Digital Signature as a Cloud-based Service

Wojciech Kinastowski
Institute of Control and Information Engineering
Poznan University of Technology
Poznan, Poland
wojtek@kinastowski.pl

***Abstract*—Cloud-based digital signature can be seen as a model for reliable, convenient, on-demand network access to security infrastructure that performs cryptographic operations of digital signature. This study proposes a protocol for data exchange between signer and signing-enabled cloud environment in the cloud-based digital signature model. It also covers performance results and implementation notes of Signer entity.**

Keywords-Digital Signature; Cloud Computing; Cryptography.

I. INTRODUCTION

Recently, cloud has become a new paradigm for delivering computing as a utility. Although the theory behind cloud computing is based on decades of the existing technologies and research, enthusiastic response from developers and widespread acceptance among users confirms that cloud computing is here to stay and likely to play an even more important role as a concept in many fields of information technology, including encryption. Defining cloud computing as a “model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [1], and digital signature as “the result of a cryptographic transformation of data which, when properly implemented, provides the services of: origin authentication, data integrity and signer non-repudiation” [2], cloud-based digital signature can be seen as a model for reliable, convenient, on-demand network access to security infrastructure that performs cryptographic operations of digital signatures.

The main difference between a standard digital signature system and a cloud-based one is that, while the first operates in the “close” environment of a personal computer and plugged-in dedicated devices (microchip card and card reader), the cloud-based system involves network data exchange between signer and signing-enabled cloud environment. This paper proposes a protocol for this data exchange and, as a result, outlines Software as a Service (SaaS) cloud that performs digital signature.

The paper is organized as follows. Section 2 describes some basic requirements for cloud-based digital signature system. Next, in Section 3, the protocol's entities and data flow are analyzed. Section 4 details each step in the protocol. Section 5 is based on the implementation of Signer entity and covers performance results and implementation notes. Finally, the related work and motivation for future work are discussed at the end of the paper.

II. REQUIREMENTS

Requirements for cloud-based digital signature protocol are associated with the demands for newly designed public-key cryptosystems reported in the literature [4,5,6].

A. Security

Security of cloud-based digital signature system simply refers to the protection of user’s private key from being retrieved and/or used without authorization. Each time the private key is restored in the cloud it can be extracted and used outside the system (attack on key). Other threats are related to unauthorized use of the private key inside the system, which may be affected by a modification of data sent for signing (attack on data) or being impersonated online (impersonation attack).

Considering the source of risk to the system’s security, we can identify two main groups of threats. The system can be compromised by vulnerabilities in supporting software (including operating system, web browser, web server, database server etc.). This kind of threats can be called indirect because they are not related to the process of cloud signature itself. The affected system may disclose confidential data or allow unauthorized modification to data flow. The ability to protect the system against indirect threats is obviously limited. Therefore, when designing a secure cloud signature system, it is necessary to analyze the effects of a successful attack using vulnerability in supporting software. In such a case, security of user’s private keys must be preserved.

The other group of risks is directly related to vulnerabilities in the system’s protocols and procedures (direct threats). They may occur in each component of the system and at each stage of the process. In contrast to the indirect risks, a successful attack using the features and characteristics of the protocols and procedures of designed system results in disruption of the signature process and often allows an attacker to compromise private keys restored in the cloud. Therefore, a secure cloud signature system must prove its resistance to direct threats.

When analyzing security of centralized cloud signature system, all the involved protocols and procedures need to be examined to understand the scope of potential attacks. When only a single private key can be compromised, we are talking about local-scope threats. The attacks which threaten all private keys and any signing process are considered global-scope.

B. Usability

ISO [23] defines usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" (ISO 9241-11:1998). The emphasis placed on this requirement stems from the belief that current systems do not correspond with modern standards of usability (well known from electronic payment systems and e-banking) and that high usability is always at odds with the requirement of high security level [3,7].

A radical method of achieving high usability is to eliminate dedicated devices for digital signature (microchip cards, card readers) and propose data e-signing as in-cloud service. By transferring processing logic to infrastructure provider (cloud) and providing a simple access interface, the process of digital signature can be reduced to standard authentication and secure data transfer.

C. Cross-platform and integration capabilities

In order for any kind of digital system to be considered cross-platform, it must be able to operate in any hardware and software configuration. Dedicated hardware in conventional digital signing solutions impose mandatory system requirements. It makes porting the system to new platform (e.g., mobile devices) very complicated. It also makes it difficult to integrate digital signing services with other electronic services.

Providing an interface for digital signature services through standard network protocols has multi-platform capabilities at both the hardware and software level. Transfer of processing logic to cloud also offers great opportunities for integration with other electronic services residing in the cloud.

III. Protocol Basics

We can identify four basic protocol entities:

A. Signer

Signer (User) is the client for signature service, whose private key is restored in the cloud in digital signing process. Considering the complexity of the digital signature process, the system requirements for signer are minimal. They encompass a mobile device with an active SIM card (e.g., phone) and a device with Internet access (e.g., Internet enabled PC with modern web browser). These very basic requirements allow processing regardless of hardware and software platforms. For the mobile device, it means flexibility in terms of architecture and operating system as well as services offered by the mobile operator. For the Internet enabled device, there are no operating system and web browser restrictions. Nevertheless, there are computing power and web browser supported technologies issues related to client-side cryptographic operations. This is discussed in Section 5.

The concept of moving processing to the cloud eliminates the need for dedicated hardware and software. Signer does not have to deal with a microchip card, a card reader and pre-installed software.

B. Issuer

Issuer is an entity that owns or creates data signed by Signer in digital signing process. In this paper, the most basic model is presented, which assumes that Issuer and Signer are the same user. However, it should be noted that more complex models with separation of these roles can be presented. Regardless of role separation, issuing data is also characterized by "cloud-based processing logic". Thus, the system requirement remains the same for both Signer and Issuer.

C. Proxy

Proxy provides the interface for the digital signature service in cloud. The device consists of a single server or a group of servers with software that supports HTTP communications protocol (web server), database management system and dedicated applications. The role of proxy server is reduced to managing and monitoring user access to a hardware security module (HSM) where cryptographic operations of cloud-based digital signature are implemented. Process management includes user's authentication as well as collecting and formatting data sent to the HSM. Proxy also performs monitoring and logging system events.

D. Hardware Security Module (HSM)

HSM is a device with built-in secure cryptoprocessor dedicated to managing cryptographic keys and carrying out cryptographic operations of cloud-based digital signature. The HSM certified by NIST [24] is considered tamper-resistant, which is why the environment of this protocol entity is assumed secure in both the logical and physical layer.

As mentioned earlier, the basic model of cloud-based signature service assumes that Signer signs data he owns. The model describes the interaction of three entities (Signer/Issuer, Proxy and HSM). Signer/Issuer and Proxy communicate with HTTP protocol. In order to provide a higher level of security, this communication should be made over a secure TLS channel. HSM can be connected to Proxy as a built-in device (e.g., PCI device) or reside as a standalone cryptoserver. The detailed configuration of cloud environment (Proxy and HSM) is beyond the scope of this paper.

IV. Protocol Details

User, in addition to unique identifier (name) and password (pass), has a mobile phone with active SIM card and corresponding phone number. This device is used to receive text messages, sent from the signing system, containing the value of one-time password (OTP).

Each user is assigned an asymmetric public-private key pair (k_{pub}^{user}, k_{prv}^{user}) representing electronic signature keys. Key

k_{prv}^{user} is used to digitally sign data, which is why its protection is critical from a security point of view.

Hardware security module maintains its own asymmetric key pair $(k_{pub}^{hsm}, k_{prv}^{hsm})$, symmetric key K, the value of OTP_{secret} for the one-time password generation algorithms and implements the following:

- Gen - password-based key derivation function [8],
- Sym^{enc}, Sym^{dec} - encryption and decryption algorithm of symmetric cipher working in Authenticated Encryption (AE) mode [9,10],
- Asym - asymmetric cipher,
- $Sign_{Asym}$ - digital signature algorithm,
- Gen_{OTP} - one-time passwords generator [11,12].

Proxy stores $\hat{k}^{user}$ necessary to restore the user's private key:

$$\hat{k}^{user} = Sym_K^{enc}(Sym_{Gen(pass)}^{enc}(k_{prv}^{user})) \tag{1}$$

In order to sign a document (doc), the following steps are performed:

1. User connects to the Proxy and pre-authenticates. In order to keep the protocol as simple as possible, Signer uses only one password in the system. Although the pre-authentication process is used mainly for phone number identification, it uses the same password that secures users private key. That's why security requirements for this process should be relaxed, for example, by using collision-rich functions [7]. Another idea is to allow clients to pre-authenticate to servers using zero-knowledge proofs.
2. The server identifies the phone number of the authenticated user and initiates the process of providing one-time code OTP.
3. The user downloads the software, necessary for protocol communication, as a dynamic website. Using the supplied implementation of algorithms User generates:

$$\widehat{doc} = Sym_{Gen(pass||OTP)}^{enc}(doc) \tag{2}$$

$$\widehat{pass} = Asym_{k_{pub}^{hsm}}(pass) \tag{3}$$

   and sends (login, $\widehat{pass}$, $\widehat{doc}$) to Proxy.
4. Proxy forwards ($\widehat{pass}$, $\widehat{doc}$) dataset received from user together with $\hat{k}^{user}$ suitable for an authenticated user to the security module (HSM).
5. HSM restores:

$$pass = Asym_{K_{prv}^{hsm}}(\widehat{pass}) \tag{4}$$

$$OTP = Gen_{OTP}(\hat{k}^{user} \,||\, OTP_{secret}) \tag{5}$$

$$doc = Sym_{Gen(pass||OTP)}^{dec}(\widehat{doc}) \tag{6}$$

$$k_{prv}^{user} = Sym_{Gen(pass)}^{dec}(\, Sym_K^{dec}(\hat{k}^{user})) \tag{7}$$

   As the algorithm Sym^{dec} operates in AE mode, operation (6) confirms the integrity and authenticity of the document and verifies the one-time password. Similarly, operation (7) also authenticates User by verifying (pass).
6. Security module (HSM) signs a document using the user's private key k_{prv}^{user}

$$doc_{sign} = Sign_{k_{prv}^{user}}(doc) \tag{8}$$

Fig. 1 depicts a detailed view of the protocol flow by describing the sequence of actions in a process. The key features can be summarized as follows:

- Independent proofs. Security of the user's private key relies on two independent proofs of identity: something the user has (registered SIM card and the phone receiving one-time passwords) and something the user knows (password).
- 'Sole control'. The private key remains under the user's 'sole control'. Key data is encrypted with password known only by Signer. It is impossible to restore even by the service provider. The only person who can do that is Signer. The concept of 'sole control' is discussed in detail in [14].
- Security functions in HSM. All main security functions are moved to a secure environment of Hardware Security Module. Outside the HSM private keys and data to be signed are always encrypted. Verification of independent proofs (password and one-time password) is also implemented in HSM by using a symmetric cipher in AE mode.
- High usability level. From Signer's point of view digital signature process has been reduced to standard authentication and secure data transfer (see Fig. 1). Signer does not need any dedicated devices for digital signature.
- Event logging. Proxy can be used as an event logger in the system, which meets the requirement to include generating digital signature into the security process of public key infrastructure (PKI) pointed out in [6].

V. SIGNER ENTITY IMPLEMENTATION NOTES

As mentioned earlier, there are some implementation issues related to client-side cryptographic operations that must be analyzed in order to estimate the additional computational overhead of the proposed protocol when comparing to basic server-side digital signature protocol, with no client-side encryption (e.g., one proposed in [16]).

First of all, the client-side cryptographic operations, performed in step 3 of the protocol, are executed transparently in browser environment and will probably be implemented in JavaScript. Most web programmers agree that the biggest challenge in web design lies in dealing with the variety of browsers. While the majority of active page elements are reliably rendered in most browsers, each browser has its own quirks when it comes to the implementation of JavaScript engine. This might cause different overhead for the same machine when performing cryptographic computation in different browsers. Secondly, client-side data encryption requires loading local files. Such feature is not supported by older browsers. A standard way to interact with local files was introduced in HTML5 specification, so an up-to-date, HTML5-enabled browser is required to interact in the protocol. Although this entails additional restrictions, the need to use an up-to-date browser also meets the security requirements mentioned in Section 3.

Further notes are based on Signer entity implementation, prepared as dynamic HTML page with SJCL library for cryptography in JavaScript [20]. For asymmetric encryption 256-bit ElGamal ECC was used. Symmetric encryption is performed with 128-bit AES in CCM mode. Table I shows the average execution time for step 3 (see Section 4) for different sizes in different browsers.

It has been observed that performing symmetric encryption on larger files causes browser to freeze. This behavior is unacceptable in terms of usability. To avoid this, larger files should be split into smaller parts and encrypted separately. When choosing the size of file splitter the following factors must be taken into consideration. Still, encryption of large file parts might cause the browser to freeze on older machines. Small file parts increase the number of iterations in encryption loop, which influences overall performance.

Table II shows the average execution time of encrypting 10 MB file with different splitter size. The test was performed on two different computers with high and low computing power, respectively.

In addition to computation overhead, there is also the additional download size of required scripts. Using well-known optimization techniques this size can be reduced to approximately 50kB, which is negligible from the user's point of view.

TABLE I. AVERAGE EXECUTION TIME FOR DIFFERENT DOC SIZES IN DIFFERENT BROWSERS

File size	**Execution time(ms)**		
	Chrome	***Firefox***	***IE***
100kB	688	344	186
200kB	814	392	245
500kB	1186	559	422
1MB	1521	820	688
10MB	11183	5825	5188
20MB	23634	11564	9932

TABLE II. AVERAGE EXECUTION TIME FOR DIFFERENT DOC SIZES IN DIFFERENT BROWSERS

Splitter size	**Execution time(ms)**	
	Computer 1	***Computer 2***
100kB	6246	15319
500kB	5955	14452
1MB	5884	13747
5MB	5673	freeze

VI. RELATED WORK

A secure digital signature creation environment, based on mobile devices and smart cards, is defined and analyzed by A. Mana et al. [15]. Storing private key on signer's SIM card is proposed by H. Rossnagel [16]. A more server-side approach with encrypted private keys is presented by M. Centner et al. [17]. The same authors in [18] designed a digital signature service based on smartcard-reader middleware as a Java applet. A proof-of-concept prototype of this approach has been implemented as a web-based signing service. A signing scheme for thin clients, with server based processing is presented by Y. Lei et al. [13]. J. Anderson et al. [7] proposes a protocol, which allows users to store secrets, such as private keys, in the cloud, using the services of several key recovery agents.

On-going work on novel signing service schemes is also related to European Commission's mandate M/460. The UE standardization platform is prepared by two European standardization organizations, CEN [25] and ETSI [26]. In [19], the Commission indicates new perspectives and challenges for the platform. Many of them (e.g., cross-border compliance) can be implemented with cloud-based processing logic.

Things to consider when moving digital signature model, or, more general Public Key Infrastructure into cloud are addressed by H. Kharche et al. [4]. Brown and Robinson [5] show how existing security protocols (like TLS) can derive from cloud computing. Important cloud-specific security issues are also pointed out by R. Chow et al. [22].

VII. CONCLUSION AND FUTURE WORK

The proposed cloud-based digital signature protocol meets the usability and cross-platform requirements laid down in Section 2. Although the protocol was designed taking into account the security requirements, future studies are required in order to prove its security.

As the proposed protocol is mainly focused on signer-cloud communication, further studies are require to show how such digital signature model can exploit cloud benefits. Moreover, the protocol can be extended to handle more complex models (e.g., with Signer and Issuer role separation). Advanced digital signature services can be also developed based on the proposed protocol (e.g., Forward-Time Public Key proposed in [21]).

The cloud-based digital signature can also be analyzed for compliance with law and regulations of the qualified electronic signature. When it comes to EU regulations, similar studies are presented by M. Centner et al. [17].

REFERENCES

[1] P. Mella and T. Grance, "The NIST Definition of Cloud Computing". Special Publication 800-145, NIST, Sep. 2011.

[2] Security requirements for cryptographic modules, FIPS PUB 140-2, NIST, Dec. 2002.

[3] D. Davis, "Compliance Defects in Public-Key Cryptography", Proc. 6th Usenix Security Symp., Jul. 1996, pp.171-178.

[4] H. Kharche and D. S. Chouhan, “Building Trust In Cloud Using Public Key Infrastructure -A step towards cloud trust”, International Journal of Advanced Computer Science and Applications, vol. 3, no. 3, Mar. 2012, pp. 26-31.

[5] J. Brown and P. Robinson, “PKI Reborn in the Cloud”, conference slides, RSA Conference Europe, Oct. 2011, http://365.rsaconference.com/docs/DOC-3037 [retrieved: March 2013].

[6] C. Ellison and B. Schneier, “Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure”, Computer Security Journal, vol. 16, no. 1, 2000, pp. 1–7.

[7] J. Anderson, F. Stajano, “On Storing Private Keys 'In the Cloud' Extended Abstract”, unpublished, http://www.cl.cam.ac.uk/~jra40/publications/2010-SPW-key-storage.pdf [retrieved: March 2013].

[8] B. Kaliski, “PKCS #5: Password-Based Cryptography Specification Version 2.0”, RFC 2898, IETF, Sep. 2000.

[9] D. Whiting, R. Housley, and N. Ferguson, “Counter with CBC-MAC (CCM)”, RFC 3610, IETF, Sep. 2003.

[10] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, “OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption”, ACM Transactions on Information and System Security (TISSEC), vol. 6, no. 3, Feb. 2003, pp. 365-403.

[11] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, “TOTP: Time-Based One-Time Password Algorithm”, RFC 6238, IETF, May 2011.

[12] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, “HOTP: An HMAC-Based One-Time Password Algorithm”, RFC 4226, IETF, Dec. 2005.

[13] Y. Lei, D. Chen, and Z. Jiang, “Generating Digital Signatures on Mobile Devices”, Proc. 18th International Conference on Advanced Information Networking and Applications, Mar. 2004, pp. 532-536.

[14] Public Statement on Server Based Signature Services (Forum of European Supervisory Authorities for Electronic Signatures), Forum of European Supervisory Authorities for Electronic Signatures (FESA), October 2005, http://www.fesa.eu/public-documents/PublicStatement-ServerBasedSignatureServices-20051027.pdf [retrieved: March 2013].

[15] A. Mana and S. Matamoros, “Practical Mobile Digital Signatures”, Prec. EC-WEB '02 Proceedings of the Third International Conference on E-Commerce and Web Technologies, Sep. 2002 ,pp.224-233.

[16] H. Rossnagel, “Mobile Qualified Electronic Signatures and Certification on Demand”, Proc. 1st European PKI Workshop Research and Applications, Jun. 2004, pp.274-286.

[17] M. Centner, C. Orthacker, and C. Kittl, “Qualified Mobile Server Signature”, Proc. 25th International Information Security Conference, Sep. 2010, pp. 103-111.

[18] M. Centner, C. Orthacker and W. Bauer, “Minimal-footprint Middleware for the Creation of Qualified Signatures”, Proc. WEBIST 2010 International Conference on Web Information Systems and Technologies, Apr. 2010, pp. 64-69.

[19] Proposal for a regulation of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market, Commision staff working paper, Jun. 2012.

[20] E. Stark, M. Hamburg, and D. Boneh, “Symmetric cryptography in javascript”, Proc. ACSAC '09 Annual Computer Security Applications Conference, Dec. 2009, pp.373-381.

[21] J. Riordan and B. Schneier, “Environmental Key Generation towards Clueless Agents. Mobile Agents and Security”, G. Vigna, ed., Springer-Verlag, 1998, pp. 15-24.

[22] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, “Controlling data in the cloud: outsourcing, computation without outsourcing control”, Proc. 2009 ACM workshop on Cloud computing security, Nov. 2009, pp.85-90

[23] International Organization for Standardization, http://www.iso.org [retrieved: March 2013]

[24] National Institute of Standards and Technology, http://www.nist.gov [retrieved: March 2013]

[25] European Committee for Standardization, http://www.cen.eu [retrieved: March 2013].

[26] European Telecommunications Standards Institute, http://www.etsi.org [retrieved: March 2013].

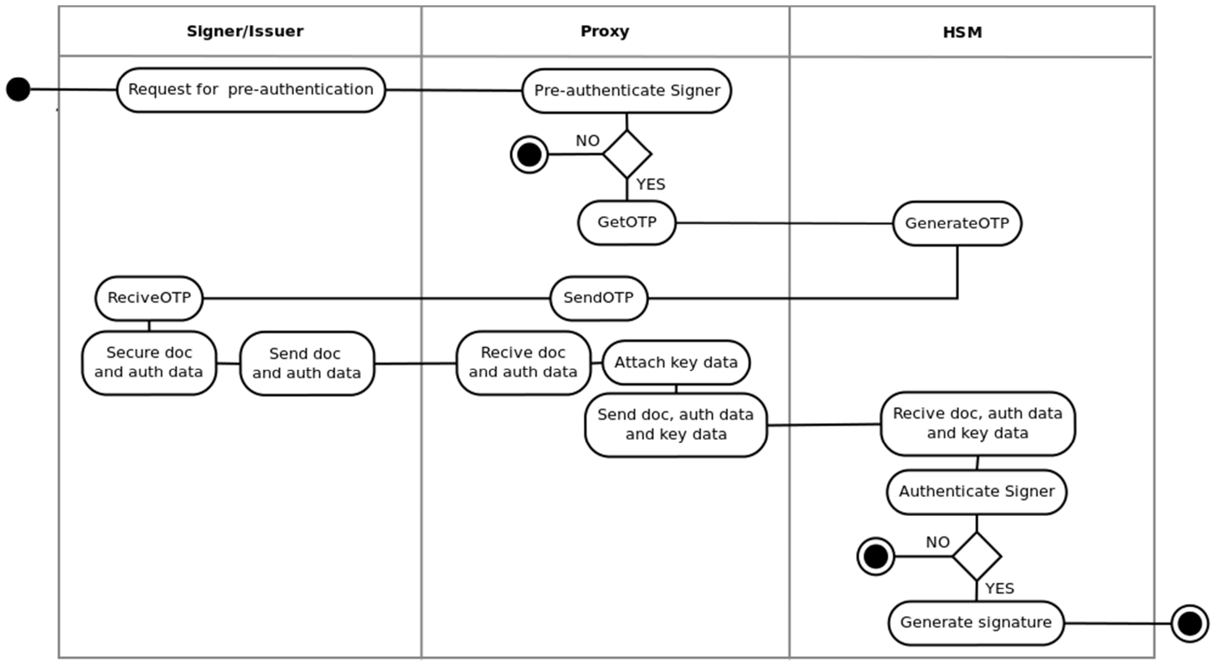

Figure 1. UML activity diagram for cloud-based digital signature protocol.