Remote Qualified Digital Signatures

Pedro do Vale[1,2]
pedroddvale@tecnico.ulisboa.pt

[1] INESC-ID, IST, Universidade de Lisboa
[2] Multicert

Abstract. Qualified digital signatures (as defined in the EU Regulation 910/2014[1]) are equivalent to hand written signatures. To assure this, they need to be created in a secure environment in order to grant them a strong legal value. The most common approach to accomplish this is with the use of secure devices, to store and use the associated private keys. To avoid users having to carry such devices, this paper presents a solution providing remote qualified signatures on mobile devices. This is accomplished by having a remote service which is able to create qualified signatures for remotely connected users. The implemented solution is able to provide a secure connection between the user device and the qualified area in the server infrastructure, supporting multiple users each with unique private keys managed by a Hardware Security Module. This solution allows for qualified signatures in a scalable and secure manner, providing a commercially and technically attractive solution.

Keywords: Qualified Signatures, Remote signatures, eIDAS, Mobile Devices, Information Security

1 Introduction

Nowadays many services are made available to the user digitally. Services such as e-Commerce and e-Government, allow users to conveniently carry out operations that otherwise would be made in person. These services, usually transactional, require privacy and integrity of sensitive data being transmitted and processed. To fulfil these security requirements, services rely on strong authentication mechanisms, such as the use of digital certificates and on digital signatures to assure authenticity.

Digital signature is a mechanism that relies on asymmetric encryption. It comprehends the knowledge of a private key, only known to its owner, and a public key for the users that wish to verify signatures computed by that owner. A signature is computed by calculating the hash value of the message to be sent and by encrypting that value with the private key. When receiving users get the signature along side the message, they obtain the hash using the public key, and compare it with a calculated hash of the received message. A positive match means that the message is authentic.

Usually public keys are stored in a digital certificate issued by trusted third-parties. This means that services and users need to trust these third-parties. Directive 1999/93/EC[2] states legal requirements for qualified digital signatures and aims to achieve a high level of trust among users and services. This type of digital signature is created inside a Secure Signature Creation Device (SSCD) and is verified using a qualified certificate, that links signature verification data to signatory and confirms its identity. Due to the secure conditions to create

qualified signatures, they are equivalent to handwritten signatures, granting them a strong legal value. One of the requirements for qualified signatures stated in the Directive is that the data necessary to create the signature is kept under sole control of the user, i.e. signature creation data cannot be located at a remote service. In a world where mobile devices usage is always increasing, this means that in order for users to create qualified signatures, users need to create them locally, thus implying the possession of the mentioned SSCD, such as a smartcards, which is not convenient since it is an extra hardware users have to carry.

Later, Regulation No. 910/2014[1], commonly known as eIDAS (IDentification and Authentication Services), replaced the previous Directive and adapted the requirements to current trends. This regulation does allow remote signature creation, as long as only the user can authorise signature creation in its name, i.e. it must not be possible for the service to create a qualified signature without prior authorisation of the user. Since 2014, there has not been many solutions that allow the user to obtain signed data in a way that he does not need to carry a SSCD, which means that there is the opportunity for new ones to be proposed (such as [3][4][5], herein discussed). Usually, remote qualified signature solutions include an Hardware Security Module (HSM) as the SSCD. An HSM is a cryptographic hardware device that because it is an isolated environment and a dedicated device, allows for cryptographic operations, such as signature creation, in a very secure way. Usually these devices, have limited resources, such as storage, meaning that newly created data may be (and usually is) persistently stored externally.

This paper proposes Remote SIGNature (RSIGN), an HSM-based solution capable of producing qualified signatures whilst the user is remotely connected to the service, through its mobile device. For the user interacting with the service, the interaction is transparent, i.e. the HSM appears to be a local one. In this solution, the user must input its authorisation code, i.e. a Personal Identification Number (PIN), to authorise signature creation. One security advantage of this solution is that this authorisation code is never stored in the service and is only known to the user. This is accomplished by protecting users' signing keys using the HSMs master key and the users' authorisation code. To successfully sign data, one has to have access to the protected signing key, the HSMs master key (only known by the HSM itself) and the user's authorisation code (only known to the user).

The entities that compose the service and respond to users' requests contains a frontend server that redirects requests/responses, a backend that deals with authentication, wrapped keys storage, certificates and communicates with the RSIGN backend, located in a restricted area. That Backend communicates with the HSM for cryptographic data creation, such as qualified signatures.

The users interact with the remote service through a mobile application. When they register themselves in the service, a signing private key is generated in their name and they are then able to choose local document(s) to be (remotely) signed. Furthermore, the solution aims at being flexible, components are well separated and offer a REST API for inter-component communication. The HSM is accessible by an PKCS#11 interface which allows for the implementation of RSIGN using different cryptographic devices. This flexibility aims at being a commercial product, since it is being developed and is intended to be deployed at Multicert, an enterprise providing security solutions and certification services.

Document Roadmap

This paper is organised as follows: Section 2 states Related Work on the subject of qualified signatures, Section 3 presents an overview of the main aspects of the solution, Section 4 further details the solution's implementation, Section 5 gives an insight on the Evaluation and Recommendations for the proposed solution and, at the end, it concludes with a few final remarks.

2 Related Work

This section starts by giving a more detailed insight on both European Union (EU) documents regarding qualified signatures[1][2]. Even though RSIGN proposes an interesting approach, other solutions capable of producing qualified signatures exist, they are also going to be discussed here.

Regarding mobile signing solutions, two main categories can be defined, namely: Server based and Mobile device based electronic signatures[6][5][7].

When discussing Server based solutions, the certificate which holds the public key necessary to verify the signature may be issued in the name of the user or in the name of the service provider to sign documents in the name of the user. In the latter, some authors argue that the user must give away its private key which goes against the requirements for qualified signatures because data is being signed in the name of the service provider and not in the name of the user, losing its qualified status[7] (and the non repudiation property by the user, for that matter). The same argument is stated in [5] and in [6]. The problem of the dubious status of qualified signatures when the signature is created at the server is justified with Art 2.2(c) of the EU Directive 1999/93/EC (the older Directive) on the requirements for qualified signature creation which states that *"it is created using means that the signatory can maintain under his sole control;"*[2]. However with the EU Regulation 910/2014[1] release as a replacement to the previous Directive, this issue is clarified. eIDAS states that:

> "(51) It should be possible for the signatory to entrust qualified electronic signature creation devices to the care of a third party, provided that appropriate mechanisms and procedures are implemented to ensure that the signatory has sole control over the use of his electronic signature creation data, and the qualified electronic signature requirements are met by the use of the device."

Plus, the aforementioned Art 2.2(c) has been updated to *"it is created using electronic signature creation data that the signatory can, with a high level of confidence, use under his sole control;"*[1]. This means that, with some security guarantees, the users may delegate the signature creation to a service provider, as long as it can maintain control of when signatures are created in their name, i.e. signatures are created only when the users explicitly authorises so. Therefore, signatures created at the server may be eligible as qualified signatures.

Server based solutions usually recur to the use of a remote HSM to store users' keys. Tailored solutions[8], i.e meant for a specific environment, also exist. [4] presents a solution which is only suited to the electronic identification system of Austria, it is an example of a tailored solution. Zefferer *et. al*[4] argues that, in general, there is a lack of proper standardisation. This lack of common solutions makes it difficult for electronic identification of citizens to be recognized in different EU member states[9][10]. This is due to the national legislations that usually differ.

On the other hand, mobile device electronic signatures imply some signing capabilities at the client side. Many of this solutions are based on smart cards. Even though smart cards are robust devices for storing users' private keys, usability wise, they are not the ideal approach, since it requires the user to have a smart card reader device[8][7].

Martínez-Montesinos *et. al*[6] argue that the main disadvantage of Mobile device based signatures that require the user possesses a mobile phone, is that the mobile network carrier (also known as mobile operator) has to establish an agreement with the entities that provide the signing capabilities which is not the case for most network carriers.

In [11], the authors argue that usability should be an important requirement when designing qualified signature solutions and that qualified signature creation at server side is "user friendlier", since the the user only authorises the signature creation and the rest of the process is performed remotely by the service layer.

Regarding existing solutions, in [3] a remote HSM-based solution is presented. It protects the signing key by wrapping it with the HSMs master key. It also protects the wrap with the user's public key. The private key corresponding to this public key (different than the signing key pair) is protected with a key derived from the user PIN. To fulfil a sign request, after a correct PIN insertion and Short Message Service (SMS) authorisation code sent by the user, the private key is decrypted by a PIN-derived key and the wrap is obtained by decrypting it. The wrap is delivered to the HSM that unwraps the signing key and then the signature is created. Authentication of the user is done by the network operator. The user must possess a Subscriber Identity Module (SIM) card for successful authentication. This solution depends in the external network operator and the possession of the SIM card. Also, some of the trust must be given on an entity other than the HSM, since the security of the PIN is verified out of its scope.

Zefferer *et. al*[4] propose a remote HSM-based solution that makes use of Austrian provided modules for qualified signature creation. It relies on the use of a widely used technology for user interaction, the SMS. However the user must input a Transaction authentication number (TAN) that is sent through in the SMS in plaintext. Despite being a convenient solution, it may not be secure. Plus, like in [3], it relies on the network operator as the it is required by the SMS technology.

Rossnagel *et. al*[5] propose a client-side signature creation that relies on a SIM card with signing capabilities, which is not common in SIM technology. To mitigate the identified problem of business lock-in (there would have to be changes in the mobile network infrastructure), certification is left to a third-parties, chosen by the user. Physically, the user would have to buy this specific card, activate signature data and require a certificate to be issued. Usability wise, this solution lacks the convenience remote signature offers.

To conclude, most solutions rely on external technologies for either authentication or signature creation, such as SIM cards. This means that there is always an hardware dependency, which should be avoided to achieve platform independence. Furthermore, in the case of remote signature solutions, authorisation of signature creation in the name of the user is always a risky procedure because it requires the authorisation data to be sent over the network. Communication channels must guarantee the authenticity, privacy and freshness of this data. Since EU Directive 910/2014[1] there has not been many solutions that could potentially be used by the general community, giving opportunity to explore innovative solutions.

3 Proposed Solution

Given the requirements set towards a service capable of remotely signing documents and the existing solutions, herein a dedicated service is presented. This solution avoids the possession of extra hardware on the client-side. It includes a remote Hardware Security Module (HSM) to perform critical security operations.

3.1 RISGN's Architecture

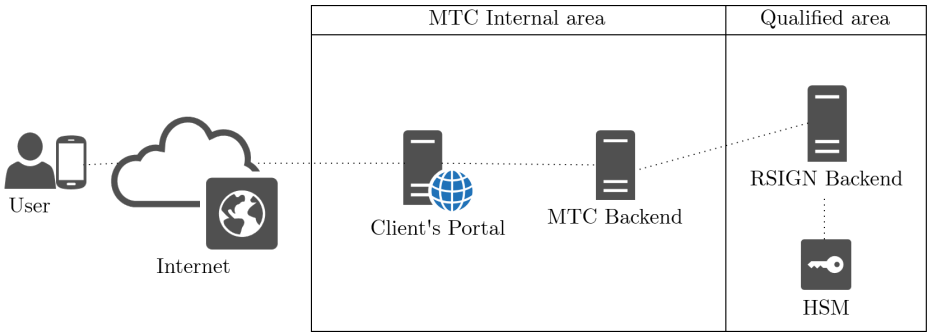

Fig. 1. RSIGN's Architecture

The architecture, depicted in Fig. 1, presents the overall components in the RSIGN's architecture. On the client-side, the user accesses the service through its mobile device, such as a smartphone or a laptop, connected to the internet. Operations triggered by users are redirected to the Client Portal, which includes the frontend of the system. This component is responsible for authenticating the user in the system. If successful, it redirects requests and responses to/from the Multicert (MTC) Backend.

The MTC Backend is responsible for storing signing keys, generate the associated certificates, and making requests to RSIGN when necessary. The RSIGN Backend receives requests and serves as the application logic supporting HSM-related operations. As mentioned before, the HSM performs the needed cryptographic operations and stores sensitive data. The RSIGN Backend communicates with the HSM using a PKCS#11 interface, thus allowing hardware independence, i.e. RSIGN can provide the desired functionality using different HSMs.

The Client Portal and the MTC Backend are composed of several components that were not displayed in this initial diagram, for simplicity's sake. The actual components, illustrated in Fig. 2, are: the Client Portal Frontend, which make the service available to client applications, the Client Portal Backend which deals users' authentication by verifying their credentials against what is stored in the Authentication Database. The MTC Backend is composed by the RADIX, a component that manages the users' certificates and is responsible for redirecting requests/responses to/from RSIGN. It relies on a database used to store users' certificates (Certificate Database) and a database to store signing keys (Key Database), once they are registered in RSIGN.

Since this is a security solution, it is important to group the architecture's components into areas with well-defined boundaries between them because the

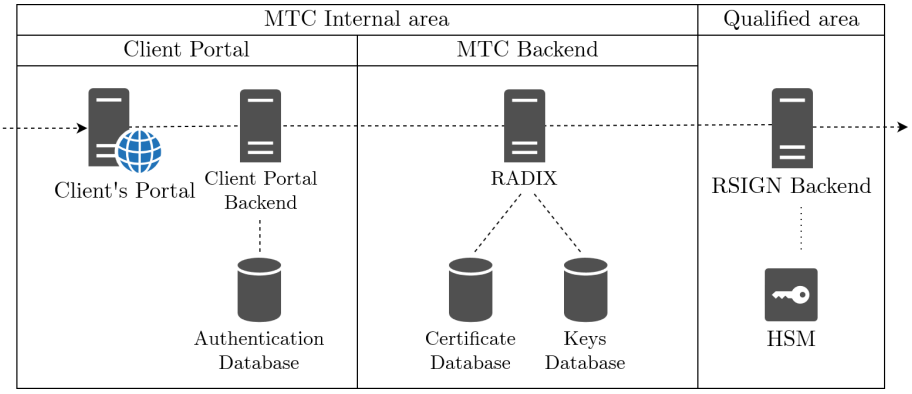

Fig. 2. Components in the MTC Internal zone.

data that enters and leaves each area needs to be protected. For instance, the PIN must must not be disclosed, i.e. it must not be accessible in plain text (user endpoint and Qualified area exempt). This facilitates compliance with eIDAS[1] in the sense that only the user has the necessary knowledge to trigger qualified signature creation and no potentially less secure components (e.g. Client Portal) can get access to sensitive data in plain text. Furthermore, this area separation allows for a modular deployment of the components in the physical infrastructure.

3.2 RSIGN's secure channels

In order to achieve a secure solution, the communication going inwards and outwards from each component must be protected in some way. The Transport Layer Security (TLS) standard provides the security properties that protect the messages exchanged between components that are directly connected. However, regarding the eIDAS[1], it must be assured that sensitive data (in this case PINs, data to be signed and signatures) exchanged between the user and the components in the Qualified area, is protected.

Fig. 3 illustrates the secure communication channels between the several system components. A TLS connection among communicating components and a secure channel on top of the TLS channels is established before any sensitive data is exchanged.

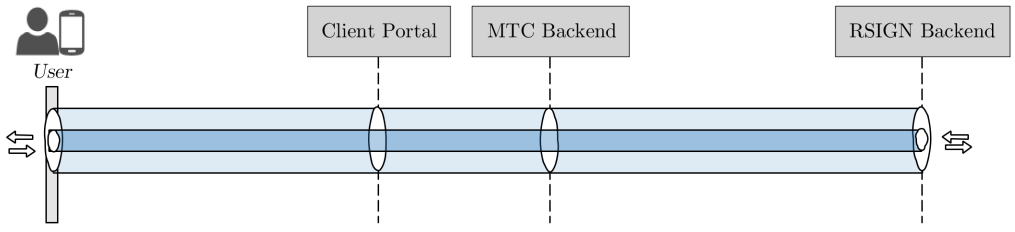

Fig. 3. Communication channels between system components, after user login

In RSIGN, the establishment of a secure channel that links the user endpoint and the RSIGN Backend is initiated by the Diffie-Hellman (DH) Key Exchange

protocol, at user log in. After the user successfully authenticates itself towards the system, the user's DH public values are sent to the RSIGN Backend and once the Backend computes the shared secret, it sends its own public value so that the user may obtain an equal shared secret. Afterwards, the user and the RSIGN Backend exchange random data to confirm the use of the newly created link/channel. From that moment on, the user and the RSIGN Backend share a secret key that may be used to protect data that not even intermediary components, i.e. the Client Portal and the MTC Backend, can have access to in plain text.

The Diffie-Hellman (DH) protocol[12] outputs a shared secret key that provides Confidentiality to the data being exchanged. However, this protocol lacks Data Authenticity and Freshness mechanisms. Therefore the Hashed Message Authentication Code (HMAC)[13] mechanism and the use of *nonces*, random sequence numbers, are added to assure the aforementioned security properties.

After the completion of the DH protocol, messages are exchanged in the following way: the sender that wishes to send a message, M, derives an encryption and integrity keys, K_c and K_a, from the agreed shared secret key, K_s, and computes the HMAC of the encrypted data plus *nonce*, N_1, to be sent. It then sends the encrypted message and the HMAC alongside it (see (1)).

$$\begin{aligned} K_c, K_a &= derive(K_s) \\ A &= \{data, N_1\}_{K_c} \\ M &= A, HMAC(A, K_a) \end{aligned} \tag{1}$$

The receiver then derives the encryption and integrity key (the same way the sender did) and computes the HMAC of the encrypted message to verify the authenticity of the data. After decrypting and processing the data, it responds to the sender with a new message, M', containing response data, $data'$, and the incremented *nonce*, $N_1 + 1$ (see (2)). If the original sender verifies that the message is authentic and the nonce is as expected, it accepts the response.

$$\begin{aligned} B &= \{data', N_1 + 1\}_{K_c} \\ M' &= B, HMAC(B, K_a) \end{aligned} \tag{2}$$

3.3 RSIGN's operations

The operations RSIGN provides are: *Data Generation*, *Signing* and *Change PIN*, which are detailed below. Before each operation, the user must log in the system, using its email and password. This means the first interaction the user performs in RSIGN is signing up, using its email and password.

Data generation The following protocol, depicted in Fig. 4, represents the *data generation*, i.e. signing key and PIN, necessary for the *signing* operation.

In this diagram, the user requests its intention to be able to sign data in the future. For that, a key pair and a PIN is generated by the HSM. Although, as stated before, the private key must not be return in plain text outside the HSM. For this, the newly generated private key, $K_{signing}$, is wrapped using a derived key (see (3)). This wrapping key, $K_{wrapping}$, is derived from the master key of the HSM, K_{MK} (which never "leaves" its premises), and in the derivation process, a hash of the PIN is added as a computation parameter. After the operation is completed, the wrapped key, the public key and the PIN are returned. Private

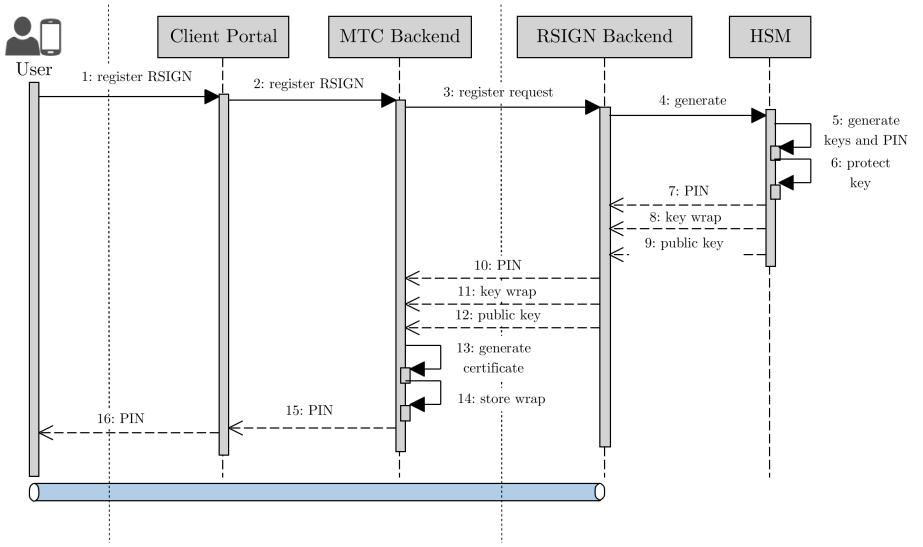

Fig. 4. Sequence digram presenting user data generation in RSIGN

and public key pair, PIN and wrapping key are deleted from the HSM, where all this operations took place. Note that the PIN is returned to the RSIGN Backend in plain text. This is due to feature limitations of the HSM's interface, PKCS#11 and is only secure because the RSIGN Backend is trusted, to that extend.

$$\begin{aligned} K_{wrapping} &= derive(K_{MK}, hash(PIN)) \\ key\ wrap &= wrap(K_{signing}, K_{wrapping}) \end{aligned} \tag{3}$$

In the RSIGN Backend, the PIN and the wrap are encrypted and, along the public key, sent to the MTC Backend. The PIN is then delivered to the end user, using the secure channel, where the PIN is obtained. The user then becomes the only entity to have knowledge of the PIN. It is not persistently stored elsewhere.

In the MTC Backend, a certificate is generated, based on the newly generated public key, and stored in a certificate database. Some information of this certificate will be used to uniquely identify the wrapped signing key in the Key database.

This data generation operation can be compared with the proposed solution in [3] in which the master key is used to wrap the signing key. In RSIGN the signing key is wrapped with a temporary key, which is destroyed after the operation and it needs to be computed again to obtain the singing key to be able to used it thus making the HSMs master key less exposed since it is not used directly to wrap keys. Furthermore, in [3], the PIN is not used directly to protect the signing key, it rather is used, outside the HSM, to protect the private key necessary for obtaining the signing key. Therefore, there are two signature objects for each user. In RSIGN storage is optimised by having only one signature object per user.

Signing Following this *data generation* interaction, RSIGN is ready to sign data in the name of the user. Diagram in Fig. 5 states a signing operation. Again, after a successful login, the user sends in the hashed data in the sign request through the secure channel, along with the authorisation PIN. After receiving the request, the MTC Backend retrieves the wrap of that user's signing key and sends a request to the RSIGN Backend. Then the HSM performs the key unwrapping

(the same way it was computed in 3). Unwrapping operation fails if the given PIN is incorrect (RSIGN Backend blocks the user after a number of unsuccessful PIN tries for a limited time). From there, the private key in the HSM is used to sign the data in the signing request. Again, private and wrapping keys are deleted from the HSM and the qualified signature is returned to the end user via the established channel.

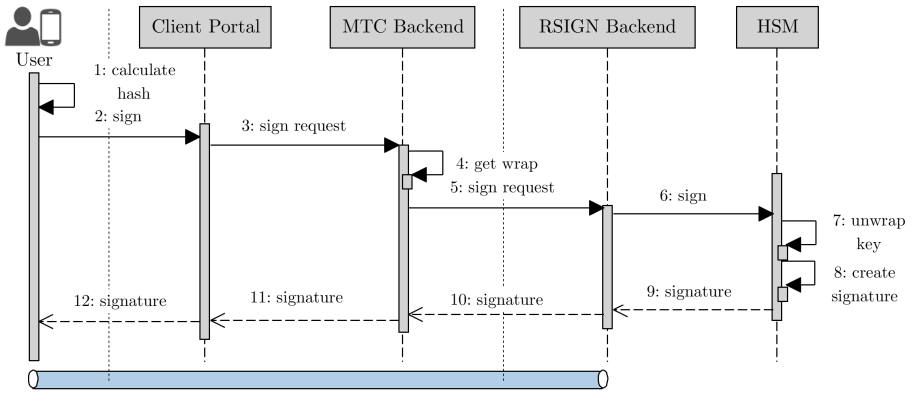

Fig. 5. Sequence diagram presenting a sign request

Change PIN The user may also change its PIN and it is advised that he/she does so after it receives the PIN in the *Data Generation*. Similarly to that initial operation, for this to happen the old PIN is used to compute the wrapping key, in the HSM, which is then used unwrap the singing key, compute a new wrapping key with the new PIN and wrap the singing key to be stored externally. Again, the unwrapping fails if the old PIN is incorrect.

4 Implementation

As mentioned before, the solution is being developed in collaboration with Multicert. This solution is meant to be deployed in an existing signature solutions which offers various options for signing. That solution includes an adapted version of the one herein presented. For simplicity sake, some components were not implemented in this prototype since they are already deployed in Multicert's infrastructure. The core of this solution consists of the Qualified area and its interaction with users. Accordingly, the implementation consists of a user with its smartphone connected to the Internet, interacting with RSIGN through an Android application which uses a server provided Representational State Transfer (REST) interface, allowing for users to signup, login, request signature *data generation* (which generates the signing key and PIN, request remote signature creation of one or more files and request its PIN to be changed. At the server side, the Client Portal relies on an Authentication Database for user validation (at the MTC Backend) and communicates directly with the RSIGN Backend redirecting requests/responses. There, the Backend, connected to a key database (to store key wraps), communicates with the HSM for the required operations.

In terms of the secure channel, that provides Confidentiality, Authenticity and Freshness to sensitive data being exchanged between the user endpoint and the RSIGN Backend, at login the user endpoint relies on the RSIGN Backend's certificate to encrypt the nonce (which later will ensure the Freshness of the communication) and to verify the signature included in the response thus proving that the DH public value of the RSIGN Backend can be trusted. After computing the shared secret key, the user endpoint verifies the HMAC value sent by the Backend which authenticates both the signature and the encrypted nonce.

5 Evaluation and Recommendations

The security analysis to the Proposed Solution is presented in this section, including what needs to be considered when using its Implementation or when extending it.

Since this is a remote service, the security properties of the communication channels that link the various components of the system are an important issue. An attack on the data being exchanged may pose a serious threat to the service. Mutual authentication and confidentiality between the Backends and between the MTC backend and Client Portal should be ensured by the TLS protocol using up-to-date certificates.

As for the secure communication between the user endpoint and RSIGN Backend, it is accomplished by an indirect channel which applies the Diffie-Hellman (DH) Key Exchange protocol, the HMAC mechanism and the use of *nonces* to assure Confidentiality, Authenticity and Freshness to sensitive data, such as PINs, signatures and hashed data. This allows for data to be sent in and out without having to trust the information in plain text to intermediate components. Note that, to avoid impersonation attacks, the public value the RSIGN Backend sends to the user endpoint in the DH protocol, should be signed. For this, the user must have access to the RSIGN Backend public key to verify the signature (e.g. stored locally in a certificate).

In terms of hardware, being the core of this solution, the qualified area must only include certified components. Specifically, the HSM and the RSIGN Backend must be secure and trusted. Without this guarantees, the solution cannot be considered a qualified service.

Regarding the HSM not storing the PIN anywhere and needing it to compute the (un)wrapping key in the HSM, it is a great advantage in granting the qualified status of the solution. On the other hand, it has one clear drawback: when the PIN is incorrect, it means that the validation must go from end to end of the components for trying again. This is a trade-off between Performance and Security, wherein Security is considered a priority. Also, the number of times the user may attempt a PIN should be limited, to avoid brute force-attacks.

To protect data meant to be signed, it is recommended that the hash of that data is sent. This means that the hash must be computed at client-side, which requires a mobile device with such capabilities. It also means that it is not in the HSM secure environment that hash values are computed, using its "bullet-proof" inward algorithms. This raises questions regarding the environment in which the client application is running. In the implemented solution, for instance, it is assumed that the android application, and the smartphone where it is hosted, is secure. Nonetheless, one advantage of this solution is that the service offers a flexible interface, i.e. porting to a different platform is possible and is not difficult.

The required initial steps, signup and login (excluding the secure channel establishment), are not the most relevant feature in the solution. This is due to the

fact that there is already strong authentication mechanisms in the infrastructure this solution was designed for (Multicert's). This paper does not propose an innovative way of user authentication.

Given the limited strength of the PIN, an alternative should be implemented. Future work includes a way to compute a user friendly authorisation code that is transformed and send to the RSIGN Backend. At the moment, this is mitigated by the use of the mentioned secure channel.

Although Performance is not the most important architectural requirement, this solution is intended for users who want to quickly obtain qualified signatures. To test the solution, the machine that was running the service has an Intel(R) Core(TM) i5-3340M CPU running at 2.70 Gigahertz (GHz), with 8 Gigabytes (GBs) of memory, and the operating system Windows 8.1 Pro. The HSM is the Crypto Server simulator[14] and both the Client Portal and the RSIGN Backend were running in separate processes. Client-side an android device version 4.4.2, physically close the service's machine, running at 1.7 GHz in a Octa-Core ARM Cortex-A7 with 2GB of memory was used as the platform to run the client application. Both devices were connected to the same internal network, at Multicert.

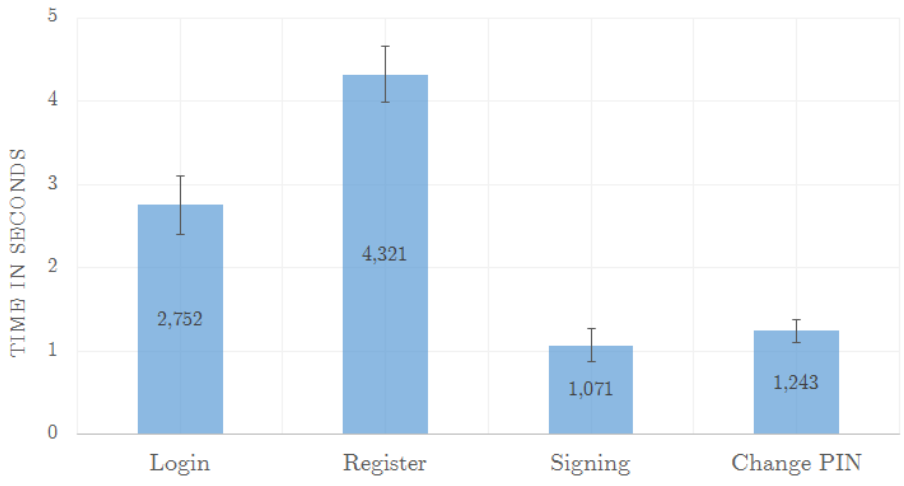

Fig. 6. Latency for each RSIGN operation, measured in seconds (standard deviation included).

Fig. 6 details the latency of the main operations in RSIGN. As can be observed, the most common operation, Signing, takes 1,071 seconds, using a 1 Megabyte (MB) file. Considering that the signing key has to be retrieved and unwrapped inside the HSM, the time necessary for a user to obtain a qualified signature for a file it chooses, this is a satisfactory latency. The Register operation takes 4,321 seconds, which has an excepted high latency since a signing key pair and a PIN are generated in the HSM and that the signing key is wrapped and stored in a database. Nonetheless, this operation only has to be requested by the user once for a signing key to be generated by RSIGN.

6 Conclusion

Being able to create signed data remotely is useful for mobile users but it requires the signature creation to be created in a secure manner considering that this creation might be hindered by the network vulnerabilities. Qualified Signatures were first introduced in Directive 1999/93/EC[2]. They must be created by a Secure Signature Creation Device (SSCD), and the operation must only be authorised by the end user.

Most of the related solutions that in some way are capable of performing Qualified Digital Signatures, defined in the most recent Regulation commonly known as eIDAS[1], consider user dependency of extra hardware. Solutions independent of extra hardware promote usability, there have not been many solutions of this kind proposed and the ones that have, still require some extra hardware on the client-side to provide the intended security properties of the communication with the remote entities.

The presented solution, Remote SIGNature (RSIGN), considers different security areas in which components are distributed. It includes an Hardware Security Module (HSM), as the SSCD and a database for persistent storage of signing keys. The need for a database is driven by the limited storage capability of the HSM. However, to achieve the requirements in the current regulation[1], the private key must not be accessible in plain text outside the HSMs domain. For that reason, the private signing key is protected by both user and HSMs private data. The user private value, a PIN, is never stored in the service, meaning that only the user may authorise qualified signature creation.

The proposed solution is a secure and remote environment where qualified signatures may be created in the name of the user. The strength of this solution is in the secure and scalable way the users' signing keys are managed as well as how protected the data that users' send and receive during transmission.

Acknowledgments

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

Bibliography

[1] The European Parliament and the Council of the and European Union. Directive No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC, 2014.

[2] 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., 1999.

[3] Clemens Orthacker, Martin Centner, Christian Kittl, Clemens Orthacker, Martin Centner, Christian Kittl, Qualified Mobile, Server Signature, Vijay Varadharajan, Christian Weber Security, and Privacy Silver. Qualified Mobile Server Signature To cite this version :. 2014.

[4] Thomas Zefferer, Arne Tauber, and Bernd Zwattendorfer. Harnessing Electronic Signatures to Improve the Security of SMS-based Services. 101(January), 2012.

[5] Heiko Rossnagel. Mobile Qualified Electronic Signatures and Certification on Demand. *Public Key Infrastructure*, pages 274–286, 2004.

[6] María Martínez-Montesinos, Daniel Sánchez-Martínez, Antonio Ruiz-Martínez, and Antonio F Gómez-Skarmeta. A Survey of Electronic Signature Development in Mobile Devices. *Journal of Theoretical and Applied Electronic Commerce Research*, 2(2), 2007.

[7] Lothar Fritsch, Johannes Ranke, and Heiko Rossnagel. Qualified mobile electronic signatures: Possible, but worth a try. *Information Security Solutions Europe (ISSE) 2003 Conference, Vienna Austria*, pages 1–6, 2003.

[8] Manuel Schallar Christof Rath, Simon Roth and Thomas Zefferer. Design and Application of a Secure and Flexible Server-Based Mobile eID and e-Signature Solution. *International Journal On Advances in Security*, 7(3 & 4):50–61, 2014.

[9] Jessica Schroers Colette Cuijpers. eIDAS as guideline for the development of a pan European eID framework in FutureID. *GI-Edition Lecture Notes in Informatics*, 2014:23–38, 2015.

[10] Alexander Marsalek and Thomas Zefferer. Leveraging the Adoption of Electronic Identities and Electronic-Signature Solutions in Europe. pages 69–71, 2017.

[11] Thomas Zefferer and Vesna Krnjic. Usability Evaluation of Electronic Signature Based E-Government Solutions. *Proceedings of the IADIS International Conference WWW/INTERNET 2012*, pages 227–234, 2012.

[12] Martin E. Hellman Whitfield Diffie. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(NO.6):644–654, 1976.

[13] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication. RFC 2104, RFC Editor, February 1997. `http://www.rfc-editor.org/rfc/rfc2104.txt`.

[14] Utimaco. `https://hsm.utimaco.com/cryptoserver/cryptoserver-sdk/`. (retrieved: 17 September of 2017).