SCIS 2016 2016 Symposium on
Cryptography and Information Security
Kumamoto, Japan, Jan. 19 - 22, 2016
The Institute of Electronics,
Information and Communication Engineers

A Note on Authenticated Key Exchange in Cryptocurrency

Tatsuya Kyogoku * Masayuki Abe † Tatsuaki Okamoto †

Abstract: In this paper we focus on how to realize end-to-end secure communication between cryptocurrency users. While a cryptocurrency users have not only financial transactions but also commercial transactions (e.g., acknowledgment of receipt, follow-up correspondence, etc), a cryptocurrency only guarantees the security of financial transactions. If we want to get end-to-end security, we need another long term key to execute authenticated key exchange(AKE), but It is wasteful to require two long term keys. In order to solve this problem, a bitcoin-based authenticated key exchange protocol was proposed by Patrick McCorry et al. This protocol executes all interactions by the same keys. However, this protocol has the following inefficiencies; 1) Request a unnecessary payment to generate a session key, and 2) unable to prove the security in a formal model. To overcome the above inefficiencies, we present a new protocol that achieves secure commercial transactions as well as secure financial transactions. Our protocol never requests any extra long term keys, any extra payments and any trusted third parties. We prove the security of our protocol in the Canetti-Krawczyk model.

Keywords: Authenticated Key Exchange, Cryptocurrency, Canetti-Krawczyk model, ECDSA, Schnorr signature

1 Introduction

An electronic payment system that differs from a physical currency needs to address reliability of the trading partner and double spending problem. In commercial applications, merchants are wary, and demands more information from their customers than they usually need, and must detect double spending in some way because the movement of the currency cannot be confirmed as physical trading. In order to solve these two problems typically a financial transaction over the Internet requires a trusted third party. While the system works well, it still suffers from the inherent weaknesses of the trust-based model. Therefore Satoshi Nakamoto proposed an electronic payment system Bitcoin [1] in which not requires a trusted third party. This system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Satoshi Nakamoto proposes a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes. While the security of cryptocurrency transactions has been extensively studied, little attention has been paid to the security of post-transaction correspondence. The merchant and the customer often need to engage in commercial transactions after a financial transactions is completed. So they have to generate a new long term key pair to establish a session key by AKE protocol. It is wasteful to require another long term keys. In order to solve this problem Bitcoin-based authenticated key exchange (AKE) [2] was proposed. Bitcoin-based AKE is the first study on how to realize end-to-end secure communication between Bitcoin users in a post-transaction scenario without requiring any trusted third party or additional authentication credentials. However this method is not perfect. We point out two matters.

1. Blockchain has to contain both customer and merchant's signature in order to complete AKE session, but only the customer's signature is added to blockchain when the customer pays Bitcoin to the merchant. Therefore, this method needs extra payment that send from the merchant to the customer to get another signature.

2. In [2] they show a proof of security about their scheme in the informal model. They consider three security requirements, (Private key secrecy, Full forward secrecy,and Session key secrecy) that informally defined. Private key secrecy means that the adversary is unable to gain any extra information about the private key of an honest party by eavesdropping her communication with other parties. Full forward secrecy means that the adversary is unable to determine the shared secret of an eavesdropped AKE session in the past between a pair of honest parties, even if their private key are leaked subsequently. Session key secrecy means that the adversary is unable to determine the shared secret between two honest parties by eavesdropping their communication or changing their message.

* Graduate School of Informatics, Kyoto University
† NTT Secure Platform Labolatories, NTT Corporation

Our contributions:
In this paper, we present a new protocol Π that achieves the following.

1. Eliminate extra payment that send from the merchant to the customer.
2. We focus on how to prove the security in formal model. Concretely we are now trying to obtain proof of security in the Canetti-Krawczyk(CK) model [3].
   Bitcoin uses Elliptic Curve Digital Signature Algorithm(ECDSA) [5] when a user pays some bitcoins, however ECDSA is unable to simulate a signature without a secret key. This mean that if we use ECDSA, we can't prove the security. To solve this problem, we prepare an electronic payment system, and an AKE protocol Both protocol based on the Schnorr signature [6] which can simulate without knowing a secret key, and we make end-to-end secure communication between cryptocurrency users by proving the security in the CK model with a sining oracle when both protocols executed by the same long term key pair.

Organization of the paper:
This paper is organized as follows. In Section 2, we recall the notions of a signature scheme and a security model, and present some useful definitions and notations. In Section 3, we introduse related works. In Section 4, we give a new protocol Π and denote proof of security in Section 5. Finally we denote conclusion.

2 Preliminaries

In this section we introduce some technics that we use in our scheme.

2.1 ECDSA

Domain parameters D = (CURB : Using elliptic curb, G : Generator for the elliptic curve, n : integer order of G).
H = hash function . Alice creates private key $d \in [1, n-1]$, and a public key curve point $Q = d \times G$. For Alice to sign a message m, she follows these steps:

1. Choose $k \in_R [1, n-1]$.
2. Compute $kG = (x_1, y_1)$ where $x_1 \in_R [0, q-1]$
3. Compute $r = x_1 \bmod n$.
   If $r = 0$, then go to Step 1.
4. Compute $e = H(m)$.
5. Compute $s = k^{-1}(e + dr) \bmod n$.
   If s = 0, then go to Step 1.
6. Return (r, s).

Verification :

1. Verify $r \in [1, n-1]$ $s \in [1, n-1]$.
2. Compute $e = H(m)$.
3. Let z be the L_n left most bits of e..
4. Compute $w = s^{-1} \bmod n$.
5. Compute $u_1 = zw \bmod n$.
6. Compute $u_2 = rw \bmod n$.
7. Compute $(x_1, x_2) = u_1 \times G + u_2 \times Q_A$.
8. The signature is valid if $r \equiv x_1 \pmod n$, invalid otherwise.

2.2 Schnorr signature

Public key : p, q, g, A
p and q are prime number q | p -1.
$g^q \equiv 1 \pmod p$.
$A = g^a \bmod p$.
Secret key : $a \overset{R}{\in} \mathbb{Z}_q$
When sign a message m, follows these steps:

1. Choose $r \overset{R}{\in} \mathbb{Z}_q$.
2. Compute $e = H(g^r, m)$
3. Compute $s = (r - ae)$
4. Return (e, s).

Verification :

1. Compute $e' = H(g^s A^e)$
2. The signature is valid If e' = e , invalid otherwise.

2.3 The discrete logarithm(DL) assumption

p: Prime number.
G: Cyclic group of order p.
g: Generator of G.
The DL assumption in G states that, given $A = g^a \in G$, it is computationally infeasible to compute the discrete logarithm a of A.

2.4 Computational Diffie-Hellman(CDH) assumption

p: Prime number.
G: Cyclic group of order p.
g: Generator of G. The CDH assumption in G states that, given (g, g^x, g^y) for randomly chosen generator g and randomly chosen points, it is computationally infeasible to compute g^{xy}.

2.5 Gap Diffie-Hellman(GDH) assumption

p: Prime number.
G: Cyclic group of order p.
g: Generator of G.
DDH: The Decisional Diffie-Hellman oracle that is able to decide whether a tuple $(g, g^{x'}, g^{y'}, g^{z'})$ is such that $z' = x'y'$ or not.
The GDH assumption in G states that, given (g, g^x, g^y) for randomly chosen generator g and randomly chosen points, it is computationally infeasible to compute g^{xy} with the help of DDH.

2.6 HMQV [7] and sHMQV [8] protocols

We use a sHMQV protocol to execute authenticated key exchange.

$\hat{A}(a, A)$	$\hat{B}(b, B)$
$x \overset{R}{\in} \mathbb{Z}_q$	$y \overset{R}{\in} \mathbb{Z}_q$
$X = g^x$	$Y = g^y$
send X to $\hat{B}$	send Y to $\hat{A}$
$Z_{\hat{A}} = (YB^e)^{x+da}$	$Z_{\hat{B}} = (XA^d)^{y+eb}$

Figure 1: AKE protocol. $\hat{A}$ and $\hat{B}$ are the identifier. (a, A) and (b, B) are the key pairs (secret key, public key). In HMQV, $d = (X, \hat{B})$, $e = (Y, \hat{A})$ and both party sets the session key $k = H(Z_{\hat{A}}) = H(Z_{\hat{B}})$. in sHMQV $d = (X, \hat{B}, Y)$, $e = (Y, \hat{A}, X)$ and both party sets the session key $k = H(Z_{\hat{A}}, \hat{A}, \hat{B}, X, Y) = H(Z_{\hat{B}}, \hat{B}, \hat{A}, Y, X)$.

2.7 Canetti-Krawczyk(CK) model adopted to cryptocurrency transaction

The strongest security definition for AKE protocols was formalized by Canetti and Krawczyk [3] [4].This model guarantees that the leaking of a session key or session state information will have no effects on the security of other sessions. We introduce the security environment, the execution model and the definition of security about a key exchange protocol.

Unauthenticated network model

The AKE security environment involves multiple honest n parties $[P_1 \cdots P_n]$ and each party has k sessions $[s_1 \cdots s_k]$ and an adversary M controls all sessions schedule. (We stress that an AKE session is executed by a single party.)

In this model, M delivers party's messages. Therefore listens all party's messages and controls what messages will reach a honest party or not. Additionally, M can change these messages or inject its own generated arbitrary messages. In addition to the above basic attacker capabilities, M can execute following queries.

1. Corrupt query : Revealing all secret information, and taking full control of any party.
2. Session reveal query : Revealing the session key of any session.
3. Session state reveal query : Revealing the session specific secret information of any session without revealing the long-term secret key.

Execution model

When M requires P_i to establish a session with P_j, we describe (P_i, P_j, s). s means a session identifier, and same parties never have a same session identifier twice. If the session (P_j, P_i, s') which have a same session identifier (s = s') exists, we define that these two sessions (P_i, P_j, s) and (P_j, P_i, s') are matching, and when we focus on the session (P_i, P_j, s), we refer to (P_j, P_i, s') as a matching session.

Definition of security

In security analysis we use the term clean state. If the session is clean, its meets the following conditions.

1. The adversary M never executes Session reveal query and Session state reveal query to the session and a matching session.
2. The adversary M never corrupt parties that executes the session.

The adversary M can select any clean completed session (completed means that the party computed a session key, and we refer to this session as the test session), and executes Test query to the test session just only once at any moment. We describe test query as following. When the party receive test query

1. Choose $b \overset{R}{\leftarrow} [0, 1]$
2. If b = 0
   Return random string.
3. If b = 1
   Return test session's session key.

After receiving a value, the adversary M must hold clean state about the test session and the matching session, and at the close of this query, output some value b'. The adversary ' s goal is to guess correctly which of the cases was selected. If two parties which are not corrupted complete a matching session, then their session keys are the same, and if result of the test session query is

$$Pr[b' = b] \leq \frac{1}{2} + \epsilon$$

, then this key exchange protocol is session key security.

3 Abstraction of cryptocurrency transaction using DL-type key

To prove the security of our protocol Π, we have to create a sining oracle which returns the corresponding signature when the sining oracle receive a message and the public key. To generate a Schnorr signature on an input message($T_{\hat{B}}$), the sining oracle O_S run in the following way. (We are modeling the hash function H as a random oracle)

1. Receive a message $T_{\hat{B}}$ and the public key B.
2. Choose $s_{T_B}, e_{T_B} \overset{R}{\in} \mathbb{Z}_q$.
3. Compute $g^r = g^{s_{T_B}} B^{e_{T_B}}$.
4. Define $e_{T_B} = H(g^r, T_{\hat{B}})$.
5. Return (e_{T_B}, s_{T_B}).

In this paper we give this oracle O_S to the adversary that attacks our protocol Π in the CK model.

4 Protocol Π

Our protocol uses an electronic payment system that only change bitcoin's sining algorithm from ECDSA to Schnorr, and sHMQV [8] (a variant of HMQV) as a AKE protocol. We first introduce some preliminaries used in our protocol Π.
$\hat{A}, \hat{B}$: Parties identifier.
a, b: Secret keys.
A, B : Public keys.
T_A: A's transaction.
H, H_1, H_2: Hash functions.

4.1 session activation

1. Payment($\hat{A}$, $\hat{B}$): $\hat{A}$ performs the following.
   (a) Select $r \overset{R}{\in} \mathbb{Z}_q$.
   (b) Compute $R = g^r$.
   (c) Compute $e_{T_A} = H(R, T_A)$.
   (d) Compute $s_{T_A} = r - ae_{T_A}$.
   (e) Select $x \overset{R}{\in} \mathbb{Z}_q$.
   (f) Compute $X = g^x$.
   (g) Send X and signature (e_{T_A}, s_{T_A}) to $\hat{B}$ with identifier $(\hat{A}, \hat{B}, X)$.
2. Respond($\hat{B}$, $\hat{A}$, X)
   (a) $\hat{B}$ responds if (e_{T_A}, s_{T_A}) is in the authorized block chain.
   (b) Select $y \overset{R}{\in} \mathbb{Z}_q$.
   (c) Compute $Y = g^y$.
   (d) Send Y to $\hat{A}$.
   (e) Compute $e = H_1(Y, \hat{A}, X)$.
   (f) Compute $s_B = y - eb$
   (g) Compute $Z_B = (XA^{-d})^{s_B}$.
   (h) Compute a session key.
   $K_2 = H_2(Z_B, \hat{A}, \hat{B}, X, Y)$
   and complete session with identifier $(\hat{B}, \hat{A}, Y, X)$
3. Complete($\hat{A}, \hat{B}, X, Y$)
   (a) $d = H_1(X, \hat{B}, Y)$
   (b) Compute $s_A = x - da$
   (c) Compute $Z_A = (YB^{-e})^{s_A}$
   (d) Compute a session key
   $K_1 = H_2(Z_A, \hat{A}, \hat{B}, X, Y)$
   and complete session with identifier $(\hat{A}, \hat{B}, X, Y)$

5 Proof of Security

5.1 AKE security

Theorem 1
Under the GDH assumption, the protocol Π with it's hash functions modeled as a random oracle, is a secure key exchange protocol in the Canetti Krawczyk model with the sining oracle.

lemma 1
If two parties $\hat{A}$, $\hat{B}$ complete the matching session, then their session keys are the same.
Proof.

$$\begin{aligned} Z_A &= (YB^{-e})^{s_A} \\ &= (g^y g^{-be})^{s_A} \\ &= g^{(y-be)(x-ad)} \end{aligned} \tag{1}$$

$$\begin{aligned} Z_B &= (XA^{-d})^{s_B} \\ &= (g^x g^{-ad})^{s_B} \\ &= g^{(x-ad)(y-be)} \end{aligned} \tag{2}$$

therefore $H_2(Z_A, \hat{A}, \hat{B}, X, Y) = H_2(Z_B, \hat{B}, \hat{A}, Y, X)$

lemma 2
Under the GDH assumption, there is no feasible adversary that succeeds in distinguishing the session key from a random value with nonnegligible probability in the test query.

Proof.
The adversary M has only two possible strategies to distinguish $H_2(Z, \hat{A}, \hat{B}, X_0, Y_0)$ from a random value. (We are modeling H_1, H_2 as a random oracle and $(\hat{A}, \hat{B}, X_0, Y_0)$ as a identifier of the test session.)

1. Forging attack
   The adversary M succeeds in computing the value $(Z, \hat{A}, \hat{B}, X_0, Y_0)$, and gets the session key to use hash function H_2 on it.
2. Key replication attack
   The adversary M succeeds in establishing a session which is not the test session but has the same session key as the test session. In this case M can distinguish a session key from random value without knowing the value $(Z, \hat{A}, \hat{B}, X_0, Y_0)$.

5.1.1 Infeasibility of forging attack

In this strategy, the forger is denoted by F, and the adversary that succeed in forging is denoted by M. The test session $(\hat{A}, \hat{B}, X_0, Y_0)$ considers $\hat{A}$ alone from a definition of the CK model, but $\hat{A}$ surely receives a message (B, A, Y_0), because the test session parties are never corrupted. The generation of a value Y_0 that delivered to $\hat{A}$ can fall under one of the following cases.

1. Y_0 was generated by $\hat{B}$ in the matching session.
   $(\hat{B}, \hat{A}, Y_0, X_0)$
2. Y_0 was never generated by $\hat{B}$.
3. Y_0 was generated by $\hat{B}$ with another party.
   $(\hat{B}, \hat{A}', Y_0, X')$ $(A' \neq A)$
4. Y_0 was generated by $\hat{B}$ with $\hat{A}$ but in another session.
   $(\hat{B}, \hat{A}, Y_0, X')$ $(X' \neq X)$

We denote the probability that the adversary successes in the forging attack is negligible in each case. For each of the cases we build a forger F which given access to a DDH oracle DDH, and solves the GDH problem.

Case 1:
In this case forger F takes input $(X_0, Y_0) \in G^2$, and works as follows.

1. Make parties $P_1, \cdots, P_n$.
2. Select two integers $i, j \leftarrow [1, \ldots, n]$.
3. Select two integers $t_1, t_2 \leftarrow [1, \ldots, k]$.
4. Select two honest parties $p_i = \hat{A}$ and $p_j = \hat{B}$
5. Set a key pairs(secret key, public key) $p_i \leftarrow (a, A)$ $p_j \leftarrow (b, B)$, and assign random static key pairs to other parties.

Note that, F guesses that M will select t_1-th session at P_i or t_2-th session at P_j as the test session.
M makes queries as follows.

1. Payment($\hat{P}_1, \hat{P}_2$):
   $\hat{P}_1$ executes the Payment() activation of the protocol. However if the session being created is t_1-th session at $\hat{A}$ (or t_2-th session at $\hat{B}$), F checks whether $\hat{P}_2$ is $\hat{B}$(or $\hat{A}$). If so, F sets the ephemeral public key to X_0(orY_0) from the input of F. Otherwise, F aborts.
2. Respond($\hat{P}_1, \hat{P}_2, Y$)
   $\hat{P}_1$ executes the Respond() activation of the protocol. However if the session being created is t_1-th session at $\hat{A}$ (or t_2-th session at $\hat{B}$), F checks whether $Y = Y_0$ (or $Y = X_0$). If so, F sets the ephemeral public key to X_0(orY_0) from the input of F, and completes the session without computing a session key. Otherwise, F aborts.
3. Complete($\hat{P}_1, \hat{P}_2, X, Y$)
   $\hat{P}_1$ executes the Complete() activation of the protocol. However if the session being created is t_1-th session at $\hat{A}$ (or t_2-th session at $\hat{B}$), F checks whether $Y = Y_0$ (or $Y = X_0$). If so, F completes the session without computing a session key. Otherwise, F aborts.
4. Session State Reveal (s):
   F return to M the schnorr signature and the shared secret Z. if s is t_1-th session at $\hat{A}$ (or t_2-th session at $\hat{B}$), F aborts.
5. Session Key Reveal (s)
   F return to M the session key of s. If s is t_1-th session at $\hat{A}$(or t_2-th session at $\hat{B}$), F aborts.
6. Corrupt($\hat{P}$)
   F gives M the private key of $\hat{P}$ and state information for current sessions and session keys at $\hat{P}$. From the moment of corruption M takes full control over $\hat{P}$. If M tries to corrupt $\hat{A}$ or $\hat{B}$, F aborts.
7. $H_1(X, \hat{P}_1, Y)$:
   If the input value was previously asked in other times, returns the same value.
   If the input value was not previously asked, F simulates a random oracle on it.
8. $H_2(Z, \hat{P}_1, \hat{P}_2, X, Y)$:
   If the input value was previously asked in other times, returns the same value.
   If the input value was not previously asked, F simulates a random oracle on it.

If F asked H_2 with $\hat{P}_1 = \hat{A}$, $\hat{P}_2 = \hat{B}$, $X = X_0$, $Y = Y_0$ for some Z value, and $DDH(X_0 A^d, Y_0 B^e) = 1$ where $d = H_1(X_0, \hat{B}, Y_0)$ and $e = H_1(Y_0, \hat{A}, X_0)$, then F can compute

$$CDH(X_0, Y_0) = Z X_0^{eb} Y_0^{da} g^{-deab}.$$

It means that If M succeeds in forging Z, F solves GDH problem. Therefore $Pr[F] \leq \epsilon$, and if M selects t_1-th session at P_i or t_2-th session at P_j as the test session and the matching session, F perfectly simulates M's environment except with negligible probability.
The probability of guess right is $\frac{2}{(nk)^2}$. Therefore the success probability of M is negrigible.

$$Pr[F] \geq \frac{2}{(nk)^2} Pr[M].$$

Case 2:

In this case F takes input $(X_0, B) \in G^2$ and works as follows.

1. Make parties $P_1, \cdots, P_n$.
2. Select two integers $i, j \leftarrow [1, \ldots, n]$.
3. Select a integer $t \leftarrow [1, \ldots, k]$.
4. Select two honest parties $p_i = \hat{A}$ and $p_j = \hat{B}$
5. Set a as a $\hat{A}$'s secret key, and A as a $\hat{A}$'s public key.
   Set B as a $\hat{B}$'s public key.
   Assign random static key pairs to other parties.

Note that, F guesses that M will select t-th session at P_i as the test session. To simulate M's environment, F uses oracle O_S with an input message $T_{\hat{B}}$ and creates an Oracle $O_{\hat{B}}$. and proceeds as follows:

Oracle $O_{\hat{B}}$ runs in the following way.

1. When invoked to generate an ephemeral key:
   (a) input B.
   (b) Choose $s, e \overset{R}{\in} \mathbb{Z}_q$.
   (c) Compute $Y = g^s B^e$.
   (d) return Y.

2. When invoked to generate a Schnorr signature on an input message($\hat{P}$, X):

   (a) define $e = H_1(Y, \hat{P}, X)$.
   if $H_1(Y, \hat{P}, X)$ was previously defined in different values $e' \neq e$, then define $e = H_1(Y, \hat{P}, X)$ once again instead of e'.

   (b) return (Y, s).

S simulates all session activations at $\hat{B}$ for M with the help of O_S and $O_{\hat{B}}$. M makes queries except for B as follows.

1. Payment($\hat{P}_1, \hat{P}_2$):
   $\hat{P}_1$ executes the Payment() activation of the protocol. However if the session being created is the t-th session at $\hat{A}$, S checks whether $\hat{P}_2$ is $\hat{B}$. If so, F sets the ephemeral public key to X_0 from the input of F and doesn't compute the session key. Else, F aborts.

2. Respond($\hat{P}_1, \hat{P}_2, Y$)
   $\hat{P}_1$ executes the Respond() activation of the protocol. However if the session being created is the t-th session at $\hat{A}$, F checks whether $\hat{P}_2$ is $\hat{B}$. If so, F sets the ephemeral public key to X_0 from the input of F, and completes the session without computing the session key. Otherwise, F aborts.

3. Complete($\hat{P}_1, \hat{P}_2, X, Y$)
   $\hat{P}_1$ executes the Complete() activation of the protocol. However if the session being created is the t-th session at $\hat{A}$, F completes the session without computing the session key.

4. Session State Reveal (s):
   F return to M the schnorr signature and the shared secret Z. However, if F is t-th session at $\hat{A}$, F aborts.

5. Session Key Reveal (s)
   F return to M the session key of s. If s is t-th session at $\hat{A}$, F aborts.

6. Corrupt($\hat{P}$)
   F gives M the private key of $\hat{P}$ and state information for current sessions and session keys at $\hat{P}$. From the moment of corruption M takes full control over $\hat{P}$. If M tries to corrupt $\hat{A}$ or $\hat{B}$, F aborts.

7. $H_1(X, \hat{P}_1, Y)$:
   If the input value was previously asked in other times, returns the same value.
   If the input value was not previously asked, F simulates a random oracle on it.

8. $H_2(Z, \hat{P}_1, \hat{P}_2, X, Y)$:
   If the input value was previously asked in other times, returns the same value.
   If the input value was not previously asked, F simulates a random oracle on it.

If F asked H_2 with $\hat{P}_1 = \hat{A}$, $\hat{P}_2 = \hat{B}$, $X = X_0$, $Y = Y_0$ for some Z value, and $DDH(X_0A^d, Y_0B^e) = 1$ where $d = H_1(X_0, \hat{B}, Y_0)$ and $e = H_1(Y_0, \hat{A}, X_0)$, then F can output

$$Z(= g^{x_0y_0}g^{-x_0eb}g^{-ady_0}g^{abde}).$$

In this case F can't compute $g^{x_0y_0}$ and g^{-xeb}. Thus, F is unable to output $CDH(X_0, B)$. To solve $CDH(X_0, B)$, following the Forking Lemma[9] approach, F runs M again in a same environment but F responds to $H_1(Y, \hat{P}, X)$ with a value $e' \overset{k}{\in} \mathbb{Z}_q(e' \neq e)$. If M succeeds in the second run, F computes

$$Z'(Y_0B^{-e'})^{da} = g^{x_0y_0}g^{e'x_0b}.$$

and therefore obtains

$$CDH(X_0, B) = (\frac{Z}{Z'})^{\frac{1}{e'-e}}B^{da}.$$

It means that If M succeeds in forging Z, F solve GDH problem. Therefore $Pr[F] \leq \epsilon$, and if M selects t-th session at $\hat{A}$ and peer is $\hat{B}$ as the test session, F perfectly simulates M's environment except with negligible probability.
The probability is at least $\frac{1}{n^2k}$. Therefore the success probability of M is negrigible.

$$Pr(S) \geq \frac{q_{H_1}^{-1}}{(n)^2k}Pr(M).$$

(The loss factor of $q_{H_1}^{-1}$ is from the Forking lemma.)

Case 3, 4: The simulation for M's environment for case 3 and 4 is the same as case 2. If the adversary M wins the forging attack, its computes the 5 tuples $(Z, \hat{A}, \hat{B}, X_0, Y_0)$. Therefore F takes the value

$$Z = g^{(x_0-da)(y_0-eb)}.$$

In this case $\hat{B}$ has a session which uses Y_0 as the ephemeral public key. So O_B must output (Y_0, s) with $Y_0 = g^sB^{e'}$ where $e' = H_1(Y, \hat{A}, X')$ or $H_1(Y, \hat{A}', X)$. ($\hat{A}' \neq \hat{A}, X' \neq X_0$, X is any given value, and $e' \neq e$). Then F takes Z' from the corresponding query to H_2. It then holds that

$$Z' = g^{(x_0-da)(y_0-e'b)}.$$

From Z and Z', F obtains

$$CDH(X_0, B) = (\frac{Z}{Z'})^{\frac{1}{e'-e}}B^{da}.$$

Therefore the success probability of M is negrigible.

$$Pr[F] \geq \frac{1}{(n)^2k}Pr[M].$$

Note that, if we consider the case that cryptocurrency payment and AKE protocol share not only the long term key but also the ephemeral key, we unable to prove the security. We propose the Payment query (instead of subsection 4.1),

1. Payment($\hat{A}$, $\hat{B}$): $\hat{A}$ performs the following.

   (a) Select $x \overset{R}{\in} \mathbb{Z}_q$.
   (b) Compute $X = g^x$.
   (c) Compute $e_{T_A} = H(X, T_A)$.
   (d) Compute $s_{T_A} = x - ae_{T_A}$.
   (e) Send X and signature (e_{T_A}, s_{T_A}) to $\hat{B}$ with identifier $(\hat{A}, \hat{B}, X)$.

In this case, while the forger F can simulate ephemeral public key Y and d, F is unable to simulates s_A and Z. Therefore we are unable to prove the security when we share the ephemeral key pairs.

5.1.2 Infeasibility of Key replication attack

Since this protocol uses a hash function to output a session key as a random oracle, If input value is difference, hash function never return same value. Therefore, as long as the session identifier $(\hat{A}, \hat{B}, X, Y)$ is hashed together with the shared secret value Z, the success probability of adversary against Key replication attack is negrigble. This completes the proof of Lemma 2. Together with Lemma 1, we complete the proof of Theorem 1.

5.1.3 Other security properties

The AKE protocol is desirable to achieve following two security properties in addition to the session key security. So we prove the security that our protocol resists to these attacks.

1. Resistance to Key-Compromise Impersonation (KCI) attacks
Resistance to KCI attacks mean that, if the adversary M gets the private key of $\hat{A}$, then M can't distinguish the session key of a complete session at $\hat{A}$ from a random value under conditions that the session peer is not corrupted and the session and its matching session (If its exist) are clean.

Theorem 2.
Under the GDH assumption, the protocol Π, with hash functions modeled as random oracles, resists KCI attacks..

proof.
We assume that the protocol Π is secure even if the adversary M gets the secret key of $\hat{A}$. The only change to the proof is that removes the corruption of $\hat{A}$ from the description of forger F as a reason for F to abort. The proof remains valid since

1. In test session the ephemeral public key of $\hat{A}$ is given from the input to forger F. This means that the adversary M is unable to select a ephemeral public key freely.
2. M is unable to compute $Z_B (= g^{(s_A)(s_B)})$ even If M gets the secret key of $\hat{A}$,
3. The above abort operation is never used in the proof. Therefore, simulation is still perfect.

2. Resistance to Weak Perfect Forward Secrecy(PFS)
Forward secrecy means that the adversary is unable to determine the shared secret of an eavesdropped AKE session in the past between a pair of honest parties, even if their private keys are leaked subsequently. And the weak PFS ensures that the session key of the session whose incoming and outgoing messages X, Y are not chosen by the adversary enjoys forward secrecy.

Theorem 3.
Under the CDH assumption, the protocol Π, with hash functions modeled as random oracles, provides weak PFS.

proof.
Given an adversary M that breaks the weak PFS property, we construct a CDH solver S as follows. In this case S takes input $(X_0, Y_0) \in G^2$, and works as follows.

1. Make parties $P_1, \cdots, P_n$.
2. Select two integers $i, j \leftarrow [1, \ldots, n]$.
3. Select a integer $t \leftarrow [1, \ldots, k]$.
4. Select two honest parties $p_i = \hat{A}$ and $p_j = \hat{B}$
5. Set key pairs(secret key, public key) $p_i \leftarrow (a, A)$ $p_j \leftarrow (b, B)$, and assign random static key pairs to other parties.

S set t-th session of $\hat{A}$ $(\hat{A}, \hat{B}, X_0, Y_0)$. If M select $(\hat{A}, \hat{B}, X_0, Y_0)$ as a test session and distinguishes the session key of the test session from random, then M must query H_2 with $(Z, \hat{A}, \hat{B}, X_0, Y_0)$. Therefore, F can output Z $(= g^{x_0 y_0} g^{-x_0 eb} g^{ady_0} g^{abde})$, $g^{-x_0 eb}$, g^{ady_0} and g^{abde}. Then F can compute $CDH(X_0, Y_0)$. Therefore the success probability of M is negrigible.

5.2 Cryptocurrency security

Theorem 4.
If the Schnorr signature is unforgeable, the protocol Π with it's hash functions modeled as a random oracle, is a secure cryptocurrency payment .

lemma 3.
Under the DL assumption, after execute the protocol Π, there is no feasible adversary that succeeds in gaining an existential forgery of the Schnorr signature, under an adaptively chosen message attack with nonnegligible probability.

Proof.
Given the adversary M that succeeds in forging the Schnorr signature under an adaptively chosen message attack, we construct a DL solver S as follows. In this case S takes input $A \in G$, and works as follows.

1. Make parties $P_1, \cdots, P_n$.

2. Select integer $i \leftarrow [1, \ldots, n]$.

3. Select a honest party $p_i = \hat{A}$.

4. Set A as a $\hat{A}$ ' s public key. and assign random static key pairs to other parties.

S simulates all the session activation at $\hat{A}$ for M with the help of the oracle O_S and $O_{\hat{B}}$, and other parties session activations are freely set by M. To solve $DL(A)$, following the Forking Lemma[9] approach, S runs M again in a same environment but S responds to $H(X, T_{\hat{A}})$ with a value $e' \overset{R}{\in} \mathbb{Z}_q (e' \neq e)$. If M succeeds in the second run, S computes

$$R = g^{s'} A^{e'}.$$

and therefore obtains

$$DL(A) = \frac{s' - s}{e - e'}$$

It means that If M succeeds in an adaptively chosen message attack, S solves the DL problem. Therefore $Pr[S] \leq \epsilon$, and if M selects $\hat{A}$, S perfectly simulates M's environment except with negligible probability. The probability is at least $\frac{1}{n}$. Therefore, the success probability of M is negrigible.

$$Pr(S) \geq \frac{q_{H_1}^{-1}}{n} Pr(M).$$

(The loss factor of $q_{H_1}^{-1}$ is from the Forking lemma.)

6 Conclusions

In this paper, we consider end-to-end secure communication between cryptocurrency users. When both the AKE protocol and the cryptocurrency system share the long term key pair, the signature simulation of the cryptocurrency payment and the AKE protocol are stand alone, so that we can perfectly simulate the adversary's environment. Therefore, we realize end-to-end secure communication between cryptocurrency users without extra long term keys in CK model. On the other hands, we have to prepare the ephemeral key pair for both sides, because when we share as well as the ephemeral key, the signature simulation of the cryptocurrency payment and the AKE protocol are unable to separate, so that we can't simulate the adversary's environment.

References

[1] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System Cryptology-Proceedings of EUROCRYPT ' 99, in: Lecture Notes in Comput. Sci., vol. 1592, Springer-Verlag, 1999, pp. 295-310.

[2] Patrick McCorry, Siamak F. Shahandashti, Dylan Clarke, Feng Hao, Authenticated Key Exchange over Bitcoin

[3] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, Advances in Cryptology-EUROCRYPT 2001 volume 2045 of Lecture Notes in Computer Science, pages 453-474. Springer, May 2001.

[4] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, ProvSec 2007: 1st International Conference on Provable Security, volume 4784 of Lecture Notes in Computer Science. pages 1-16. Springer, November 2007.

[5] D. Hankerson, S. Vanstone, and A. Menezes. Guide to Elliptic Curve Cryptography. Springer Professional Computing. Springer, 2004.

[6] D. Pointcheval and J. Stern. Security proofs for signature schemes. In Advances in Cryptology. EURO- CRYPT96, pages 387-398. Springer, 1996.

[7] H. Krawczyk. HMQV: A High-Performance Secure Dieffe-Hellman Protocol. In Advances in Cryptology-CRYPTO 2015, pages 546-566. Springer, 2005.

[8] Shijun Zhao and Qianying Zhang. sHMQV: An Efficient Key Exchange Protocol for Power-limited Devices.

[9] D. Pointcheval and J. Stern. Security proofs for signature schemes. In Advances in Cryptology - EURO- CRYPT ' 96, pages 387-398. Springer, 1996.

[10] Shijun Zhao and Qianying Zhang. sHMQV: An Efficient Key Exchange Protocol for Power-limited Devices