A Generic Method of Detecting Private Key Disclosure in Digital Signature Schemes

Feng Bao
Institute for Inforcomm Research
1 Fusionopolis Way, #21-01 Connexis
Singapore 138632

Abstract-**Digital signature is very critical and useful for achieving security features such as authentication, certification, integrity and non-repudiation etc. In digital signature schemes, private keys play the most fundamental role of security and trust. Once a private key is compromised, the key owner loses all of the protection to himself so that he can be impersonated. Hence it is crucial for a private key owner to know whether his key has been stolen. The first study toward detecting private key disclosure is [4], where the schemes based on the time-division and private key updating are presented. The approach is similar to the forward-secure signature in the key-update style. In this paper we propose a completely different approach for a user to detect whether his private key for signing digital signatures is compromised. The solution satisfies the four attractive properties: 1) the user need not possess another cryptographic key and what he has are his private key and a memorable password; 2) the signature schemes are not in the update of the private key in time-divided manner and our method can be applied to the existing signature schemes; 3) although a trusted party (TP) is required in our method, the user and the TP need not share any secret; and 4) the user is stateless, i.e., he does not need to record all the messages and the signatures he has signed before.**

I. Introduction

Public key cryptosystems have been widely adopted as very important and effective tools for information security. Digital signature is the most powerful technology for authentication, integrity and non-repudiation etc. It has been accepted as a legal evidence of commitment in the legislation of many countries.

All the security systems which take digital signature as a building-block component establish the security mechanism based on the secrecy of the private signing keys. The security is completely compromised if the private keys are stolen or get lost. In the real world people do lose things for various reasons. However, the damage can be reduced to minimum if the disclosure of the private keys is detected as soon as possible so that the key owners can cancel the compromised keys and replace them with new ones.

The detecting task is not trivial since in the existent digital signature schemes, there is no difference between the signatures signed by the key owners and the signatures signed by the hackers who have stolen the private keys. A trivial solution is that the signer records all the messages and the signatures he has signed before. Apparently, such a solution introduces cost in memory and the checking of the suspected signature needs the comparison with all the previous record. A stateless solution is desired that has no such record.

Our method generates differences that distinguish the real signatures by owners from the signatures signed by hackers. It is obviously that the users must know something which the hackers do not have. We also want to limit the size of such unique secrets owned only by users such that they can be remembered by human-being and hence need not be stored in user's machines. Otherwise, it makes no sense as the hacker could steal both of the keys. In our methods, those secrets can be taken as memorable passwords as short as 4 digits only.

Our method is very generic, which can be applied to all the practically being used digital signature schemes. The detecting procedure is very simple and easy to implement. The methods are aimed at helping users to determine whether

CHINACOM 2010, August 25-27, Beijing, China

DOI 10.4108/chinacom.2010.138

their signing keys are compromised or not, rather than helping the users to prove to other people that their keys are compromised.

The first paper in detecting key disclosure is [4], where the method is to update the private key periodically. That is the so-called time-divided manner in which time is divided into many intervals and a different private key is used for each interval. The techniques employed in [4] are similar to the forward-secure signature schemes. The basic idea of that scheme is that the private key is updated periodically, say, SK_0, SK_1, SK_2, …, SK_n, where SK_i is used only for the i-th period. The private key update algorithm is a randomized algorithm in the sense that different people would obtain different SK_{i+1} even if they update the same SK_i. If a hacker succeeds in stealing SK_i, he will still have a different SK_{i+1} and hence be caught if he signs with SK_{i+1}. But this method cannot be used to distinguish the signatures generated by the hacker from the signatures generated by the owner in the i-th period. Another drawback of the method is that it cannot be applied to existent signature schemes. Users must give up the schemes they are using and turn to the time-divided schemes, which have not been proved to be accepted by practical users. Our methods can be applied directly to the existent schemes. In other words, the users can still use their private keys and signature schemes they have been using in our approach. But in the method of [4], they must change their private keys and the signature schemes.

The basic idea of our method is to utilize a memorable password, which is not stored in any of user's devices. It is remembered by the user. Hence the password must be short enough. In that case, it could be subject to dictionary attack. The research on the password-based authentication key exchange (PAKE) against dictionary attack has been studied for many years, see [1, 2, 3, 5, 6, 7, 8, 9]. Our purpose is to achieve the same security as PAKE schemes, i.e., the password cannot be compromised by dictionary attack. As a result, an attacker has to try many times with a server in order to catch the user's password. The security can be guaranteed by limiting the number of times of false trials.

As an analogy, our method is like to attach a unique gene to a private key. The gene is the password that is remembered by the user. Such a gene makes the signature signed by the user different from the signature by others with the same private key. However, the difference can only be detected by DNA test, which can only be conducted by a trusted centre.

The paper is organized as follows. We describe the model of our schemes in Section 2. We describe the new generic method in Section 3. In Section 4, we present some concrete examples of the signature schemes. Section 5 concludes the paper.

II. The Model of Detecting Key Disclosure

Entities involved
Users – there are multiple users, each user has a unique id
TP – trusted party

Keys involved
User with id x has a pair of public/private keys (*PKx, SKx*) for signature verification and generation respectively; in addition, he has a memorable password *PWx,* which is not stored anywhere except in his brain. TP has a public/private key pair (*PKt, SKt*) for public key encryption.

Signing
When user x wants to sign message *M*, he inputs his password *PWx* into his machine, the signature *s* is computed by the signing function sign:

$$s=\mathbf{sign}(M, SKx, PKt, PWx)$$

The password *PWx* is removed from the machine after signing while *SKx* is still stored in the machine.

Verification
Verification requires only the public key of the signer. It can be conducted by any party. This is simply the same as in the ordinary signature schemes

$$\mathbf{veri}(M, s, PKx)=\text{yes or no}$$

Identifying
When user x sees a signature *s* which is his signature on message *M*, x wants to identify whether this signature was signed by himself or by other people. If *s* is identified as not signed by himself, x knows that his key *SKx* must have been compromised. The identification procedure is very simple. The user x just sends a request along with the signature he wants to identify to the TP. TP returns an answer yes or no. If the

answer is yes, the tested signature was signed by user x. If the answer is no, the tested signature was generated by someone else and in this case user x knows that his private key has been compromised.

Security of identification

The identification procedure is dictionary-attack-resistant. No one can get the password *PWx* by dictionary attack even if he is allowed to intercept all the communications between user x and TP. A hacker who has not compromised user x's private key cannot produce valid request. The TP does not answer invalid request. A hacker who has compromised *SKx* can generate valid request but can exclude only one possibility of *PWx* by implementing the identification request once. The TP should inform the user if the number of requests it receives that have negative answers exceeds a small limit. This is to prevent the hacker who has *SKx* from finding *PWx* by online attack. The user knows that *SKx* must have been compromised if someone else can generate valid requests with negative answers.

III. Description of Our Method

We describe our method by 4 figures. They clearly illustrate our method.

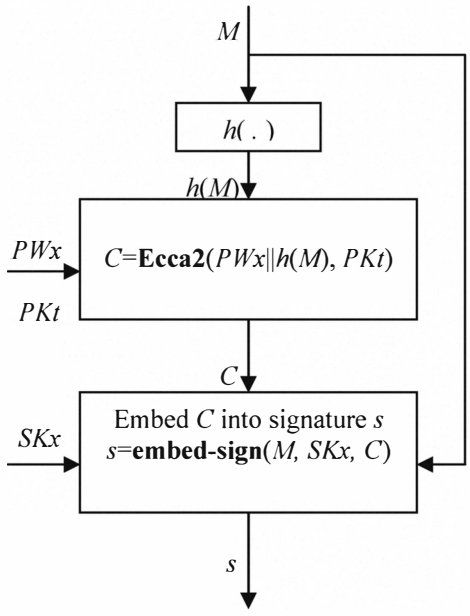

Figure 1. Flowchart of signing.

FIGURE 1 is a model of signing process, where **Ecca2**(*M, K*) denotes the public key encryption of *M* by encryption key *K* in CCA2 manner and **Esym**(*M, K*) denotes the encryption of *M* by key *K* with a symmetric key cryptosystem, such as DES or AES and *h*(*M*) denotes the hash of *M*. embed-sign will be illustrated later.

FIGURE 2 depicts the identification process related to Figure 1.

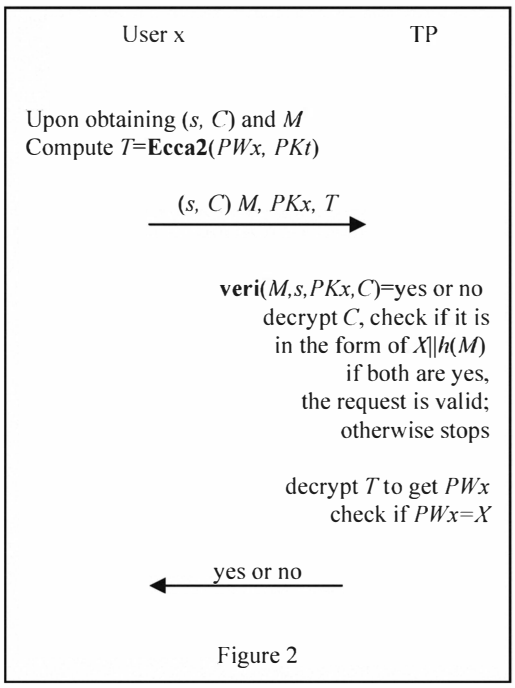

Figure 2

FIGURE 3 is an improved version of the scheme in Figure 1.

In the scheme of Figure 1, everyone can retrieve *C* from the signature. This is not a desired feature in some situations. For example, in this case the TP can obtain any user's password without the user's permission. In the improved version in Figure 3, the TP cannot obtain a user's password unless the user seeks to proceed the identification process with TP. We denote by **Esym**(*M, K*) the symmetric key encryption with secret key *K*, such as DES, 3-DES, AES etc.

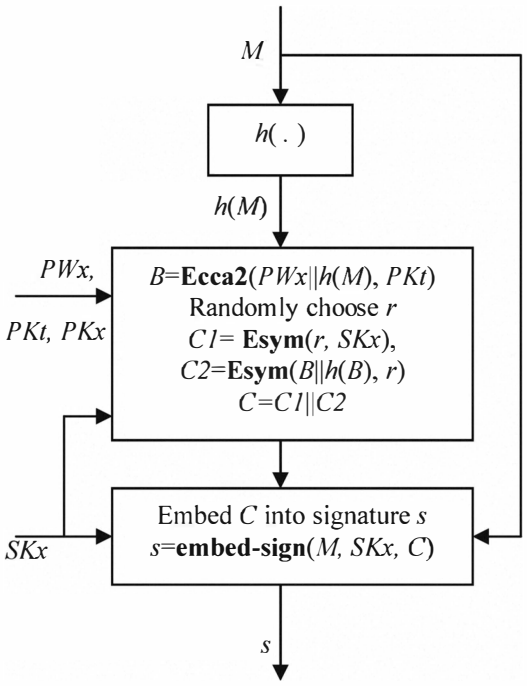

FIGURE 3 Flowchart of the signing process

FIGURE 4 depicts the identification process related to Figure 3.

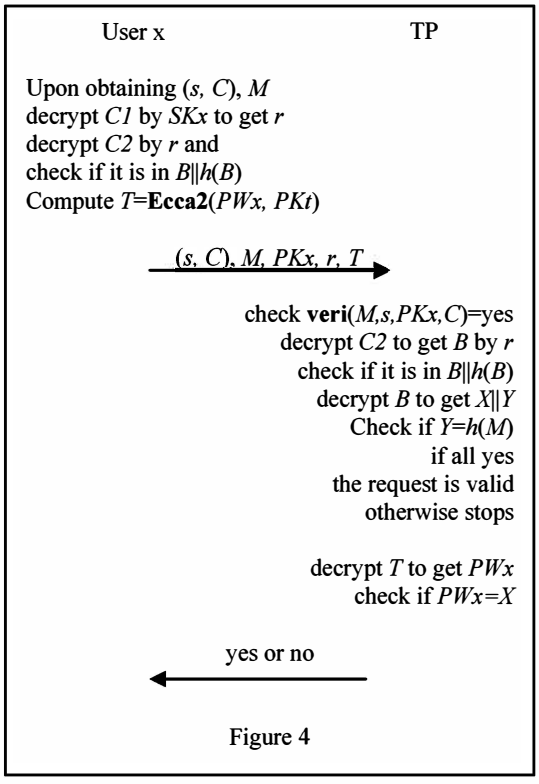

Figure 4

Security Discussion

Why we need a trusted party. The password must be protected by a cryptographic key when it is embedded into the signature. Otherwise the password can always be found by brute force search.

Password will not be compromised. Although *C* is disclosed to everyone, it is impossible to get *B* without knowing the private key *SKx*. If *SKx* is compromised by a hacker, he can find out *B*. But he has no chance to find out *PWx* because the encryption is CCA2. In other words, even if the hacker obtains *B* and he knows *PWx* is either 0 or 1, he cannot tell from *B, M, PKt* whether *PWx* is 0 or 1. Offline dictionary attack is impossible due to the adoption of CCA2 public key encryption.

Replay attack. Suppose a hacker steals the private key *SKx*. He can retrieve *B* from *C*. If he reuses the *B* for the message other than *M*, the signature will be identified by TP as forged signature. This is because *M* is embedded in *B* and it will be checked by TP. If the hacker signs the same message *M* with the retrieved *B*, it is impossible to distinguish the forged signature from the real one. But it makes no sense for the hacker to sign a message which has already been signed by the owner.

Online dictionary attack. The hacker with *SKx* may impersonate the key owner and go to TP repeatedly to request Identification. He can guess the password and try one by one. Each time the TP would have an invalid request. The TP can set up an uplimit on the number of invalid requesst. Once the number of invalid requests succeeds the limit, offline resolution will be sought.

IV. Concrete Embed-Sign Implementation

Embodiment of embed-sign

There are many ways to embed a message *C* into a signature. The most general method is to concatenate *h*(*C*) to the signed message *M* and sign *M*||*h*(*C*) instead of *M* only. The *C* is attached to the signature s such that (*s, C*) is the final signature. The new signature is tamper-resistant in the sense that no one can modify the signature (e.g., to change *C* or remove *C*) such that it is still valid.

For the signature schemes which are converted from interactive Zero-Knowledge Proof, a better method of embedding is to embed the C into the hash function. In this sort of signature schemes, the random challenge is replaced by the hash value of message, public key and a randomly chosen number. The detailed description is as follows.

Preferred embodiment of embed-sign

Let user x take Schnorr signature or any other signature schemes which are transferred from a zero-knowledge proof based on the random oracle of hash function. The message C can be embedded into the hash function. The concrete construction by Schnorr signature is as follows.

- C – the message to be embedded
- M – the message to be signed
- P – the prime modulo in Schnorr signature
- Q – a large prime factor of P-1, $|Q|>160$
- G – a number of order Q
- SKx – the private key of x, a random number from $\{1,2\ldots,Q-1\}$
- PKx – the public key of x, $PKx=G^{SKx} \bmod P$

Embed-sign:

Randomly choose w from $\{1,2\ldots,Q-1\}$, set $W=G^w \bmod P$

Set $r=h(C, M, W)$ and $s=w+rSKx \bmod Q$

(s, r, C) is the signature of M.

Verification:

Given (s, r, C), M and PKx, check if $r=h(C, M, G^s(PKx)^{-r} \bmod P)$

V. Conclusions

In this paper we propose a new approach to detect whether a private signing key has been compromised. Of course the assumption is that the hacker signs new messages with the stolen private key. There is no solution if the hacker keeps the private key without using it. Our approach is different from the previous solutions by taking a different model. The advantage of our solution is that we do not need key update and there is no time-division. The proposed method applies to practically used signature schemes. The scheme needs a trusted party, which is unable to compromise user's private key and password from the signature. The trusted party can be a service to users, who can choose from which trusted party to request the service.

References

[1] S. M. Bellovin and M. Merritt, "Encrypted key exchange: password based protocol secure against dictionary attack", Proceedings of 1992 IEEE Symposium on Security and Privacy, pp. 72-84, IEEE Computer Society, 1992.

[2] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attack", Proceedings of Eurocrypt 2000, LNCS, Springer-Verlag, 2000.

[3] V. Boyko, P. MacKenzie and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman", Proceedings of Eurocrypt 2000, pp. 156-171, LNCS, Springer-Verlag, 2000.

[4] Gene Itkis, "Analysis and verification: Cryptographic tamper evidenceT", Proceedings of the 10th ACM conference on Computer and communication security (ACM CCS), pp. 355 – 364, Oct 2003.

[5] D. P. Jablon, "Strong password-only authenticated key exchange", Computer Communication Review, ACM, 26(5), pp. 5-26, 1996.

[6] J. Katz, R. Ostrovsky and M. Yung, "Efficient password-authenticated key exchange using human memorable passwords", Proceedings of Eurocrypt 2001, LNCS 2045, Springer-Verlag, 2001.

[7] S. Lucks, "Open key exchange: How to defeat dictionary attacks without encrypting public keys", Proceedings of the security Protocols Workshop, LNCS 1361, pp. 79-90, Springer-Verlag, 1997.

[8] P. MacKenzie, S. Patel, and R. Swaminathan, "Password-authenticated key exchange based on RSA", Proceedings of Asiacrypt 2000, pp. 599-613, LNCS, Springer-Verlag, 2000.

[9] T. Wu, "The secure remote password protocol", Proceedings of 1998 Internet Society Symposium on Network and Distributed System Security, pp. 97-111, 1998.