deployment Guide, v1.2.0.0. 2018. Available online: <https://manuals.ascertia.com/SigningHub/8.6/Architecture-Deployment/> (accessed on 10 December 2023).
25. Methics. Mobile Id and Signature Solutions Presentation. 2022. Available online: https://www.methics.fi/wp-content/uploads/2022/06/Methics_Presentation_2022_brief.pdf (accessed on 12 December 2023).
26. CEN TS 419 241-5; Protection Profiles for TSP Cryptographic Modules—Part 5 Cryptographic Module for Trust Services. European Standards Organizations: Brussels, Belgium, 2016.
27. CEN TS 419 241-2; Trustworthy Systems Supporting Server Signing—Part 2 Protection Profile for QSCD for Server Signing. European Standards Organizations: Brussels, Belgium, 2018.
28. Orthacker, C.; Centner, M.; Kittl, C. Qualified mobile server signature. In Proceedings of the IFIP International Information Security Conference, Brisbane, Australia, 20–23 September 2010; Springer: Berlin/Heidelberg, Germany, 2010.
29. Rath, C.; Roth, S.; Bratko, H.; Zefferer, T. Encryption-Based Second Authentication Factor Solutions for Qualified Server-Side Signature Creation. In Proceedings of the 2015 International Conference on Electronic Government and the Information Systems Perspective, Valencia, Spain, 1–3 September 2015; Springer: Cham, Switzerland, 2015; Volume 9265.
30. Rath, C.; Roth, S.; Schallar, M.; Zefferer, T. Design and Application of a Secure and Flexible Server-Based Mobile eID and e-Signature Solution. *Int. J. Adv. Secur.* **2014**, *7*, 50–61.
31. Zefferer, T. A server-based signature solution for mobile devices. In Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia, Kaohsiung, Taiwan, 8–10 December 2014.
32. Theuermann, K.; Tauber, A.; Lenz, T. Mobile-only solution for server-based qualified electronic signatures. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; IEEE: New York, NY, USA, 2019.
33. Wojciech, K. Digital Signature as a Cloud-based Service. In Proceedings of the Cloud Computing 2013: The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization IARIA 2013, Seville, Spain, 27 January–1 February 2013.
34. Verheul, E. SECDSA: Mobile Signing and Authentication Under Classical Sole Control. *Cryptol. Eprint Arch.* **2021**.
35. Erdogan, O.; Saran, N.A. A survey on server-based electronic identification and signature schemes to improve eIDAS: With a new proposal for Turkey. *PeerJ Comput. Sci.* **2021**, *7*, e734. [[CrossRef](#)] [[PubMed](#)]
36. Göransson, A. Electronic Identification as an Enabling or Obstructive Force: The General Public's Use and Reflections on the Swedish e-ID. Master's Thesis, Linnaeus University, Växjö, Sweden, 2018.
37. Rosca, V. Exploring Barriers to Mobile e-ID Adoption: A Government Perspective on Republic of Moldova Mobile e-ID. Master's Thesis, Umeå University, Umeå, Sweden, 2017.
38. Lenz, T.; Bernd, Z. Towards cross-border authorization in European eID federations. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; IEEE: New York, NY, USA, 2016.
39. Kubach, M.; Leitold, H.; Roßnagel, H.; Schunck, C.H.; Talamo, M. SSEDIC 2020 on Mobile eID. In Proceedings of the Open Identity Summit 2015, Berlin, Germany, 10–11 November 2015.
40. ETSI EN 319 142-1, v1.1.1; Electronic Signatures and Infrastructures (ESI); PAdES Digital Signatures; Part 1: Building Blocks and PAdES Baseline Signatures. ETSI: Sophia Antipolis, France, 2016.
41. ETSI EN 319 142-2, v1.1.1; Electronic Signatures and Infrastructures (ESI); PAdES Digital Signatures; Part 2: Additional PAdES Signatures Profiles. ETSI: Sophia Antipolis, France, 2016.
42. Casalicchio, E.; Iannucci, S. The state-of-the-art in container technologies: Application, orchestration and security. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5668. [[CrossRef](#)]
43. Gohwong, S. The State of the Art of Cryptography-Based Cyber-Attacks. *Int. J. Crime Law Soc. Issues* **2019**, *6*. [[CrossRef](#)]
44. Lou, X.; Zhang, T.; Jiang, J.; Zhang, Y. A Survey of Microarchitectural Side-channel Vulnerabilities, Attacks, and Defenses in Cryptography. *ACM Comput. Surv.* **2021**, *54*, 122. [[CrossRef](#)]
45. Luo, W.; Cao, L.; Shi, Y.; Wan, L.; Zhang, H.; Li, S.; Chen, G.; Li, Y.; Li, S.; Wang, Y.; et al. Recent progress in quantum photonic chips for quantum communication and internet. *Light Sci. Appl.* **2023**, *12*, 175. [[CrossRef](#)] [[PubMed](#)]

46. Pirandola, S.; Andersen, U.L.; Banchi, L.; Berta, M.; Bunandar, D.; Colbeck, R.; Englund, D.; Gehring, T.; Lupo, C.; Ottaviani, C.; et al. Advances in Quantum Cryptography. *arXiv* **2019**, arXiv:1906.01645.
47. Scarani, V.; Bechmann-Pasquinucci, H.; Cerf, N.J.; Dušek, M.; Lütkenhaus, N.; Peev, M. The security of practical quantum key distribution. *Rev. Mod. Phys.* **2009**, *81*, 1301–1350. [[CrossRef](#)]
48. Portmann, C.; Renner, R. Security in quantum cryptography. *Rev. Mod. Phys.* **2022**, *94*, 025008. [[CrossRef](#)]
49. Quantum—Safe Cryptography—Fundamentals, Current Developments and Recommendations. Available online: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.pdf?__blob=publicationFile&v=6 (accessed on 1 November 2023).
50. Network Working Group. *Internet X.509 Public Key Infrastructure—Time-Stamp Protocol (TSP)*; IETF: Fremont, CA, USA, 2001.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.