

## HOW TO STRENGTHEN CERTIFICATE ENROLLMENT

TOBIAS STRAUB

**ABSTRACT.** Many PKIs implement certificate enrollment using a registration and a certification authority. However, the standard protocol has a weakness since the intended sequence of steps in the process cannot be enforced rigorously. In the current paper, we present a new enrollment protocol to remedy this flaw. Our method provides security by means of shared signatures and distributed key generation while at the same time producing standard-compliant certificates signed with RSA or DSA.

**Introduction** *Certificate enrollment* denotes the process of initializing a new user in a public key infrastructure (PKI) which is a routine yet security-critical task. To underline a weakness in the standard protocol, we sketch a typical realization below. In this paper, we focus on *hierarchical* PKIs and assume that certificates are issued by a central and trustworthy party, the *certification authority* (CA).

To enroll in the PKI, a user generates a key pair e.g. by means of his web browser and sends the corresponding *certification request* (e.g. a PKCS#10 file) to the *registration authority* (RA). This message needs to be authenticated, e.g. by the user presenting himself at the RA, to establish a binding between the entity and the public key. The RA collects and verifies requests which are then transferred to the CA in a digitally signed container like a PCKS#7 file. Finally, the CA issues the corresponding certificates and publishes them in an online directory.

Security reasons motivate the RA/CA distinction: Firstly, to protect the PKI's certification key, the CA uses it only in a safe environment, typically on a computer disconnected from the network. The second reason is that a certificate can only be as "strong" (i.e. meaningful) as the underlying registration process. Thus, PKIs often decentralize this task using a number of RAs to be closer to the end-user. The implicit idea behind sharing responsibilities is to achieve a *four-eye* or *double verification principle*. Instead of enforcing this rule with cryptography, PKIs nowadays rely on organizational and procedural controls and – in the end – on a supposed RA/CA trust relationship. This exposes the PKI to danger since it does not effectively prevent the CA from deviating from the protocol. For instance, a fraudulent employee having access to the certification key is able to issue certificates even if there are no corresponding requests signed by an RA.<sup>1</sup> This weakness is intolerable, especially when RA and CA are operated by two different organizations as it is the case in an outsourced PKI (see [10] for a thorough treatment of this subject).

**The New Enrollment Protocol** For the sake of simplicity, we assume that there is a single certification authority (denoted CA) which certifies all end-entities using the key pair  $\langle \text{priv}, \text{pub} \rangle$ . Extending the model to multiple CAs is straightforward. We now seek a solution meeting the following requirements:

- (R1) The protocol must support multiple registration authorities  $RA_i, 1 \leq i \leq n$  assigned to CA.
- (R2) Every creation of a certificate signed by *priv* requires the cooperation of CA and  $RA_i$  for an arbitrary  $i \in \{1, \dots, n\}$ . This condition is also sufficient.
- (R3) A trusted third party (i.e. different from CA,  $RA_i$ ) must not be involved in parts of the protocol where secret information is handled.
- (R4) The protocol must output a standard-compliant certificate which current PKI-clients can verify using *pub* as a trust anchor.

---

Keywords: Certificate enrollment, cryptographic protocols, digital signatures, PKI, threshold cryptography

<sup>1</sup>A severe security incident illustrating this problem was reported in 2001 when attackers impersonated Microsoft using VeriSign code-signing certificates ([www.microsoft.com/technet/security/bulletin/MS01-017.asp](http://www.microsoft.com/technet/security/bulletin/MS01-017.asp)).

Let us briefly discuss the motivation behind the requirements: (R1) is a natural condition and (R2) is due to the observation that, in PKIs using the standard protocol, CA is always able to issue certificates completely on its own, i.e. without a legitimate certificate request. An external party being substantially involved in the process interferes with the idea of a four-eye principle. This motivates (R3). Consider a scenario where a company outsources CA to a service provider, but remains in charge of the registration. From the company's viewpoint, there is no difference in having to trust – to a large extent – the service provider or another third party (both is undesirable). To be useful in existing PKI environments, the protocol must support standard techniques to avoid the deployment of additional client software (R4).

The basic idea of the new approach is to apply a secret sharing of  $\text{priv}$  and to provide the parties with the corresponding shares such that (R2) is satisfied. More precisely, a *threshold function sharing scheme* [4] is required since  $\text{priv}$  must not be reconstructed during signature generation (otherwise, a single party would be able to subsequently create certificates on its own, contradicting (R2)). Appropriate schemes are available for RSA [3] and DSA [6], we describe them below. During an initialization phase, a private key and the secret shares for the respective parties have to be generated. We cannot delegate this task to a *trusted dealer* because of (R3). The parties have to use a shared key generation protocol instead because the dealer may ally himself with one of the parties in order to discover  $\text{priv}$ . Fortunately, protocols for shared key generation are available, too. We show how the system can be initially set up only by CA and  $\text{RA}_1$  which subsequently provide  $\text{RA}_i, i > 1$ , with key material. This allows dynamically adding new RAs to the PKI. In the following, we assume that every two parties are able to exchange messages confidentially and mutually authenticated.

**DSA** Let  $((p, q, g, x), (p, q, g, y \equiv_p g^x))$  denote a DSA key pair.<sup>2</sup> Initially, CA and  $\text{RA}_1$  construct  $q, p, g$  (in this order).<sup>3</sup> CA and  $\text{RA}_1$  select secret random values  $w_1$  and  $x_1$  which determine the private key via a *multiplicative sharing*  $x \equiv_q w_1 x_1$ . The values  $y_1 \equiv_p g^{w_1}$  and  $z_1 \equiv_p g^{x_1}$  are then published such that both parties can compute  $y \equiv_p y_1^{x_1} \equiv_p z_1^{w_1}$ .  $\text{RA}_i, i > 1$ , obtains its share  $x_i$  by a technique similar to that used in *proactive and verifiable secret sharing*:  $\text{RA}_1$  picks  $u_i \in \mathbb{Z}_q^*$  at random and sends  $(i, u_i, z_1^{u_i^{-1}})$  to CA and  $(x_i \equiv_q x_1 u_i^{-1}, g^{u_i})$  to  $\text{RA}_i$ . CA and  $\text{RA}_i$  can verify that the pair  $(w_i \equiv_q w_1 u_i, x_i)$  is indeed another multiplicative sharing of  $x$  by exchanging the values  $y_i \equiv_p g^{w_i}$  and  $z_i \equiv_p g^{x_i}$ .

The computation of a distributed DSA signature requires slightly more effort. In our realization,  $\text{RA}_i$  and CA use a 4-move protocol which is due to MacKenzie and Reiter [6]. At first, the parties generate an ephemeral key pair  $\langle k \in \mathbb{Z}_q^*, r \equiv_p g^k \rangle$  where  $k \equiv_q k_{\text{CA}} k_{\text{RA}_i}$  is shared multiplicatively. To sign the (SHA-1) hash value  $h$  of the certificate body, the parties compute  $s \equiv_q k^{-1}(h + xr)$ . This is done with the help of a homomorphic encryption system<sup>4</sup> in order to keep the inputs  $k_{\text{CA}}, w_i, k_{\text{RA}_i}$ , and  $x_i$  secret. A client verifying  $(r \bmod q, s)$ , which is a standard DSA signature, is ignorant of how it was generated.

**RSA** There is an efficient algorithm of Boneh and Franklin [2] for shared generation of an RSA key pair  $\langle (N, d), (N, e) \rangle$  in the case of at least three parties, but this protocol cannot be used here because of (R3). We use a two-party version instead which is due to Gilboa [5]. During this protocol, CA and  $\text{RA}_1$  create an RSA modulus of the form  $N = (p_{\text{CA}} + p_{\text{RA}_1})(q_{\text{CA}} + q_{\text{RA}_1})$  where the indices indicate secrets of the respective party. The computation of a candidate  $N$  requires a homomorphic encryption system, too. On average, there are a lot of unsuccessful trials since the probability for  $N$  having exactly two prime factors is proportional to  $\log(N)^{-2}$ . Therefore, this step takes significantly longer than choosing the two primes independently. (The quadratic slowdown can be avoided by using a multi-prime modulus [9].)

<sup>2</sup> $p$  and  $q$  are primes where  $q$  divides  $p - 1$ ,  $g \in \mathbb{Z}_p^*$  is of order  $q$ , and  $x \in \mathbb{Z}_q^*$  is chosen uniformly at random. Equivalence modulo  $M$  is denoted by  $\equiv_M$ .

<sup>3</sup> $q$  can be defined as the smallest prime exceeding a random lower bound which was agreed on using distributed coin-tossing. The construction of  $p$  and  $g$  is similar.

<sup>4</sup>For instance, the Naccache/Stern, Paillier, or Benaloh cryptosystem have this property.

## How to Strengthen Certificate Enrollment

Having found an appropriate  $N$ , the parties agree on a public exponent  $e$  and compute an *additive sharing*  $d_{CA} + d_{RA_1} = d$  of the private exponent. The creation and distribution of the CA/RA <sub>$i$</sub>  shares is analogous to DSA. A standard RSA signature can be obtained by simply multiplying the two *partial signatures*  $s_{CA} \equiv_N h^{d_{CA}}$  and  $s_{RA_1} \equiv_N h^{d_{RA_1}}$  modulo  $N$ . This method requires only a 2-move communication.

**Security Considerations** A common security model for two-party protocols is the *honest-but-curious* scenario where participants follow the protocol to the letter, but are allowed to do arbitrary extra computations.<sup>5</sup> Let us first consider the case  $n = 1$ : It can be shown that neither of the two parties involved in key and signature generation does learn anything (or only a few bits which can easily be guessed) about the other party's secret provided that the underlying homomorphic encryption scheme is semantically secure and the DSA resp. RSA intractability assumption holds [2, 5, 6]. The computation of new shares of  $x$  for party RA <sub>$i$</sub> ,  $i > 1$ , does not leak information since  $u_i$  (and  $x_i$ ) is uniformly distributed.

**Prototype** We have implemented a JAVA prototype which produces standard-compliant X.509 certificates. Using two machines connected via a local area network, this operation takes 2.0s for RSA ( $\log_2 N = 1024$ ) and 2.9s for DSA ( $\log_2 p = 1024, \log_2 q = 160$ , Paillier cryptosystem with 1024 bit modulus). The time for shared generation of a DSA key is 120s, whereas the time for RSA is about 3 hours. We expect that optimizations like the multi-prime RSA approach will lower this number.

**Related Work** Threshold cryptography is already used in applications, e.g. to protect the signing key in SET (see [www.setco.org](http://www.setco.org)). Another example is a "virtual smart card" where one of the two shares of the private key is derived from a user password and the other one is stored on a network server (see [7] and [www.singlesignon.net](http://www.singlesignon.net)). Threshold RSA signatures combined with distributed key generation (for three or more parties) is also used in the context of a time stamping service [1]. The ITTC-Projekt [11] is a framework to secure online systems like the Apache web server with the help of threshold cryptography.

## REFERENCES

- [1] H. Appel et al. Ein sicherer, robuster Zeitstempeldienst auf der Basis verteilter RSA-Signaturen, DuD Fachbeiträge, vieweg, 2000. (in German)
- [2] D. Boneh and M. Franklin. Efficient generation of shared RSA keys, CRYPTO '97.
- [3] C. Boyd. Digital multisignatures, Cryptography and Coding, Clarendon Press, 1989.
- [4] A. De Santis et al. How to share a function securely, ACM Symposium on the Theory of Computing, 1994.
- [5] N. Gilboa. Two-party RSA key generation, CRYPTO '99.
- [6] P. MacKenzie and M. K. Reiter. Two-Party Generation of DSA Signatures, CRYPTO 2001.
- [7] P. MacKenzie and M. K. Reiter. Networked Cryptographic Devices Resilient to Capture, International Journal of Information Security 2(1), 2003.
- [8] G. Poupard and J. Stern. Generation of Shared RSA Keys by Two Parties, ASIACRYPT '98.
- [9] T. Straub. Efficient Two-Party Multi-Prime RSA Key Generation, IASTED CNIS, 2003.
- [10] T. Straub. Zur Absicherung von PKI-Outsourcing mit Hilfe verteilter digitaler Signaturen, D·A·CH Security 2004. (in German)
- [11] T. Wu et al. Building Intrusion Tolerant Applications, USENIX Security Symposium, 1999.

---

<sup>5</sup>Weaker assumptions suffice for the case of DSA when two zero-knowledge proofs are added, cf. [6].