Proving Key Usage*

Malek Bechlaghem[1] and Vincent Rijmen[1,2]

[1] Cryptomathic NV
Interleuvenlaan 62/19, B-3001 Heverlee, Belgium
{malek.bechlaghem,vincent.rijmen}@cryptomathic.com
[2] IAIK, Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria
vincent.rijmen@iaik.tugraz.at

Abstract. In the context of a validation server, or more general PKI server, it could be interesting to give the server the possibility to check whether a certain public key was used. We investigate here some possibilities. We also present an escrow system as an application of the technology.

1 Introduction

In this paper we consider an IT system using a *PKI server*. Similar to the idea of authentication servers, which can provide centralized services with respect to access control, authentication and authorization, PKI servers provide centralized PKI services to client applications. After proper authentication to the server, the client can request PKI services. Such systems have been proposed in the past, for different kinds of reasons.

The PKI server introduced here has the capability to check whether a given ciphertext was produced with a given public key, without having access to the message plaintext. Both purely asymmetric and hybrid encryption schemes are considered.

This work is related to the concept of *verifiable encryption* [4]. Verifiable encryption schemes allow to prove certain properties about an encrypted message, without revealing the message. We aren't concerned with properties of the message however, but only examine the key used to encrypt it.

2 Background

We distinguish two classes of PKI servers. In both classes, we can further distinguish between partially trusted servers and completely trusted servers. In general, it is more easy to design systems using completely trusted servers. One design challenge is to reduce the required trust in centrally managed servers as much as possible.

* This work was supported by the funding of IWT Flanders, project: "ITEA 02011 SATURN: Security applications and technologies for universal information networks."

2.1 Voluntary PKI servers

By voluntary PKI servers, we mean servers that provide services which could in principle be provided as well by the client applications. In order to reduce the complexity of PKI applications, the functionality is centralized. We could imagine a system where the functionality is provided by both the client applications and the server, and there is some criterion to select at run-time to perform the service locally or centrally.

A good example are validation servers that relieve the applications from complicated tasks like certificate chain construction, certificate status check, certificate attribute check (e.g. key usage) etc. [9, 1].

2.2 Mandatory PKI servers

By mandatory PKI servers, we mean servers that can't be ignored by the client applications. The PKI trust model is based on the fact that the server plays its role. This type of servers is introduced not only in order to solve practical problems, but also some of the theoretical problems inherent to a distributed PKI model, like for instance providing both non-repudiation and revocation services.

A good example here are server-aided signatures like mediated RSA [3], a variant of RSA where the private key of the user is split into two parts, one owned by the user and the other by a trusted server so that both parts are used during an on-line protocol in order to create an RSA signature. On-line Trusted third parties (TTP) involved in non-repudiation and certified email protocols [5, 11] are another good example of mandatory PKI servers. Such TTP's are involved in order to provide mandatory non-repudiation evidence. The centralized PKI system described in this paper is of this type.

2.3 Setting

When used in an organization, it might be desired to configure the PKI server in a way that it not only provides services, but it also checks on the users/applications. For instance, it can be a policy that users/applications may not accept a signed message unless the PKI server has validated the signature. In such a setting, it may also be desirable that the server checks on outgoing messages. For instance, the server could verify whether the messages are encrypted using the proper keys.

In this paper, we study this problem: how can we give a PKI server the power to check whether outgoing messages are encrypted using the proper key material, without being able to read the messages.

Finding a solution to this problem can be use be useful in several settings. One typical setting could be such that there is outgoing data to destinations with different clearances and encryption keys at different strength levels. The PKI server acting for instance as an enterprize secure messaging gateway, can then check whether outgoing data has been encrypted with a key of the correct strength.

3 Proving Usage of a Public Key

We first discuss the situation where encryption is done with asymmetric encryption schemes. For this situation, we can reuse protocols that haven been proposed in the literature. The schemes will mainly be useful in the next section, where we discuss the situation of hybrid encryption.

In an asymmetric encryption setting, the PKI server sees a ciphertext c and wants to ascertain that c is the encryption of an unknown message m under a given public key. In many public-key schemes, an arbitrarily generated bit string will be a valid encryption of *some* message. Hence, we can rephrase the goal of the server to the verification of the fact that the sender *knows* the plaintext message m.

3.1 RSA encryption

In an RSA setting, usage of a public key can be proven in the following way. Suppose Alice wants to send a message m, which encrypts to $c = m^e \bmod n$. Alice generates a random value r, computes the witness $x = r^e \bmod n$ and submits the tuple (c, x). Subsequently, the PKI server generates a challenge $u \in \{0, 1\}$ and submits it to Alice. Alice then computes the response $y = rm^u \bmod n$ and returns it to the server. Finally, the PKI server verifies whether $y^e = xc^u$.

This protocol corresponds to the Guillou-Quisquater identification protocol [7], but here Alice proves her knowledge of the message m instead of the secret accreditation data. Alice should not send out the same message m to many different recipients, nor should she re-use the random values r.

3.2 Rabin encryption

If the Rabin encryption scheme [10] is used, the ciphertext c is constructed as $c = m^2 \bmod n$. The Fiat-Shamir identification protocol [6] can be used to prove knowledge of m without revealing m.

4 Proving Key Usage in a Hybrid Encryption Scheme

Purely asymmetric encryption is rarely used to protect messages. Instead, hybrid encryption schemes are used. We develop here a scheme that can be used to prove key usage in a hybrid encryption scheme.

Let P be a message to be sent out, consisting of the blocks $P_1, P_2, P_3, \ldots, P_n$. In a encryption hybrid scheme, a random session key is generated and used in a symmetric scheme to encrypt the message. The session key itself is encrypted using an asymmetric encryption scheme.

We present now our scheme in several steps, starting from a very unpractical setting and progressing to a more practical scheme.

4.1 First version

Let the message blocks be split in shares A_i, B_i with $P_i = A_i \oplus B_i$. The strings A and B are composed by concatenating the blocks A_i, respectively B_i. At the end of each string, a *redundancy block* is added, e.g.

$$A_{n+1} = \text{SHA-1}(A_1 A_2 \dots A_n), \tag{1}$$

$$B_{n+1} = \text{SHA-1}(B_1 B_2 \dots B_n). \tag{2}$$

Two random symmetric keys K_A, K_B are generated. Each key is used to encrypt one string, producing the ciphertext strings D and E.

$$D = \text{enc}_{K_A}(A) \tag{3}$$

$$E = \text{enc}_{K_B}(B) \tag{4}$$

The encrypted message consists of the two strings D, E plus the two session keys K_A, K_B encrypted separately under the recipient's public key.

When the message passes by the PKI server, it first checks whether the encrypted session keys are indeed encrypted with the correct public key. This can be done with the asymmetric schemes described in Section 3. Subsequently, the server still needs to ascertain that the symmetric keys encrypted with the public key were indeed used to encrypt the message shares. The server will do this by choosing A or B and asking the sender to reveal the corresponding key K_A or K_B. Suppose the server chooses A. Then the sender is asked to reveal K_A. The server verifies whether K_A encrypts to a value it has seen. Subsequently, the server decrypts D with K_A and checks whether the last block of the decryption indeed obeys the redundancy rule (1).

The server may ask for only one key, hence it will not learn anything about the message. Assuming that it is not feasible to produce a string D and a key K_A that will result in a plaintext with the correct redundancy by any other means than following the scheme, it follows that a sender who doesn't follow the scheme will be caught with probability at least 50%.

Note that this version of the protocol can be simplified somewhat: the PKI server can decide not to use the protocols of Section 3 and only verify the encryption of the session key it selected to be revealed. However, the public-key protocols will be useful in the next version, described below.

4.2 Second version

The first thing we need to improve upon, is the probability to catch cheaters, which may be as low as 50% in the first version of our protocol. We can improve the probability to catch cheaters by splitting up the message P in sub-messages, each containing a part of the blocks P_i. Each sub-message is extended with a redundancy block and its shares are encrypted under a new pair of random keys.

For each sub-message, the server decides between A and B and asks to reveal one key. The probability to catch a cheater increases now with the number of sub-messages that are cheated with: for t malformed sub-messages, the probability becomes $1 - 2^{-t}$.

The protocol can also be extended by having more than two shares for each sub-message. Using s shares, from which the server may ask to reveal any set of $s-1$, the probability to cheat and not be detected reduces to s^{-t}.

Increasing s and/or t results in having to generate and encrypt more session keys (st in total). Hence we need a scheme to derive many session keys from a few values. Let l denote the number of session keys we send over, then we want a derivation scheme that allows to derive st keys in such a way that revealing $(s-1)t$ keys will not disclose any information on the remaining t keys. Note that this demand is sufficient, but not necessary: not all selections of $(s-1)t$ keys can be revealed in practice. We can define t sets of s keys from which at most $s-1$ are revealed. However, we can easily meet the more strict demands.

Let k be the size of a session key. Let X, Y be an l-dimensional respectively st-dimensional (row) vector with coordinates in $\mathrm{GF}(2^k)$. We choose $l = (s-1)t+1$ and compute Y as $Y = X \cdot A$. Here A is a (fixed) $l \times (st)$ matrix with the property that all $l \times l$ sub-matrices are of full rank. If $2^k > st$, then matrices satisfying these constraints can be constructed from linear codes over $\mathrm{GF}(2^k)$, with length st, dimension l and minimal distance t. The constraint $2^k > st$ is easy to meet in practical situations.

Now, in order to save on the number of session keys that need to be generated, protected and transmitted, the following is done. The sender generates a random value K of length l, encrypts it with the receiver's public key and transmits it with the message. The st session keys used to encrypt the message parts are derived from $K \cdot A$. This allows to save the encryption and transmission of $t-1$ k-bit values.

5 Application in a Key Escrow System

The type of PKI server introduced in this paper, can also be used as part of an escrow system.

5.1 Principle

The escrow system consists of two independently operated servers. The first escrow server is a PKI server as described above. It is on-line, sees all communication passing by and has the capability to check whether the messages are encrypted under the proper keys. The second escrow server is off-line. It only receives the messages that need to be decrypted. The public key of the second server is published and available to all users.

Users who want to transmit a message, are obliged to encrypt it twice: once with the key for the intended recipient, and once with the public key of the second escrow server. If the users perform this double encryption dutifully, the second escrow server can decrypt messages when required. The task of the first escrow server is to verify whether the users follow the protocol.

5.2 Message format

Users who want to transmit a message P are obliged to encrypt it in a way similar to discussed in Section 4. We describe the scheme using the version of the protocol described in Section 4.1.

Let the message P consist of blocks $P_i, i = 1, \ldots, n$. The blocks are split into shares $A_i, B_i, i = 1, \ldots, n$. Redundancy blocks aren't necessary here. Four symmetric keys are generated: K_A, K_B, L_A, L_B. The string A is encrypted with a symmetric cipher, once under key K_A to produce the ciphertext string D, and once under key L_A to produce the ciphertext string F. Similarly, B is encrypted under K_B to produce the string E, and under L_B to produce the string G.

The keys K_A, K_B are encrypted under the public key of the recipient, and the keys L_A, L_B under the the public key of the second escrow server. This allows both the recipient and the second escrow server to recover the message P.

The PKI server executes the following steps in order to determine whether the sender has produced messages of the correct format.

1. (Optional) Check whether the symmetric keys L_A, L_B have been encrypted under the public key of the second escrow server.
2. Choose A or B and ask the sender to reveal the keys K_A, L_A, respectively K_B, L_B.
3. Check whether the revealed symmetric key L_A or L_B encrypts to the value that was received with the encrypted message.
4. Use the revealed keys to decrypt D and F, respectively E and G. Check whether the decrypted values are equal. If the values are not equal, then the sender didn't follow the required format.

Senders that don't follow the required format, are caught with probability 50%. The system can be extended in a similar way as described in Section 4.2.

6 Related Work

In the asymmetric encryption setting, our proof of usage of a certain public key, is in fact a proof of knowledge of the plaintext corresponding to the ciphertext under examination. Proving knowledge of the plaintext corresponding to a submitted ciphertext, has been called *plaintext-aware encryption* before [2].

Regarding proving of public key usage for encrypting messages, without having access to the plaintext message, we know of no published papers addressing this issue.

Our protocols are related to protocols based on verifiable encryption [4] since we describe an entity (PKI server in our settings) that tries to find out some property on an encrypted message while the message is given in an encrypted form. Verifiable encryption is meant to ensure that a entity accepts the encryption of an invalid message only with negligible probability. Typically verifiable encryption are used with digital signatures representing hence a way to encrypt a message under a designated public key and subsequently prove that the resulting ciphertext indeed contains such signature. Contrarily to protocols using

verifiable encryption, we are only interested in proving that a designated public key has been used to encrypt a message without having access to the plaintext and to the decryption private key.

The schemes that we developed in this paper allow also implementing a powerful non-repudiated message delivery authority. Such authority has been defined in the literature [8] as providing the sender and recipient of a message with signed, time-stamped proof of non-repudiation of origin, of receipt and of submission. The sender and recipient can use such proofs in resolving disputes occurring between them. Our schemes allow extending such delivery authority with an additional feature which is making sure that the sender and recipient exchange confidential messages so that the messages are encrypted with valid designated keys.

7 Conclusions and Further Work

We presented the idea of a PKI server that can check which key has been used to encrypt messages, without having access to the message plaintext. We described asymmetric and hybrid protocols for achieving this goal. Finally, we explained as a special application how an escrow system can be based on this PKI server.

The hybrid scheme presented in this paper is provided as a first step in a new direction. More research is required to get more certainty about the security level and to develop schemes that are more usable.

Secondly, we think it will be interesting to look at new applications for the primitive introduced in this paper. We are particulary interested in developing new non-repudiation and fair-exchange protocols.

References

1. D. Barbecaru, A. Lioy, "Towards Simplifying PKI Implementation: Client-Server based Validation of Public Key Certificates," in Proceedings of IEEE ISSPIT 2002, 2002.
2. M. Bellare and P. Rogaway, "Optimal asymmetric encryption– how to encrypt with RSA," in Advances in Cryptology, Proceedings of EUROCRYPT '94, LNCS 950, Springer-Verlag, 1995, pp. 92–111.
3. D. Boneh, X. Ding, G. Tsudik, and B. Wong, "Instanteneous revocation of security capabilities," in Proceedings of the USENIX Security Symposium 2001, 2001.
4. J. Camenish, and I. Damgaard, "Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes," in Advances in Cryptology, Proceedings of ASIACRYPT 2000, LNCS 1976, Springer-Verlag, 2000, pp. 331–345.
5. T. Coffey, P. Saidha, "Non-repudiation with mandatory proof of receipt," in ACM-CCR: Computer Rommunication Review 26.
6. A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in Advances in Cryptology, Proceedings of CRYPTO '86, LNCS 263, Springer-Verlag, 1987, pp. 186–194.

7. L.C. Guillou and J.-J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory," in Advances in Cryptology, Proceedings of EUROCRYPT '88, LNCS 330, Springer-Verlag, 1988, pp. 123–128.
8. A. Herzberg, "Guaranteed delivery for secure electronic commerce and payments," unpublished manuscript.
9. M. Jalali-Sohi, P. Ebinger, "Towards Efficient PKIs for Restricted Mobile Devices," in Proceedings of IASTED International Conference Communication and Computer Networs, 2002.
10. M.O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.
11. J. Zhou, D. Gollman, "Certified electronic mail," in Proceedings of Computer Security - ESORICS'96, 1996.