2016 IEEE 40th Annual Computer Software and Applications Conference

Interactive Incontestable Signature for Transactions Confirmation in Bitcoin Blockchain

Yan Zhu, Ruiqi Guo, Guohua Gan
School of Computer and Communication Engineering
Univeirsity of Science and Technology Beijing, 100083
Email:yanzhu@ustb.edu.cn

Wei-Tek Tsai
School of Comp., Info. and Dec. Systems Engineering
Arizona State University, 85287
Email: wtsai@asu.edu

***Abstract*—Blockchain is a radical innovation that has a significant impact on payments, stock exchanges, cybersecurity, and computational law. However, it has significant limitations regarding uncertainty for a transaction to be confirmed. This paper proposes a new system for exact confirmation of transactions in a block. Replacing original signature, a new Interactive Incontestable Signature (IIS) scheme is used between dealer and owner to confirm a transaction. By this signature, the dealer can assure the owner that a transaction will be included into blockchain in a non-repudiation way. The scheme is proved to be secure for owner's unforgeability and dealer's incontestability.**

***Index Terms*—Blockchain, Signature, Interactive Proof.**

I. Introduction

Blockchain, as the core technology behind Bitcoin, has seen widespread use recently. It provides a decentralized and consistent mechanism, and promises to become an infrastructure for various application fields, such as online payments, stock exchange, cybersecurity and computational law. This technology has been a primary focus of interest from many financial institutions, e.g., venture capital invested in blockchain-related companies has accelerated considerably over the past three years and is on track to top $600 million in 2015 [1]. Currently, the foundational layer and infrastructure necessary to support a rich ecosystem of blockchain-based applications and services is being established.

Despite its potential, blockchain faces many barriers as well. At present, the major kind of faults produced in blockchain is *block conflict*, which indicates that a *fork* in the blockchain can occur if two blocks are published nearly simultaneously. The current solution for conflict is based on "*Longest Chain Rule*" (LCR) [2]: if you see multiple blocks, treat the longest chain as legitimate. This means that node follows the protocol rule that they will only try to extend the longest branch they know about. This rule causes a few of transactions on the wrong side of the fork to be delayed since they would be reorganized into new blocks (called blockchain reorganization). It also faces risks if double-spending attack is attempted. In the current implementation, a new block is generated about every 10 minutes [2]. The uncertainty of whether an established block is in the prevailing branch leads to a common rule that a given transaction is not confirmed until it is at least 6 blocks deep in the chain.

Though the conflict could be resolved, "*instant confirmation*" is still hard to reach. The confirmation is a verification process that offers a final proof after validating a certain transaction. It is necessary, whenever a transaction is received, to get confirmation from other nodes on the network that the transaction is indeed valid. In Bitcoin, however, there is no notion of "instant confirmation", due to the following reasons:

- A transaction is implicitly confirmed through being included into a block which is followed up by approximate 6 blocks (as described above). This process, taking at least one hour, causes a high confirmation latency.
- Even after such a long wait, the transaction is not confirmed in total finality (which means the transaction is permanently included into the blockchain). In fact, it just offers 99.9999% finality [3] after two hours as does Bitcoin.
- To check whether a transaction has been validated, client needs to search the most recent 6 blocks (6 additional blocks for an implicit confirmation, also called confirmed by 6 blocks) after he/she publishes this transaction one hour later. This is a large computational overhead because the search volume is about 24,000 transactions.

The search volume discussed above is computed by 24,000= 6×4,000 according to maximum of roughly 4,000 transactions per block (the average transaction size is around 200-250 bytes and the block size is at most 1MB in Bitcoin currently).

To conclude the above discussions, "instant confirmation" is still a challenge for the current blockchain. The important takeaway though is that there is no absolute notion of "permanently included" and the blockchain simply uses a reasonably safe policy of considering transactions confirmed when they are included with very high probability. The confirmation time is quite variable, taking from tens of minutes to over two hours, and on average it will take about an hour.

Contribution. In this paper we address the problem of implementing the instant confirmation with incontestability in blockchain. Based on two basic assumptions, we propose an interactive signature protocol to achieve our goal. This protocol, replacing the original signature scheme of transactions, is called Interactive Incontestable Signature (IIS), and it works between dealer and owner to implement the instant confirmation with incontestability. By this signature, the dealer can assure the owner that a transaction will be permanently included in the blockchain in a non-repudiation way. In addition, this signature is short and easy-to-build in a 3-move simple way.

Our signature scheme is constructed on general bilinear map group system. We also prove the security of scheme under the unforgeability of owner and the incontestability of dealer based on two extended computational bilinear Diffie-Hellman assumptions, eCBDH_1 and eCBDH_2, respectively. Our experimental results shown that the scheme has good properties: short signature, high performance, and low key storages.

Organization. The rest of this paper is organized as follows: Section II presents our system model and requirements. Our construction and security analysis of Interactive Incontestable Signature (IIS) is presented in Section III. The performance evaluation is shown in Section IV. Finally, the conclusion and future extensions are discussed in Section V.

II. System Model

A. Design Objectives

Our model addresses the problem of building a blockchain with more exact confirmation and higher performance than existing blockchain. In order to achieve instant confirmation, it is required that each transaction will receive feedback from the block generator after it passes the verification process, and generators cannot deny their prior verification behavior at any time. Exactly, our work focuses on the following properties:

- **Confirmability:** represents the ability that a transaction will not face the risk of becoming invalid later, once it is appended into a new block.
- **Incontestability:** refers to the ability to ensure that the block generator (*Dealer*) cannot deny the prior behavior after the transaction is included into block in blockchain.

B. Our System

Our system is built on the existed Bitcoin Blockchain with some improvements. The improvements contain two aspects: one is that the dealer in our system is designated before the beginning of generating block and unique at the same time, and another is that the situation of ”blockchain reorganization” is not expected to happen once new block is generated.

In order to design such a system, there is a need to introduce a new concept, called *accounting cycle*, into our system, which indicates the time interval between two contiguous generated blocks in blockchain. The accounting cycle corresponds to two basic assumptions: **(1) there is one single dealer in each accounting cycle**, and **(2) block conflict won’t occur in blockchain (no fork)**. Obviously, these two assumptions fail in existing blockchain protocol in Bitcoin. However, some work necessary to hold them has already been done. Eyal et al. [4] proposed a blockchain protocol that includes two block types: the key blocks for leader election and the microblocks for transaction storage. In their approach, leader election ensures that only one node will be elected as the dealer to generate a block in each accounting cycle. As a result, it further eliminates the possibility of fork problem.

Our system model is shown in Fig.1. In each accounting cycle, a set of transaction owners (e.g., $O_1, \cdots, O_n$) transmit their transactions to the designated dealer; then the dealer authenticates the transactions’ owners and validates transactions respectively before processing confirmed transactions into new block. This new block complies with the blockchain structure in Bitcoin, which includes hash pointer between blocks and merkle tree among transactions.

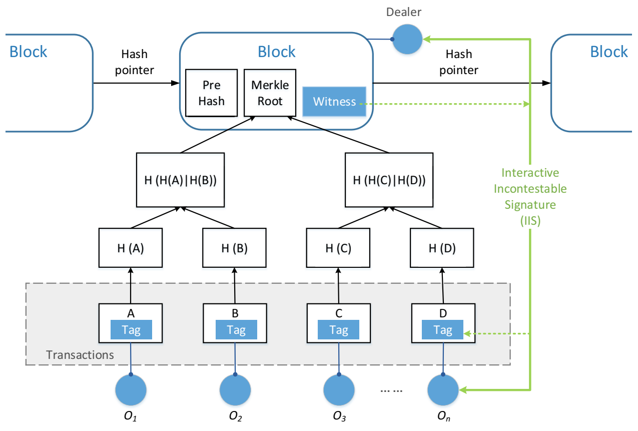

Fig. 1. Our System Model on Interactive Incontestable Signature (IIS).

In our system, we add a new part *Witness*, which stores identifiable information of the dealer, into each block. Once a transaction is confirmed, a proof of validation (*Tag*) will be generated and attached to the transaction. Any nodes in the network can validate this *Tag* with *Witness* published in the block. Transaction with a valid *Tag* indicates that it is already confirmed by the dealer, and it ensures the incontestability of dealers by establishing relationship between valid transactions handled by the dealer and identity of the dealer. We construct an Interactive Incontestable Signature (IIS) scheme to create such a relationship, and the formal definition of IIS is given as follows:

Definition 1: (Interactive Incontestable Signature, IIS): An Interactive Incontestable Signature (IIS) scheme consists of a tuple of algorithms (Setup, OKeyGen, DKeyGen, WitGen, Sign, Verify), as follows:

- **Setup**: This algorithm outputs a parameter as the master public key.
- **OKeyGen**: For each owner, it outputs a owner’s public-private key pair by himself, where the owner holds the secret key but the public key is public.
- **DKeyGen**: This algorithm outputs a dealer’s public-private key pair, and the manager appends the public key into the master public key.
- **WitGen**: This algorithm generates a witness of a secret number and outputs this witness.
- **Sign**: This is an interactive proof protocol for yielding signature between the dealer and the owner for a certain transaction, and they interactively generate a message-signature pair and outputs this signature.
- **Verify**: This algorithm outputs 1 if it is a valid message-signature pair, otherwise output 0.

Compared with the transitional signature, the above signature has several differences:

1) Either the owner or the dealer can generate the public/secret key by himself for easy to use;
2) the process of generating signature is an interactive proof process between two parties: dealer and owner;
3) the verification of signature requires two public keys of both dealer and owner at the same time, which means that this signature is permitted by both of them;
4) the witness is unique in each block and will be shared in all transactions in this block, that sets up a strong membership between all transactions and this block.

C. Security Requirements

In our system, there are three types of possible forgery attacks as follows: (1) forge signatures by owners, (2) forge signatures by dealers and (3) forge signatures by external attackers. Note that in our system, the dealer need to authenticate the owner of a certain transaction before they running IIS to generate a signature. Therefore the unforgeability of external attackers can be ensured by the process of authentication. We therefore focus our security model on the other two security requirements.

- **Unforgeability of Owner.** In this case, the owner of a transaction may aim at skipping validation process to forge a signature by himself/herself. This security requires that the owner cannot produce any valid signature even with unlimited computational resources.
- **Incontestability of Dealer.** This requirement guarantees that the dealer cannot deny his/her signature once he/she has generated this signature interactively with owners.

We expect that our signature is provably secure for the above requirements. Moreover, this signature must have a small impact on the original blockchain structure, as well as high performance and easy-to-implementation.

III. Our Construction

We set up our scheme using a bilinear map group system which is obtained from general bilinear pairings. A bilinear map group is a tuple $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, p, e)$ where $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of the same order p. We say that the function e is a computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ if for any $g, h \in \mathbb{G}, a, b \in \mathbb{Z}_q^*$, then $e(g^a, h^b) = e(g, h)^{ab}$. We now describe a concrete IIS scheme in Fig. 3, which is based on a bilinear map system $\mathbb{S} = (\mathbb{G}, \mathbb{G}_T, p, e)$. In this scheme, the algorithm of Sign including five steps is interactive proof process between dealer and owner, that is shown in Fig. 2.

Based on bilinear map, the correctness of signature σ_i generated interactively can be proved as follow:

$$\begin{aligned}
&e((pk_d)^{H(ID_i)}, pk_i) \cdot e(H(T_i), wit_2) \\
&= e((g^d)^{H(ID_i)}, g^{x_i}) \cdot e(H(T_i), h^{ad}) \\
&= e(g^{x_i d H(ID_i)}, h) \cdot e(H(T_i)^{ad}, h) \\
&= e(g^{x_i d H(ID_i)} \cdot H(T_i)^{ad}, h) = e(\sigma_i, h)
\end{aligned}$$

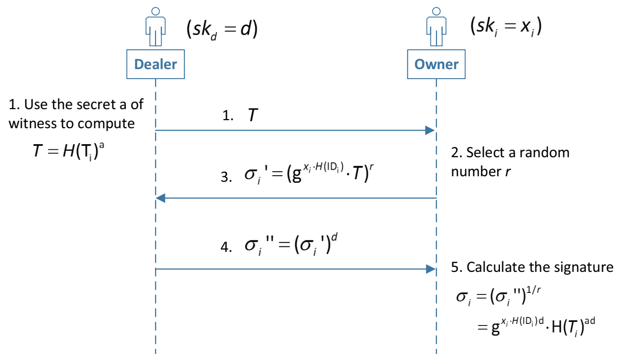
$(sk_d = d)$
Dealer
$(sk_i = x_i)$
Owner
1. Use the secret a of witness to compute
$T = H(T_i)^a$
1. T
2. Select a random number r
3. $\sigma_i' = (g^{x_i \cdot H(ID_i)} \cdot T)^r$
4. $\sigma_i'' = (\sigma_i')^d$
5. Calculate the signature
$\sigma_i = (\sigma_i'')^{1/r}$
$= g^{x_i \cdot H(ID_i) d} \cdot H(T_i)^{ad}$

Fig. 2. Signature Generating Protocol.

- **Setup**$(1^\kappa) \to mpk$: This algorithm first generates the bilinear group $\mathbb{G}, \mathbb{G}_T$ of prime order p. Let g be the generator of $\mathbb{G}$. It chooses a random $h \in \mathbb{G}$ and outputs $mpk = (g, h)$.
- **OKeyGen**$(mpk, u_i) \to (sk_i, pk_i)$: it selects a random element $x_i \in \mathbb{Z}_p^*$ as sk_i and calculates $pk_i = h^{x_i}$.
- **DKeyGen**$(mpk, dealer_d) \to (sk_i, pk_i)$: For a certain dealer, it selects a random element $d \in \mathbb{Z}_p^*$ as sk_d and calculates $pk_d = g^d$.
- **WitGen**$(mpk, sk_d) \to W$: For a certain dealer, it randomly picks a secret element $a \in \mathbb{Z}_p^*$. Then it calculates $wit_1 = g^a, wit_2 = (h^a)^{sk_d} = h^{ad} \in \mathbb{G}$ and outputs $W = (wit_1, wit_2)$.
- **Sign**$(D(a, sk_d) \leftrightarrow O(sk_i))(mpk, W) \to \sigma_i$: It takes as input the master public key mpk and witness W, secret a, sk_d possessed by the dealer and sk_i possessed by the owner. To sign a transaction T_i, dealer and this owner carry out a two-party protocol to calculate the signature, as follows:
  1) The dealer calculates the hash value $T = H(T_i)^a$ of transaction T_i together with his secret $a \in \mathbb{Z}_p^*$, and then transmits T to the owner.
  2) The owner randomly selects a number $r \in \mathbb{Z}_p^*$,
  3) The owner transmits $\sigma_i' = (g^{x_i H(ID_i)} \cdot T)^r = (g^{x_i H(ID_i)} \cdot H(T_i)^a)^r$ to the dealer, where ID_i is the ID of T_i.
  4) The dealer calculates $\sigma_i'' = (\sigma_i')^d$ with his private key sk_d and transmits $\sigma_i'' = (g^{x_i H(ID_i)} \cdot H(T_i)^a)^{rd}$ to the owner.
  5) Finally, the owner removes the random r and obtains the signature $\sigma_i = (\sigma_i'')^{1/r} = g^{x_i H(ID_i) d} \cdot H(T_i)^{ad}$. It outputs a signature σ_i for T_i, where
  $$\sigma_i = g^{x_i d H(ID_i)} \cdot H(T_i)^{ad}$$
- **Verify**$(mpk, pk_i, pk_d, W, (T_i, \sigma_i)) \to \{0, 1\}$: Given mpk, the witness W and a message-signature pair (T_i, σ_i), this algorithm checks the equation
  $$e(\sigma_i, h) = e((pk_d)^{H(ID_i)}, pk_i) \cdot e(H(T_i), wit_2)$$
  Output 1 if it is holds. Otherwise output 0.

Fig. 3. The full construction of Interactive Incontestable Signature (IIS).

A. Integrating with Blockchain

While the construction of the previous section gives an overview of our approach, we have yet to describe how our scheme integrates with blockchain. The general overview of our approach is straightforward. To initiate a transaction to Bob, Alice first constructs a transaction infor-

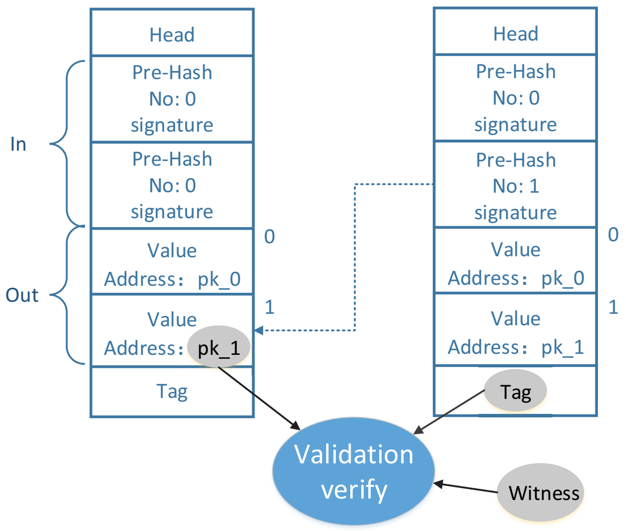

Fig. 4. How other nodes verify a certain transaction contained in block.

mation T_A. T_A contains pk_B (generated by Bob running OKeyGen(mpk, u_B)), transaction content and a simple signature produced by Alice (only used between Alice and dealers who generate the next block). Then she broadcasts T_A to the network.

To create a new block, the dealer runs WitGen(mpk, sk_d) and stores witness W in the block for validation. The dealer searches for all transactions collected in this accounting cycle and verifies transaction content. Here, we take T_A for example, the dealer first authenticates Alice's identity through its original signature and checks whether T_A is a valid transaction. If these conditions hold and the referenced transaction is not claimed as an input into a different transaction (avoid double-spending), the dealer and Alice then run interactively the Sign protocol to generate a new signature σ_A as a Tag, replacing the original one.

When this block has been built, other nodes in the network examine whether T_A is a valid transaction by running Verify$(mpk, pk_A, pk_d, W, (T_A, \sigma_A))$ algorithm. Fig. 4 illustrates this verification process in detail. They first get Tag attached to T_A, the $witness$ stored in the block and Alice's pk_A in the corresponding out field of the referenced transaction which can be found through the hash chain. If Verify$(mpk, pk_A, pk_d, W, (T_A, \sigma_A)) = 1$ and the referenced transaction is not claimed as an input into a different transaction, T_A is regarded as valid and they continue to examine other transactions in this block; otherwise, if verify algorithm return 0 or the referenced transaction has been claimed, T_A is regarded invalid and they discard this block. Finally, if all transactions pass this validation process, the network accepts this block as valid and appends it to the blockchain.

B. Security Analysis

We now analyze the security of our construction. Considering that the unforgeability of external attackers is ensured by the authentication process as we discussed above, we focus the attention on two types of security properties (unforgeability of owner and incontestability of dealer). We will analyze these two properties as follows:

1) Unforgeability of Owner: At first, we define the unforgeability of owner based on the general security definition of signature, as follows:

Definition 2: A signature scheme is $(t, q_O, q_D, q_H, \epsilon)$-secure against the unforgeability of owner if any adversary $\mathscr{A}_1$ breaks our scheme with a negligible probability ϵ, the advantage

$$Adv_{\mathscr{A}_1} = \Pr\left[\begin{array}{c} Verify(mpk, pk_i, pk_d, W, T_i^*, \sigma_i^*) = 1 : \\ Setup(1^\kappa) = mpk; \\ \{pk_i\} \leftarrow \mathscr{A}_1^{OKeyGen(mpk,u_i)=(sk_i,pk_i)}, \\ \{pk_d\} \leftarrow \mathscr{A}_1^{DKeyGen(mpk,dealer_j)=(sk_d,pk_d)}, \\ WitGen(mpk, sk_d) = W, \\ \mathscr{A}_1(\{pk_i\}, \{pk_d\}, W) = (T_i^*, \sigma_i^*) \end{array}\right] \leq \epsilon,$$

for t time, q_O and q_D times queries for dealer and owner, and q_H times queries for the hash Oracle.

The unforgeability of owner of our proposed scheme is based on extended Computational Bilinear Diffie-Hellman (called eCBDH$_1$) assumption, which is defined as follow:

Definition 3: (extended-CBDH$_1$ Assumption) *Given $G, H, G^a, H^a, H^b \in \mathbb{G}$ for unknown $a, b \in \mathbb{Z}_q^*$, the eCBDH$_1$ assumption states that it is computationally intractable to compute the value of G^{ab}.*

Next, we prove that our scheme is unforgeability of owner according to the following theorem:

Theorem 1: Let $\mathbb{G}$ be a (t', ϵ') group for Hiffie-Hellman of order p. Then the signature scheme on $\mathbb{G}$ is $(t, q_O, q_D, q_H, \epsilon)$-secure against the unforgeability of owner, where

$$t \leq t' - 2(\log p)(q_O + q_D + q_H), \quad \epsilon \geq \epsilon'$$

We assume an adversary $\mathcal{A}$ breaks our interactive signature scheme. We will use $\mathcal{A}$ to construct a simulator $\mathcal{B}$ that breaks two types of extended-Computational Bilinear Diffie-Hellman problems which we will define later in this section. The proof process is shown in Fig. 5.

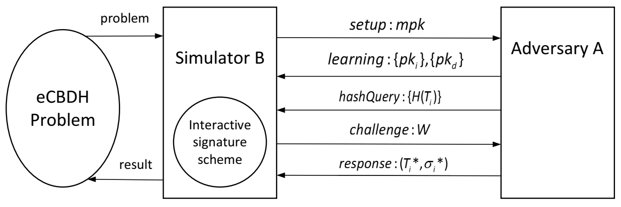

Fig. 5. The diagram of proof process in Theorem 1.

Proof: Suppose there exists an PPT adversary $\mathscr{A}_1$ that outputs a forged signature for the interactive signature scheme with a non-negligible advantage ϵ. We can use the algorithm $\mathscr{A}_1$ to construct a PPT algorithm $\mathscr{B}_1$ that can break the eCBDH$_1$ problem: for $x, y \in \mathbb{Z}_q^*$, given $G, H, G^x, H^x, H^y \in \mathbb{G}$, compute $G^{xy} \in \mathbb{G}$. Algorithm $\mathscr{B}_1$ is described as follows:

- **Setup**: Given an eCBDH$_1$ problem $(G, H, G^x, H^x, H^y \in \mathbb{G})$, the algorithm $\mathscr{B}_1$ runs the $Setup$ to generate the msk, and calculates $mpk = (g = G, h = H)$.

- **Learning**: $\mathscr{A}_1$ can issue up to q_O owner-key queries and q_D dealer-key queries. In response, $\mathscr{B}_1$ runs as follows:
  - Owner-key queries. Given an owner's index i, $\mathscr{B}_1$ chooses a random number $\lambda_i \in \mathbb{Z}_p^*$. Assuming that $sk_i = y\lambda_i$, the public key is computed as $pk_i = (H^y)^{\lambda_i}$. $\mathscr{B}_1$ sends pk_i to $\mathscr{A}_1$.
  - Dealer-key queries. Given an dealer's index j, $\mathscr{B}_1$ chooses a random number $\zeta_j \in \mathbb{Z}_p^*$. Assuming that $sk_d = x\zeta_j$, the public key is computed as $pk_d = (G^x)^{\zeta_j}$. $\mathscr{B}_1$ sends pk_d to $\mathscr{A}_1$.
- **Hash Query**: $\mathscr{A}_1$ can query a Hash Oracle up to q_H times as $H(T_i) = G^{h(T_i)}$, where there is a map: $\{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $h(T_i) \in \mathbb{Z}_p^*$.
- **Challenges**: $\mathscr{B}_1$ runs the $WitGen$ to generate $W = (wit_1, wit_2)$. It computes $wit_1 = g^a = G^a, wit_2 = h^{ad} = (H^x)^a$, and sends W to $\mathscr{A}_1$ as a challenge.
- **Response**: The adversary $\mathscr{A}_1$ calculates a message-signature pair (T_i^*, σ_i^*) in a polynomial time, and then sends it to $\mathscr{B}_1$.
- **Output**: $\mathscr{B}_1$ checks whether it is a valid pair by examining

$$e(\sigma_i^*, h) \overset{?}{=} e(pk_d^{H(ID_i^*)/\zeta_j}, pk_i^{1/\lambda_i}) \cdot e(H(T_i^*), wit_2).$$

  If this equation holds, this means that σ_i^* is a valid signature. And then $\mathscr{B}_1$ computes $G^{xy} = (\sigma_i^*/(G^x)^{h(T_i^*)a})^{1/H(ID_i^*)}$ and returns this value as the final result.

We now analyze the validate of the above construction as follows: The last equation used to check the validity of forged signature according to the equation:

$$\begin{aligned}
&e(pk_d^{H(ID_i^*)/\zeta_j}, pk_i^{1/\lambda_i}) \cdot e(H(T_i^*), wit_2)\\
&= e((G^x)^{\zeta_j H(ID_i^*)/\zeta_j}, (H^y)^{\lambda_i/\lambda_i}) \cdot e(G^{h(T_i^*)}, (H^x)^a)\\
&= e(G^{xyH(ID_i^*)+h(T_i^*)xa}, H).
\end{aligned}$$

Also, the above algorithm $\mathscr{B}_1$ is a probabilistic polynomial time (PPT) algorithm only if the adversary $\mathscr{A}_1$ can return the result within a polynomial time. This also means the PPT algorithm $\mathscr{B}_1$ can solve the eCBDH$_1$ problem with a non-negligible probability. This contradicts the hypothesis that the eCBDH$_1$ problem is hard for any PPT algorithm. That is, the advantage of any probabilistic polynomial time algorithm $\mathscr{B}_1$ in solving the the eCBDH$_1$ is negligibly small.

$$\begin{aligned}
Adv_{\mathscr{B}_1}^{eCBDH_1} &= \Pr[\mathscr{B}_1(G, H, G^x, H^x, H^y) = G^{xy}]\\
= \;&\Pr\left[\begin{array}{c} Verify(mpk, pk_i, pk_d, W, T_i^*, \sigma_i^*) = 1 : \\ \mathscr{A}_1(\{pk_i\}, \{pk_d\}, W) = (T_i^*, \sigma_i^*) \end{array}\right] \leq \epsilon.
\end{aligned}$$

The algorithm $\mathscr{B}_1$'s running time includes the running time of forgery. The additional overhead imposed by $\mathscr{B}_1$ is dominated by the need to evaluate group exponentiation for each signature, key request and hash request. Any one such such exponentiation may be computed by using at most $2\log p$ group action [5], and thus at most $2\log p$ time units on $\mathbb{G}$. $\mathscr{B}_1$ may need to answer as many as $q_O + q_D + q_H$ such requests, so its overall running time is $t' \leq t + 2(\log p)(q_O + q_D + q_H)$. ■

2) Incontestability of Dealer: Similarly to the above definition, we also define the incontestability of dealer as follows:

Definition 4: A signature scheme is $(t, q_O, q_D, q_H, \epsilon)$-secure against the incontestability of dealer if any adversary $\mathscr{A}_2$ breaks our scheme with a negligible probability ϵ, the advantage

$$Adv_{\mathscr{A}_2} = \Pr\left[\begin{array}{c} Verify(mpk, pk_i, pk_d, W, T_i^*, \sigma_i^*) = 1 : \\ Setup(1^\kappa) = mpk; \\ \{sk_i, pk_i\} \leftarrow \mathscr{A}_2^{OKeyGen(mpk, u_i) = (sk_i, pk_i)}, \\ \{pk_d\} \leftarrow \mathscr{A}_2^{DKeyGen(mpk, dealer_j) = (sk_d, pk_d)}, \\ WitGen(mpk, sk_d) = W, \\ \mathscr{A}_2(\{sk_i, pk_i\}, \{pk_d\}, W) = (T_i^*, \sigma_i^*) \end{array}\right] \leq \epsilon,$$

for t time, q_O and q_D times queries for dealer and owner, and q_H times queries for the hash Oracle.

Compared with Definition 2, it is easy to find a difference that the adversary $\mathscr{A}_2$ holds some owner's secret keys sk_i. This means that any owner cannot forge the signature of dealer. The incontestability of dealer of our proposed scheme is based on extended Computational Bilinear Diffie-Hellman 2(eCBDH$_2$) assumption, which is defined as follow:

Definition 5: (extended-CBDH$_2$ Assumption) *Given $G, H, G^a, G^b, H^{ab} \in \mathbb{G}$ for unknown $a, b \in \mathbb{Z}_q^*$, the eCBDH$_2$ assumption states that it is computationally intractable to compute the value of G^{ab}.*

Based on this assumption, we prove our scheme is incontestability of dealer, as follows:

Theorem 2: Let $\mathbb{G}$ be a (t', ϵ') group for Hiffie-Hellman of order p. Then the signature scheme on $\mathbb{G}$ is $(t, q_O, q_D, q_H, \epsilon)$-secure against the unforgeability of dealer, where

$$t \leq t' - 2(\log p)(q_O + q_D + q_H), \quad \epsilon \geq \epsilon'$$

Proof: Suppose there exists an PPT adversary $\mathscr{A}_2$ that outputs a forged signature for the above scheme with a non-negligible advantage ϵ. We can use the algorithm $\mathscr{A}_2$ to construct a PPT algorithm $\mathscr{B}_2$ that can break the CBDH$_2$ problem: for $x, y \in \mathbb{Z}_q^*$, given $G, H, G^x, G^y, H^{xy} \in \mathbb{G}$, compute $G^{xy} \in \mathbb{G}$. Algorithm $\mathscr{B}_2$ is described as follows:

- **Setup**: Given an eCBDH$_2$ problem $(G, H, G^x, G^y, H^{xy} \in \mathbb{G})$, the algorithm $\mathscr{B}_2$ runs the $Setup$ to generate the msk, and calculates $mpk = (g = G, h = H)$.
- **Learning**: $\mathscr{A}_2$ can issue up to q_O owner-key queries and q_D dealer-key queries. In response, $\mathscr{B}_2$ runs as follows:
  - Owner-key queries. Given an owner's index i, $\mathscr{B}_2$ runs the $OKeyGen$ to generate $(sk_i = x_i, pk_i = H^{x_i})$ for any a x_i, and sends (sk_i, pk_i) to $\mathscr{A}_2$.
  - Dealer-key queries. Given an dealer's index j, $\mathscr{B}_2$ chooses a random number $\zeta_j \in \mathbb{Z}_p^*$. Assuming that $sk_d = y\zeta_j$, the public key is computed as $pk_d = (G^y)^{\zeta_j}$. $\mathscr{B}_2$ sends pk_d to $\mathscr{A}_2$.
- **Hash Query**: $\mathscr{A}_2$ can query a Hash Oracle up to q_H times as $H(T_i) = G^{h(T_i)}$, where there is a map: $\{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $h(T_i) \in \mathbb{Z}_p^*$.
- **Challenges**: Assuming that $a = x$, $\mathscr{B}_2$ computes $wit_1 = g^a = G^x, wit_2 = h^{ad} = H^{xy}$. Let $W = (wit_1, wit_2)$ and sends W to $\mathscr{A}_2$ as a challenge.

- **Response**: The adversary $\mathscr{A}_2$ calculates a message-signature pair (T_i^*, σ_i^*) in a polynomial time, and then sends it to $\mathscr{B}_2$.
- **Output**: $\mathscr{B}_2$ checks whether it is a valid pair by examining

$$e(\sigma_i^*, h) \overset{?}{=} e(pk_d^{H(ID_i^*)/\zeta_j}, pk_i) \cdot e(H(T_i^*), wit_2)$$

This equation used to check the validity of forged signature holds as:

$$\begin{aligned} & e(pk_d^{H(ID_i^*)/\zeta_j}, pk_i) \cdot e(H(T_i^*), wit_2) \\ & = e((G^y)^{\zeta_j H(ID_i^*)/\zeta_j}, H^{x_i}) \cdot e(G^{h(T_i^*)}, H^{xy}) \\ & = e(G^{yH(ID_i^*)x_i + h(T_i^*)xy}, H) \end{aligned}$$

If σ_i^* is a valid signature, $\mathscr{B}_2$ computes $G^{xy} = (\sigma_i^*/(G^y)^{H(ID_i^*)x_i})^{1/h(T_i^*)}$ which means the PPT algorithm $\mathscr{B}_2$ can solve the eCBDH$_2$ problem with a non-negligible probability. This contradicts the hypothesis that the eCBDH$_2$ problem is hard for any PPT algorithm.

That is, the advantage of any probabilistic polynomial time algorithm $\mathscr{B}_2$ in solving the the eCBDH$_2$ is negligibly small.

$$\begin{aligned} & Adv_{\mathscr{B}_2}^{eCBDH_2} = \Pr[\mathscr{B}_2(G, H, G^x, G^y, H^{xy}) = G^{xy}] \\ & = \Pr\left[\begin{array}{c} Verify(mpk, pk_i, pk_d, W, T_i^*, \sigma_i^*) = 1 : \\ \mathscr{A}_2(\{sk_i, pk_i\}, \{pk_d\}, W) = (T_i^*, \sigma_i^*) \end{array}\right] \leq \epsilon. \end{aligned}$$

The algorithm $\mathscr{B}_2$'s running time is similar to the above analysis in unforgeability by owner and its overall running time is same to $t' \leq t + 2(\log p)(q_O + q_D + q_H)$. ∎

IV. Performance Evaluation

A. Performance Analysis

Our interactive signature scheme is constructed on bilinear map system from elliptic curve pairings. For simplification, we give several notations to denote the time for various operations in our signature scheme. $E(\mathbb{G})$ is used to denote the exponentiation in $\mathbb{G}$ and B to denote the pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We neglect the operations on $\mathbb{Z}_p^*$, the hash function $H : \{0,1\}^* \to \mathbb{G}$ and the multiplication in $\mathbb{G}$ and $\mathbb{G}_T$, since they are much efficient than exponentiation and pairing operation. We analyze the computation and communication complexity for each phase, where $l_{\mathbb{Z}_p^*}$, $l_{\mathbb{G}}$ denote the length of elements in $\mathbb{Z}_p^*$ and $\mathbb{G}$ respectively.

TABLE I
Complexity Analysis of our scheme

	Computation Complexity
OKeyGen	$1 \cdot E(\mathbb{G})$
DKeyGen	$1 \cdot E(\mathbb{G})$
WitGen	$3 \cdot E(\mathbb{G})$
SigGen	Dealer: $2 \cdot E(\mathbb{G})$ | Owner: $4 \cdot E(\mathbb{G})$
Verify	$1 \cdot E(\mathbb{G}) + 3 \cdot B$

In Tables I and II, we analyze the performance of our interactive signature scheme from two aspects: computation and communication/storage costs. Note that in Table I, there is no exponentiation and pairing operations in the *setup* phase, we thus do not list this algorithm here. In Table II, we use sk to denote sk_i and sk_d since both of them has the same element length. Similarly, we use pk to denote pk_i and pk_d.

TABLE II
Communication/Storage Analysis of our scheme

	Communication/Storage Complexity
Master public key (msk)	$2 \cdot l_{\mathbb{G}}$
Private key (sk)	$1 \cdot l_{\mathbb{Z}_p^*}$
Public key (pk)	$1 \cdot l_{\mathbb{G}}$
Witness (W)	$2 \cdot l_{\mathbb{G}}$
Signature (σ)	$1 \cdot l_{\mathbb{G}}$

B. Performance Evaluation

We report experimental results to demonstrate the performance of the interactive signature scheme. We build a simple demo program simulating the interactive process between two participants. This demo is implemented in Java and built upon the Java Pairing Based Cryptography Library (JPBC) cryptographic library. In table III the detail data are listed for the above experiments. We use type A elliptic curve parameter to generate bilinear pairing. In type A pairing, let l_q be the length of some prime q, l_r be the length of the order r, where r is some prime factor of $q + 1$. We imply five elliptic curves parameters in the experiments, in which each of l_q, l_r has different value: a_160 (l_q=512, l_r=160), a_200 (l_q=640, l_r=200), a_240 (l_q=768, l_r=240), a_280 (l_q=896, l_r=280) and a_320 (l_q=1024, l_r=320).

TABLE III
Computational Costs for different elliptic curve type

Type	Setup	OKeyGen	WitGen	SigGen	Verify
a_160	0.01324	0.02538	0.07569	0.15141	0.11066
a_200	0.02335	0.04598	0.13500	0.27503	0.20850
a_240	0.03725	0.07072	0.28572	0.47010	0.32450
a_280	0.06605	0.10510	0.32277	0.64518	0.48799
a_320	0.07756	0.14998	0.44662	0.91350	0.71197

V. Conclusion

In this paper, we present a new system model in blockchain for implementing "*Instant Confirmation with Incontestability*". As the core of our system, a signature scheme IIS is proposed and implemented to ensure transactions get dealer's confirmation incontestability. We integrate our scheme into blockchain, analyze the security of it and evaluate its performance.

Acknowledgment

This work is supported by the National 973 Program (Grand No. 2013CB329601) and the National Natural Science Foundation of China (Grant No. 61472032).

References

[1] Spencer Bogart and Kerry Rice. The blockchain report: Welcome to the internet of value, 2015.

[2] Ghassan O Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 906–917. ACM, 2012.

[3] Kaylash Chaudhary, Ansgar Fehnker, Jaco van de Pol, and Mariëlle Stoelinga. Modeling and verification of the bitcoin protocol. In *Proceedings Workshop on Models for Formal Analysis of Real Systems, MARS 2015, Suva, Fiji, November 23, 2015.*, pages 46–60, 2015.

[4] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. *CoRR*, abs/1510.02037, 2015.

[5] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology – ASIACRYPT 2001*, pages 514–532. Springer, 2001.