On the Ambiguity of Concurrent Signatures

Yi Mu[1], Fangguo Zhang[2], and Willy Susilo[1]

[1] School of Information Technology and Computer Science,

University of Wollongong, Wollongong, NSW 2522, Australia

[2] Fangguo Zhang: School of Information Science and Technology,

Sun Yat-Sen University, Guang Zhou, 510275, China

Email: {ymu,wsusilo}uow.edu.au, isdzhfg@zsu.edu.cn

Abstract

We point out that the notion of *ambiguity* introduced in the concurrent signatures proposed in [4] is incorrect. Any third party who observed two signatures can differentiate who has/have produced the signatures by performing the verification algorithm. We note that the model proposed in [4] is sound, but the concrete scheme does not really provide what is required in the model.

1 Introduction

The issue of fair exchange of digital signatures is a fundamental problem in cryptography. Fair exchange of digital signatures has many applications such as fair contract signing, fair exchange of digital goods, etc. Fairness in exchanging signatures is normally achieved with the aid of a trusted third party (TTP) (e.g., [3, 2]). Recently, Chen, Kudla, and Paterson in Eurocrypt 2004 [4] introduced a new notion called "concurrent signature". The concurrent signature scheme allows two parties to produce two signatures in such a way that, from any third party's point of view, both signatures are *ambiguous* with respect to the identity of the signer until a *keystone* is released by one of the parties. Upon releasing the keystone, both signatures become binding to their true signers concurrently. Actually, the current signature scheme falls just short of solving the full TTP problem in fair exchange of signatures. In concurrent signatures, the participants' public keys must be certified by a TTP beforehand, but the TTP can be inactive afterwards. Moreover, there is no requirement that TTP will help solving the problem in the case of dispute (c.f. optimistic fair exchange constructions in [3, 2, 5]).

We note that the notion of concurrent signature is novel and sound. As pointed out in the original concurrent signature paper, concurrent signatures can be constructed by using a variant of the ring signature scheme by Abe et. al. [1]. Conceptually, they are two-party ring signatures with extra information (so-called the "keystone"). Both signers sign with a ring signature scheme, respectively. Due to the signer's ambiguity of ring signatures, the signer's identity is ambiguous to any third party. It is claimed in [4], such ambiguity can be removed and the signer's identity becomes clear by releasing the keystone. This model is very attractive and in fact, this would provide a good mechanism for providing an optimistic fair exchange (although it has some additional requirements namely the certification of the public keys by the TTP at the initialization phase).

Chen *et. al.* provides a concrete concurrent scheme in their seminal paper in [4]. Unfortunately, as we shall show in this paper, the concrete scheme does not provide the signer's ambiguity as required by their model. Any third party can distinguish who the actual signer is by performing the verification algorithm.

2 Concurrent Signatures

In this section, we briefly review the concrete concurrent signature scheme proposed by Chen, Kudla, and Paterson [4]. We omit the formal model introduced in [4], and we refer the reader to [4] for more complex account. The protocol consists of four phases: SETUP, ASIGN, AVERIFY, VERIFY.

- SETUP is probabilistic algorithm that sets up all parameters including keys. It selects two large primes p, q for $q|p-1$ and a generator $g \in \mathbb{Z}_p^*$ of order q. It also generates two cryptographic hash functions $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$. Say, Alice and Bob are two parties involved in the system. Upon completion of the setup, Alice obtains her private key $x_A \in \mathbb{Z}_q$ and the corresponding public key $X_A = g^{x_A} \bmod p$ and Bob obtains $x_B \in \mathbb{Z}_q$, $X_B = g^{x_B} \bmod p$ as his private key and public key.

- ASIGN is a probabilistic algorithm that takes as input (X_A, X_B, x_i, f), where $i \in (A, B)$ and $f = H_1(k)$ for the keystone $k \in \{0,1\}^*$ and outputs an ambiguous signature σ_i. To sign a message M_A, Alice picks a random $r \in \mathbb{Z}_q$ and a keystone k and then computes $f = H_1(k)$ and

$$h = H_2(g^r X_B^f \bmod p \| M_A),$$

$$h_A = h - f \bmod q,$$

$$s_A = r - h_A x_A \bmod q.$$

The output from the algorithm is the signature on M: $\sigma_A = (s_A, h_A, f)$, which is then sent to Bob.

- AVERIFY is an algorithm that takes as input $S_i = (\sigma_i, X_i, X_j, M_i)$ and outputs *accept* or *reject*. Given $S_A = (\sigma_A, X_A, X_B, M)$, Bob checks the equality:

$$h_A + f \stackrel{?}{=} H_2(g^{s_A} X_A^{h_A} X_B^f \bmod p \| M_A) \bmod q. \tag{1}$$

If it holds, accept; otherwise, reject. If the signature is accepted, Bob signs message M_B by using (X_A, X_B, x_B, f). The resulting signature is $\sigma_B = (s_B, h_B, f)$, where

$$h' = H_2(g^r X_A^f \bmod p \| M_B),$$

$$h_B = h' - f \bmod q,$$

$$s_B = r' - h_B x_B \bmod q.$$

r' is a random number selected from $\mathbb{Z}_q$. He then sends σ_B to Alice. Upon receiving σ_B, Alice checks whether or not f is the same as the one used by herself. If not, abort. Otherwise, Alice checks the equality:

$$h_B + f \stackrel{?}{=} H_2(g^{s_B} X_B^{h_B} X_A^f \bmod p \| M_B) \bmod q. \tag{2}$$

If it holds, Alice sends k to Bob.

- VERIFY is an algorithm that takes as input (k, S_i) and checks if $H_1(k) = f$. It not, it terminates the process. Otherwise, it runs $AVERIFY(S_i)$.

In the original paper, it was claimed that both σ_A and σ_B hold ambiguity and after k is released by one of parties, the ambiguity is removed. Therefore, the goal of the protocol is achieved.

3 On the Identity Ambiguity of the Concrete Concurrent Signatures

Firstly, we show that that the following statement from [4] is not satisfied in the concrete scheme provided.

> "Signer identities of exchanged signatures are ambiguous to any third party before releasing the key stone."

Proof. Assume that Alice has constructed her signature σ_A on M_A and sent it to Bob. We show that Bob can prove to any third party that Alice has indeed signed M_A without any aid of the keystone. In other words, the ambiguity will be broken after Alice has sent her signature to Bob. We observe that

- $h_A \neq h_B$.
- The verification equation (1) shows that X_B^f has to be computed for the verification of Alice's signature. Similarly to that, equation (2) shows that X_A^f has to be computed for the verification of Bob's signature.

In order to prove that Alice has indeed constructed $\sigma_A = (s_A, h_A, f)$, Bob signs a random message M, which could be given by a third party, to construct $\sigma_B = (s'_B, h_B, f)$, where s'_B is Bob's signature on M. Following the observation above, the third party is sure that σ_B was signed by Bob and σ_A was signed by Alice because f now becomes clear. In other words, if X_B^f is used in the verification, the related signature must be constructed by Alice; otherwise, if X_A^f is used, the signature must be constructed by Bob. □

We would also like to point out that the following statement provided in [4] is not satisfied by their concrete scheme.

> "From a third party point of view, before the keystone is released, both parties could have produced both signatures, so the signatures are completely ambiguous."

Proof. Assume that Alice has constructed two signatures $\sigma_{1A} = (s_{1A}, h_{1A}, f)$ and $\sigma_{2A} = (s_{2A}, h_{2A}, f)$, then the verifications of both signatures require to compute X_B^f. This therefore reveals the identity of the signer without using the keystone.

We note that there is nothing wrong when a single signature is constructed from a ring signature scheme, since h_i and f are undistinguishable to the verifier. The problem main appears when two signatures are constructed. Observe that f is the hash value of the keystone. If f is associated with the public key of the other party during the verification, then k must also be associated with the same public key. Since f is made public, revealing k becomes redundant. If f does not break ambiguity as claimed in [4] (i.e., f is not associated with any public key in the verification), then revealing k will only reveal f and will not help to break the ambiguity of signer's identity. □

We also note that in the concrete scheme proposed in [4], the keystone is redundant. In fact, the keystone can be treated as a confirmation key. That is, the signatures are valid only when the keystone is revealed (say, after Alice has verified Bob's signature).

However, this clearly was not the original idea. In fact, such signature confirmation can be achieved by using a second signature. Namely, Alice signs σ_A after she has verified Bob's signature.

Theorem 1 *The proposed scheme provided in [4] does not satisfy the identity ambiguity as required in their model.*

Proof. Can be deduced from our arguments above. □

Another issue in the concurrent signature scheme is that the signer, either Alice or Bob, can produce multiple signatures by using the same keystone. Taking Alice as an example, she can generate two signatures, $\sigma_A = (s_A, h_A, f)$ on m_A and $\sigma'_A = (s'_A, h'_A, f)$ on m'_A. She sends only σ_A to Bob. After the keystone is released, she can claim that she actually commit to σ'_A. The same method also applies to Bob. A multiple signing can be done, even if the keystone has been released. It really makes the situation worse that any signature can be claimed to be legitimate.

4 Conclusion

We showed that the concrete scheme in [4] does not satisfy the model provided in their paper, namely the identity ambiguity. The model of concurrent signatures is novel and sound, but the construction remains as an open problem at this stage.

References

[1] M. Abe, M. Ohkubo, and K. Suzuki, "1-out-of-n signatures from a variety of keys," in *Advances in Cryptology–ASIACRYPT 2002,* LNCS 2501, pp. 415–432, Springer-Verlag, Berlin, 2002.

[2] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *IEEE Journal on Selected Areas in Communications*, vol. 18, 2000.

[3] F. Bao, R. Deng, and W. Mao, "Efficient and practical fair exchange protocols," in *Proceedings of 1998 IEEE Symposium on Security and Privacy, Oakland*, pp. 77–85, 1998.

[4] L. Chen, C. Kudla, and K. G. Paterson, "Concurrent signatures," in *Advances in Cryptology, Proc. EUROCRYPT 2004,* LNCS, Springer-Verlag, Berlin, pp.287–305, 2004.

[5] Y. Dodis and L. Reyzin. Breaking and Repairing Optimistic Fair Exchange from PODC 2003. *ACM Workshop on Digital Rights Management (DRM)*, 2003.