Journal of Mobile, Embedded and Distributed Systems, vol. II, no. 2, 2010
ISSN 2067 – 4074

Key Recovery for Certification Authorities

Eduard TRIC
Vexilla
Dionisie Lupu 35, sector 2, Bucharest, ROMANIA
ed@vexilla.org
http://www.vexilla.org

Abstract: The root key is the most important asset of a certification authority. There are several ways to protect it and to make a secure backup copy. We are describing one of the most elegant protection techniques that will be used by a real Certification Authority.

Key-Words: certification authority, key recovery, split secret.

1. Introduction

Within a PKI, the most sensible system is the certification authority (CA). For protecting the CA's private key, there are procedures which don't necessarily guaranty security, but largely increase the cost of breaking it. Our use case exploits such a security enforcement procedure, and consists of keeping the CA or a copy of the CA private key offline and sharing it between several holders. When needed, the key is reconstituted, used and then removed from the system.

2. Objectives

The objective of the key ceremony is to generate and reconstruct the private key of a certification authority in a secure and open manner.
The hardware security modules (Hsm) from the market allow the secure generation and recovery of the private key, but the biggest problem is that one cannot bring an external key and import it in the Hsm. Once you've choose a vendor, you have to use the provided technology. The lifetime of a root key (typically 15 to 20 years) is not compatible with the actual lifetime of an IT product (2 to 5 years), unless we can save and reconstruct the keys independently from the Hsm sellers.

This is a post conference paper. Parts of this paper have been published in the Proceedings of the 2nd International Conference on Security for Information Technology and Communications, SECITC 2009 Conference (printed version).

3. Methodology

The existing hardware security modules allow the generation and backup of the private keys, but lack a main functionality: the import of a key generated by software or by a third-party device. Once the root key is generated, the certification authority cannot change the hardware vendor. By using well-known algorithms, we demonstrate that is possible to generate and recover a private key without being bound to a particular hardware security module vendor.
The most popular and powerful algorithm to achieve the sharing of secrets is the Shamir algorithm, as described in "Handbook of Applied Cryptography" [6]. The usage of "secret sharing" and the strength of the Shamir algorithm are described in this book in detail. For the purpose of our use case, we would like to emphasize that this algorithm is: "perfect, ideal, extendable for new users, varying levels of control possible" and, above all, offers "no unproven assumption".

4. Technology Description

Our implementation is Java based and brings innovation by considering the whole secret as a big number. The goal is to distribute a secret Scr to n users in such a manner that any group of k users which pool their shares can recover Scr

(threshold scheme k from n). The whole secret is processed in few steps, usually 1, by splitting input data in large chunks S. For that purpose, a unique prime p longer than S is chosen from a short list of large primes. Then, there are selected k – 1 random, independent coefficients, between 1 and p - 1, defining the random polynomial over Zp, f(x) = S + a1 * x + a2 * x^2 + a3 * x^3 + ... Every Secret Holder will receive: public index i (between 1 and n), the id of p, and the results Si = f(i) mod p - 1 for every chunk. The distinct points (i; Si) allow computation of the coefficients of f(x) by Lagrange interpolation, based on p retrieved from the list. The secret is recovered by noting f(0) = S.

Remarks:

1. Implementation doesn't make assumptions about the actual algorithm. Only condition is 1 < k < n < 256.

2. The whole chunk of secret (regardless how many bits) is shared as a big number.

3. Because S is used as a number and not as bit array, to avoid losing eventual first "0" bits and to ensure that it is a positive number, the chunk is prefixed by byte 1 and after that it is processed, recovery taking care to lose first byte (it has to be 1).

4. The prime p depends of the length of S, but the range of list induces such redundancy making impossible to get useful data about S. The fact that p is known is not reducing the strength of algorithm.

5. Even if the work with big numbers is difficult, the advantages of this implementation are: the output length is compatible with the input length, the number of mathematical operations and the number of random source initializations is categorically reduced.

6. The list of large prime numbers contains 3 numbers:
   - 47911*2^5568+1, suitable for input with maximum length 697 bytes (which covers the encryption PKCS#8 of a RSA private key 1024 bits length)
   - 139191930*(10^3120-1)/99999999+1, suitable for input with maximum length 1295 bytes (which covers the encryption PKCS#8 of a RSA private key 2048 bits length)
   - 4974*10^4796+1, suitable for input with maximum length 1992 bytes (which covers the encryption PKCS#8 of a RSA private key 3200 bits length)

   These are surely prime numbers, as shown on: primes.utm.edu [4] with formulas shown above or with "database id" 38335, 29333, respectively 26086.

7. Any larger input will be split in chunks of maximum length 1992 bytes.

8. The shares have the format: v, pId, i, len0, data0[len*, data*...], where v is 1 byte version (0 in this moment), pId is 1 byte big prime id, i is 1 byte public index (it has to be considered unsigned), following a number of chunks defined by 2 bytes, big-endian length len and the proper chunk.

5. Developments

The implementation of the technology is using java smartcards.
In the book "Handbook of Applied Cryptography" [6], we find "One method is to prevent group members themselves from accessing the value of the recovered secret, as may be done by using a trusted combining device.
This is appropriate for systems where the objective is shared control, and participants need only see that an action is triggered, rather than have access to the key itself. For example, each share might be stored on a chip card, and each user might swipe its card through a trusted card reader which computes the secret, thereby enabling the critical action of opening an access door." So, the first scenario is to protect the chunks by writing them to smartcards. The private key is shared on "n" smartcards to the Secret Holders. A

ISSN 2067 – 4074

number of minimum "k" Secrets Holders, carrying their smartcards, has to come together in order to recombine the private key. The "Retriever" simply recomputes the secret without storing the content of the card. A second scenario is the usage of smartcard as holder and manager of private key. The application aims to sign a bunch of data and to retrieve a certificate chain.

6. Results

At the time of writing the proposal, the method was successfully tested with Axalto egate smartcards, but any java card can be used to implement the algorithm.

The key recovery can be used by any other certification authority in Europe, in order consolidate the cross-recognition processes needed for European interoperability of certificates, the goal being to ensure that a certificate released in one country by a trusted third party can be accepted all over Europe. One of the Wspes objectives is to maintain a European list of accredited CA's. A European directive could legalise this list. In this case, WSPES could play the role of a neutral registry maintainer, like Eurid does today for the .eu domain names. It would be up to every country to send their list of accredited national Certification Authorities, according to the process described below, in *Figure 1.*

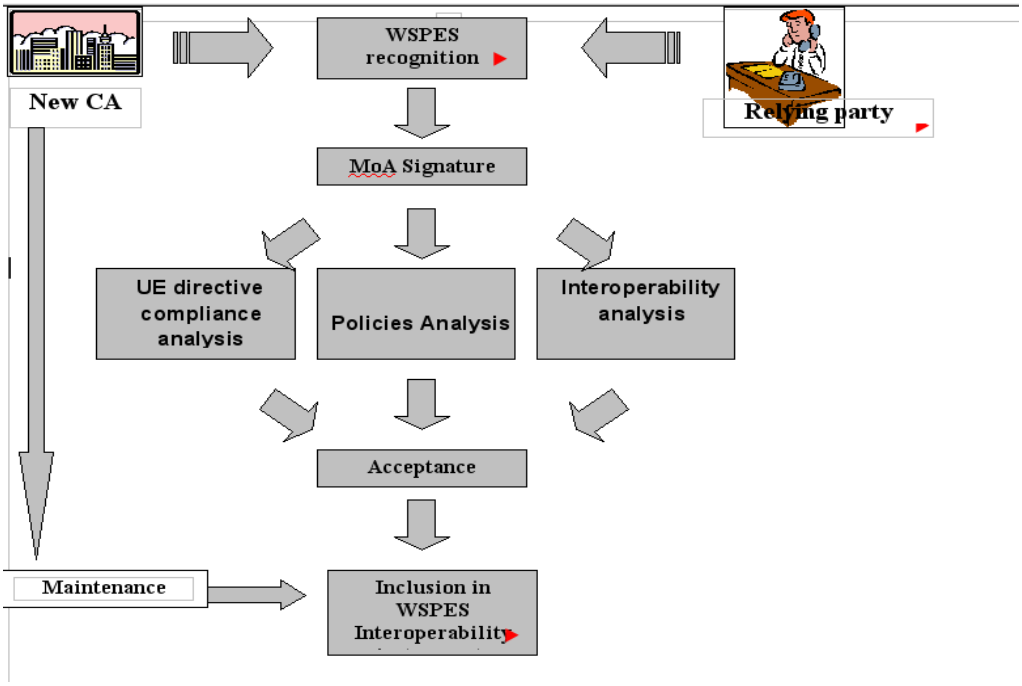

Figure 1. Process

7. Business Benefits

This technique can also be used in other "split secrets" applications.
When a majority agreement is required for taking an action, the access to a secret private key has to be granted based on a "k from n" schema. Let's suppose that an Administration Council composed by 9 members has to decide and to electronically sign a contract. To "unlock" the signature will require 5 members' secret shares! Imagine a shared value is locked by a group of interested parts. A Van Gogh painting belonging to Toshiba company, ensured by Lloyds, showed in Louvre Museum. There are situations when some parts are not likely to participate (why a Toshiba manager has to be available in

Paris to "unlock", or a museum employee in Tokyo?). The "lock" could be open by a smaller number of parts, but never only by one.
Another possible application is the secure access to a website content. The secret is split in three fragments. One fragment stays on the server, another one is distributed to the user, and the third one is kept for safety. We can reconstruct the key with 2 out of 3 fragments. All the data on the website is encrypted with the secret key. After authentication, the user presents the one fragment in his possession, the server adds the second one; the document is either, decrypted on the fly and presented for download on the server, either decrypted locally by a standalone application, taking the second fragment from the server.

8. Conclusions

The method described is easy to implement with existing security devices. The implementation using javacards already won the Innovation prize at Card Tech, Secure tech, awarded by Axalto, St Microelectronics and Sun Microsystems.
Keynectis root key is included in the major operating systems and browsers (Microsoft, Mozilla/Netscape, Linux, Adobe), easing the exchanges between different CA's. Keynectis root key can be used by W-SPES and external partners and administrations in two different ways:

- for cities already having an internal CA root key ,Keynectis will collect all the regional CA's keys, and support the cross-recognition service;
- for Cities that don't yet have a Certification Authority hierarchy, Keynectis will distribute sub-CA certificates.

References

[1] http://www.keynectis.com - Commercial site

[2] http://www.europepki.org - Demonstration site

[3] http://www.wspes.org - Eten project site

[4] http://primes.utm.edu/primes/page.php - Prime numbers

[5] http://www.schneier.com/book-applied.html - Applied Cryptography Book

[6] Menezes, A., Van Oorschot, P., Vanstone, S., *Handbook of Applied Cryptography*, CRC Press, 1996, page 526