Theoretical Computer Science 562 (2015) 194–212

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Concurrent signature without random oracles

Xiao Tan [a,c,*], Qiong Huang [b], Duncan S. Wong [c]

[a] School of Information Science and Engineering, Hangzhou Normal University, China
[b] College of Informatics, South China Agricultural University, China
[c] Department of Computer Science, City University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 17 May 2014
Accepted 23 September 2014
Available online 30 September 2014
Communicated by G. Ausiello

Keywords:
Concurrent signature
Fair exchange
Ambiguity
Anonymity
Accountability
Provable security
Multi-user setting

ABSTRACT

A concurrent signature provides an efficient way to exchange digital signatures between parties in a fair manner. Since its introduction in Eurocrypt 2004, removing the random oracle heuristic in the security analysis of a concurrent signature scheme has become an open problem, and the security of all the existing provably secure schemes could have only been done in the random oracle model, while it has been known that the security in the random oracle model may not be guaranteed when the underlying random oracles are replaced by real-life hash functions. In this paper, we solve this open problem by proposing a new concurrent signature scheme, which allows us to prove its security without random oracles. The security model we consider in this paper also slightly differs from previous works. Signatures before revealing the keystone are strongly ambiguous (or *anonymous*) in the sense that everyone is able to produce signatures that are indistinguishable from those generated honestly by the parties involved in the exchange, while signatures after revealing the keystone remain unforgeable without sacrificing the fairness property. In the multi-user setting and without random oracles, we prove the security of our scheme based on the intractability of Computational Diffie–Hellman (CDH) problem and collision resistance of hash functions.

1. Introduction

Concurrent Signature, introduced by Chen, Kudla and Paterson [8], allows untrusted parties to exchange their digital signatures efficiently in a fair manner, that is, either allowing all the parties to get each other's signatures simultaneously or letting none of them get any counterpart's signature, in an all-or-nothing fashion. A concurrent signature scheme between two communicating parties, usually being referred to as an initial signer A with a public/private key pair (pk_A, sk_A) and a matching signer B with (pk_B, sk_B), is typically carried out interactively in the following three phases.

(1) **Keystone Generation Phase**: A sets a secret k called *keystone*, then generates a *keystone-fix* f as a commitment of k, and sends f to B.
(2) **Ambiguous Signature Generation Phase**: A generates an *ambiguous signature* σ_A on a message m_A with respect to the keystone-fix f, and sends σ_A to B. σ_A is generated in such a way that no one except B can tell whether σ_A is indeed generated correctly by A under the reason that B himself did not generate σ_A. After the verification of σ_A, B then

[*] Corresponding author.
E-mail addresses: xiaotan.cs@gmail.com (X. Tan), csqhuang@gmail.com (Q. Huang), duncan@cityu.edu.hk (D.S. Wong).

http://dx.doi.org/10.1016/j.tcs.2014.09.044
0304-3975/© 2014 Elsevier B.V. All rights reserved.

generates his own ambiguous signature σ_B on a message m_B with respect to the same keystone-fix f and sends it back to A.

(3) **Signature Binding Phase**: A reveals the keystone k after verifying σ_B. The keystone k will bind the authorship of σ_A (resp. σ_B) to A (resp. B) concurrently. An ambiguous signature together with the released keystone is referred to as a *binding signature* (e.g. (k, σ_A) or (k, σ_B)).

In the Signature Binding Phase, the concurrency of binding upon the reveal of keystone ensures that either both parties get each other's binding signature, or neither of them does. Since the introduction of concurrent signature [8], it has been considered as a type of fair exchange protocols [13,15,2,18,16]. If we compare concurrent signature with other fair exchange solutions such as timed-release fair exchange [13,15], or optimistic fair exchange (OFE) [2,18,17], we will notice that concurrent signature usually achieves higher computational and communication efficiency, and does not rely on any trusted or semi-trusted third party for dispute resolution or assume computational balance between the parties.

It is worth noticing that concurrent signature is not so-called *abuse-free*. For a typical concurrent signature interaction between an initial signer A and a matching signer B, A has the full control on *when* and *whether* to reveal the keystone k in the final Signature Binding Phase. This might give A certain extent of advantage over B in some applications [39]. However, in many other applications as initially proposed in [8], concurrent signature is a very useful tool for realizing fair exchange of signatures. For example, it can be applied for trading new artworks via e-market websites that benefit both art fans and emerging artists. Without loss of generality, suppose two customers X and Y intend to buy an original painting from the seller Z. The fairness should guarantee that both the customers can offer a price for the painting of their own will, and the seller is able to make a deal with the customer who offered a higher price. Then X (resp. Y) can run a concurrent signature with Z and sign his/her offer as σ_X (resp. σ_Y) independently. After receiving the ambiguous signatures from both X and Y, Z only completes the session with the party whose offer is better. In this application scenario, X and Y are competitors, so they have no motivation to show their own keystone to each other. Besides, Z can always get the keystone from the customer who gave the better offer, because the customer needs to use Z's binding signature (say (k, σ_Z)) to take the painting from Z.

1.1. Motivation and contributions

Since Chen et al. published their seminal paper in 2004 [8], there has been a number of concurrent signature schemes proposed [30,33,24,9,32,39,29,31]. Some work have the ambiguity model improved [30,24] and the fairness requirement further enhanced [33]. Some others focus on extending concurrent signature to multi-party setting [32,31] or identity-based setting [9], or balancing the capability of controlling the release of keystone between the initial signer and the matching signer [39], or evaluating the scenarios for which concurrent signature is free of abused usage [29]. On the security analysis, however, all of those schemes have their security shown under the random oracle assumption [4] only. It is known that the random oracle model is a heuristic methodology which assumes that all the involving parties have oracle access to some truly random functions. However in [6], Canetti et al. pointed out that the random oracle model is potentially having some structural imperfectness. They gave an example showing that a scheme secure in the random oracle model can become insecure if the underlying random oracles are replaced with some real-life hash functions in practice. As a result, for the security analysis of concurrent signature, it has been a well-known yet long-lasting open problem to construct a new concurrent signature scheme, and show its security without random oracles.

Ambiguity is an important feature to achieve in many variants of signature schemes [7,38,21,37,19] as well as in concurrent signature. Though the current ambiguity model for concurrent signature [8,30] requires that the ambiguous signatures are non-self-authenticating [20,40] when the keystone is not revealed yet, the ambiguous signatures already leak the following information to the public: given an ambiguous signature σ_A or σ_B, anyone can tell that *at least one of (A, B) must have involved*. Consider the following scenario: suppose X and Y form a coalition to sign a joint statement which is to be verified by a third party Z. For privacy, X and Y do not want to let Z tell whether X and/or Y have/has indeed got involved before the statement was concurrently signed by both of them via a concurrent signature scheme interaction. If X and Y terminate the signing protocol before the Signature Binding Phase, they should be able to disavow that any of them has ever involved in the signing of the statement even after their ambiguous signatures have been revealed to Z. The current ambiguity model available in the literature does not consider this scenario. In this paper, we will also focus on enhancing the ambiguity model in such a way that the model also captures the requirement that *nothing about the authorship of ambiguous signatures would be revealed before the keystone is released*. In particular, the ambiguous signature should be *anonymous* in the sense that anybody should have the capability of producing (indistinguishable) ambiguous signatures for any user. On the other hand, in order to preserve unforgeability and fairness, those ambiguous signatures produced by the public should be dummy ones that cannot be converted into valid binding signatures.

Most of existing concurrent signature schemes allow A and B to generate multiple ambiguous signatures on different messages with respect to the same keystone-fix. For example, given a keystone-fix f, A can generate an ambiguous signature of either σ_A on a message m_A, or alternatively, σ'_A on another message m'_A. It may cause problems if the pair of exchanged binding signatures $\{(k, \sigma_A), (k, \sigma_B)\}$ are to be delivered by A or B for verification by a third party C. In particular, A can show $\{(k, \sigma'_A), (k, \sigma_B)\}$ to C and convince C that A and B exchanged signatures on the messages m'_A and m_B rather than m_A and m_B. In [22], Li et al. proposed a new notion called *accountability* for concurrent signature in order to solve