International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **6**, June **2012**

 ISSN 1349-4198
pp. 3875–3890

P2P FAIR CONTENT EXCHANGE WITH OWNERSHIP TRANSFER

Chun-I Fan[1,*], Wen-Shenq Juang[2] and Ming-Te Chen[1]

[1]Department of Computer Science and Engineering
National Sun Yat-sen University
No. 70, Lienhai Road, Kaohsiung 80424, Taiwan
*Corresponding author: cifan@faculty.nsysu.edu.tw; ecsemtchen@gmail.com

[2]Department of Information Management
National Kaohsiung First University of Science and Technology
No. 2, Jhuoyue Road, Nanzih, Kaohsiung 811, Taiwan
wsjuang@ccms.nkfust.edu.tw

Received February 2011; revised June 2011

Abstract. *In recent years, due to the maturity of network and information technologies, p2p (peer to peer) networks grow rapidly. Users (called peers) can upload or download files such as digital contents to or from other peers in a p2p network. Usually, the digital contents can contain the watermarks for ownership claim and protection. To date, a lot of buyer-seller watermarking schemes were proposed, but none of them is suitable for peers to exchange their digital contents in a p2p network. By the way, if peers attempt to exchange their digital contents by using these schemes directly, none of them can guarantee that the digital content exchange is efficient and secure between peers. In addition, how to exchange the digital contents fairly in a p2p network is another problem since a peer may not honestly forward her/his digital content to another peer. On the other hand, due to the feature of p2p networks, there may be a free-rider situation happening and it causes the unfair result in the content exchange process among peers. In this paper, we propose an efficient and fair p2p digital content exchange scheme for solving all the above problems. Our proposed scheme can exchange the watermarked digital content securely and keep the exchanging process fair to each other.*
Keywords: Digital right, Digital watermark, Secure fair content exchange, Mutual authentication, P2P

1. **Introduction.** P2P (peer to peer) networks recently grow and are used rapidly due to the maturity of network technologies. In a p2p network, there are lots of independent servers (called super node) and personal computers (called peers). Some p2p servers also can form a self-organized entity and serve as the server peers to build a p2p network with many client peers.

In a p2p network, some peers play the role of the client peer and the others play the role of the server peer. A client peer can connect to the server peer and the server peer accepts this connection with the client peer after receiving this query and obtaining the client peer information. If this client peer connects to the server peer successfully, it can query the server peer about the information of the desired data including the IP addresses of the peers which may contain the desired files, the desired files' status, and so on. When a client peer obtains this information from the server peer, it can ask the server peer to redirect its connection to another client peer containing the desired data. Then a client peer can negotiate with other peers to exchange their digital content after authenticating with each other successfully. Although, there were some authentication schemes proposed

[5, 20, 34, 35, 39, 45, 51], but none of them is suitable in a p2p network for peers doing the authentication before peers agree to exchange their digital content.

In addition, if a client peer has authenticated with another client peer successfully, then it may face that anyone of these peers may be the "altruistic" one during doing the digital content exchange process. If a peer is an "altruistic" one which disconnects with the other peers after it has gotten the intended files from them, then it causes them the unfair result in this situation. This will make that the other peers only get some part of the intended files. Then these peers may not obtain the complete files from other peers in this time. This situation is called "free-riders" [8].

When some peers agree to perform the content exchange with each other, they may encounter this free-riders situation during performing their digital content exchange. Due to this free-rider problem, to the best of our knowledge, no secure and efficient fair digital content exchange scheme has been proposed for peers in a p2p network during doing the digital content exchange process. Moreover, when a client peer performs her/his digital content exchange with another peer, it may receive the fake digital content files which may contain the viruses or other malicious codes inside. In this situation, it usually also causes them the unfair situation.

On the other hand, for offering the digital content exchange in a p2p network, a trivial approach is to perform the traditional buyer-seller watermarking protocols [15, 26, 28, 47, 52]. However, the computational cost of this approach is high. Moreover, most of them adopt the exponential operation and none of them can avoid the unfair situation, solve the free-rider problem, and be suitable for a p2p network. In order to solve all the above problems, we propose our novel p2p fair and efficient content exchange scheme. Our scheme can provide efficient and secure fair digital content exchange between peers. In addition, our proposed scheme can trace the distributor in the forensic purpose with help of the methods in [29, 36, 46].

2. **Related Works.** There are a lot of different types of networks such as the p2p network or the cloud computing network used today. In a p2p network, each network node (called peer) can share files with the help of a "super node" peer, who is a server peer and can maintain each peer's connection information and redirect peers' connections to other peers. In addition, a client peer can become a "super node" peer when its equipment is more powerful than the others in some other special situations.

In a p2p network, a peer can exchange files with another peer. If there are some peers that may attempt to perform the digital contents exchange, they have to retrieve the connection information from the server peer first and then connect to each other with the help of the server peer. When they have connected to each other, each peer must perform the authentication with the others before they start to exchange their digital contents. However, to our best knowledge, there is no suitable authentication scheme for peers performing the exchanging digital contents after authenticating with each other in a p2p network. In addition, when a peer has received the intended files from the others in the digital content exchange phase, it may cut off the connection with these communicating ones. This will cause some peers only getting some piece of the intended files. This results in the unfair situation at this time. However, even if there were some p2p content exchange schemes [9, 30] proposed in literature, any one of them only offers the payment mechanism and cannot avoid this unfair situation in a p2p network. Moreover, none of them can cope with the fake digital content problem during the digital content exchange transaction. If there is a malicious peer that forwards the fake digital contents which contain the viruses or some malicious codes inside, then this malicious peer can obtain the complete files from the honest peers but they only get the fake files containing the

viruses or the malicious codes inside after exchanging their digital contents with each other. It will also cause them the unfair situation at this time.

In [9], their scheme focuses on the payment mechanism for peers after they purchased the digital contents from the content provider peer in a p2p network. The content provider peer and the original content creator peer can get their corresponding commission and payment from the bank peer, respectively. However, it is unreasonable that any powerful peer can be upgraded to be a bank peer when the current bank peer malfunctions now. However, this new bank peer may not be the trusted by the other peers without doing any authentication approach with the other client peers. By the way, how it maintains all transactions of the previous bank peers without knowing the information about each client peer is another problem. On the other hand, their scheme does not guarantee the payment transaction fairly. If the current bank peer fails when the client peer is paying for its downloaded digital contents, then the new bank peer may not have the payment record of this client peer and this client peer may be asked to pay for the downloaded digital contents again. In this situation, the client peer cannot be able to ask the dispute resolution without the help of the trusted party in their scheme. Moreover, in this scheme, the client peer cannot avoid the unfair situation that it received the fake content from the content provider peer and free-rider problem.

In [30], they proposed a fair exchange p2p file market scheme with their payment mechanism. In their scheme, each peer can share its files to another peer and she/he has to pay the content provider peer before downloading the intended digital contents. Their approach adopts the certificate and the digital signature as the payment token. In the exchanging protocol of their proposed scheme, the content provider peer has to forward the decryption key K to the downloading peer without any encryption after confirming the payment with the bank peer. If there exists a malicious attacker between the content provider and the downloading peer, she/he may intercept this key K and then can use it to decrypt all the intercepted ciphertexts in the same transaction. Moreover, in this scheme, they also suffer the unfair situation when the client peer received the fake content from the malicious content provider peers and the free-rider problem.

In order to cope with all the above problems, we propose a novel efficient and secure fair p2p digital exchange scheme. Our proposed scheme provides efficient and secure digital content exchange and keeps the fair property in a p2p network.

3. **Preliminaries.**

3.1. **Bilinear pairing.** Let G_1 be a cyclic additive group of a prime order q and G_2 be a cyclic multiplicative group of the same order q. A bilinear pairing is a map $e : G_1 \times G_1 \longrightarrow G_2$ which satisfies the following properties:

1. Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$.
2. Non-degeneracy: There exists $P \in G_1$ such that $e(P, P) \neq 1$.
3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

3.2. **Security assumptions.** First, we introduce the security assumptions as follows:

Assumption 1 k-CAA assumption. We say that the k-CAA Assumption (for G_1) holds if for any probabilistic polynomial time adversary E the probability that E on input $\left(P, sP, h_1, h_2, \ldots, h_k \in Z_q^*, \frac{1}{s+h_1}P, \frac{1}{s+h_2}P, \ldots, \frac{1}{s+h_k}P\right)$ outputs Q_1 for some $h \in Z_q^*$ such that $Q_1 = \frac{1}{s+h}P$ is negligible. The probability of success of E is taken over the uniform random choice of $h \in Z_q^*$ and the coin tosses of E.

Assumption 2 k-mBDH assumption [44]**.** We say that the k-BDHI Assumption (for G_2) holds if for any probabilistic polynomial time adversary E the probability that E

on input $\left(P, sP, tP, h, h_1, h_2, \ldots, h_k \in Z_q^*, \frac{1}{s+h_1}P, \frac{1}{s+h_2}P, \ldots, \frac{1}{s+h_k}P\right)$ outputs Q_2 for some t such that $Q_2 = e(P,P)^{\frac{1}{s+h}t}$ is negligible.

4. **The Proposed Scheme.** In order to solve the unfair content exchange and the free-riding problem, we use the ID-based authentication key agreement scheme [23] as our building block in a p2p network. Our proposed scheme contains the following properties including mutual authentication and key agreement, without WCA (Watermark Certificate Authority), low computation cost, fair exchange, and optional usage of the robust watermark or the reversible watermark. First, each peer has to register with the server peer and the server peer can provide the dispute resolution between peers. In our scheme, there are four phases including the setup phase, the key agreement phase, the content exchange phase and the resolution phase.

4.1. **Notations.** Notations are as follows:

- q: a prime number
- G_1: a cyclic additive group of a prime order q
- G_2: a cyclic multiplicative group of the same order q
- P: a generator of the group G_1
- e: a Weil mapping $G_1 \times G_1 \longrightarrow G_2$
- Alice (A for short): a client peer in a p2p network
- Bob (B for short): a client peer in a p2p network
- Server (S for short): a server peer in a p2p network
- m_i: an original digital content without any watermark embedded, where $i \in \{\text{A, B}\}$
- M_i: a watermarked object after the watermark embedding operation by performing the reversible watermark/robust watermark method, where $i \in \{\text{A, B}\}$
- s_i: a secret key $s_i \in Z_q^*$ of each peer in the system and $i \in \{\text{A, B, S}\}$
- P_i: a public key $P_i = s_iP$ of each peer in the protocol, where $i \in \{\text{A, B, S}\}$
- W_i: a watermark for exchanging and embedding operation, where $i \in \{\text{A, B}\}$
- ssk_{ij}: a session key which is used in each transaction, where $i, j \in \{\text{A, B, S}\}$
- $h(\cdot)$: a secure one-way hash function $h : \{0,1\}^* \longrightarrow Z_q^*$
- $E_{ssk_{i,j}}(\cdot)$: a symmetric encrypting function with the temporary symmetric shared key $ssk_{i,j}$, where $i, j \in \{\text{A, B, S}\}$
- $D_{ssk_{i,j}}(\cdot)$: a symmetric decrypting function with the temporary symmetric shared key $ssk_{i,j}$, where $i, j \in \{\text{A, B, S}\}$
- Exc_i: an exchange document description on m_i, where $i \in \{\text{A, B}\}$
- $\oplus$: a general partial encryption operation on m_i which can be applied to homomorphic encryption such as RSA encryption, where $i \in \{\text{A, B}\}$
- $\otimes$: a watermark embedding operation respect to the partial encryption operation $\oplus$

4.2. **The setup phase.** In this phase, we assume that there exists a trusted party KGC (Key Generation Center) in our proposed scheme and KGC runs the setup algorithm with the security parameters l and t to generate the master key s and the security parameter param in the following.

1. Select two groups G_1 and G_2 of the prime order $q \geq 2^l$ and a bilinear pairing $e : G_1 \times G_1 \longrightarrow G_2$.
2. Assume that P is a generator of G_1 and let $g = e(P,P)$ be a generator of G_2. Then KGC chooses two secure hash functions $H_1 : \{0,1\}^* \longrightarrow \{0,1\}^t$ and $H_2 : \{0,1\}^* \longrightarrow \{0,1\}^k$, where t is the security parameter and k is the length of a session key.

3. After setting these functions, KGC picks up a secret value $s \in Z_q^*$ as the system private key and it also computes $P_{pub} = sP$ as the system public key. On the other hand, KGC produces the secret key of each parties. KGC generates $q_A = h(ID_A)$ and $S_A = \frac{1}{s+q_A}P$ for Alice. Then it computes $q_B = h(ID_B)$, $S_B = \frac{1}{s+q_B}P$, $q_S = h(ID_S)$ and $S_S = \frac{1}{s+q_S}P$ for Bob and the server, respectively. After generating these secret keys, it forwards them to Alice and Bob by a secure channel.

KGC publishes the system parameter $\mathsf{param} = (l, t, e, q, g, G_1, G_2, P, P_{pub}, h, H_1, H_2)$ and keeps the master-key s in secret.

4.3. **The registration and key-agreement phase.** In this phase, we assume that there exist three parties Alice (A for short), Bob (B for short) and the server (S for short) in a p2p network. The KGC generates the server's key pairs and publishes the system parameter $\mathsf{param} = (l, t, e, q, g, G_1, G_2, P, P_{pub}, h, H_1, H_2)$. Then A and B do the followings, respectively.

1. First, A generates the exchange digital content in the following.
   1.1 Prepare the exchange description Exc_A on the intended digital content m_A.
   1.2 Compute the hash value $t_A = H_1(Exc_A||ID_A||N_A)$ by using H_1 with a fresh nonce N_A and two random numbers $a, b, l \in Z_q^*$.
   1.3 Produce the hash value $q_s = h(ID_S)$ and the public key $Q_S = P_{pub} + q_sP$. Then she computes $G_A = abQ_S$, $t_u = e(abP, P) = g^{ab}$ and $G_l = l * P$.
   1.4 Compute $S_A = (ab + t_A)\frac{1}{s+q_A}P$ and forward $(Exc_A, N_A, ID_A, S_A, Q_S, G_A, t_A, t_u, G_l)$ to S.

2. On the other hand, B also generates his exchange digital content as the same process as A does.
   2.1 Prepare his exchange description Exc_B on his exchange object m_B.
   2.2 Compute the hash value $t_B = H_1(Exc_B||ID_B||N_B)$ by using H_1 with a nonce N_B and two random numbers $c, d, n \in Z_q^*$.
   2.3 Produce the hash value $q_s = h(ID_S)$ and the public key $Q_S = P_{pub} + q_sP$. Then he computes $G_B = cdQ_S$, $t_r = e(cdP, P) = g^{cd}$ and $G_n = n * P$.
   2.4 Compute $S_B = (cd + t_B)\frac{1}{s+q_B}P$ and forward $(Exc_B, N_B, ID_B, S_B, Q_S, G_B, t_B, t_r, G_n)$ to S.

3. After receiving these tuples from A and B, S does the following.
   3.1 Compute $q_s = h(ID_S)$ and $Q_S = P_{pub} + q_sP$. After generating these parameters, it computes two hash values $t_A = H_1(Exc_A||ID_A||N_A)$ and $t_B = H_1(Exc_B||ID_B||N_B)$ and the following equations to verify G_A, S_A, G_B and S_B.
   (i) $e(G_A, S_S) \stackrel{?}{=} e\left(abQ_S, \frac{1}{s+q_s}P\right) = e(abP, P) = t_u = g^{ab}$.
   (ii) $e(G_B, S_S) \stackrel{?}{=} e\left(cdQ_S, \frac{1}{s+q_s}P\right) = e(cdP, P) = t_r = g^{cd}$.
   (iii) $e(S_A, P_{pub} + q_AP) \stackrel{?}{=} e\left((ab + t_A)\frac{1}{s+q_A}P, sP + q_AP\right) = e((ab + t_A)P, P) = g^{ab+t_A} = t_u g^{t_A}$.
   (iv) $e(S_B, P_{pub} + q_BP) \stackrel{?}{=} e\left((cd + t_B)\frac{1}{s+q_B}P, sP + q_BP\right) = e((cd + t_B)P, P) = g^{cd+t_B} = t_r g^{t_B}$.
   3.2 After verifying these equations, it selects $e, f \in_R Z_q^*$ and computes $t_v = e * P$ and $t'_v = f * P$ and computes $z = H_1(t_u||t_v||N_A + 1||N_S||e * G_l)$ and $z' = H_1(t_r||t'_v||N_B + 1||N'_S||f * G_n)$. S forwards (z, t_v, N_S) and (z', t'_v, N'_S) to A and B, respectively.

4. When A receives (z, t_v, N_S) from S, she performs the following verification.
   4.1 Produce the hash value $z_1 = H_1(t_u||t_v||N_A + 1||N_S||l * t_v)$ and compare it with z. If it is valid, she generates her response $H_1(t_u||t_v||N_S + 1||l * t_v)$ and sends it back to S. After this checking, she can compute the session key $ssk_{A,S} = H_2(t_u||t_v||e * l * P)$.
   4.2 On the other hand, B also performs the verification and computation as the same process as A does. If the result is invalid, he can abort the transaction with S. Otherwise, B computes his session key $ssk_{B,S} = H_2(t_r||t'_v||n * t'_v)$. By the way, S also computes two session keys $ssk_{A,S} = H_2(t_u||t_v||e * G_l)$ and $ssk_{B,S} = H_2(t_r||t'_v||f * G_n)$ shared with A and B, respectively.

4.4. **The content exchange phase.**

1. After the authentication with S, A and B compute $\delta'_A = m_A \oplus H_1(t_v * i)$ and $\delta'_B = m_B \oplus H_1(t'_v * j)$, where $i, j \in_R Z_q^*$. Let R_i, R_j be the random numbers such that $R_i = i * P$, $R_j = j * P$, respectively.
2. Then, they also produce two ciphertexts $E_{ssk_{A,S}}(\delta'_A, Exc_A, R_i, h(\delta'_A||Exc_A||ID_A||R_i))$ and $E_{ssk_{B,S}}(\delta'_B, Exc_B, R_j, h(\delta'_B||Exc_B||ID_B||R_j))$ on their digital contents. They send these ciphertexts to S for fair exchange usage. After receiving and decrypting these two ciphertexts, S verifies these tuples with two hash values $h(\delta'_A||Exc_A||ID_A)$ and $h(\delta'_B||Exc_B||ID_B)$. If they are valid, S generates the corresponding watermarks and the signatures in the following.
   2.1 Compute a new watermark $W_A = h(ID_A||Exc_A||r_3)$ and $W_B = h(ID_B||Exc_B||r_4)$, where $r_3, r_4 \in_R Z_q^*$.
   2.2 Decrypt δ'_A and δ'_B by using $H_1(e*R_i)^{-1}$ and $H_1(f*R_j)^{-1}$ and obtain the original message m_A and m_B, respectively.
   2.3 Produce the encrypted exchange digital contents $m'_A = (m_A \oplus k_2) * H_1(r'_6 * R_j)^{-1}$ and $m'_B = (m_B \oplus k_1) * H_1(r'_5 * R_i)^{-1}$ by two random keys $k_1, k_2 \in Z_q^*$, where $r'_5, r'_6 \in_R Z_q^*$. S also computes two encrypted watermarks $W'_A = W_A \oplus k_1$ and $W'_B = W_B \oplus k_2$.
3. After generating the corresponding hash values, S computes the hash value $t_{A_1} = H_1(Exc_A||ID_A||h(m'_A)||h(W'_B)||R'_6)$ and $t_{B_1} = H_1(Exc_B||ID_B||h(m'_B)||h(W'_A)||R'_5)$ on A and B's digital contents, where $R'_5 = r'_5 * P$, $R'_6 = r'_6 * P$.
4. Then S generates the corresponding signatures $S_{A_1} = \frac{t_{A_1}}{s+q_A}P$ and $S_{B_1} = \frac{t_{B_1}}{s+q_B}P$ on two hash values t_{A_1} and t_{B_1}, respectively.
5. After computing these hash values and the signatures, S produces two ciphertexts $E_{ssk_{A,S}}(Exc_B, S_{A_1}, t_{A_1}, W'_A, h(W'_B), m'_A, R'_6)$ and $E_{ssk_{B,S}}(Exc_A, S_{B_1}, t_{B_1}, W'_B, h(W'_A), m'_B, R'_5)$ for A and B.
6. A computes the tuples in the following.
   6.1 Decrypt $E_{ssk_{A,S}}(Exc_B, S_{A_1}, t_{A_1}, W'_A, h(W'_B), m'_A, R'_6)$. Then A uses the following equation to see whether the signature S_{A_1} is valid or not

$$e(S_{A_1}, P_{pub} + q_A P) \stackrel{?}{=} e\left(\frac{t_{A_1}}{s+q_A}P, sP + q_A P\right) = e(t_{A_1}P, P) = e(P, P)^{t_{A_1}} = g^{t_{A_1}}.$$

   6.2 After verifying S_{A_1}, A also produces the hash value $t'_{A_1} = H_1(Exc_A||ID_A||h(m'_A)||h(W'_B)||R'_6)$ and compares it with t_{A_1}. On the other hand, B also performs the same verification as A does. If the signature and the hash value are valid, A prepares to authenticate with B. A prepares her nonce N'_A and computes the following equations.
      6.2.1 Choose a random number $\tau_A = \tau_a * P$, where $\tau_a \in_R Z_q^*$.
      6.2.2 Compute the hash value $h_A = H_1(N'_A||\tau_A||ID_A||Exc_A)$.

6.2.3 Generate the exchange message $C_1 = (h_A, ID_A, N'_A, Exc_A)$.

7. After generating the above tuples, A forwards $(N'_A, Exc_A, \tau_A, h_A, C_1)$ to B. When B receives this tuple, he checks that If $h_A = H_1(N'_A||\tau_A||ID_A||Exc_A)$ is valid or not. If yes, then he computes his response in the following.
   7.1 Compute the authentication token $T_{A,B} = h(\tau_A||\tau_B||\tau_b * \tau_A)$ and the hash value $h_B = H_1(N'_A+1||N'_B||ID_A||ID_B||Exc_A||Exc_B||\tau_B)$, where $\tau_B = \tau_b * P$, $\tau_b \in_R Z_q^*$.
   7.2 Generate the temporary key $k_4 = H_1(T_{A,B})$ and produce the ciphertext $C_2 = E_{k_4}(h_B, ID_B, N'_B, Exc_B)$.
   After generating the above tuples, B forwards (N'_B, τ_B, C_2) to A. When receiving this tuple from B, A decrypts C_2 and checks N'_B as the same process as B does. If it is valid, A can make sure that the current communicating party is B. Then A computes its response to B in the following.
   7.3 Compute the hash value $h'_A = H_1(N'_B + 1||Exc_B||T_{A,B})$.
   7.4 Compute the session key $ssk_{A,B} = H_2(\tau_A||\tau_B||\tau_a * \tau_b * P)$.
   After generating the above tuples, she forwards h'_A back to B. When B receives h'_A, he can verify it with the shared token $T_{A,B}$ and N'_B. If they are valid, B also computes its session key $ssk_{A,B} = H_2(\tau_A||\tau_B||\tau_b * \tau_A)$.
8. After preparing its session key $ssk_{A,B}$, A generates the signature $S'_{A_1} = \frac{t'_{A_1}}{s+q_A}P$ and the hash value $t'_{A_1} = H_1(t_A||S_{A_1}||R_7)$, where $R'_7 = r'_7 * P$ and $r'_7 \in_R Z_q^*$. Then she generates the ciphertext $E_{ssk_{A,B}}(S_{A_1}, t_{A_1}, S'_{A_1}, t'_{A_1}, Exc_A, Exc_B, h(W'_B), h(m'_A), R'_6, R'_7)$ and sends it back to B.
9. B decrypts the ciphertext after he has gotten the ciphertext $E_{ssk_{A,B}}(S_{A_1}, t_{A_1}, S'_{A_1}, t'_{A_1}, Exc_A, Exc_B, h(W'_B), h(m'_A), R'_6, R'_7)$. He produces the session key $ssk_{A,B}$ and decrypts $E_{ssk_{A,B}}(S_{A_1}, t_{A_1}, S'_{A_1}, t'_{A_1}, Exc_A, Exc_B, h(W'_B), h(m'_A), R'_6, R'_7)$.
   9.1 He performs the checking of the signatures (S_{A_1}, S'_{A_1}) and the hash values (t_{A_1}, t'_{A_1}) to see whether they are valid or not.
   9.2 If yes, then he produces $H_1(Exc_A||ID_A||h(m'_A)||h(W'_B||R'_6))$ and $H_1(t_{A_1}||S_{A_1}||R'_7)$ and compares them with t_{A_1} and t'_{A_1}, respectively. If they are valid, B believes that Exc_A was confirmed by the server S. Then B computes the ciphertext $E_{ssk_{A,B}}(S_{B_1}, Exc_B, t_{B_1}, h(W'_A), h(m'_B), m'_B, R'_5)$ and forwards it to A.
10. After receiving it, A does the followings.
    10.1 A decrypts $E_{ssk_{A,B}}(S_{B_1}, Exc_B, t_{B_1}, h(W'_A), h(m'_B), m'_B)$ and performs the checking as the same process as B does.
    10.2 If they are valid, A forwards $E_{ssk_{A,B}}(m'_A)$ to B. If B cannot decrypt it or it can not pass the verification, then B forwards $E_{ssk_{B,S}}(S_{A_1}, t_{A_1}, S'_{A_1}, t'_{A_1}, Exc_A, Exc_B, h(W'_B), h(m'_A), R'_6, R'_7)$ to S and could ask S for a dispute resolution.
11. After A and B obtain the desired digital contents, respectively, A and B can perform the following watermarking embedding operations, respectively.
    11.1 A decrypts m'_B with her random value r_1, i.e., $m'_B * H_1(i * R'_5) = m_B \oplus k_1$ and prepares $W'_A = W_A \oplus k_1$. Then she inputs $m_B \oplus k_1$ and W'_A into her watermarking algorithm under the $\otimes$ operation and obtains the final result $M'_A = m_B \otimes W'_A$.
    11.2 B also performs the same action as A does. He also can produce his own digital content $M'_B = m_A \otimes W'_B$.

4.5. **The resolution phase.** In this phase, B forwards A's signature to the server S and asks S for the dispute resolution.

1. Firstly, B prepares the partial signature $E_{ssk_{B,S}}(S_{A_1}, t_{A_1}, S'_{A_1}, t'_{A_1}, Exc_A, Exc_B, h(W'_B), h(m'_A), R'_6, R'_7)$ and forwards it to S.

2. S checks if each part of this tuple is valid or not. If they are valid, S accepts B's accuse and prepares A's digital content $E_{ssk_{B,S}}(k_2 \oplus m_A)$.
3. Finally, S forwards this ciphertext $E_{ssk_{B,S}}(k_2 \oplus m_A)$ to B. As receiving it from S, B decrypts $E_{ssk_{B,S}}(k_2 \oplus m_A)$ and performs his watermarking embedding operation on A's digital content.

5. **Security Analysis.**

- **Mutual-authentication.** In the authentication phase, A selects her nonce to authenticate the server S and B performs the same action as A does. If an attacker replays the nonce N_A or N_B, then it will be discovered by S. On the other hand, S also answers its nonce back to A and B. Then A and B check whether it is a fresh response of the server S or not. We have the formal security proof in the appendix.
- **Key-agreement.** In the key agreement phase, we know that the attacker cannot obtain the session key $ssk_{A,B}$ without knowing τ_a or τ_b in this phase. We claim that the attacker cannot have the non-negligible probability to guess the session key $ssk_{A,B}$ correctly under the k-mBDHI and k-CAA assumptions. We also have the formal security proof in the appendix.
- **Fair-exchange.** In the resolution phase, if A does not honestly forward her digital content to B, then B can forward A's signature to S and ask the server S to resolve this unfair situation. If it is valid, S can search A's digital content in its database and forward it to B. Hence, B can also obtain A's digital content in this phase. This approach can resolve the free-rider problem and can guarantee the fair exchange delivery in the p2p network. On the other hand, A only forwards its signature to B during the exchange phase. If B does not forward his digital content to A, then B also cannot be able to extract A's digital content from A's signature or hash value. The fair exchange also can be guaranteed in this situation.
- **Content-exchange.** In the content exchange phase, A generates her own digital content M'_A by embedding her watermark W'_A into the original digital content m_A. We assume that A and B's watermarks are robust that attackers cannot extract/embed their watermarks without knowing the embedding operation $\otimes$ in a polynomial time. We assume that the partial encryption operation $\oplus$ is secure and it can be applied with a homomorphic encryption operation such as the RSA encryption.

6. **Performance and Functionality Comparisons.** We assume that q is of 160 bits for security consideration [31]. Assume that H is the computation time of one hashing operation, Exp is the computation time of one modular exponential operation in a 1024-bit modulo, and M is the computation time of one modular multiplication in a 1024-bit modulo, EC_M is the computation time of the multiplication of an element over an elliptic curve, EC_P is the computation time for the bilinear pairing operation of two elements over an elliptic curve, and EC_A is the computation time of the addition of two elements over an elliptic curve [2, 21, 32]. By the way, we assume that the schemes [9, 30] whose encryption operation is about 1 Exp RSA encryption operation and let Sig, $SymEnc$ and $SymDec$ be the signature operation, symmetric encryption and symmetric decryption, respectively. Assume that an elliptic curve over a 163-bit field has the same security level of 1024-bit public key cryptosystems such as the RSA or the Diffie-Hellman cryptosystem [21]. Assume that $Exp \cong 8.24EC_M$ for the implementation with the StrongARM processor in 200MHz as referenced in [21]. We also can find in the relationship $Exp \cong 240M$, $Exp \cong 600H$, $Exp \cong 3.2EC_P$ and $EC_A \cong 5M$ [3, 4, 6], [10, 22, 42, 53]. In [51], we find the relation that a public key encryption/decryption in an elliptic curve is about 1 EC_A

TABLE 1. Properties comparisons

	P1	P2	P3	P4	P5	P6	P7	P8
Ours	Yes	Yes	Yes	Yes	$Medium$	$Optional$	Yes	Yes
[9]	No	No	No	No	N/A	N/A	No	Yes
[30]	No	No	No	No	$High$	N/A	Yes	Yes

P1: Content Exchange; P2: Authentication and Key Agreement
P3: Without Watermark Certificate Authority; P4: Watermark Ownership
P5: Computation Cost (Low: $\leq 500M$ | $Medium$: $1000M \geq$ and $\geq 500M$ | $High$: $> 1000M$)
P6: $Optional$($Robust$ or $Fragile$ Watermark)/$Robust$ Watermark
P7: Fairness; P8: P2P Environment
Yes: Satisfied; No: Not Satisfied; N/A: Not Provided

TABLE 2. Efficiency comparisons

	Registration and Key-agreement Phase	Content Exchange Phase	Resolution Phase	Total Costs	Approximation
Ours	$11H$+$3EC_A$ +$4EC_M$+$4EC_P$	$35H$+$1EC_A$ +$9EC_M$+$1EC_P$ +$17SymEnc$ +$6\oplus$+$2\otimes$	$3SymEnc$ +$1\otimes$	$46H$+$4EC_A$ +$13EC_M$+ $20SymEnc$ +$5EC_P$+$6\oplus$+$2\otimes$	$834M$+$6\oplus$+$2\otimes$
[30]	N/A	$7Exp$+$2H$	N/A	$7Exp$+$2H$	$1881M$

Exp: Modular Exponential Operation; M: Modular Multiplication Operation
$SymEnc/Dec$: Symmetric Encryption or Decryption over An Elliptic Curve
EC_P: Bilinear Pairing Operation over An Elliptic Curve; EC_M: Multiplication over An Elliptic Curve
EC_A: Addition over An Elliptic Curve; H: Hash Operation
$\otimes$: Watermark Embedding Operation; $\oplus$: Partial Encryption Operation; N/A: Not Provided

and assume that $PubEnc_{ec}/PubDec_{ec}$ is the public key encryption in an elliptic curve, respectively.

In [30], we find that their scheme does not have the formal security analysis. Their scheme also assumes that each peer has to apply a "capital certificate" before performing their fair exchange protocol. It also needs a Certificate Authority (CA) to manage the capital certificate list as the PKI infrastructure in the p2p network. We think that it is unsuitable and unreasonable for a p2p network since most of p2p networks work in the anonymous status without deploying and checking the certificate of each peer. By the way, the computation cost of their scheme is about $7Exp + 2H \cong 1881M$. In our scheme, the cost is about $834M$+$6\oplus$+$2\otimes$. Table 1 and Table 2 are the functionalities and performance comparisons.

7. **Conclusions.** In this paper, we offer the lightweight authentication method to provide fair and secure digital content exchange and avoid the free-rider problem in a p2p network. In our proposed scheme, the client peer also can authenticate the server peer in a p2p network, exchange her/his digital content fairly with the other peer and also provide many nice properties.

Acknowledgment. This work was partially supported by the National Science Council of the Taiwan under grants NSC 100-2221-E-327-019-MY2, NSC 100-2219-E-110-003, NSC 100-2219-E-110-005 and "Aim for the Top University Plan" of the National Sun Yat-sen University and Ministry of Education, Taiwan.

REFERENCES

[1] A. Alaraj and M. Munro, An efficient e-commerce fair exchange protocol that encourages customer and merchant to be honest, *Proc. of Computer Safety, Reliability, and Security, LNCS*, vol.5219, pp.193-206, 2008.

[2] A. Jurisic and A. J. Menezes, Elliptic curves and cryptography, *Dr. Dobb's Journal*, pp.23-36, 1997.

[3] A. Ramachandran, Z. Zhou and D. Huang, Computing cryptographic algorithms in portable and embedded devices, *Proc. of the IEEE International Conference on Portable Information Devices*, pp.1-7, 2007.

[4] B. Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons, 1996.

[5] C.-L. Chen, Y.-Y. Chen and Y.-H. Chen, Group-based authentication to protect digital content for business applications, *International Journal of Innovative Computing, Information and Control*, vol.5, no.5, pp.1243-1251, 2009.

[6] D. Hankerson, A. Menezes and M. Scott, Software implementation of pairings, in *Identity-Based Cryptography, Cryptology and Information Security Series*, M. Joye and G. Neven (eds.), 2008.

[7] D. Pointcheval and J. Stern, Security arguments for digital signatures and blind signatures, *Journal of Cryptology*, vol.13, pp.361-396, 2000.

[8] E. Adar and B. Huberman, Free riding on gnutella, *First Monday*, vol.10, no.5, 2000.

[9] G. Arora, M. Hanneghan and M. Merabti, P2P commercial digital content exchange, *Electronic Commerce Research and Applications*, vol.4, no.3, pp.250-263, 2005.

[10] G. Bertoni, L. Breveglieri, L. Chen, P. Fragneto, K. Harrison and G. Pelosi, A pairing SW implementation for smart-cards, *Journal of Systems and Software*, vol.81, no.7, pp.1240-1247, 2008.

[11] H. Du and Q. We, Efficient and provably-secure certificateless short signature scheme from bilinear pairings, *Computer Standards & Interfaces*, vol.31, no.2, pp.390-394, 2009.

[12] H. L. Jin, M. Fujiyoshi and H. Kiya, Lossless data hiding in the spatial domain for high quality images, *IEICE Trans. Fundamentals*, vol.E90-A, no.4, pp.771-777, 2007.

[13] IEEE P1363, *Standard Specifications for Public-Key Cryptography*, Draft Version D22, 2005.

[14] I. K. Jeong, O. Kwan and D. H. Lee, A Diffie-Hellman key exchange protocol without random oracles, *Proc. of CANS 2006, LNCS*, vol.4301, pp.37-54, 2006.

[15] I. M. Ibrahim and S. H. N. El-Din, An effective and secure buyer-seller watermarking protocol, *Proc. of IAS*, pp.21-28, 2007.

[16] J. Fridrich, M. Goljan and R. Du, Lossless data embedding-new paradigm in digital watermarking, *EURASIP J. Applied Signal Process.*, vol.2002, no.2, pp.185-196, 2002.

[17] J. Liu, R. Sun, W. Ma, Y. Li and X. Wang, Fair exchange signature schemes, *Proc. of Advanced Information Networking and Applications – Workshops*, pp.422-427, 2008.

[18] J. Tian, Reversible data embedding using a difference expansion, *IEEE Trans. Circuits Syst. Video Techno.*, vol.13, no.8, pp.890-896, 2003.

[19] J. Walkerdine, D. Hughes, P. Rayson, J. Simms, K. Gilleade, J. Mariani and I. Sommerville, A framework for P2P application development, *Computer Communications*, vol.31, no.2, pp.387-401, 2008.

[20] J.-W. Lo, C.-C. Lee, M.-S. Hwang and Y.-P. Chu, A secure and efficient ECC-based AKA protocol for wireless mobile communications, *International Journal of Innovative Computing, Information and Control*, vol.6, no.11, pp.5249-5258, 2010.

[21] K. Lauter, The advantages of elliptic curve cryptography for wireless security, *IEEE Wireless Communications*, vol.11, no.1, pp.62-67, 2004.

[22] K. Takashima, Scaling security of elliptic curves with fast pairing using efficient endomorphisms, *IEICE Trans. on Fundamentals*, vol.E90-A, no.1, pp.152-159, 2007.

[23] K. Y. Choi, J. Y. Hwang, D. H. Lee and I. S. Seo, ID-based authenticated key agreement for low-power mobile devices, *Proc. ACISP 2005, LNCS*, vol.3574, pp.494-505, 2005.

[24] M. Abdalla, M. Bellare and P. Rogaway, The oracle Diffie-Hellman assumption and an analysis of DHIES, *Proc. of CT-RSA01*, pp.143-158, 2001.

[25] M. A. Strangio, Efficient Diffie-Hellman two-party key agreement protocols based on ellptic curves, *Proc. of the 2005 ACM Symposium on Applied Computing*, pp.324-331, 2005.

[26] M. H. Shao, A privacy-preserving buyer-seller watermarking protocol with semi-trust third party, *Proc. of TrusBus, LNCS*, vol.4657, pp.44-53, 2007.

[27] M. U. Celik, G. Sharma, A. M. Teklp and E. Saber, Lossless generalized-LSB data embedding, *IEEE Trans. Image Process.*, vol.14, no.2, pp.253-266, 2005.

[28] M. Deng and B. Preneel, On secure and anonymous buyer-seller watermarking protocol, *Proc. of ICIW*, pp.524-529, 2008.
[29] M. J. Lee, K. S. Kim and H. K. Lee, Forensic tracking watermarking against in-theater piracy, *Proc. of the 11th International Workshop of Information Hiding 2009, LNCS*, vol.5806, pp.117-131, 2009.
[30] M. Zuo and J. Li, Constructing fair-exchange P2P file market, *Proc. of the Grid and Cooperative Computing, LNCS*, vol.3795, pp.941-946, 2005.
[31] NIST FIPS PUB 186-2, *Digital Signature Standard*, National Institute of Standards and Technology, U. S. Department of Commerce, 2001.
[32] N. Koblitz, A. Menezes and S. Vanstone, The state of elliptic curve cryptography, *Designs, Codes and Cryptography*, vol.19, pp.173-193, 2000.
[33] N. Hopper, D. Molnar and D. Wagner, From weak to strong watermarking, *Proc. of Theory of Cryptography, LNCS*, vol.4392, pp.362-382, 2007.
[34] N. W. Lo and K.-H. Yeh, A novel authentication scheme for mobile commerce transactions, *International Journal of Innovative Computing, Information and Control*, vol.6, no.7, pp.3093-3103, 2010.
[35] N. W. Lo and K.-H. Yeh, A practical three-party authenticated key exchange protocol, *International Journal of Innovative Computing, Information and Control*, vol.6, no.6, pp.2469-2483, 2010.
[36] P. Gilbert, Forensic analysis of digital image tampering, *Proc. of the IFIP International Federation for Information Processing*, pp.259-270, 2006.
[37] Q. Huang, G. Yang, D. S. Wong and W. Susilo, Ambiguous optimistic fair exchange, *Proc. of Advances in Cryptology-ASIACRYPT 2008, LNCS*, vol.5350, pp.74-89, 2008.
[38] R. Aringhieri and D. Bonomi, A simulation model for trust and reputation system evaluation in a P2P network, *Proc. of Computational Intelligence, Theory and Applications*, pp.169-180, 2006.
[39] R.-C. Wang, W.-S. Juang and C.-L. Lei, A robust authentication scheme with user anonymity for wireless environments, *International Journal of Innovative Computing, Information and Control*, vol.5, no.4, pp.1069-1080, 2009.
[40] R. Tso, X. Yi and X. Huang, Efficient and short certificateless signature, *Proc. of CANS 2008, LNCS*, vol.5339, pp.64-79, 2008.
[41] S. Han, M. Fujiyoshi and H. Kiya, An efficient reversible image authentication method, *IEICE Trans. Fundamentals*, vol.E91-A, pp.1907-1914, 2008.
[42] S. Hohenberger, *Advances in Signatures, Encryption, and E-Cash from Bilinear Groups*, Ph.D. Thesis, Massachusetts Institute of Technology, 2006.
[43] S. Siersdorfer and S. Sizov, Automatic document organization in a P2P environment, *European Conference on IR Research, LNCS*, vol.3936, pp.265-276, 2006.
[44] S. Mitsunari, R. Sakai and M. Kasahara, A new traitor tracing, *IEICE Trans. Fundamentals*, vol.E85-A, no.2, pp.481-484, 2002.
[45] T.-S. Wu, H.-Y. Lin, C.-L. Hsu and K.-Y. Chang, Efficient verifier-based authenticated key agreement protocol for three parties, *International Journal of Innovative Computing, Information and Control*, vol.6, no.2, pp.755-762, 2010.
[46] V. D. V. Michiel, N. L. Aweke and K. Ton, Electronic content delivery and forensic watermarking, *Multimedia Systems*, vol.11, no.2, pp.174-184, 2005.
[47] V. V. Das, Buyer-seller watermarking protocol for an anonymous network transaction, *Proc. of ICETET*, pp.807-812, 2008.
[48] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, vol.IT-22, no.6, pp.644-654, 1976.
[49] W. S. Jaung, Efficient three-party key exchange using smart cards, *IEEE Transactions on Consumer Electronics*, vol.50, no.2, pp.619-624, 2004.
[50] W.-S. Juang, RO-cash: An efficient and practical recoverable pre-paid offline E-cash scheme using bilinear pairings, *Journal of Systems and Software*, vol.83, no.4, pp.638-645, 2010.
[51] W.-S. Juang, C.-L. Lei, H.-T. Liaw and W.-K. Nien, Robust and efficient three-party user authentication and key agreement using bilinear pairings, *International Journal of Innovative Computing, Information and Control*, vol.6, no.2, pp.763-772, 2010.
[52] Y. Hu, A watermarking protocol for privacy tracing, *Proc. of ISECS*, pp.882-885, 2008.
[53] Z. Li, J. Higgins and M. Clement, Performance of finite field arithmetic in an elliptic curve cryptosystem, *Proc. of the 9th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pp.249-256, 2001.

[54] Z. Zhang, S. Zhou, W. Qian and A. Zhou, KEYNOTE: Keyword search by node selection for text retrieval on DHT-based P2P networks, *Proc. of the 11th International Conference on Database Systems for Advanced Applications, LNCS*, vol.3882, pp.797-806, 2006.

Appendix A. Security Analysis.

A.1. **Mutual authentication.** We define the security of the proposed protocol. Assume that each party's identity is denoted as p_i and each p_i holds a pair of private/public keys, where $1 \leq i \leq I$ and I denotes the set of the identities of the parties who can participate in our proposed protocol.

Assume Π_i^k represents the k-instance of the party p_i. If p_i accepts with its partner p_j, then it means that p_i holds a session key $ssk_{i,j}$ which is shared with party p_j. Let a session identifier of the instance Π_i^k be denoted as sid_i^k that presents the k-th session which is different from other sessions in this protocol, where $k \in N$ and N being the set of positive integers. In the following, we model the capabilities of an adversary. We allow the adversary $\mathcal{A}$ that she/he can control all communication in the proposed protocol via accessing to the oracles. Following are the queries that an adversary can query.

- A query $Send_C(\Pi_i^k, M)$ is used that an attacker $\mathcal{A}$ sends a message to the instance Π_i^k such that i believes the message has been sent from server S. If $\mathcal{A}$ makes a $Send_C(\Pi_i^k, M)$ query with $M =$ "start" to Π_i^k, then the Π_i^k will be instructed to initiate a protocol run. An initial oracle is that if the first message is "start". Otherwise, it is a responder oracle.
- A query $Send_S(\Pi_S^k, M)$ is used that a server oracle received message M and it computes the response message and returns it back to the attacker $\mathcal{A}$.
- A query $Reveal(i(j), k)$ is used to send a session key of Π_i^k (or $\Pi_j^{k'}$) to the adversary.
- A query $Corrupt(i)$ is used to expose the private key of the oracle Π_i.
- A query $Test(\Pi_{i,j}^t)$ is used to define the advantage of an adversary. When an adversary $\mathcal{A}$ asks a $Test$ query to the "fresh" instance $\Pi_{i,j}^t$ in the t-th session, where a coin B is flipped. If b is 1, then return the session key hold by $\Pi_{i,j}^t$. Otherwise, return a random string chosen uniformly from $\{0,1\}^*$ to $\mathcal{A}$.

Next, we give the security definitions of our proposed scheme. These definitions are defined in the following.

Definition A.1. *(Partner) First, we define what is the partner function. We assume that there exists an instance Π_i^k of the party p_i and the partner of Π_i^k is the instance $\Pi_j^{k'}$ of the party $p_j (\neq p_i)$ who believes that it is interacting. We can say that two instances Π_i^k and $\Pi_j^{k'}$ are partnered if the following properties are satisfied:*

1. $sid_i^k = sid_j^{k'}$.
2. *p_j is the partner of Π_i^k.*
3. *p_i is the partner of $\Pi_j^{k'}$.*

Definition A.2. *(Matching Conversation) Fix a number of flows $R = 2\rho - 1$ and an R-flow protocol Π. Run Π in the presence of an adversary E and consider two oracles $\Pi_{i,j}^s$, an initiator oracle, and $\Pi_{j,i}^t$, a responder oracle, that engage in conversation C and C', respectively.*

1. *We say that C' is a matching conversation to C if there exist $\tau_0 < \tau_1 < \ldots < \tau_{R-1}$ and $a_1 = a'_1, \ldots, a_\rho = a'_\rho$ and $\alpha_1 = \alpha'_1, \ldots, \alpha_{\rho-1} = \alpha'_{\rho-1}$ such that C is prefixed by:*

$$(\tau_0, \lambda, (m_1, a_1)), (\tau_2, (\mu_1, \alpha'_1), (m_2, a_2)), \ldots, (\tau_{2\rho-2}, (\mu_{\rho-1}, \alpha'_{\rho-1}), (m_\rho, a_\rho))$$

and C' is prefixed by:

$$(\tau_1,(m_1,a'_1),(\mu_1,\alpha_1)),(\tau_3,(m_2,a'_2),(\mu_2,\alpha_2)),\ldots,(\tau_{2\rho-3},(m_{\rho-1},a'_{\rho-1}),(\mu_{\rho-1},\alpha_{\rho-1})).$$

2. *We say that C is a matching conversation to C' if there exist $\tau_0<\tau_1<\ldots<\tau_R$ and $a_1=a'_1,\ldots,a_\rho=a'_\rho$ and $\alpha_1=\alpha'_1,\ldots,\alpha_{\rho-1}=\alpha'_{\rho-1}$ such that C' is prefixed by:*

$$(\tau_1,(m_1,a'_1),(\mu_1,\alpha_1)),(\tau_3,(m_2,a'_2),(\mu_2,\alpha_2)),\ldots,(\tau_{2\rho-3},(m_{\rho-1},a'_{\rho-1}),$$
$$(\mu_{\rho-1},\alpha_{\rho-1}),(\tau_{2\rho-1},(m_\rho,a'_\rho),*)$$

and C is prefixed by:

$$(\tau_0,\lambda,(m_1,a_1)),(\tau_2,(\mu_1,\alpha'_1),(m_2,a_2)),\ldots,(\tau_{2\rho-2},(\mu_{\rho-1},\alpha'_{\rho-1}),(m_\rho,a_\rho)).$$

If C is a matching conversation including appendices to C' and C' is a matching conversation including appendices to C, $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ are said to have had matching conversations including appendices.

Definition A.3. *($No\text{-}matching_E(k)$) Let $No\text{-}matching_E(k)$ denote the event that, there exist (i,j,s) such that an oracle $\Pi^s_{i,j}$ accepted but there is no oracle $\Pi^t_{j,i}$, which has had a matching conversation to $\Pi^s_{i,j}$ under the presence of a polynomial time attacker E, with $i,j\notin I$ (where I denotes the set of the parties corrupted by E) and $s,t\in N$.*

Definition A.4. *($Good\text{-}Guess_E(k)$) We say $Good\text{-}Guess_E(k)$ that is the event that, there exist an polynomial time attacker E that she/he can correctly guess that she/he is given the real session key or a random number after performing the Π protocol successfully, where k is the system security parameter.*

Definition A.5. *A protocol Π is a secure mutual authentication protocol if for every adversary E:*

1. *If $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ have matching conversation, then both oralces accept.*
2. *The probability of $No\text{-}matching_E(k)$ is negligible, where k is the system security parameter.*

Definition A.6. *A protocol Π is a secure authentication and key agreement protocol if the following requirements are satisfied:*

1. *If Π is a secure mutual authentication protocol.*
2. *If $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ hold the same session key after performing Π protocol successfully.*
3. *The probability of $Good\text{-}Guess_E(k)$ is negligible.*

Definition A.7. *(Freshness) A instance Π^k_i is fresh if the following conditions are true at the conclusion of the simulation of the proposed protocol:*

1. *If $\Pi^k_{i,j}$ has not been queried, $Reveal(i,k)$.*
2. *If there existed $\Pi^{k'}_{j,i}$ partner with $\Pi^k_{i,j}$ where $\Pi^{k'}_{j,i}$ has not been queried, $Reveal(i,k)$.*
3. *The partner of $\Pi^k_{i,j}$ is not an insider attacker.*

Definition A.8. *(Forward Security(FS)) An adversary $\mathcal{A}$ is now allowed to ask all queries including $Corrupt(i)$ (or $Corrupt(j)$) which returns the secret key of party p_i (or p_j) to $\mathcal{A}$ during the simulation. A protocol Π is forward secure that if $\mathcal{A}$ cannot compromise the past information even the $Corrupt(i)$ (or $Corrupt(j)$) are queried.*

Theorem A.1. *Our p2p fair content exchange scheme (P2PFCES for short) is a (t_0, ε_0, q_h, q_{H_1}, q_s, q_E)-secure mutual authentication and key agreement protocol under the assumption that k-CAA problem is intractable.*

Lemma A.1. *We assume that the hash functions h, H_1 are random oracles. Suppose that there exists a Forger E for given ID_A, ID_B, ID_S with running time t_0 and advantages ε_0. We suppose that E can make the following q_h times h query, q_{H_1} times H_1 query, q_s times Send query and q_E times Extract query and total q times query, respectively. If $\varepsilon_0 > 10q_{H_1}^2(q_s+1)(q_s+q_{H_1})/q$, there exists a attacker $\mathcal{B}$ that can solve the k-CAA problem with time $t_1 \leq 120686q_{H_1}t_0/\varepsilon_0$.*

Proof: Suppose $No\text{-}matching_E(k)$ is non-negligible and is not less than ε_0. We can construct a simulator $\mathcal{B}$ from a challenger ψ. $\mathcal{B}$ is given the k-CAA problem instance $\left(P, sP, q_1, \ldots, q_k, \frac{1}{s+q_1}P, \frac{1}{s+q_2}P, \ldots, \frac{1}{s+q_k}P\right)$ from ψ, where $k \geq q_h, q_s$. $\mathcal{B}$'s goal is to compute $\frac{1}{s+q_0}P$ for some q_0. Then $\mathcal{B}$ takes E as its subroutine and simulates attacking environment for E. $\mathcal{B}$ first prepares the system parameters (e, G_1, G_1) and (g, P, h, H_1, H_2) with $g = e(P, P)$. $\mathcal{B}$ publishes these parameters to E.

Before simulation, we assume that E can query the following queries, h-query, H_1 query, *Send* query and *Extract* query at most once. $\mathcal{B}$ can response *Send* and *Extract* queries with the help of the saved record of the h-query. $\mathcal{B}$ also creates the list L_h and L_{H_1} for simulation usage and avoids collision during the simulation. In order to response the above queries, $\mathcal{B}$ can prepare the following queries.

- h-query: When E makes the h-query to $\mathcal{B}$ with ID_i and if $ID_i = ID_A$, $\mathcal{B}$ chooses q_0 from the random oracle h as the hash value of ID_A. Otherwise, $\mathcal{B}$ randomly chooses a q_i, returns $< q_i, ID_i >$ from the k-CAA problem instance to E and also adds it to list L_h.
- H_1-query: When E makes the H_1-query to $\mathcal{B}$ with message m, $\mathcal{B}$ chooses $\mathbf{h}$ from the H_1 oracle and returns it back to E. Then $\mathcal{B}$ adds $< m, \mathbf{h} >$ to list L_{H_1}.
- *Send*-query: When E makes the $Send(\Pi_i)$-query to $\mathcal{B}$, if $ID_i = ID_A$, $\mathcal{B}$ computes $(Exc_i, N_i, ID_i, G_i, S_i)$, where $G_i = ab(sP + q_0P)$, $a, b \in Z_q^*$ and generates $S_i = h^*P$ from the random oracle h, where it implies $h^* = \frac{ab+\mathbf{h}}{s+q_0}$. Otherwise, $\mathcal{B}$ finds $< q_i, ID_i >$ and $< m, \mathbf{h} >$ and prepares the hash value q_i and $\mathbf{h}$ from list L_h and L_{H_1}, respectively. Then $\mathcal{B}$ computes $G_i = ab(sP + q_iP)$ and $S_i = (ab + \mathbf{h})\frac{1}{s+q_i}P$ and returns it back to E.
- *Extract*-query: When E makes the *Extract*-query on $ID_i \notin < ID_A, ID_B, ID_S >$, $\mathcal{B}$ returns $\frac{1}{s+q_i}P$ to E.

After response these queries to E, if E can generate a tuple $< ID_i, G_i, S_i >$ without querying other oracle except H_1, then $\mathcal{B}$ can replay the same tape with different choices of H_1 as done in the forking lemma [7]. Finally, E outputs two different messages $< ID_i, G_i = ab(sP + q_0P), S_i = (ab + \mathbf{h})\frac{1}{s+q_0}P >$ and $< ID_i, G_i' = ab(sP + q_0P), S_i' = (ab + \mathbf{h}')\frac{1}{s+q_0}P >$, such that $\mathbf{h} \neq \mathbf{h}'$. Then $\mathcal{B}$ can take E's two outputs and solve the k-CAA problem by computing $\frac{S_i'-S_i}{\mathbf{h}'-\mathbf{h}} = \frac{1}{s+q_0}P$ for some $q_0 \in Z_q^*$.

The probability that $\mathcal{B}$ correctly guesses $\mathbf{h}$ and $\mathbf{h}'$ is $1/q_{H_1}^2$. By the forking lemma [7], the probability that $\mathcal{B}$ solves the k-CAA problem is $\varepsilon_0 > 10q_{H_1}^2(q_s + 1)(q_s + q_{H_1})/q$. On the other hand, the total time t_1 of $\mathcal{B}$ is equal to the running time of the forking lemma that is bounded by $120686q_{H_1}t_0/\varepsilon_0$. □

Theorem A.2. *The proposed protocol P2PFCES is said (t, ε, q_s, q_h, q_c)-forward-secure against existential forgery attack in the random oracle model if there is no probability polynomial time adversary $\mathcal{B}$ who can (t', ε')-break the k-CAA and k-mBIDH problem, where $\varepsilon' \geq \left((1-\varepsilon)\frac{1}{2^{2l}} \cdot \frac{1}{q_{H_2}} \cdot \frac{1}{q_s} - \frac{1}{2}\right) + \frac{10q_{H_1}^2(q_s+1)(q_s+q_{H_1})}{q}$, $t' = t + (q_s + q_{H_2})t_{rn} + \frac{120686q_{H_1}t_0}{\varepsilon_0}$, t_{rn} denotes the time to produce a random number, t_0 denotes the time that attacker E*

breaks the k-CAA problem with the probability ε_0 at most q_s send queries, q_h hash queries, and q_c corrupt queries.

Proof: First, we consider an adversary $\mathcal{A}$ attacking the $P2PFCES$ in the sense of the forward security. Let $Forge$ be the event that there exists at least one forged signature in the $P2PFCES$. We can derive that

$$\Pr_{\mathcal{A}}[\textit{Good-Guess}_E(k)] \geq \Pr_{\mathcal{A}}[Forge] + \Pr_{\mathcal{A}}[\textit{Good-Guess}_E(k) \wedge \overline{Forge}],$$

where b and b' are the coin flips chosen by the simulator and the attacker, respectively. Beginning this proof, we also define other two lemmas to complete this proof. □

Lemma A.2. *We claim that there is no attacker $\mathcal{A}$ that can forge the authentication transcripts with the non-negligible probability $\varepsilon_0 = \Pr_{\mathcal{A}}[Forge] \geq \frac{10q_{H_1}^2(q_s+1)(q_s+q_{H_1})}{q}$ in the polynomial time bound $t' \leq \frac{120686q_{H_1}t_0}{\varepsilon_0}$.*

Proof: First, we consider the following case that when there exists a forger that she/he can forge the authentication transcripts between user A or B. It means that she/he can impersonate a client to authenticate with server S. In order to compute the advantage of $\mathcal{A}$ from this case, we use $\mathcal{A}$ to construct a challenger $\mathcal{F}$ that generates a valid message $< ID_i, G_i, S_i >$. $\mathcal{F}$ generates all key pairs for users A, B and server S. Then it simulates all oracle queries to $\mathcal{A}$ as in Theorem A.1. Let $Forge$ be the event that $\mathcal{A}$ generates a new and valid message tuple $< ID_i, G_i, S_i >$. The probability of $\mathcal{A}$ satisfies

$$\Pr_{\mathcal{A}}[Forge] \geq \frac{10q_{H_1}^2(q_s+1)(q_s+q_{H_1})}{q},$$

where $t' \leq \frac{120686q_{H_1}t_0}{\varepsilon_0}$. By Lemma A.1, we can conclude that the probability $\Pr_{\mathcal{A}}[Forge]$ is negligible. □

Lemma A.3. *We claim that there is no attacker $\mathcal{A}$ that can correctly guess the session key with the non-negligible probability*

$$\Pr\left[\textit{Good-Guess}_E(k) \wedge \overline{Forge}\right] \geq (1-\varepsilon)\frac{1}{2^{2l}} \cdot \frac{1}{q_{H_2}} \cdot \frac{1}{q_s}$$

in the polynomial time $t' \leq t + (q_s + q_{H_2})t_{rn}$, where t_{rn} denotes the time to produce a random number.

Proof: In the second case, $\mathcal{A}$ attempts to get the session key $ssk_{A,B}$ in the random oracle model, she/he must ask the $H_2(\tau_A||\tau_B||\tau_a{*}\tau_b{*}P)$ H_2 query in the A's aspect. On the other hand, it is the same situation in B's aspect. So we just consider the communication between A and S. We construct the attacker $\mathcal{B}$ which breaks the k-mBIDH problem using $\mathcal{A}$ as the subroutine with non-negligible probability. First, $\mathcal{B}$ is given an instance of the k-mBIDH problem $\left(e, G_1, G_2, P, sP, tP, h, h_1, h_2, \ldots, h_k, \frac{1}{s+h_1}P, \frac{1}{s+h_2}P, \ldots, \frac{1}{s+h_k}P\right)$, where $k \geq q_{H_2}, q_s$. Its goal is to compute $e(P,P)^{\frac{1}{s+h}t}$. $\mathcal{B}$ sets the public system parameters $< e, G_1, G_2, P, P_{pub}, g, h', H_1, H_2 >$ which it lets $P_{pub} = sP$, $g = e(P,P)$, $S_S = \frac{1}{s+h_1}P$ and $Q_S = sP + h_1P$ accompanying with h', H_1 and H_2 random oracles. Then $\mathcal{B}$ gives these parameters to $\mathcal{A}$. In order to compute the advantage of $\mathcal{A}$, $\mathcal{B}$ guesses α correctly such that $\mathcal{A}$ asks its $Test$ query in the α-session. After preparing these system parameters, $\mathcal{B}$ also simulates the oracle queries of $\mathcal{A}$ in the following.

- h'-query: When $\mathcal{A}$ makes the query with ID_i and if $ID_i = ID_A$, then $\mathcal{B}$ chooses h from the k-mBIDH problem instance and sets $h = h(ID_A)$ and returns it to $\mathcal{A}$. Otherwise, it forwards h_i and adds $< ID_i, h_i >$ into the empty list $L_{h'}$, where $i = 2 \sim k$.

- H_1 query: If $\mathcal{A}$ makes H_1 query on input m with $ID_i = ID_A$, $\mathcal{B}$ sets tP to be the hash value of ID_A, where it implies $t = H_1(m)$ and adds it to the list L_{H_1}. Otherwise, $\mathcal{B}$ generates a random number $\mathbf{h}$ and adds $< m, \mathbf{h} >$ into the list L_{H_1} and returns $\mathbf{h}$ to $\mathcal{A}$.
- H_2 query: If $\mathcal{A}$ makes H_2 query on input m with $ID_i \neq ID_A$ and $ID_j \neq ID_B$, $\mathcal{B}$ searches the list L_{H_2} and sees if there exists a record $< m, k >$ in it. If yes, $\mathcal{B}$ returns k to $\mathcal{A}$.
- $Send_S(\Pi_i^k, M)$ query: In this query, we define it as client-to-server type query. When $\mathcal{A}$ makes a $Send_S(\Pi_i^k, M)$ query with $ID_i = ID_A$ and if the query is in the α session, $\mathcal{B}$ fails.

  Otherwise, $\mathcal{B}$ searches to see whether if $< ID_i, h_i >$ is in $L_{h'}$ and $\mathbf{h}$ is in L_{H_1} or not, respectively. If yes, $\mathcal{B}$ selects two random numbers a, b and computes $G_i = (ab+\mathbf{h})Q_S$, $t_{u_i} = e((ab + \mathbf{h})P, P) = e(P, P)^{ab+\mathbf{h}}$ and $S_i = \frac{1}{s+h_i}P$, where $i = 2 \sim k$. $\mathcal{B}$ appends $< ID_i, G_i, S_i, t_{u_i} >$ into the empty list L_{Send_S} and returns $< ID_i, G_i, S_i, t_{u_i} >$ to $\mathcal{A}$.
- $Send_C(\Pi_i^k, M)$ query: In this query, we define it as server-to-client type query. When $\mathcal{A}$ makes a $Send_C(\Pi_i^k, (ID_i, G_i, S_i))$ query, $\mathcal{B}$ chooses two random numbers h^* and t_{v_i} and sets $z_i = h^*$. Then it returns (z_i, t_{v_i}) to $\mathcal{A}$. $\mathcal{B}$ also stores $< M, z_i, t_{v_i} >$ into the empty list L_{Send_C}, where $i = 2 \sim k$.
- *Execute* query: When $\mathcal{A}$ asks an $Execute(A, S)$ query, then $\mathcal{B}$ returns the records $< (ID_i, G_i, S_i, t_{u_i}), (z_i, t_{v_i}) >$ by using the simulation result of above $Send_S$ and $Send_C$ queries.
- $Corrupt(i)$ (or $Corrupt(j)$) query: When $\mathcal{A}$ asks an $Corrupt(i)$ query on ID_i, where $ID_i \neq ID_A$, $\mathcal{B}$ finds the record $< ID_i, h_i >$ in list $L_{h'}$. Then $\mathcal{B}$ returns $\frac{1}{s+h_i}P$ to $\mathcal{A}$.
- $Reveal(\Pi_{i,j}^k)$ query: When $\mathcal{A}$ asks an $Reveal(\Pi_{i,j}^k)$ query and if the session is not the α session, then $\mathcal{B}$ returns a random number to $\mathcal{A}$. Otherwise, $\mathcal{B}$ delays to response in the *Test* query.
- *Test* query: When $\mathcal{A}$ asks an *Test* query and if the query is not asked in the α-th session, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ flips a coin b; if $b = 1$, it returns the real session key, else it returns a random number to $\mathcal{A}$.

The success probability of $\mathcal{B}$ depends on the event that $\mathcal{A}$ can forge a target value $t_u = e(P, P)^{\frac{1}{s+h}t}$ with the signature $S_A = \frac{t}{s+h}P$ and the hash value $t = H_1(m)$ and $\mathcal{B}$ correctly guesses the session key $ssk_{A,B}$ in the α session. In the above simulation, the probability that $\mathcal{B}$ correctly guesses α is $\frac{1}{q_s}$. Let *Forge* be the event that $\mathcal{A}$ can forge the signature S_A in the α session with a non-negligible probability ε. Then we can have $\Pr[Forge] = \varepsilon$.

Without querying the hash oracle H_2, the probability of $\mathcal{A}$ correctly guessing the hash oracle value is no longer than $\frac{1}{2^{2l}}$ on the random number $\tau_a * \tau_b * P$. We define this event as the N_{H_2} and its probability is $\Pr[N_{H_2}] \leq \frac{1}{2^{2l}} \cdot \frac{1}{q_{H_2}}$. Then we can derive that the probability of $\mathcal{A}$ without outputting a signature value S_A is $\Pr[\overline{Forge}] \geq 1 - \varepsilon$. Thus, $\mathcal{B}$ does not forge the signature S_A and guesses the session key $ssk_{A,B}$ in the α session correctly with the probability $\Pr[\textit{Good-Guess}_E(k) \wedge \overline{Forge}] \geq (1 - \varepsilon)\frac{1}{2^{2l}} \cdot \frac{1}{q_{H_2}} \cdot \frac{1}{q_s}$. □

Therefore, we summarize that the advantage of $\mathcal{B}$ of Lemma A.2 and Lemma A.3 is $\Pr[\textit{Good-Guess}_E(k) \wedge \overline{Forge}] - \frac{1}{2} \geq \left((1 - \varepsilon)\frac{1}{2^{2l}} \cdot \frac{1}{q_{H_2}} \cdot \frac{1}{q_s} - \frac{1}{2}\right)$, where $t' = t + (q_s + q_{H_2})t_{rn}$. Hence, we have

$$\varepsilon' \geq \left(\frac{(1 - \varepsilon)}{2^{2l} q_{H_2} q_s} - \frac{1}{2}\right) + \frac{10 q_{H_1}^2 (q_s + 1)(q_s + q_{H_1})}{q},$$

where $t' \leq t + (q_s + q_{H_2})t_{rn} + \frac{120686 q_{H_1} t_0}{\varepsilon_0}$. □