IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 November 2022, accepted 6 December 2022, date of publication 9 December 2022,
date of current version 14 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3228109

RESEARCH ARTICLE

Comprehensive Formal Modeling and Automatic Vulnerability Detection for a Bitcoin-Compatible Mixing Protocol

XIANGLIN BAO, XIAOFENG XU, (Member, IEEE), PING ZHANG, (Member, IEEE), AND TAO LIU
School of Computer and Information, Anhui Polytechnic University, Wuhu 241000, China

Corresponding author: Xiaofeng Xu (xuxiaofeng@ahpu.edu.cn)

This work was supported in part by the Provincial Natural Science Foundation of Anhui under Grant 2108085QF264 and Grant 2108085QF268, in part by the Advanced Research Project of Anhui Polytechnic University under Grant Xjky2020119 and Grant Xjky2020122, and in part by the Research Start-Up Fund for Talent Introduction of Anhui Polytechnic University under Grant 2020YQQ062.

ABSTRACT Blindcoin applies a Bitcoin-compatible mixing protocol with a blind signature scheme to improve the design of the popular Mixcoin. Given the openness of Bitcoin and the decentralization of the P2P network, it is imperative to formally analyze whether the malicious can break the security goals of the Blindcoin protocol. This work proposes a symbolic model for Blindcoin and conducts comprehensive formal verification. Fine-grained security goals of Blindcoin are formalized and subsequently encoded as model lemmas. However, it is challenging to verify the Blindcoin in a formal and automatic way. To tackle the challenges, we propose a tool-friendly symbolic model that can capture the semantics of multi-layers of Bitcoin and the features of Blindcoin. Our formal verification covers real-world security scenarios and discovers the Blindcoin vulnerabilities without human interaction. Furthermore, we offer several suggestions to fix the detected Blindcoin vulnerabilities and discuss the generalization of the proposed model.

INDEX TERMS Bitcoin, Blindcoin, formal analysis, protocol vulnerabilities, symbolic model.

I. INTRODUCTION

Bitcoin is one of the best-performing cryptocurrencies. The users trade with their pseudonymous addresses and manage huge assets autonomously [1]. However, the blockchain security [2] and the anonymity of Bitcoin users cannot be fully guaranteed [3], [4], [5]. After requesting the Bitcoin transaction list, deanonymization attacks can be launched based on the address analysis [6], [7], [8]. Mixcoin, as a landmark of the Bitcoin mixing service, hides the address links with the middle mix to prevent the deanonymization attack [9]. While the middle mix still has access to the address links and can deanonymize Bitcoin users [10]. To improve the design of Mixcoin, the Blindcoin protocol is proposed to hide the address links from the middle mix by applying a Bitcoin-compatible mixing protocol with a blind signature scheme [11].

Although Blindcoin improves the Mixcoin and enhances Bitcoin anonymity, it is imperative to analyze the Blindcoin security. The malicious can cause the irrecoverable loss of Bitcoin users who adopt mixing service [12]. Due to the openness of Bitcoin and the decentralized P2P network, the malicious can also pollute the network traffic and manipulate the protocol communication. Since no authority is responsible for the middle mix selection, the malicious can act as the middle mix to destroy protocol execution and steal Bitcoin [8]. Therefore, Bitcoin users require a efficient and convincing security analysis to detect vulnerabilities of the Bitcoin-compatible mixing protocols. Related works [6], [8], [9], [11], [12] have provided the informal security analysis of Bitcoin mixing service with manual reasoning and offered suggestions for users to mix their Bitcoin. Compared to

The associate editor coordinating the review of this manuscript and approving it for publication was SK Hafizul Islam.

TABLE 1. Related works of blockchain formalization.

Category	Ref.	Contribution
Bitcoin backbone	[14]	Formalizing and proving basic properties of Bitcoin core
	[15]	Proving wellformedness properties of Bitcoin Blockchain
	[16]	Formalizing Bitcoin and discovering the computational power stealing attack
Bitcion payment	[17]	Proving security goals for the Bitcoin payment protocol
Ethereum smart contracts	[18]	Proving security properties of smart contracts
	[19]	Controlling the complexity of formal analysis of smart contracts
	[20]	Formalizing Ethereum-based service smart contracts with Coq
	[21]	Formalizing smart contracts via ATL model checking
Cryptocurrency wallets	[22]	Formalizing and verifying the security properties of the hardware wallets
	[23]	Formalizing and performing an automatic analysis of the Bitcoin wallet

formal verification, informal security analysis needs human efforts but is hard to discover hidden vulnerabilities.

Few previous works have considered formal verification of protocol built for Bitcoin mixing services. It remains an open problem to formally model the Bitcoin-compatible mixing protocol and verify its security properties. Meanwhile, the automatic verifiable model can avoid error-prone reasoning and the advanced tool can construct the security proof of the symbolic model automatically [13]. In this paper, we address this gap by proposing an automatic verifiable model for Blindcoin in the symbolic settings.

A. RELATED WORKS

There are rigorous related works that concentrate on blockchain formalization as shown in Table 1. Garay et al. provided a formal model of the Bitcoin backbone [14]. Atzei et al. proposed a formal model for Bitcoin transactions [15]. Bao et al. formalized Bitcoin core protocols in the symbolic setting, and discovered the computational power stealing attack of Bitcoin stratum protocol with the machine-verified proofs [16]. Modesti et al. proved security goals for the Bitcoin payment protocol [17]. Hirai et al. [18], Amani et al. [19], Nam et al. [20], and Yang et al. [21] focused on the formal analysis of Ethereum smart contracts. Arapinis et al. [22] and Das et al. [23] provided the modelling and verification of cryptocurrency wallets.

With the symbolic settings and advanced verification tools, some researchers have formally verified hundreds of protocols [24], including major deployed protocols such as FIDO UAF protocol [25], TLS protocol [26], 5G authentication protocol [27], noise protocols [28], e-voting protocols [29], authentication and key agreement protocol, e-cash protocols [30], etc. They have proved the advantages of discovering protocol vulnerabilities with faithful symbolic models and advanced verification tools [13], [24], [31]. The revenue stealing attack of Bitcoin was discovered in our previous work [16], which proposed the symbolic model of Bitcoin core protocols. However, the advanced verification tools provide limited support for efficient reasoning when the symbolic model has a huge state size and advanced algebraic properties. Therefore, the pragmatic balance of the model comprehensiveness and complexity is required by the convincing and efficient formal verification.

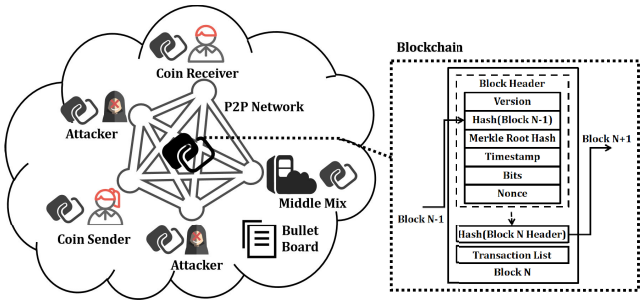
B. CHALLENGES AND CONTRIBUTIONS

This work proposes a tool-friendly symbolic model for the Bitcoin-compatible mixing protocol and provides machine-verified proof for the security of the Bitcoin mixing service. We introduce the formalization details of Blindcoin protocol and discuss the generalization of the proposed formal model.

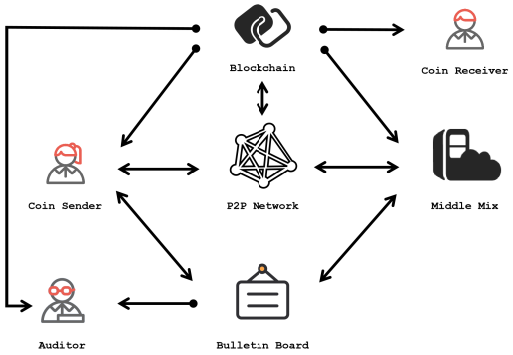
Blindcoin is built upon Bitcoin, and the modeling of Blindcoin requires the analysis of a huge amount of Bitcoin codes and documents. To achieve comprehensive formalization, the modeling requires considerable interpretation of codes and documents to cover the entity interactions of multiple Bitcoin layers and capture the semantics of structural transactions, growing blockchain and Blindcoin mixing service. Due to the implicit security settings of Blindcoin, we should consider all the possible scenarios with proper security assumptions, and fine-grained formalize the required security properties of mixing service, including the authentication, accountability, anonymity, and balance consistency. To capture the features of Bitcoin-compatible mixing protocols, we should model the sophisticated Blindcoin operation with the standard cryptographic primitives [32]. At the same time, the Bitcoin-compatible mixing protocols have plenty of entity interactions and complex state transitions. An automatic and efficient model reasoning of these protocols is outside the scope of state-of-the-art verification techniques and tools, let alone manual reasoning, as the number of interactions is too large to explore [27]. With a pragmatic balance of model complexity, we give special treats for cryptographic interactions, concurrent sessions, and dynamic blockchain to realize efficient reasoning. We faithfully verify Blindcoin's security and efficiently detect its vulnerabilities with machine-proofs.

To our knowledge, this paper is the first work that provides a faithful and comprehensive formal model for the Bitcoin mixing service. The proofs of the violated Blindcoin properties and the detected Blindcoin vulnerabilities show the comprehensiveness and efficiency of our modeling. The main contributions of this work are summarized as follows:

- We formalize all sorts of security properties of Bitcoin-compatible mixing services, including authentication, accountability, privacy, and balance consistency, while no previous work provides the fine-grained formalization.



(a) The architecture of Blindcoin protocol.



(b) Blindcoin entities and communications.

FIGURE 1. Blindcoin protocol architecture, entities and communications.

- We propose an efficient and reliable formalization that balances the model's complexity and comprehensiveness. Experimental results show that the proposed model is sufficient for automation capability and can verify the target lemmas in an efficient way.
- Model verification results disclose the previously found attacks and the newly detected vulnerabilities. To this end, we extract the real-world security scenarios of Blindcoin and explicitly capture the multi-layers semantics of Bitcoin and the features of Blindcoin.
- The proposed model definitions are easy to be expanded to other Bitcoin-compatible mixing protocols. Moreover, we use the Coinswap protocol as an example to introduce how to expand the proposed model.

C. ORGANIZATION

The rest of this paper is organized as follows. Section II introduces the overview of the Blindcoin protocol, Section III presents the formalization of security goals, Section IV presents the model restrictions and security assumptions, and Section V introduces formalization of state transition. Verification results are reported in Section VI. Finally, discussion is given in Section VII and conclusion is drawn in Section VIII.

II. OVERVIEW OF THE BLINDCOIN PROTOCOL

A. ARCHITECTURE

The Blindcoin protocol architecture, entities and communications are shown in Figure 1. The coin sender initializes

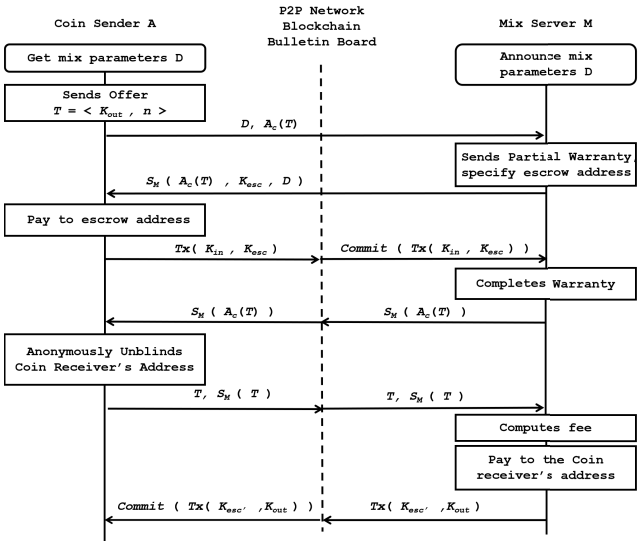


FIGURE 2. Protocol procedures of Blindcoin.

the Blindcoin protocol and sends the Blindcoin offer, pays to the escrow address of the middle mix, and unblinds the address of the coin receiver. The middle mix handles the Blindcoin offer, sends the Blindcoin partial warranty, completes the Blindcoin warranty, and transfers Bitcoin to the address of the coin receiver. The coin receiver receives the Bitcoin that is transferred anonymously if the protocol executes successfully. P2P Network relays the Blindcoin traffic, Bitcoin blockchain records the Bitcoin transaction, and the bulletin board publishes the Blindcoin partial warranty and warranty. Due to the openness of Bitcoin, any protocol entity of Blindcoin can be the attacker.

B. PROTOCOL PROCEDURES

Figure 2 illustrates the Blindcoin protocol procedures. Blindcoin protocol execution requires the public log supported by the P2P network, Blockchain, and the bulletin board. In protocol initialization, the Blindcoin middle mix announces mix parameter $\mathtt{D}$, and the coin sender gets $\mathtt{D}$ from the bulletin board. Then, the coin sender generates its Blindcoin offer $\mathtt{T}=<\mathtt{K}_{out},\mathtt{n}>$, where $\mathtt{K}_{out}$ is the address of the coin receiver and $\mathtt{n}$ is the user parameter to decide the randomized mix fees. Upon the Blindcoin offer is generated, the coin sender communicates with the middle mix with the parameter $\mathtt{D}$ and the blinded address $\mathtt{A}_C(\mathtt{T})$ to the middle mix, where $\mathtt{A}_C$ is the blinding function. The unblinding function is $\mathtt{A}_{C'}$, the equation $\mathtt{A}_{C'}(\mathtt{S}(\mathtt{A}_C(\mathtt{x})))=\mathtt{S}(\mathtt{x})$ is satisfied, and $\mathtt{S}(\mathtt{x})$ is the signature of message $\mathtt{x}$. After receiving the mix parameter $\mathtt{D}$ and the blinded address $\mathtt{A}_C(\mathtt{T})$, the middle mix returns a partial warranty $\mathtt{S}_M(\mathtt{A}_C(\mathtt{T}),\mathtt{K}_{esc},\mathtt{D})$, where $\mathtt{K}_{esc}$ is an escrow address for the middle mix to receive payment and $\mathtt{S}_M$ is the mix's signature.

To pay to the escrow address $\mathtt{K}_{esc}$, the coin sender publish transaction $\mathtt{Tx}(\mathtt{K}_{in},\mathtt{K}_{esc})$ to transfer Bitcoin from the coin sender's address $\mathtt{K}_{in}$ to $\mathtt{K}_{esc}$. Then, the transaction is

verified by P2P network, the Bitcoin blockchain confirms the transaction $\mathtt{Tx}(\mathtt{K}_{in}, \mathtt{K}_{esc})$ and the Bitcoin balance of the two address is updated. After the escrow address $\mathtt{K}_{esc}$ receives Bitcoin, the middle mix computes the warranty $\mathtt{S}_M(\mathtt{A}_C(\mathtt{T}))$ and posts it to the bulletin board. Upon the bulletin board publishes the warranty to the public, the coin sender unblinds the warranty with the unblinding function $\mathtt{A}_{C'}$ to get the mix's signature $\mathtt{S}_M(\mathtt{T})$, which is posted to the bulletin board along with the Blindcoin offer $\mathtt{T}=<\mathtt{K}_{out}, \mathtt{n}>$. After the bulletin board publishes the Blindcoin offer, the middle mix transfers Bitcoin from its escrow address $\mathtt{K}_{esc'}$ to the unblinded address $\mathtt{K}_{out}$, namely Bitcoin transaction $\mathtt{Tx}(\mathtt{K}_{esc'}, \mathtt{K}_{out})$. Upon Bitcoin transaction is published, the P2P network verifies the transaction, the blockchain confirms this transaction $\mathtt{Tx}(\mathtt{K}_{esc'}, \mathtt{K}_{out})$, and the Bitcoin balance of the two address is updated.

III. ASSUMPTIONS AND THREAT MODEL

We use the Dolev-Yao adversary as our basic threat model, in which the Blindcoin attackers have the ability to read, modify, intercept, and inject any message communicated in the Bitcoin network. And in the specified security scenarios, we need to endow more abilities to Blindcoin attackers or remove some abilities from Blindcoin attackers. The assumptions of Blindcoin's adversary abilities are stated below.

A. ASSUMPTIONS ON CRYPTOGRAPHIC PRIMITIVES

Corresponding with the Dolev-Yao adversary, we assume that only the one with the private key can recover the encrypted message and create the corresponding signature. The attacker cannot destroy the security of the cryptographic primitives.

B. ASSUMPTIONS ON ENTITIES AND CHANNELS

1) HONEST P2P NETWORK

If the honest blockchain nodes are the majority, it means the P2P network is honest. In the scenario of the honest P2P network, we assume that the attacker cannot modify, intercept, and inject the message sent by the P2P network. It means we need to remove the corresponding ability of Blindcoin attackers by defining the authentication channel for the P2P network. The model rules to define the authentication channel are stated in Section V.C. The attacker cannot initiate or respond to the communication as the role of the P2P network. Since blockchain traffic is open-access, we assume that the attacker can eavesdrop on the communication between the coin sender, the P2P network, and the middle mix. The honest P2P network is assumed to guarantee the growth of blockchain and only accept the transaction that transfers the unspent Bitcoin. It means we need to remove the corresponding ability of Blindcoin attackers by defining the restrictions for blockchain growth and transaction confirmation. The related model restrictions are stated in Section III.D.

2) DISHONEST P2P NETWORK

If the dishonest blockchain nodes are the majority, it means the P2P network is dishonest. In the scenario of the dishonest network, we assume that the attacker not only can intercept and manipulate the message sent by the P2P network but also can initiate and respond to the communication as the role of the P2P network. The attacker can eavesdrop on the communication between the coin sender, the P2P network, and the middle mix. If the blockchain fork appears, the dishonest P2P network allows the chain length to decrease and the transaction that transfers the spent Bitcoin. It means we remove the model restrictions of blockchain growth and transaction confirmation stated in Section III.D.

3) LAZY MIDDLE MIX

If the middle mix does not check the uniqueness of the escrow address, it means the middle mix is lazy. In the scenario of the lazy middle mix, we assume that the lazy middle mix stores its private key securely. The attacker cannot modify, intercept, or inject the message sent by the lazy middle mix. So we need to remove the corresponding ability of Blindcoin attacker to control the message of the lazy middle mix by applying the authentication channel stated in Section V.C. The attacker can eavesdrop on the communication between the P2P network and the lazy middle mix. We assume that the lazy middle mix stores its private key securely. The attacker cannot initiate or respond to the communication as the role of the lazy middle mix. It means the Blindcoin attacker cannot get the private key of the lazy middle mix and we apply the restriction on the lazy middle mix's private key. We assume the lazy middle mix allows the same escrow address to receive the Bitcoin. It means we remove the restriction on the escrow address uniqueness and the attacker can replay the same escrow address to the lazy middle mix.

4) UNLAZY MIDDLE MIX

If the middle mix checks the uniqueness of the escrow address, it means the middle mix is unlazy. In the scenario of the unlazy middle mix, we assume that the unlazy middle mix stores its private key securely. The attacker cannot intercept and manipulate the message sent by the unlazy middle mix. So we also need to remove the corresponding ability of Blindcoin attacker to control the message of the unlazy middle mix by applying the authentication channel stated in Section V.C. The attacker can eavesdrop on the communication between the P2P network and the unlazy middle mix. We assume that the unlazy middle mix stores its private key securely. The attacker cannot initiate or respond to the communication as the role of the unlazy middle mix. It means the Blindcoin attacker cannot get the private key of the unlazy middle mix and we apply the restriction on the private key secrecy of the unlazy middle mix. We assume the unlazy middle mix does not allow the same escrow address to receive the Bitcoin. It means we apply the restriction on the

escrow address uniqueness and the attacker cannot replay the same escrow address to the lazy middle mix.

5) LAZY COIN SENDER

If the coin sender does not guarantee the uniqueness of its account address that transfers different Bitcoin, it means the coin sender is lazy. In the scenario of the lazy coin sender, we assume that the lazy coin sender stores its private key securely. The attacker cannot intercept and manipulate the message sent by the lazy coin sender. So we apply the authentication channel stated in Section V.C. to remove the corresponding ability of Blindcoin attacker to control the message of lazy coin sender. The attacker can eavesdrop on the communication between the P2P network and the lazy coin sender. The attacker cannot initiate or respond to the communication as the role of the lazy coin sender. It means the Blindcoin attacker cannot get the private key of the lazy coin sender and we apply the restriction on the private key secrecy of the lazy coin sender. We assume the lazy coin sender allows the same Bitcoin account address to transfer different Bitcoin. It means we remove the restriction on the Bitcoin account address uniqueness and the attacker can replay the same escrow address to the lazy coin sender.

6) UNLAZY COIN SENDER

If the coin sender guarantees the uniqueness of its account address that transfers different Bitcoin, it means the coin sender is unlazy. In the scenario of the unlazy coin sender, we assume that the unlazy coin sender stores its private key securely. The attacker cannot intercept and manipulate the message sent by the unlazy coin sender. So we also apply the authentication channel stated in Section V.C. to remove the corresponding ability of Blindcoin attacker to control the message of unlazy coin sender. The attacker can eavesdrop on the communication between the P2P network and the unlazy coin sender. The attacker cannot initiate or respond to the communication as the role of the unlazy coin sender. It means the Blindcoin attacker cannot get the private key of the unlazy coin sender and we apply the restriction on the private key privacy of the unlazy coin sender. We assume the unlazy coin sender does not allow the same Bitcoin account address to transfer different Bitcoin. So we apply the restriction on the Bitcoin account address uniqueness and the attacker cannot replay the same escrow address to the lazy coin sender.

7) MALICIOUS PARTICIPANT

If the attacker plays one of the roles of the Blindcoin entities, the entity that the attacker acts as is malicious, which can be the malicious coin sender and the malicious middle mix. In the scenario of the malicious coin sender, we assume that the malicious coin sender reveals its private key to the attacker. It means the Blindcoin attacker can get the private key of the malicious coin sender. Therefore, in the protocol initialization, we define the model rule to reveal the private key of the coin sender and we remove the restriction on the private key secrecy of the coin sender. In the scenario of the malicious middle mix, we assume that the malicious middle mix reveals its private key to the attacker. It means the Blindcoin attacker can get the private key of the malicious middle mix. So, in the protocol initialization, we define the model rule to reveal the private key of the middle mix and we remove the restriction on the private key secrecy of the middle mix. The attacker can intercept and manipulate the message sent by the malicious participant. The attacker can eavesdrop on the communication of the malicious participant. The attacker can initiate or respond to the communication as the role of the malicious participant.

C. ASSUMPTIONS ON DATA PROTECTIONS

In our modeling, the attacker has the knowledge of the data fields of blockchain identification, the current block height, the previous transaction id of the Bitcoin, the input script with signature to unlock the previous transaction, the current transaction id to transfer the unlocked Bitcoin, the output script with the public key to lock current transaction, the partial warranty, and the completed warranty. We assume the data fields of the mix parameters and the required block periods of different protocol instances as the same that the attacker cannot distinguish. We verify if the security properties of Blindcoin hold when the relationship between the coin sender and the blinded address is compromised or leaked. We assume the adapted randomness to generate the blinded address is not guessable and the attacker has no knowledge of the adapted randomness. The attacker can monitor the activities of the coin sender and cannot guess the unused account address of the coin sender. The attacker can collect the published escrow address of the middle mix and cannot guess the future escrow address to be used. We assume the private key of the honest participants is secret when protocol starts.

D. MODEL RESTRICTIONS

We define and apply model restrictions for different Blindcoin security scenarios and blockchain semantics. The restrictions are introduced below, and the term specified in the restrictions are introduced in Section V.

The asynchronous nature of Bitcoin transaction confirmation results in ambiguous blockchain updating [33] and blockchain forking [34]. To model the scenario of the honest P2P network, restriction `NoDoubleSpending` is specified to prevent the transaction that transfers the spent Bitcoin. It ensures the transaction that transfers bitcoin from `x` to `a`, and the transaction transferring bitcoin from `x` to `b` cannot be both confirmed and recorded. Restriction `OneChain` is specified to eliminate cross-chain transactions and only one blockchain is initialized. It can be removed if we expand our model to verify the mixing protocol provides cross-chain service. Restriction `LedgerHeight` is specified to limit the longest chain length and prevent the unstopping of the model reasoning. Restriction `LowerThan` is specified to guarantee the growth of the Bitcoin blockchain. It ensures the tail block of the blockchain has a lower block height than the newly generated block. Restriction `Equality` and

InEquality are specified for the transaction unlocking to check the signature validity check.

To model the security scenarios of the unlazy middle mix restriction UniqueX is applied and term x in action Unique(x) represent the escrow address used by the middle mix. It ensures the escrow address only appears in one mixing offer during protocol execution. To model the security scenarios of the unlazy coin sender, restriction UniqueX is applied and term x in action Unique(x) represent the account address used by the coin sender. It ensures the account address of coin sender appears only once in Bitcoin transaction input script during protocol execution. For the security scenarios of the honest Blindcoin entity, the restriction No_keyReveal is specified to limit the private key of the honest entity to be secure and unknown to the attacker. Moreover, to model the scenario of the honest coin sender, restriction Spendable is specified to limit that a Bitcoin is locked before it becomes spendable. It ensures that the coin sender cannot lock a spendable Bitcoin in the previous transaction again.

At the same time, the defined model restrictions can not only fit each considered security scenario but also reduce the space size of the model state to exploit Bitcoin blockchain semantics and get the security proofs. The open-access blockchain records enlarge the state space size of the Blindcoin model and increase the difficulty of trace guarding to achieve model verification. These restrictions restrict the considered transition traces in the protocol analysis and reduced the space size of trace exploration to balance the model complexity and the reasoning efficiency of Blindcoin security verification.

```
restriction NoDoubleSpending:
"All BC x n m t1 t2 #i #j.
   Spend(BC,x,n,t1)@i & Spend(BC,x,m,t2)@j ==>#i=#j "

restriction OneChain:
"All #i #j.Initchain('1')@i & Initchain('1')@j ==>#i=#j "

restriction LedgerHeight:
"not Ex  BC height #i.
   ChainHeight(BC,'1'+'1'+'1'+'1'+'1'+height)@i"

restriction LowerThan:
"All BC t1 t2 #i. LowerThan(BC,t1,t2)@i ==>Ex x .t2=t1+x "

restriction Spendable:
  "All BC x n t1 #i #j.
   ((Unspent(x) @i) & (Spend(BC,x,n,t1) @j)) ==> (#i<#j)"
restriction Equality:
"All x y #i.Eq(x,y) @i ==> x=y "

restriction InEquality:
"All x #i. InEq(x, x)@i ==> F "

restriction No_keyReveal:
 "All #i A ltkA .!PK(A,pk(ltkA))@i
  ==>not Ex #j .!KU(ltkA)@j "

restriction UniqueX:
  "All x #i #j. ((Unique(x) @i) & (Unique(x) @j))
   ==> (#i=#j)"
```

IV. FORMALIZATION OF SECURITY GOALS

In this section, we fine-grained formalize the security goals of Bitcoin mixing services and provide the definitions of authentication property, accountability property, anonymity property, and balance consistency property required by the Blindcoin protocol. There is a formal relationship between the definitions of these security properties and their mathematical definitions will be encoded as our model lemmas [27].

Here, we present our definition of Blindcoin security properties along with the corresponding mathematical definitions of Blindcoin security properties. The encoded model lemmas are presented in Section IV.E. The fact term meanings in our mathematical formulas are introduced in Section V. The symbol # in our mathematical formulas denotes the temporal variables. The symbol @ in our mathematical formulas is the sort prefix for the temporal variables and works for the action constraints.

A. AUTHENTICATION PROPERTY

Blindcoin makes informal claims about authentication properties in its paper. This work formally defines the Blindcoin authentication based on Lowe's taxonomy of authentication properties [35] to clarify the ambiguity of Blindcoin authentication and cover the security requirements of data exchange in the Blindcoin protocol. To achieve the fine-grained formalization, we specify two levels of Blindcoin authentication: *Non-injective Agreement* and *Injective agreement*.

Definition 1: Given the protocol instance with the middle mix, entity a with role A and entity b with role B, if the Blindcoin protocol ensures **Non-injective Agreement**, we have that if, whenever data exchange initiator A completes the exchange of data ts apparently with data exchange responder B, then B has previously been running the protocol apparently with A, and the two agents agreed on the data values corresponding to all the variables in ts.

1) NON-INJECTIVE AGREEMENT ON BLINDCOIN OFFER BETWEEN THE COIN SENDER AND THE MIDDLE MIX

First, the coin sender must have the assurance that whenever it completes the data exchange of Blindcoin offer apparently with the middle mix, then the middle mix has previously been running the protocol apparently with the coin sender, and they agreed on the data values of Blindcoin offer. The following formula is specified for the *Non-injective Agreement* on Blindcoin offer between coin sender and the middle mix, where a and b are the Blindcoin entity names of the role coin sender 'CS' and the role middle mix 'Mix', i and j denote the appearance time of claim actions, and ts represents the terms of Blindcoin offer.

```
∀ a b ts #i.
   Claim_commit_agr(a,b,<'CS','Mix',<'Offer',ts>>)@i
⟶(∃ #j.
   Claim_running_agr(b,a,<'CS','Mix',<'Offer',ts>>)@j)
```

2) NON-INJECTIVE AGREEMENT ON BLINDCOIN ESCROW ADDRESS BETWEEN THE MIDDLE MIX AND COIN SENDER

The middle mix must have the assurance that whenever it completes the data exchange of Blindcoin escrow address apparently with the coin sender, then the coin sender has previously been running the protocol apparently with the middle mix, and they agreed on the data values of Blindcoin escrow address. The following formula is specified for the *Non-injective Agreement* on Blindcoin escrow address between the middle mix and coin sender, where a and b are the Blindcoin entity names of the role middle mix 'Mix' and the role coin sender 'CS', i and j and i2 denote the appearance time of claim actions, and ts represents the terms of Blindcoin escrow address.

```
∀ a b ts #i.
  Claim_commit_agr(a,b,<'Mix','CS',<'Esc',ts>>)@i
⟶ (∃ #j.
  Claim_running_agr(b,a,<'Mix','CS',<'Esc',ts>>)@j)
```

3) NON-INJECTIVE AGREEMENT ON BLINDCOIN DESTINATION ADDRESS BETWEEN THE COIN SENDER AND THE MIDDLE MIX

The coin sender must have the assurance that whenever it completes the data exchange of Blindcoin destination address apparently with the middle mix, then the middle mix has previously been running the protocol apparently with the coin sender, and they agreed on the data values of Blindcoin destination address. The following formula is specified for the *Non-injective Agreement* on Blindcoin destination address between the coin sender and the middle mix, where a and b are the Blindcoin entity names of the role coin sender 'CS' and the role middle mix 'Mix', i and j denote the appearance time of claim actions, and ts represents the terms of Blindcoin destination.

```
∀ a b ts #i.
  Claim_commit_agr(a,b,<'CS','Mix',<'Dst',ts>>)@i
⟶ (∃ #j.
  Claim_running_agr(b,a,<'CS','Mix',<'Dst',ts>>)@j)
```

Definition 2: If Blindcoin ensures **Injective Agreement** for any protocol instance with the middle mix to blind Bitcoin transaction, any entity a with role A and any entity b with role B, we have that if, whenever data exchange initiator A completes the exchange of data ts apparently with data exchange responder B, then B has previously been running the protocol apparently with A, and the two agents agreed on the data values corresponding to all the variables in ts, and for each data exchange of ts initialized by A, there is a unique matching data exchange of B.

4) INJECTIVE AGREEMENT ON BLINDCOIN OFFER BETWEEN THE COIN SENDER AND THE MIDDLE MIX

First, the coin sender must have the assurance that whenever it completes the data exchange of Blindcoin offer apparently with the middle mix, then the middle mix has previously been running the protocol apparently with the coin sender, and they agreed on the data values of Blindcoin offer, and there is a unique matching data exchange of the middle mix for each data exchange of Blindcoin offer initialized by the coin sender. The following formula is specified for the *Injective Agreement* of Blindcoin offer between coin sender and the middle mix, where a and b are the Blindcoin entity names of the role coin sender 'CS' and the role middle mix 'Mix', i and j and i2 denote the appearance time of claim actions, and ts represents the terms of Blindcoin offer.

```
∀ a b ts #i.
  Claim_commit_agr(a,b,<'CS','Mix',<'Offer',ts>>)@i
⟶ (∃ #j.
  Claim_running_agr(b,a,<'CS','Mix',<'Offer',ts>>)@j
  & j < i
  & ¬ (∃ a2 b2 #i2.
  Claim_commit_agr(a2,b2,<'CS','Mix',<'Offer',ts>>)@i2
  & ¬ (#i2 = #i)))
```

5) INJECTIVE AGREEMENT ON BLINDCOIN ESCROW ADDRESS BETWEEN THE MIDDLE MIX AND COIN SENDER

The middle mix must have the assurance that whenever it completes the data exchange of Blindcoin escrow address apparently with the coin sender, then the coin sender has previously been running the protocol apparently with the middle mix, and they agreed on the data values of Blindcoin escrow address, and there is a unique matching data exchange of the coin sender for each data exchange of Blindcoin escrow address initialized by the middle mix. The following formula is specified for the *Injective Agreement* of Blindcoin escrow address between the middle mix and coin sender, where a and b are the Blindcoin entity names of the role middle mix 'Mix' and the role coin sender 'CS', i and j and i2 denote the appearance time of claim actions, and ts represents the terms of Blindcoin escrow address.

```
∀ a b ts #i.
  Claim_commit_agr(a,b,<'Mix','CS',<'Esc',ts>>)@i
⟶ (∃ #j.
  Claim_running_agr(b,a,<'Mix','CS',<'Esc',ts>>)@j
  & j < i
  & ¬ (∃ a2 b2 #i2.
  Claim_commit_agr(a2,b2,<'Mix','CS',<'Esc',ts>>)@i2
  & ¬ (#i2 = #i)))
```

6) INJECTIVE AGREEMENT ON BLINDCOIN DESTINATION ADDRESS BETWEEN THE COIN SENDER AND THE MIDDLE MIX

The coin sender must have the assurance that whenever it completes the data exchange of Blindcoin destination address apparently with the middle mix, then the middle mix has previously been running the protocol apparently with the coin sender, and they agreed on the data values of Blindcoin destination address, and there is a unique matching data exchange of the middle mix for each data exchange of Blindcoin destination address initialized by the coin sender. The following formula is specified for the *Injective Agreement* of Blindcoin destination address between the middle mix and coin sender, where a and b are the Blindcoin entity names of the role

coin sender 'CS' and the role middle mix 'Mix', i and j and i2 denote the appearance time of claim actions, and ts represents the terms of Blindcoin destination.

```
∀ a b ts #i.
  Claim_commit_agr(a,b,<'CS','Mix',<'Dst',ts>>)@i
⟶ (∃ #j.
  Claim_running_agr(b,a,<'CS','Mix',<'Dst',ts>>)@j
  & j < i
  & ¬ (∃ a2 b2 #i2.
  Claim_commit_agr(a2,b2,<'CS','Mix',<'Dst',ts>>)@i2
  & ¬ (#i2 = #i)))
```

B. ACCOUNTABILITY PROPERTY

Blindcoin provides the informal claims of accountability in its paper. Based on Blindcoin's judgment of whether the entity should be blamed, we define *Judging Procedures* for Blindcoin protocol. The defined *Judging Procedures* should rely only on the open-access and tamper-resistant data on the bulletin board so that any mixing entity and the external auditor can run the procedures. To formally verify whether Blindcoin can identify the malicious entity individually, we define the *individual accountability* motivated by [36].

Definition 3: The **Judging Procedures** of Blindcoin are defined as the following procedures, where A represents the coin sender, B represents the coin receiver, and M represents the middle mix.

(1) If the account value of the first escrow address x does not increase when expected, then the coin sender A is judged as malicious because A does not claim the transaction to pay for the middle mix M who uses the first escrow address to receive payment.

(2) If the destination cannot be blinded by the coin sender A when expected, then A publishes the partial warranty and the middle mix M is judged as malicious because M refuses to sign and claim the correct warranty.

(3) If the destination cannot be paid by the middle mix M when expected due to the invalid warranty, the coin sender A is judged as malicious because A refuses to unblind and claim the correct destination for M.

(4) If the value of the coin receiver account, which is the destination, does not increase when expected, then the valid warranty claimed by M can be logged on the bulletin board and the middle mix M is judged as malicious because M refuses to claim the payment to the destination.

Definition 4: If the Blindcoin protocol ensures **Individual Accountability** for any Blindcoin protocol instance, we have that at least one malicious mixing entity can be blamed correctly and individually when protocol failure occurs, and the malicious entity is identified based on the **Judging Procedures**.

1) INDIVIDUAL ACCOUNTABILITY OF THE MALICIOUS COIN SENDER WHO DOES NOT CLAIM THE TRANSACTION TO PAY FOR THE MIDDLE MIX

The malicious coin sender who does not claim the transaction to pay for the middle mix can be identified and no wrong identification when protocol failure occurs. The following formula is specified for the *individual accountability* of malicious coin sender A who sends the Blindcoin offer with blinded coin destination b and refuses to make payment to the middle mix M from the coin sender's address *s* to the middle mix's escrow address pw.

```
∀ A s r M b x pw #t1 #i #t2 #j.
 ( Secret(A,s,r)@t1 & Claim_Offer(A,M,b)@i
 & UseEsc1(M,x,b,pw)@t2 & Claim_PWarranty(A,M,pw)@j
 & ¬(∃ #t4. Inc(s,x)@t4))
 ⟶ ¬(∃ #k. Claim_PayEsc(A,M,pw)@k)
```

2) INDIVIDUAL ACCOUNTABILITY OF THE MALICIOUS MIDDLE MIX WHO REFUSES TO SIGN AND CLAIM THE CORRECT WARRANTY

The malicious middle mix who refuses to sign and claim the correct warranty can be identified and no wrong identification when protocol failure occurs. The following formula is specified for the *individual accountability* of malicious middle mix M who received the payment from the coin sender A and refuses to sign the warranty w.

```
∀ A s r M b x pw #t1 #i #t2 #j #t4.
 ( Secret(A,s,r)@t1 & Claim_Offer(A,M,b)@i
 & UseEsc1(M,x,b,pw)@t2 & Claim_PWarranty(A,M,pw)@j
 & Inc(s,x)@t4
 & ¬(∃ A2 w #t5 #q. UnblindDst(A2,w,r)@t5
       & Claim_Dst(A2,M,r)@q))
⟶ (∃ #h1. Reveal_PWarranty(M,pw)@h1)
    & ¬(∃ w #s #p. SignWarranty(A,M,b,pw,w)@s
          & Claim_Warranty(A,M,b,w)@p)
```

3) INDIVIDUAL ACCOUNTABILITY OF THE MALICIOUS COIN SENDER WHO REFUSES TO UNBLIND AND CLAIM THE CORRECT DESTINATION

The malicious coin sender who refuses to unblind and claim the correct destination can be identified and no wrong identification when protocol failure occurs. The following formula is specified for the *individual accountability* of malicious coin sender A who get the warranty w from the middle mix M and refuses to unblind the coin destination r.

```
∀ A s r M b x pw #t1 #i #t2 #j #t4.
 ( Secret(A,s,r)@t1 & Claim_Offer(A,M,b)@i
 & UseEsc1(M,x,b,pw)@t2 & Claim_PWarranty(A,M,pw)@j
 & Inc(s,x)@t4
 & ¬(∃ #t. Claim_PayDst(A,M,r)@t))
⟶ ¬(∃ A2 w #t5 #q. UnblindDst(A2,w,r)@t5
       & Claim_Dst(A2,M,r)@q)
```

4) INDIVIDUAL ACCOUNTABILITY OF THE MALICIOUS MIDDLE MIX WHO REFUSES TO CLAIM THE PAYMENT TO THE DESTINATION

The malicious middle mix who refuses to claim the payment to the destination can be identified and no wrong identification when protocol failure occurs. The following formula is specified for the *individual accountability* of malicious middle mix M who gets the unblinded destination r from the

coin sender A and refuses to make a payment to the coin destination r.

```
∀ A s r M b x pw w A2 #t1 #i #t2 #j #t4 #p #t5 #q.
 ( Secret(A,s,r)@t1 & Claim_Offer(A,M,b)@i
 & UseEsc1(M,x,b,pw)@t2 & Claim_PWarranty(A,M,pw)@j
 & Inc(s,x)@t4
 & Claim_Warranty(A,M,b,w)@p
 & UnblindDst(A2,w,r)@t5 & Claim_Dst(A2,M,r)@q
 & ¬(∃ y #t7. Inc(y,r)@t7)
⟶ (∃ #h2. Log_Warranty(M,w)@h2)
    & ¬(∃ #t. Claim_PayDst(A,M,r)@t))
```

C. ANONYMITY PROPERTY

Blindcoin protocol informally declares its anonymity. To formalize the Blindcoin anonymity, this work defines *Secrecy* and *Unlinkable Coin Receiver* to specify the secrecy of address links and the unlinkability of the coin receiver.

Definition 5: The Blindcoin protocol ensures **Secrecy**, if no other peers, including the middle mix, know the address links between the coin sender s and the coin receiver r for any protocol instance.

1) SECRET_MAPPINGS

There are no Bitcoin peers know the mapping between the coin sender account address and the Blindcoin destination address. The following formula is specified for the *Secret_Mappings*, where A is the coin sender who transfers Bitcoin from address s to address r secretly, and *x* is the middle address of the middle mix.

```
∀ A s r #i . Secret(A,s,r)@i
 ⟶ ¬(∃ x #p #j. Known(s,x)@p & (Known(x,r)@j))
```

Definition 6: The Blindcoin protocol ensures **Unlinkable Coin Receiver** if the attacker cannot distinguish the coin receiver from other peers for any protocol instance with the middle mix to blind Bitcoin transactions.

2) UNLINKABLE COIN RECEIVER

The attacker cannot distinguish the two systems with different coin receivers so that the coin sender transfers Bitcoin to the target destination address is unlinkable. To prove *Unlinkable Coin Receiver*, we consider two different instances of Blindcoin protocol using the operator diff to differ the key terms of the two instances to check the *Observational Equivalence* of Blindcoin protocol. *Observational Equivalence* has been proved to be an effective way to prove the privacy of cryptographic protocols. The detailed handling of our model to check the *Observational Equivalence* is shown by the explanation model rule *unblindReceiver_sync* in Section V.

D. BALANCE CONSISTENCY PROPERTY

For the asset-related protocols, it is required that one asset cannot be spent twice and the sum of the account balance should be consistent [30]. To formalize the balance consistency property of Blindcoin protocol, this work defines *No Double Spending*, *Safe Transfer*, and *Safe Middle Mix*.

Definition 7: The Blindcoin protocol ensures **No Double Spending** if any peer cannot spend one Bitcoin twice for any protocol instance.

1) NO DOUBLE SPENDING

A Bitcoin locked in the previous transaction cannot be spent twice. The following formula is specified for the *no double spending*, where BC is the blockchain label, and x denotes one asset of an account, *n* and m are the destinations of the asset, t1 and t2 are the blockchain heights when the coin sender transferred the asset.

```
∀ BC x n m t1 t2 #i #j .
 Spend(BC,x,n,t1)@i  & Spend(BC,x,m,t2)@j
 ⟶ #i=#j
```

Definition 8: The Blindcoin protocol ensures **Safe Transfer** if there is one Bitcoin sent by coin sender s, there must be one Bitcoin received by coin receiver r for any protocol instance with the middle mix.

2) SAFE TRANSFER

There is only one trace of the account balance increase of the coin receiver that corresponds with the account balance decrease of the coin sender so that the Bitcoin transfer is safe for the coin sender and the coin receiver. The following formula is specified for the *safe transfer*, where A denotes coin sender with account s whose account balance decreases when the balance of account x increases, and r denotes coin destination whose account balance increases when the balance of account y (z) decreases.

```
∀ A s y r #i #j. ((Secret(A,s,r)@i) & (Inc(y,r)@j))
 ⟶ (∃ x #p. Dec(s,x)@p
    & ¬ (∃ z #q.(Inc(z,r)@q) & ¬(#j=#q)))
```

Definition 9: The Blindcoin protocol ensures **Safe Middle Mix** if the middle mix spends one Bitcoin during mixing, there must be one Bitcoin received by the middle mix during mixing for any protocol instance with the middle mix.

3) SAFE MIDDLE MIX

The account balance of the middle mix is consistent so that the Bitcoin of the middle mix is safe. The following formula is specified for the *safe middle mix*, where A1 (A2, A3) denotes the entity playing the role of coin sender, M denotes the entity playing the role of the middle mix, s denotes the account of A2, r denotes coin destination, x denotes the escrow address, and b is the blinded destination.

```
∀ A1 M r #i.
 (Claim_PayDst(A1,M,r)@i)
 ⟶ (∃ A2 s x b #j. ((DstPW(A2,s,r,x,b)@j)) )
    & ¬(∃ A3 #q.(Claim_PayDst(A3,M,r)@q) & ¬(#i=#q))
```

E. MODEL LEMMAS

We have defined Blindcoin security goals as traced properties with guarded first-order logic formulas, and the observational equivalence property based on the diff operator. Based on the above definitions, we encode Blindcoin property into our model lemmas which are detailed listed in this section.

Authentication is a trace property, and we encode authentication based on definitions 1-2. For *Non-injective Agreement*, we specify `lemma Agreement_nia_Offer` to guard the data exchange of Blindcoin offer between the coin sender and the middle mix, `lemma Agreement_nia_Esc` to guard the data exchange of the escrow address between the middle mix and the coin sender, and `lemma Agreement_nia_Dst` to guard the data exchange of coin destination between the coin sender and the middle mix. We specify `lemma Agreement_ia_Offer`, `lemma Agreement_ia_Esc`, and `lemma Agreement_ia_Dst`, to guard insecure states against *Injective Agreement*.

```
lemma Agreement_nia_offer:
"All a b ts #i.
   Claim_commit_agr(a,b,<'CS','Mix',<'Offer',ts>>)@i
 ==> (Ex #j.
   Claim_running_agr(b,a,<'CS','Mix',<'Offer',ts>>)@j) "
lemma Agreement_nia_Esc:
"All a b ts #i.
   Claim_commit_agr(a,b,<'Mix','CS',<'Esc',ts>>)@i
 ==> (Ex #j.
   Claim_running_agr(b,a,<'Mix','CS',<'Esc',ts>>)@j) "
lemma Agreement_nia_Dst:
"All a b ts #i.
   Claim_commit_agr(a,b,<'CS','Mix',<'Dst',ts>>)@i
 ==> (Ex #j.
   Claim_running_agr(b,a,<'CS','Mix',<'Dst',ts>>)@j) "
lemma Agreement_ia_Offer:
"All a b ts #i.
   Claim_commit_agr(a,b,<'CS','Mix',<'Offer',ts>>)@i
 ==> (Ex #j.
   Claim_running_agr(b,a,<'CS','Mix',<'Offer',ts>>)@j
   & j < i
   & not (Ex a2 b2 #i2.
   Claim_commit_agr(a2,b2,<'CS','Mix',<'Offer',ts>>)@i2
   & not (#i2 = #i))) "
lemma Agreement_ia_Esc:
"All a b ts #i.
   Claim_commit_agr(a,b,<'Mix','CS',<'Esc',ts>>)@i
 ==> (Ex #j.
   Claim_running_agr(b,a,<'Mix','CS',<'Esc',ts>>)@j
   & j < i
   & not (Ex a2 b2 #i2.
   Claim_commit_agr(a2,b2,<'Mix','CS',<'Esc',ts>>)@i2
   & not (#i2 = #i))) "
lemma Agreement_ia_Dst:
"All a b ts #i.
   Claim_commit_agr(a,b,<'CS','Mix',<'Dst',ts>>)@i
 ==> (Ex #j.
   Claim_running_agr(b,a,<'CS','Mix',<'Dst',ts>>)@j
   & j < i
   & not (Ex a2 b2 #i2.
   Claim_commit_agr(a2,b2,<'CS','Mix',<'Dst',ts>>)@i2
   & not (#i2 = #i))) "
```

We encode accountability property based on definitions 3-4. For each decision made by *Judging Procedures* of Blindcoin, four security lemmas are specified for SmartVerif to guard against insecure trace that violates *individual accountability*. We specify `lemma Accountable_A_1` to capture the insecure trace that wrongly blames the coin sender who refuses the payment for the middle mix, `lemma Accountable_M_1` to capture the insecure trace that wrongly blames the coin sender who refuses to claim the correct warranty, `lemma Accountable_A_2` to capture the insecure trace that wrongly blames the coin sender who refuses to unblind the coin destination, and `lemma Accountable_M_2` to capture the insecure trace that wrongly blames the middle mix who refuses the payment for the coin destination.

```
lemma Accounable_A_1:
  "All A s r M b x pw #t1 #i #t2 #j.
      ( Secret(A,s,r)@t1 & Claim_Offer(A,M,b)@i
      & UseEsc1(M,x,b,pw)@t2 & Claim_PWarranty(A,M,pw)@j
      & not(Ex #t4. Inc(s,x)@t4))
   ==> not(Ex #k. Claim_PayEsc(A,M,pw)@k)"
lemma Accounable_M_1:
  "All A s r M b x pw #t1 #i #t2 #j #t4.
      ( Secret(A,s,r)@t1 & Claim_Offer(A,M,b)@i
      & UseEsc1(M,x,b,pw)@t2 & Claim_PWarranty(A,M,pw)@j
      & Inc(s,x)@t4
      & not(Ex A2 w #t5 #q. UnblindDst(A2,w,r)@t5
      & Claim_Dst(A2,M,r)@q)
      & not(Ex w #s #p. SignWarranty(A,M,b,pw,w)@s
           & Claim_Warranty(A,M,b,w)@p)
  ==> Ex #h1. Reveal_PWarranty(M,pw)@h1)"
lemma Accountable_A_2:
  "All A s r M b x pw #t1 #i #t2 #j #t4.
    ( Secret(A,s,r)@t1 & Claim_Offer(A,M,b)@i
    & UseEsc1(M,x,b,pw)@t2 & Claim_PWarranty(A,M,pw)@j
    & Inc(s,x)@t4
    & not(Ex #t. Claim_PayDst(A,M,r)@t))
  ==> not(Ex A2 w #t5 #q. UnblindDst(A2,w,r)@t5
        & Claim_Dst(A2,M,r)@q)"
lemma Accountable_M_2:
  "All A s r M b x pw w A2 #t1 #i #t2 #j #t4 #p #t5 #q.
    ( Secret(A,s,r)@t1 & Claim_Offer(A,M,b)@i
    & UseEsc1(M,x,b,pw)@t2 & Claim_PWarranty(A,M,pw)@j
    & Inc(s,x)@t4
    & Claim_Warranty(A,M,b,w)@p
    & UnblindDst(A2,w,r)@t5 & Claim_Dst(A2,M,r)@q
    & not(Ex y #t7. Inc(y,r)@t7)
  ==> (Ex #h2. Log_Warranty(M,w)@h2)
      & not(Ex #t. Claim_PayDst(A,M,r)@t))"
```

We define `lemma Secrecy` for Blindcoin model to guard against insecure trace that violates *Secret_Mappings* based on the definition 5. We check *Unlinkable Coin Receiver* based on the definition 6 using the observational equivalence and the operator `diff`. The required special treatment of Blindcoin model is described in the next section.

```
lemma Secrecy:
   "All A s r #i . Secret(A,s,r)@i
    ==> not(Ex x #p #j. Known(s,x)@p & (Known(x,r)@j))"
```

For the balance consistency property, we specify three security lemmas for SmartVerif to guard against the insecure traces of Blindcoin executions. To guard against the violation of *No Double Spending*, we specify `lemma NoDoubleSpending` based on the definition 7 to show the security of the modeled blockchain. We specify `lemma Consistency_SR` for the Blindcoin model based on the

definition 8 to guard against the violation of *Safe Transfer*, and `lemma Consistency_Fund` based on the definition 9 to guard against the violation of *Safe Middle Mix*.

```
lemma NoDoubleSpending:
 " All BC x n m t1 t2 #i #j .
   Spend(BC,x,n,t1)@i  & Spend(BC,x,m,t2)@j
   ==>#i=#j "
lemma Consistency_SR:
 "All A s y r #i #j. ((Secret(A,s,r)@i) & (Inc(y,r)@j))
  ==> (Ex x #p. Dec(s,x)@p
      & not(Ex z #q.(Inc(z,r)@q) & not(y=z)))"
lemma Consistency_Fund:
 "All A1 M r #i.
  (Claim_PayDst(A1,M,r)@i)
  ==> (Ex A2 s x b #j. ((DstPW(A2,s,r,x,b)@j)) )
      & not(Ex A3 #q.(Claim_PayDst(A3,M,r)@q) & not(#i=#q))"
```

V. FORMALIZATION OF STATE TRANSITION

In this section, we demonstrate the state transitions modeled for the Blindcoin protocol and the encoded symbolic model rules in SmartVerif [13].

A. TERMS, FUNCTIONS AND EQUATIONS

To capture the Blindcoin behaviors covering the Bitcoin data layer, network layer, and application layer, we define terms, functions, and equations for the Blindcoin model to represent Bitcoin structural transactions, growing blockchain, and Blindcoin mixing service. Table 2 lists the function symbols and equations applied in the Blindcoin model.

In our model, we specify Bitcoin structural transaction as `<l,txin,s,e,txid,h>`, where `l` is the label of blockchain, `txin` is the previous transaction id,`s` is the input script to unlock the previous transaction, `e` is the output script to lock current transaction, `txid` is current transaction id that equals to `hash(<txin,s,e>)`, `h` is current block height. The updated chain length is specified as a string with an increasing length that starts with '1', and so does the height of the newly added block. In the Blindcoin model, a newly produced empty block is empty at first, whose block height h+'1' is the current chain length concatenated with '1'. And the newly produced empty block is the premise for the Bitcoin blockchain to record the newly generated transaction.

The defined model rules force the growth of the blockchain and limit no more than one Bitcoin transaction in each block. Then, we can capture the blockchain semantics and achieve the automatic verifiability. To keep the Bitcoin blockchain growing and the new transaction can be recorded, if the block with height h+'1' is added to the chain, then a new empty block with height h+'1'+'1' is produced at the same time. If the newly produced block is added to the Bitcoin blockchain, the blockchain is updated as the model rules specify. To achieve the finite model reasoning time and catch the whole Blindcoin protocol procedures, in our model, we specify that the bitcoin blockchain only grows in a limited period, and the execution of the Blindcoin protocol is done before the chain stops growing.

TABLE 2. Function symbols and description.

Function	N-ary	Description
`hash`	1-ary	hash value calculation
`pk`	1-ary	public key generation
`sign`	2-ary	signature calculation
`verify`	3-ary	signature verification
`true`	0-ary	truth value
`blind`	2-ary	blinding value
`unblind`	2-ary	unblinding value
Equation (Model function property):		
`verify(m,sign(m,k),pk(k))=true`		
`unblind(blind(m,r),r)=m`		
`unblind(sign(blind(m,r),k),r)=sign(m,k)`		

TABLE 3. Blindcoin model fact and meaning.

Fact symbol	Fact meaning
`Ltk`	The private key
`Fr`	New fresh value
`Pk`	The public key
`RecordedTx`	Recorded transaction
`Basechain`	Blockchain basic data
`ChainHeight`	Current blockchain height
`EmptyBlock`	New empty block
`Out`	Insecurely send
`In`	Insecurely receive
`Out_Auth`	Authentically send
`Auth`	Authentic message
`In_Auth`	Authentically receive
`Out_Sec`	Securely send
`Sec`	secure message
`In_Sec`	Securely receive
`BulletBoard`	Records on bulletin board
`A_i`	Coin sender's i^{th} state
`A_i_sync`	Coin sender's i^{th} state in sync
`M_i`	The i^{th} state of the middle mix
`B_i`	The i^{th} state of the coin receiver
`NormalTx`	Signature locked transaction

We also define Blindcoin model functions and equations to capture different mixing services. To model transaction scripts used to protect the randomly mix Bitcoin shown in Figure 2, this work defines the equation `verify(x,sign(x,d),pk(d))=true` to capture the functionality of signature lock. For ease of understanding, we take the RSA-based signature as an example to explain our modeling of Bitcoin ownership. In RSA, every public key `e` is generated along with its paired private key `d`, namely `e=pk(d)`, and requires `e*d=1`. The signature `sign(x,d)` that equals x^d in RSA, where `x` is the signed message and `d` is the private key of the signer. The function `verify` needs the signed message `x`, the signature `s`, and the public key `e` as its three arguments, and `verify` is defined to represent signature verification. Any party can verify a signature by comparing `x` with s^e, where `e*d=1` and s^e equals `x` if the signature is valid. For Bitcoin address, namely, the public key of the coin owner, local macro `e=pk(x)` is defined to let `e` equal to the public key of the coin receiver and `s` be the signature of the coin receiver.

This work defines the blinding function `blind` and the paired unblinding function `unblind` used to hide the

TABLE 4. Blindcoin model action and meaning.

Action Symbol	Action Meaning
Known	Know pseudonym mapping
LowerThan	Restrict block height
NewCoinbase	Generate new Bitcoin
Unspent	Restrict spendable Bitcoin
Inc	Increase spendable Bitcoin
Spent	Restrict unspendable Bitcoin
Update	Update blockchain
SteadyTx	Comfirm transaction
FirstBlock	Genenrate first block
ChainHeight	Receive current chain height
EmptyBlock	Generate empty block
ChOut_A	Output authentic message
ChIn_A	Input authentic message
ChOut_S	Output secure message
ChIn_S	Input secure message
SHTLTx	Generate SHT-lock transaction
Bob_i	Retrive coin receiver's i^{th} state
Claim_Lock	Claim lock transaction
Alice_i	Retrive coin sender's i^{th} state
Unique	Restrict unique participant
Dec	Decrease spendable Bitcoin
Carlie_i	Retrive middle mix's i^{th} state
Claim_Unlock	Claim unlock the transaction
Claim_Offer	Claim Mixing offer
UseEsc1	Use the first escrow address
Claim_PWarranty	Claim partial warranty
Get_PWarranty	Get partial warranty
DstPW	Retrive destination and partial warranty
Claim_PayEsc	Claim escrow address to be paid
Earlier	Restrict transaction time
Transfer	Transfer Bitcoin
SignWarranty	Sign warranty
Claim_Warranty	Claim warranty
Log_Warranty	Log warranty
UnblindDst	Unblind destination
Claim_Dst	Claim destination
UseEsc2	Use the second escrow address
Claim_PayDst	Claim destination to be paid
Spend	Spend Bitcoin
Claim_commit_agr	Claim the commitment for agreed data
Claim_running_agr	Claim the running for agreed data

mapping between pseudonyms to model the blind signature scheme shown in Figure 2. Function `blind` needs the randomness `r` and message `x` as its two arguments, and `blind` is defined to represent blinded message `blind(x,r)`. Function `unblind` needs the randomness `r` and blinded message `y` as its two arguments, and `unblind` is defined to represent the unblinded result `unblind(y,r)`. In RSA-based blind signature scheme, `blind(x,r)`=$r^e * x$ is the blinded result of message `x`, and the unblinded result `unblind(y,r)` equals $r^{-1} * y$. The blinded message `b` equals $r^e * x$ and the blinded signature `y=sign(b,d)` equals $r * x^d$. Blindcoin model verifies the signature by comparing the original message `x` with the unblinded result of valid signature $r^{-1} * y^e$. Therefore, this work specifies the equation `unblind(sign(blind(x,r),d),r)=sign(x,d)` to model signature unblinding and verification, and equation `unblind(blind(x,r),r)=x` is specified to model the property of blinding and unblinding functions when $r=r^e$. The signature function `sign` and the paired verification function `verify` are used not only in the bitcoin transaction but also in blind signature verification.

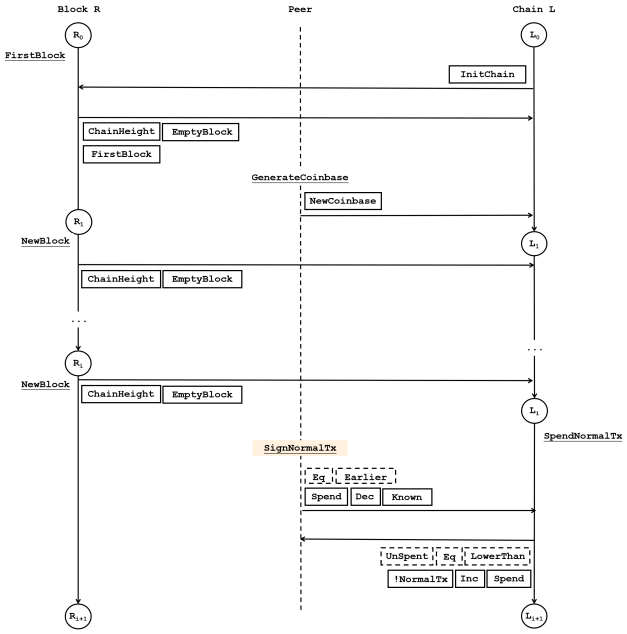


FIGURE 3. State transitions modeled for Bitcoin blockchain.

B. MODEL FACTS

We define Blindcoin model facts consumed/produced during state transitions of Blindcoin protocol and the special action facts (also called actions) guarded by the verification tool to detect insecure states. In the symbolic setting, Blindcoin model facts and actions are as the form $F(t_1,\ldots,t_n)$, where `F` is a specified fact/action symbol and t_i is one of the `n` terms composing this fact/action. The Blindcoin model fact symbols and action symbols are listed in Tables 3 and 4. The Blindcoin model fact/action terms are built from Blindcoin model functions, which are used to explore the concurrent interactions of the Bitcoin data layer, network layer, and script layer. To reduce the model size and achieve an automated-verifiable Blindcoin model, we define Blindcoin model facts to capture the features of the structural transaction, growing blockchain, and Blindcoin mixing service.

To model structural transactions and essential terms generated by the data layer, we apply the fact symbol `Ltk` to associate a private key generated by the built-in fact symbol `Fr` to a Bitcoin user. The argument of `Fr` is the fresh value used as the randomness. Fact symbol `Pk` associates the public key to a user identity, and anyone can look up the public key of the user via the fact `Fr`. Fact symbol `RecordedTx` is used to retrieve the confirmed transaction recorded by the blockchain. Action symbol `Known` labels the trace that the public gets the pseudonym mapping in a recorded transaction. Action symbol `LowerThan` restricts that the current block must be larger than the current chain length. Action symbol `NewCoinbase` labels the trace that the blockchain mines a new Bitcoin. Action symbol `Unspent` restricts that the block height when the Bitcoin becomes spendable is lower than the block height when it becomes unspendable. Action symbol

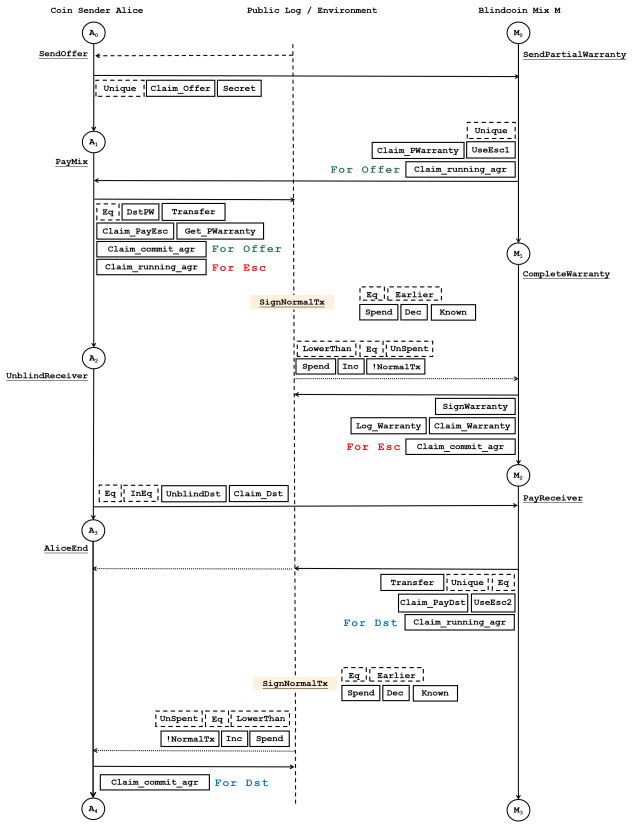


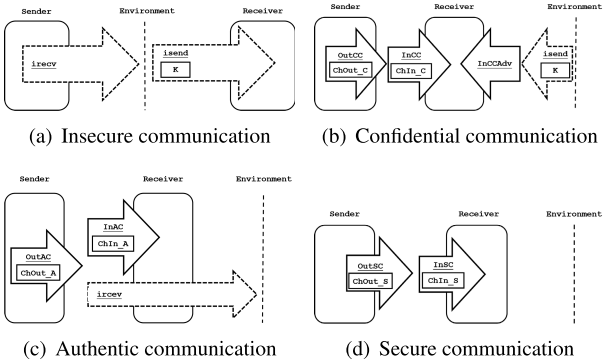
FIGURE 4. State transitions modeled for Blindcoin.

`Inc` labels the trace that the Bitcoin amount received by a Bitcoin address is increased. Action symbol `Spent` restricts that the block height when the Bitcoin becomes unspendable is higher than the block height when it becomes spendable. Action symbol `Update` labels the trace that the blockchain record is updated. Action symbol `SteadyTx` labels the trace that the blockchain confirms a transaction.

The fact symbol `Basechain` is used to specify the initialization information, which denotes the blockchain category and compatibility information to model the growing blockchain/network layer. Fact symbol `ChainHeight` is defined to specify the tail block height of the blockchain, which can only be consumed once to guarantee blockchain growth. Blindcoin model looks up the current block height via the fact `ChainHeight` and consumes this fact to determine the block height term of `EmptyBlock`. The Blindcoin model generates the newly generated block via the fact `EmptyBlock` when the chain grows and the new transaction generation consumes the fact to determine the confirmation time of the new transaction. Action symbol `FirstBlock` labels the trace of the first block generation. Action symbol `ChainHeight` labels the trace of looking up the chain height. Action symbol `EmptyBlock` labels the trace of the new empty block generation.

The messages of the newly generated public key are broadcast via the fact `Out` and received via the fact `In`. This work considers four kinds of channels that are public channel, confident channel, authentic channel, and secure channel separately. Information received/sent by the public channel with facts `In` and `Out` from/to the public, can be controlled by the Dolev-Yao adversary. Fact symbols `Out_Auth`, `Auth`, and `In_Auth` are defined to specify the authentic channel functionalities, and fact symbols `Out_Sec`, `Sec`, and `In_Sec` are defined to specify the secure channel functionalities. Some symbols can be used directly without definition, such as the fact symbol of attacker knowledge `K`, the fact symbol of fresh value generation `Fr`, and the fact symbols of public outputs `Out` and public inputs `In`. The modeling of these channel functionalities is introduced in the paragraph Channel rules.

This work symbolizes the bulletin board to record the accountable warranties published via the bulletin board to capture the features of the Blindcoin mixing service and the message terms in the script layer. The terms of the fact `BulletBoard` is open-access, representing the auditable information of the Blindcoin protocol. This work defines the fact symbol `A_i` to define the i_{th} states of the coin sender and the fact symbol `A_i_sync` to verify observational equivalence, which is detailed introduced in Section III. Fact symbol `M_i` to define the i_{th} states of the middle mix, and `B_i` to define the i_{th} states of the coin receiver. Fact symbol `NormalTx` is defined to look up the confirmed transaction with signature lock.



(a) Insecure communication (b) Confidential communication

(c) Authentic communication (d) Secure communication

FIGURE 5. Communication channels modeled for Blindcoin.

Figure 3 shows the basic Bitcoin state transitions specified by Blindcoin model facts. Figure 4 shows state transitions of the Blindcoin protocol that choose signature lock script to secure the transferred Bitcoin. In Figure 3 and Figure 4, the underlined texts denote model rules that consume and produce Blindcoin model facts to specify protocol executions, and the text framed with the solid rectangle denotes the Blindcoin model actions that are used in Blindcoin security lemmas to guard the insecure state, the text framed with the dotted rectangle denotes the Blindcoin model actions that are used in model restrictions to guide model reasoning, the circled texts denote the entity states and the texts above the horizontal arrows denote the communication message and their directions.

C. CHANNEL RULES

We model four communication channels with different security threats to consider real-world security scenarios of Blindcoin communications. The definitions of the four communication channels are stated in definition 10-13. Any entity can be the auditor to look up the bulletin board and judge the malicious. Since the audit information on the bulletin board is open-access and temper-resistant, the communication among the coin sender, the bulletin board, and the middle mix uses the authentic channel. Due to the openness of the Bitcoin network, the communication among the coin sender, the P2P network, and the middle mix uses the public channel. Meanwhile, we also consider different scenarios of P2P network. If the P2P network is dishonest, the communications between the coin sender, the coin receiver, the middle mix, and the P2P network, use the public channel. Otherwise, the communications with blockchain use the authentic channel.

Definition 10: If a Blindcoin communication channel is modeled as **public**, then assuming that the attacker i) can control communication traffic sent/received by the honest entity with message eavesdropping, interception, modification, and injection; and ii) can act as the protocol entities to sent/received communication traffic.

Definition 11: If a Blindcoin communication channel is modeled as **confidential**, then assuming that the attacker i) cannot read the communication traffic sent/received by the honest Blindcoin entity with message eavesdropping; and ii) can act as the Blindcoin entities to sent/received communication traffic.

Definition 12: If a Blindcoin communication channel is modeled as **authentic**, then assuming that the attacker i) can read the communication traffic sent/received by the honest Blindcoin entity with message eavesdropping; ii) cannot tamper the communication traffic sent/received by the honest Blindcoin entity with message interception, modification, and injection; iii) cannot act as the Blindcoin entities to sent/received communication traffic.

Definition 13: If a Blindcoin communication channel is modeled as **secure**, then assuming that the attacker i) cannot read and tamper the communication traffic sent/received by the honest Blindcoin entity with message eavesdropping, interception, modification, and injection; ii) cannot act as the Blindcoin entities to sent/received communication traffic.

As everyone knows, the Bitcoin transaction is broadcast via the public channel, namely, the Bitcoin transaction message can be received/sent via the fact symbol `In`/`Out`. Due to the built-in rules `irecv` and `isend`, the attacker can control communication traffic sent/received by the fact `Out`/`In` with message eavesdropping, interception, modification, and injection. We limit attacker abilities to control messages communicated through special channels and specify the model rules `OutCC`, `InCC` and `InCCAdv` to protect confidential communication, the rules `OutAC` and `InAC` to protect authentic communication, and the rules `OutSC` and `InSC` to protect secure communication. This work separately adopts certain attacker-limited rules in the corresponding channel assumption to remove specified abilities of the Blindcoin protocol attackers and offer channel protection to analyze Blindcoin protocol security under fine-grained channel assumptions.

```
rule irecv:
   [ Out( x ) ] ---> [ !KD( x ) ]

rule isend:
   [ !KU( x ) ]  --[ K( x ) ]-> [ In( x ) ]

rule OutCC:
   [ Out_Con($A,$B,x) ] --[ ChOut_C($A,$B,x) ]-> [ !Conf($B,x) ]

rule InCC:
   [ !Conf($B,x), In($A) ] --[ ChIn_C($A,$B,x) ]-> [ In_Con($A,$B,x) ]

rule InCCAdv:
   [ In(<$A,$B,x>) ] --> [ In_Con($A,$B,x) ]

rule OutAC:
   [ Out_Auth($A,$B,x) ] --[ ChOut_A($A,$B,x) ]-> [ !Auth($A,x), Out(<$A,$B,x>) ]

rule InAC:
   [ !Auth($A,x), In($B) ] --[ ChIn_A($A,$B,x) ]-> [ In_Auth($A,$B,x) ]

rule OutSC:
   [ Out_Sec($A,$B,x) ]
   --[ ChOut_S($A,$B,x) ]->
   [ !Sec($A,$B,x) ]

rule InSC:
   [ !Sec($A,$B,x) ] --[ ChIn_S($A,$B,x) ]-> [ In_Sec($A,$B,x) ]
```

Due to different implementations of the channel between the coin sender and the middle mix, this work not only assumes that their communications are sent/received via the facts `Out`/`In` but also assumes that they build an authentic channel and communicate Blindcoin protocol messages via the facts `In_Auth`/`Out_Auth`. So we separately consider the communication channel for the Bitcoin blockchain to send its records to be authentic and insecure. So the blockchain records can be received via the facts `In_Auth` and `In` in different scenarios. The sent/received message between the P2P network and the Blockchain is protected against the attacker, namely the communications inside the Bitcoin consensus network for chain growth are assumed as secure. This work separately considers the communication channel between the bulletin board and the user to be authentic and public, namely, the coin sender and the middle mix can receive/send the message from/to the bulletin board via the facts `In_Auth`/`Out_Auth` and `In`/`Out`. We model that the bullet broad and the blockchain can receive the Blindcoin auditable message and Bitcoin transaction via the fact `In`.

The insecure communication via the public channel is based on the built-in rule `irecv` and rule `isend` whose traces are labeled with action `K`. As Figure 5 (a) shows, we model that any message outputted from this channel can be collected and analyzed by the attacker in the public environment, and any message inputted into it can be fabricated. Confidential communication via the confidential channel is modeled by user-defined rule `OutCC` with trace label `ChOut_C`, `InCC` with trace label `ChIn_C` and `InCCAdv`. As Figure 5 (b) shows, we model that the attacker hidden in the public environment cannot read the message outputted from the confidential channel but can input fake messages into this channel. Authentic communication via the authentic channel is modeled by user-defined rule `OutAC` with trace label `ChOut_A`, and `InAC` with trace label `ChIn_A`. As Figure 5 (c) shows, we model that the attacker hidden

in the public environment cannot input the fake message into the authentic channel but can read the messages outputted from this channel. Secure communication via the secure channel is modeled by user-defined rule `OutSC` with trace label `ChOut_S`, and `InSC` with trace label `ChIn_S`. As Figure 5 (d) shows, we model that the attacker hidden in the public environment cannot read the message outputted from the secure channel and cannot input fake messages into this channel.

D. BLINDCOIN RULES

We define multiset rewriting rules to specify the premises and conclusions of protocol execution to model the state transitions of the Blindcoin protocol model. To facilitate the understanding, the rules `SendOffer` and `UnblindReceiver` in Figure 4 are explained as examples. Based on the explanations of the Blindcoin model symbols in Table 2, Table 3 and Table 4, the definitions of other Blindcoin model rules can be inferred from the state transitions in Figures 3-4.

We define `rule SendOffer` to specify the premises for a coin sender to send a Blindcoin protocol offer, the labels that trace/restrict the state transitions, and the conclusion of offer sending. The facts `Fr(n)` and `Fr(r)` are defined as part of the premises to send the Blindcoin protocol offer. For Blindcoin, it uses randomness to determine the mixing fee that the coin sender needs to pay and uses the blind signature scheme to break the address links. It requires a nonce as the parameter of the blinding function and the unblinding function. In this rule, the fresh nonce is denoted as `n` and generated via the fact symbol `Fr` to represent the nonce to randomize the mixing fee. A fresh nonce `r` is also generated by the fact symbol `Fr` to represent the nonce to blind the message and unblind the related signature. To model the premise state to be retrieved, we specify the fact `A_0` whose terms consist of the coin sender's identity `A`, another coin sender's identity to unblinding the signature `A2`, its private key `ltkA`, the coin receiver's identity `B`, and the coin receiver's public key `pkB`. Fact `A_0` denotes the first state of the coin sender after protocol initialization. To model the premise bulletin board records publishing the data of the available mix server, we specify the fact `BulletBroad` whose terms consist of the log mark `'ReadyMix'`, the mix server's identity `M`, and its public key `pkM`. It uses the fact symbol `BulletBoard` with the prefix `!` to model the append-only and arbitrarily-lookup records on the bulletin board, whose content cannot be modified and is published to the public.

```
rule SendOffer:
  let
    dst = <pkB,~n>
  in
    [ Fr(~n), Fr(~r), A_0($A,$A2,~ltkA,$B,pkB),
      !BulletBroad('ReadyMix',$M,pkM)]
  --[ Unique(~ltkA), Unique(pkB),
      Claim_Offer($A,$M,blind(dst,~r)),
      Secret($A,pk(~ltkA),pkB)]->
    [ Out(<'Offer',blind(dst,~r)>),
      A_1( $A,$A2,$M,pkM,~ltkA,pkM,pkB,~n,~r)]
```

```
rule UnblindReceiver:
  let
    dst = <pkB,~n>
    offer = blind(dst,~r)
    w = sign(offer,~ltkM)
    s = unblind(w,~r)
  in
    [ In_Auth($BB,$A,<'Warranty',w>),
      A_2($A,$A2,$M,pkM,~ltkA,pkB,~n,~r,pkM1,pw)]
  --[ Eq(s,sign(dst,~ltkM)),
      InEq($A,$A2),
      UnblindDst($A2,w,pkB),
      Claim_Dst($A2,$M,pkB)]->
    [ Out(<$A2,$M,'Unblind',<pkB,~n,s>>)]
```

These four premise facts are consumed by `rule SendOffer` to trigger the state transition, and its trace are restricted by action `Unique`, which limits the same fact term `ltkA` and the same fact term `pkB` to appear no more once during model reasoning. The state transitions triggered by `rule SendOffer` are labeled by the action `Claim_Offer` and action `Secret` to guard against insecure traces. Blindcoin requires the coin sender to blind its Blind offer message before sending the offer to the mix server. This example assumes that the coin sender sends the blinded offer via the public channel. It specifies the fact `Out` to publish the blinded offer via the public channel, whose terms consist of the coin sender's identity `A`, the message mark `'Offer'`, the blinded offer `blind(dst,~r)` where `dst` is the local macro that equals to `<pkB,~n>`. As one of the conclusions, the coin sender's state is transited into its second state `A_1`, whose terms consist of `A`, `A2`, `M`, `pkM`, `ltkB`, `n`, and `r`.

We define `rule UnblindReceiver` to specify the premise, the labels, and the conclusion for the state transition of a coin sender unblinding the destination. Blindcoin protocol states that the warranty completed by the mix server consists of a signed agreement from the mix server, and the user can get the evidence against the malicious mix server. It requires the mix server to publish the warranty to the bulletin board and the coin sender to look up the warranty record from the bulletin board. In this example, `rule UnblindReceiver` assumes that the coin sender receives the warranty from bulletin board records via the authentic channel. It specifies the premise fact `In_Auth` whose terms are the identification of bulletin board `BB`, the coin receiver's identity `A`, the record mark `'Warranty'`, and the warranty `w`, namely the signature of the message `<pkB,~n>` blinded with the randomness `r` and signed by the private key `ltkM`.

This rule specifies the premise fact `A_2` for the coin sender to retrieve, whose terms consist of `A`, `A2`, `M`, `pkM`, `ltkA`, `pkB`, `~n`, `~r`, the escrow address `pkM1`, and the announced partial warranty `pw`. Fact `A_2` is the third state of the coin receiver after protocol initialization and is generated by the rule `PayMix`. It consumes these two premise facts to trigger the state transition, and its traces are restricted by action `Eq`, which limits the warranty to be valid so that the unblinded signature can be recovered.

To consider different implementations of the coin sender, we separately assume the coin sender is unlazy that generates a new address each time and lazy that reuses the same address. In this example, we assume that the coin sender is unlazy and the traces are restricted by the action `InEq` that limits the coin sender to use a new identity `A2` that is different from the old identity `A`. The triggered state transition is labeled by the action `UnblindDst` and `Claim_Dst` to guard against the insecure traces. The coin sender publishes the unblinded warranty via the authentic channel and the fact `Out_Auth` is specified whose terms consist of `A2`, `BB`, the message mark `'Unblind'`, `pkB`, `n`, and the recovered signature `s`, namely the unblinded warranty `unblind(w,~r)`.

```
rule UnblindReceiver:
  let
    dst = <pkB,~n>
    offer = blind(dst,~r)
    w = sign(offer,~ltkM)
    s = unblind(w,~r)
  in
    [ In_Auth($BB,$A,<'Warranty',w>),
      A_2($A,$A2,$M,pkM,~ltkA,pkB,~n,~r,pkM1,pw)]
  --[ Eq(s,sign(dst,~ltkM)),
      Execute_UR(),
      Claim_Dst($A2,$M,pkB)]->
    [ A_2_sync($A2,$M,pkM,'Unblind',pkB,~n,s)]
rule UnblindReceiver_sync:
  let
    dst = <pkB,~n>
    offer = blind(dst,~r)
    w = sign(offer,~ltkM)
    s = unblind(w,~r)
  in
    [ A_2_sync(A,$M,pkM,'Unblind',pkB1,~n1,s1),
      A_2_sync(A,$M,pkM,'Unblind',pkB2,~n2,s2)]
  --[ Sync()]->
    [ Out_Auth(A,$BB,<'Unblind',
      diff(<pkB1,~n1,s1>,<pkB2,~n2,s2>)>)]
```

When we check *Unlinkable Coin Receiver* of Blindcoin, we specially treat the data of the coin receiver via the above definitions of `rule UnblindReceiver` and `rule UnblindReceiver_sync`. The fact `A_2_sync` contains the unblinded identifiable information of the coin receiver, which should be denoted as the deference between protocol execution. Therefore, we apply `rule UnblindReceiver_sync` to specify the difference between Blindcoin executions by the operator `diff`. As the two rules demonstrated below, the difference between executions is that the fact `diff(<pkB1,~n1,s1>,<pkB2,~n2,s2>)` is replaced by `<pkB1,~n1,s1>` and `<pkB2,~n2,s2>`.

VI. VERIFICATION RESULTS

A. VERIFICATION RESULT

We conduct the experiments to verify the security of the Blindcoin protocol and evaluate the model's ability of vulnerability detection. Because of our tool-friendly model, we reason our model automatically and get the proofs of the Blindcoin vulnerabilities assisted with SmartVerif. Our experiments comprehensively prove the vulnerabilities in the Blindcoin protocol and detected the attacks violating authentication, accountability, anonymity, and balance consistency. The detected insecure states against the security goals are demonstrated in our Appendix.

Tables 5-8 demonstrate our verification results of the Blindcoin security, where ✓ denotes the satisfied lemma and ✗ denotes the violated lemma. The proofs of the violated lemmas and the detection of the possible Blindcoin attacks are illustrated in the Appendix. Here, we introduce how the detected attacks violate our model lemma and explain the Blindcoin attacks detected by our model. And we offer suggestions to prevent the detected attacks. After that, we discuss the generalization of our model, and show how to verify other Bitcoin-compatible mixing protocol based on our model.

B. DETECTED VULNERABILITIES

1) ATTACKS VIOLATING AUTHENTICATION

When verifying the agreement of the Blindcoin offer, we detect that the attacker can act as the coin sender and send the offer message after replacing the coin sender field `pk(~ltkA)` with the attacker's account `pkA`. Our proof shows that the honest middle mix still declares the partial warranty as normal even if the mixing offer is sent by the attacker. Once the Blindcoin attacker receives the partial warranty from the middle mix, it acts as the middle mix and forwards the partial warranty to the honest coin sender. Then, the honest coin sender pays to the escrow address. We prove that the middle mix confirms the offer with the Blindcoin attacker, not the honest coin sender. It brings the action `Claim_running_agr(...,pkA,...,'Offer',...)` into conflict with `Claim_commit_agr(pk(~ltkA),...,'Offer',...>)`, violating our lemmas `Agreement_ nia_offer` and `Agreement_ia_offer`. The violated lemmas mean that the agreement on the data exchange of Blindcoin offer between the coin sender and the middle mix cannot be ensured by Blindcoin.

When verifying the agreement of the Blindcoin coin destination, we detect that the Blindcoin attacker can act as the coin sender and sends the unblinded destination message after replacing the coin sender field `pk(~ltkA2)` with the attacker's account `pkA2`. Our proof shows that the honest middle mix still declares payment to the destination as normal even if the attacker sends the unblinded destination message with the replayed warranty, which leads to the Bitcoin loss of the middle mix. It brought the action `Claim_running_agr(...,pkA2,...,'Dst',...)` into conflict with `Claim_commit_agr(pk(~ltkA2),...,'Dst',...>)`, violating our model lemma `Agreeme- nt_nia_Dst` and `Agreement_ia_Dst`. The violated lemmas mean that the agreement of the escrow address between the coin sender and the middle mix cannot be ensured by Blindcoin. Moreover, the Blindcoin warranties may be the same by choosing the blind randomness of different Bitcoin destinations. The attack can be

TABLE 5. Verification results of *Authentication*.

Property	Lemma	Result
Non-injective Agreement	Agreement_nia_offer	✗
	Agreement_nia_Esc	✓
	Agreement_nia_Dst	✗
Injective Agreement	Agreement_ia_offer	✗
	Agreement_ia_Esc	✓
	Agreement_ia_Dst	✗

TABLE 6. Verification results of *Accountability*.

Judging	Entity	Lemma	Result
Procedure(1)	Coin sender	Accountable_A1	✗
Procedure(2)	Middle mix	Accountable_M1	✓
Procedure(3)	Coin sender	Accountable_A2	✓
Procedure(4)	Middle mix	Accountable_M1	✓

TABLE 7. Verification results of anonymity.

Property	Lemma	Scenario	Result
Secrecy	Secret_Mapping	Unlazy mix Unlazy sender	✓
		Unlazy mix Lazy sender	✓
		Lazy mix Unlazy sender	✗
		Lazy mix Lazy sender	✗
Unlinkable CR	Observational Equivalence	Unlazy mix Unlazy sender	✗

TABLE 8. Verification results of *Balance Consistency*.

Property	Lemma	Protocol	Result
No Double Spending	NoDoubleSpending	Basechain	✗
Situations: Restrict Bitcoin blockchain to be NoDoubleSpending			
Safe Transfer	Consistency_SR	Blindcoin	✗
Safe Middle Mix	Consistency_Fund	Blindcoin	✗

conducted in a dangerous implementation of the middle mix that ignores the repeated use of the escrow address. So it is required to check the uniqueness of the signed escrow address (i.e. the partial warranty).

2) ATTACKS VIOLATING ACCOUNTABILITY

When verifying the *Individual Accountability* of the malicious coin sender who sends the offer and refuses the payment to the escrow address, our proof shows that the first escrow address of the middle mix can be tampered with by the attacker and the judge produce will blame the honest coin sender wrongly in the scenario of the public communication channel (i.e. using fact `Out` and fact `In`). So the account balance of the first escrow address does not increases even if the honest middle mix makes payment to the escrow address. We detect the trace of action `Claim_PayEsc` when no corresponding trace of action `Inc` was detected, violating our model lemma `Accountable_A1`. The violated lemma means that the honest coin sender and the malicious coin sender cannot be distinguished.

3) ATTACKS VIOLATING ANONYMITY

When verifying the *Secrecy* of Blindcoin, our proof shows that the public can get the transaction link between the coin sender, the middle mix, and the coin receiver if the middle mix is lazy and the coin sender is lazy. We both detect the trace of action `Known(pk(~ltkA),pk(~ltkM))` and the trace of action `Known(pk(~ltM),pk(~ltkB))`, violating our model lemma `Secret_Mapping`. The violated lemma means that Blindcoin may leak the relationship between the coin sender and the coin receiver.

TABLE 9. Model verification time cost of authentication.

	Agreement_nia_offer	Agreement_nia_Esc	Agreement_nia_Dst	Agreement_ia_offer	Agreement_ia_Esc	Agreement_ia_Dst
Our model	1.107s	0.791s	2.057s	1.107s	2.057s	2.051s
-untreated transaction confirmation	4.029s	2.158s	7.482s	4.029s	5.611s	7.482s
-untreated Bitcoin status	5.247s	3.748s	6.821s	3.673s	9.745s	8.770s
-untreated block height	12.302s	9.763s	17.770s	8.201s	22.847s	20.309s
Original model	Unterminated	Unterminated	Unterminated	Unterminated	Unterminated	Unterminated

TABLE 10. Model verification time cost of accountability.

	Honest coin sender; Honest mix	Honest coin sender; Malicious mix	Malicious coin sender; Honest mix	Malicious coin sender; Malicious mix
		Lemma Accountable_A1		
Our model	9.131s	8.401s	7.641s	6.711s
-untreated transaction confirmation	24.907s	26.677s	29.677s	27.877s
-untreated Bitcoin status	38.928s	34.728s	34.878s	35.528s
-untreated block height	78.873s	82.933s	86.503s	93.083s
Original model	Unterminated	Unterminated	Unterminated	Unterminated
		Lemma Accountable_M1		
Our model	12.784s	13.304s	12.384s	12.814s
-untreated transaction confirmation	58.118s	55.328s	58.028s	58.688s
-untreated Bitcoin status	48.444s	49.794s	53.794s	50.944s
-untreated block height	157.747s	154.387s	148.017s	149.137s
Original model	Unterminated	Unterminated	Unterminated	Unterminated
		Lemma Accountable_A2		
Our model	14.61s	14.15s	14.52s	13.87s
-untreated transaction confirmation	53.136s	54.276s	56.046s	56.586s
-untreated Bitcoin status	41.524s	40.174s	43.924s	48.324s
-untreated block height	180.282s	175.732s	173.002s	167.612s
Original model	Unterminated	Unterminated	Unterminated	Unterminated
		Lemma Accountable_M2		
Our model	10.349s	10.729s	9.859s	9.689s
-untreated transaction confirmation	32.933s	32.273s	30.803s	31.493s
-untreated Bitcoin status	34.314s	34.914s	38.314s	37.714s
-untreated block height	114.932s	118.503s	125.501s	130.19s
Original model	Unterminated	Unterminated	Unterminated	Unterminated

TABLE 11. Model verification time cost of anonymity.

	Unlazy mix; Unlazy coin sender	Unlazy mix; Lazy coin sender	Lazy mix; Unlazy coin sender	Lazy mix; Lazy coin sender
		Lemma Secret_Mapping		
Our model	13.481s	36.551s	38.795s	52.635s
-untreated transaction confirmation	38.316s	113.805s	124.995s	168.768s
-untreated Bitcoin status	36.774s	109.527s	119.948s	156.297s
-untreated block height	99.814s	386.089s	405.268s	492.407s
Original model	Unterminated	Unterminated	Unterminated	Unterminated
		Observational Equivalence		
Our model	6.742s	6.356s	6.529s	6.857s
-untreated transaction confirmation	22.351s	21.641s	22.824s	22.942s
-untreated Bitcoin status	30.645s	28.561s	29.694s	29.673s
-untreated block height	49.907s	48.982s	49.762s	50.243s
Original model	Unterminated	Unterminated	Unterminated	Unterminated

TABLE 12. Model verification time cost of balance consistency.

	Honest p2p network; Honest coin sender	Honest p2p network; Malicious coin sender	Dishonest p2p network; Honest coin sender	Dishonest p2p network; Malicious coin sender
		Lemma NoDoubleSpending		
Our model	2.201s	3.070s	0.211s	0.237s
-untreated transaction confirmation	5.888s	6.384s	0.636s	0.689s
-untreated Bitcoin status	7.101s	6.293s	0.640s	0.619s
-untreated block height	17.791s	16.991s	1.823s	1.795s
Original model	Unterminated	Unterminated	Unterminated	Unterminated
	Honest coin sender; Honest middle mix	Honest coin sender; Malicious middle mix	Malicious coin sender; Honest middle mix	Malicious coin sender; Malicious middle mix
		Lemma Consistency_SR		
Our model	120.914s	118.014s	125.814s	117.264s
-untreated transaction confirmation	329.815s	326.815s	305.515s	304.115s
-untreated Bitcoin status	343.647s	322.047s	305.247s	299.997s
-untreated block height	895.197s	841.597s	904.797s	846.997s
Original model	Unterminated	Unterminated	Unterminated	Unterminated
		Lemma Consistency_Fund		
Our model	66.502s	61.502s	53.102s	46.502s
-untreated transaction confirmation	272.098s	266.398s	271.198s	239.348s
-untreated Bitcoin status	315.01s	288.91s	260.41s	228.560s
-untreated block height	574.418s	494.418s	448.818s	408.868s
Original model	Unterminated	Unterminated	Unterminated	Unterminated

When we verify *Unlinkable CR* (Unlinkable Coin Receiver), we detect that the adversary can distinguish two protocol instances where the coin receivers differ with `diff`-terms. Lemma *Observational Equivalence* is proved to be violated. It means that the attacker in the public environment can tell the two protocol instances apart and link certain coin receivers by sending coins and analyzing public transactions by requesting the open-access blockchain.

4) ATTACKS VIOLATING BALANCE CONSISTENCY

When verifying the *No Double Spending* of Blindcoin, our proof shows that one Bitcoin can be spent twice. We detect the trace of action `Spend(...,~n,...)` appears twice, violating our model lemma `NoDoubleSpending`. The violated lemma means that Bitcoin can be double-spent like the real-world Bitcoin.

To eliminate the effect of the double-spent Bitcoin, this work restricts the Bitcoin blockchain to satisfy *NoDoubleSpending* during model reasoning.

Blindcoin has no requirement of the middle mix to maintain the list of the old warranty. When verifying the *Safe Transfer* of Blindcoin in the scenario of the honest P2P network, our proof shows that more than one Bitcoin is received by the coin receiver while only one Bitcoin is sent by the coin sender if the attacker replays the old warranty. We detect the trace of action `Inc(...,pk(~ltkB))` appears twice, violating our model lemma `Consistency_SR`. The violated lemma means that the Bitcoin received by the coin receiver is more than the Bitcoin sent by the coin sender, and the balance consistency between the coin sender and the coin receiver cannot be ensured.

When verifying the *Safe Mix* of Blindcoin in the scenario of the honest P2P network, our proof shows that the coin sender pay to the middle mix only once while the middle mix pay the coin destination twice if the middle mix accepts the replayed warranty. We detected that the trace of action `Claim_PayDst($A2.1,...,pk(~ltkB))` conflicts the trace of action `Claim_PayDst($A2,...,pk(~ltkB))`, while the trace of the action `DstPw(...,pk(~ltkB),...)` appeared only once. It violates our model lemma `Consistency_Fund`, so the balance consistency of Blindcoin fund cannot be ensured.

C. MODEL PERFORMANCE AND COMPARISON

The experimental results of model performance are summarized in Tables 9-12. Table 9 lists the verification time of the authentication lemmas based on different models and the considered security scenarios are the honest participants who want to achieve agreement with each other on the exchanged message. Table 10 lists the verification time of the accountability lemmas based on different models in different security scenarios with the honest P2P network. Table 11 lists the verification time of the anonymity lemmas based on different models in different security scenarios with the honest P2P network. The first part of Table 12 lists the verification time of lemma `NoDoubleSpending` based on different models in different security scenarios to check the consistency of the basechain of Blindcoin. The second part of Table 12 lists the verification time of balance consistency lemmas based on different models in different security scenarios with honest P2P network. The experience results show that the verification of the target lemmas based on the original model is unterminated and the original model cannot achieve automated verification. The verification time costs of the models that removed some of our special treatments for blockchain semantics, namely the models that removed some special restriction defined by our work, are several times the time cost of our model verification based on the treated blockchain semantics. Compared with the model that removed some of our special treatments and the original model with no special treatment, our model is sufficient for automation capability and it can verify all the lemmas in a very efficient way. At the same time, our model can check the target lemmas in different security scenarios more efficiently than the model

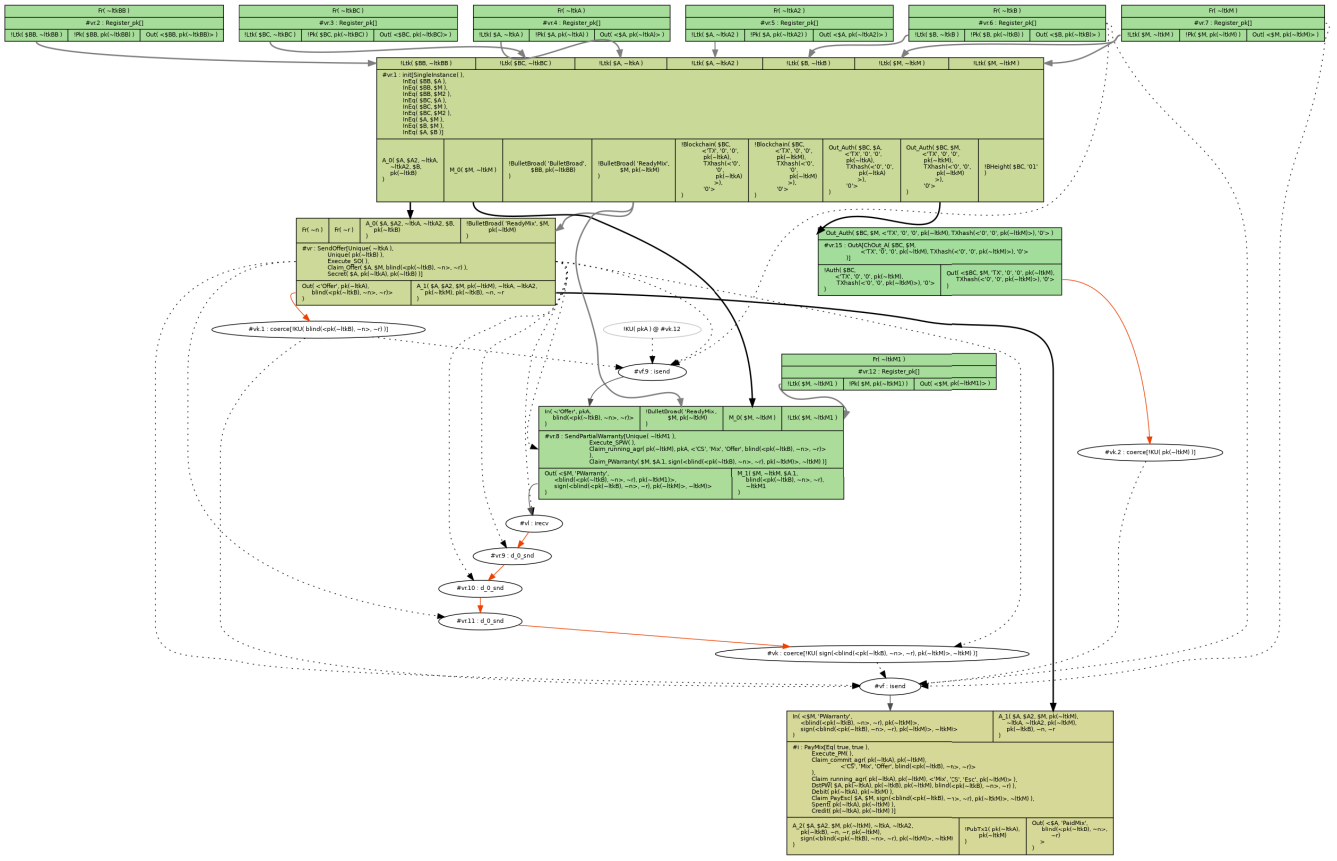
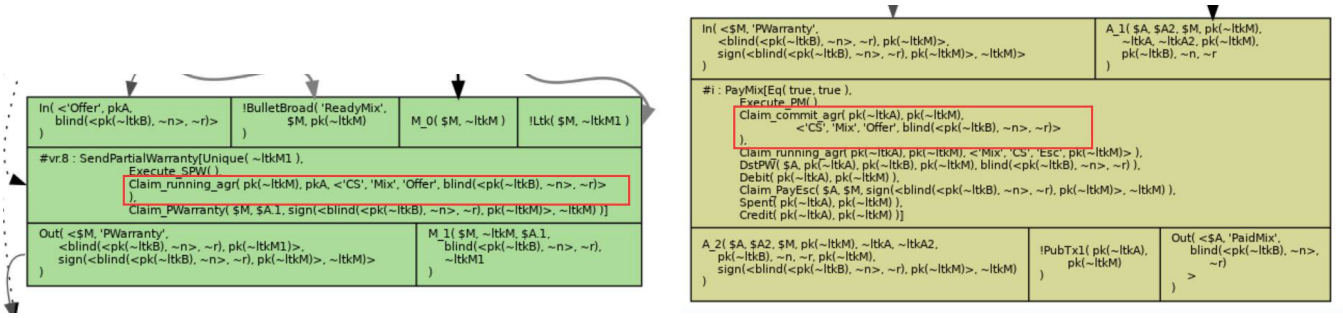


FIGURE 6. **Proof of lemma** `Agreement_nia_offer`.



(a) Claim_running_agr (b) Claim_commit_agr

FIGURE 7. **The conflicted actions detected in the proof violate the lemma** `Agreement_nia_offer`**. Lowe's taxonomy specifies injective agreement has a higher level than non-injective agreement. And our model lemma** `Agreement_ia_offer` **specifies a higher level of authentication than the lemma** `Agreement_nia_offer`**. The detected actions also violate the lemma** `Agreement_ia_offer`**.**

that removed some of our special treatments, and the original model cannot get the verification result of target lemmas in different security scenarios.

VII. DISCUSSION

Based on the verification results and the traced insecure states, we can derive some suggestions for Blindcoin protocol implementations and generalize to other Bitcoin-compatible mixing protocols.

A. SUGGESTIONS

Some suggestions for Blindcoin protocol implementations could be derived from the previous verification results. Firstly, the Blindcoin protocol should explicitly specify the negotiation procedure of mixing offer id. Blindcoin offer id should be added to the required field of mixing offer to ensure injective authentication between Blindcoin mixing entities and resist the relay attack damaging the security of Bitcoin mixing. Before requesting the mixing service of Blindcoin,

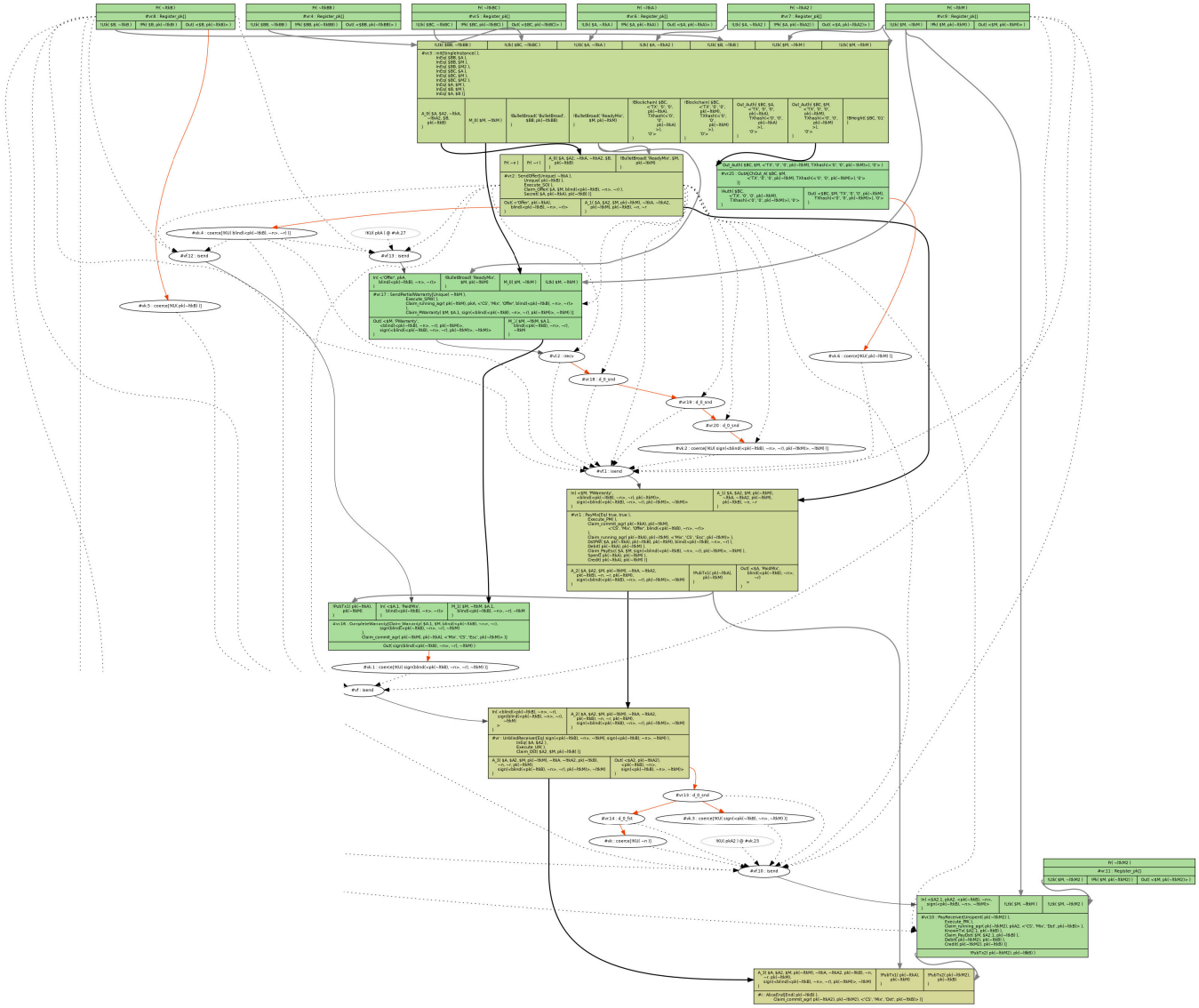
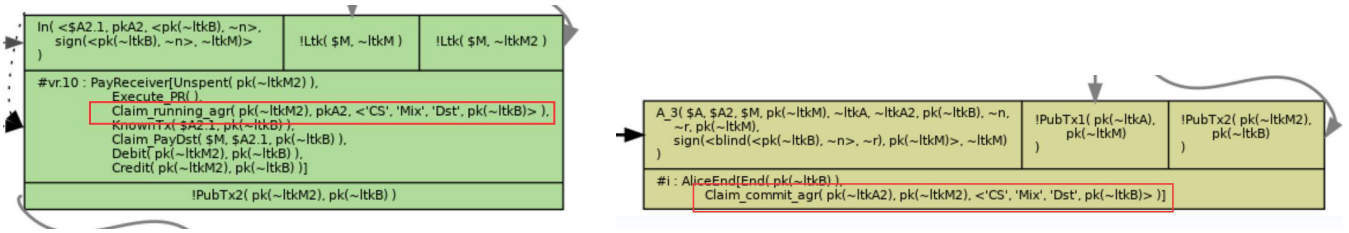


FIGURE 8. **Proof of lemma** `Agreement_nia_Dst`.



(a) Claim_running_agr (b) Claim_commit_agr

FIGURE 9. **The conflicted actions detected in the proof violate the lemma** `Agreement_nia_Dst`**. And our model lemma** `Agreement_ia_Dst` **specifies a higher level of authentication than the lemma** `Agreement_nia_Dst`**. The detected actions also violate the lemma** `Agreement_ia_Dst`**.**

the coin sender procedure should check whether there is any collision between the newly computed offer and the old offer. Before responding to the coin sender, the middle mix procedure should check the freshness of the offer id, and ensure the

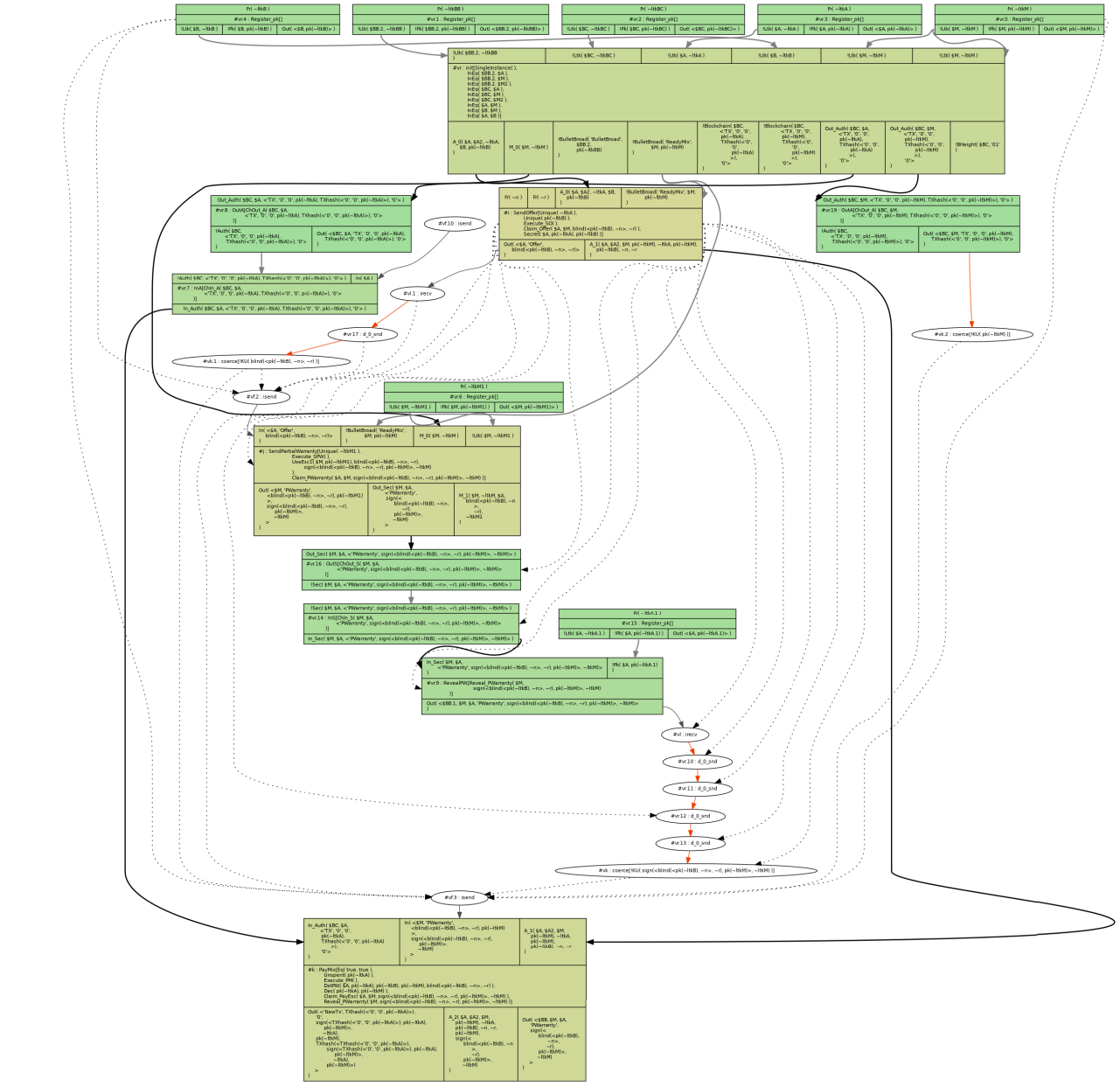
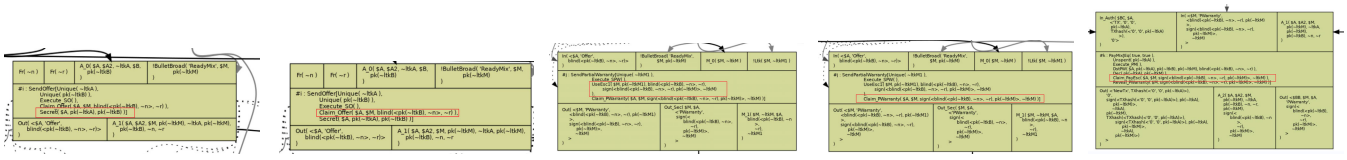


FIGURE 10. **Proof of lemma** `Accountable_A1`.



(a) Secret (b) Claim_Offer (c) UseEsc1 (d) Claim_PWarranty (e) Claim_PayEsc

FIGURE 11. **The conflicted actions detected in the proof violate the lemma** `Accountable_A1`.

unique address paired with the mixing offer. Bitcoin address used by Blindcoin protocol should be ensured one addresses one use.

Moreover, the Blindcoin protocol should provide explicit security assumptions to guide the protocol implementations, including the security assumption of the communication

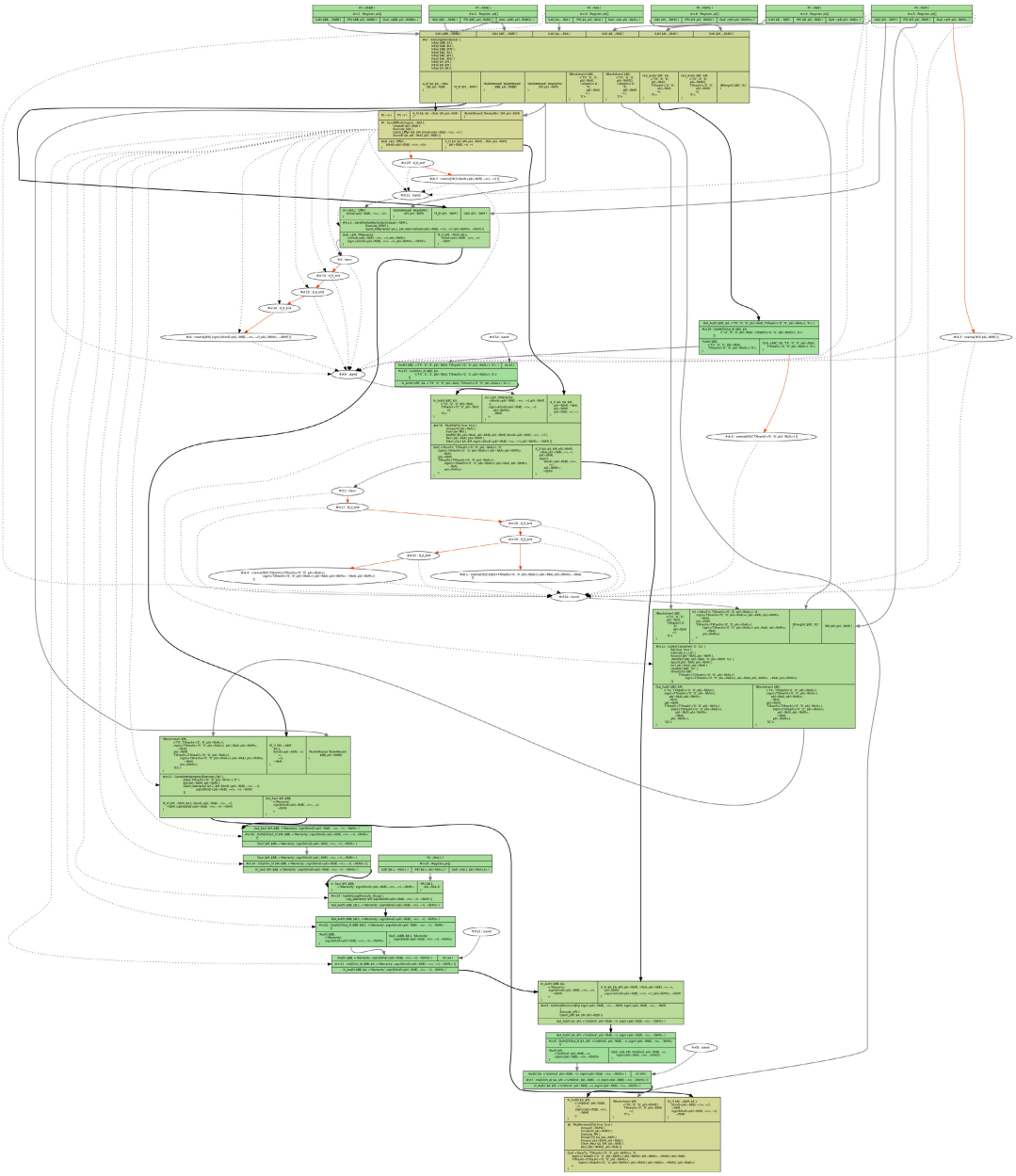
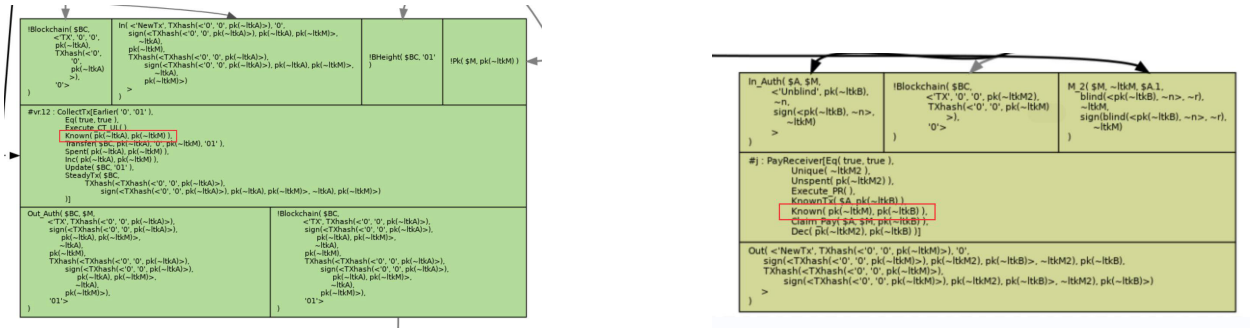


FIGURE 12. **Proof of lemma** `Secret_Mapping`.

channel. Due to the openness of the Bitcoin network and no qualification of the middle mix, the security assumptions of the Blindcoin protocol should be as minimal as possible. To further protect the Blindcoin anonymity, the encryption and randomization mechanisms are suggested to be applied to hide the coin destination.

B. GENERALIZATION OF THE PROPOSED MODEL

Our model rules and restrictions are defined as a unified form that is suitable to represent the certain procedure of different Bitcoin-compatible mixing protocols. Security analyzers can easily extend our model and verify other bitcoin-compatible mixing protocols. They can optionally apply the appropriate model rules and restrictions to specify the protocol procedures and attacker abilities, and capture the characteristics of the Bitcoin-compatible mixing protocol to be formalized. To make the security analyzer easy to capture Bitcoin transactions with different features, our model specifies Bitcoin transactions as a unified form `<l,txin,s,e,txid,h>`, where `l` is the label of blockchain, `txin` is the previous transaction id, `s` is the input script to unlock the previous transaction, `e` is the output script to lock current transaction, `txid` is current transaction id that equals to `hash(<txin,s,e>)`, `h` is current block height. When analyzing other Bitcoin-compatible mixing protocols besides Blindcoin, the security analyzer can make a supplementary



(a) The first Known

(b) The second Known

FIGURE 13. **The conflicted actions detected in the proof violate the lemma** `Secret_Mapping`.

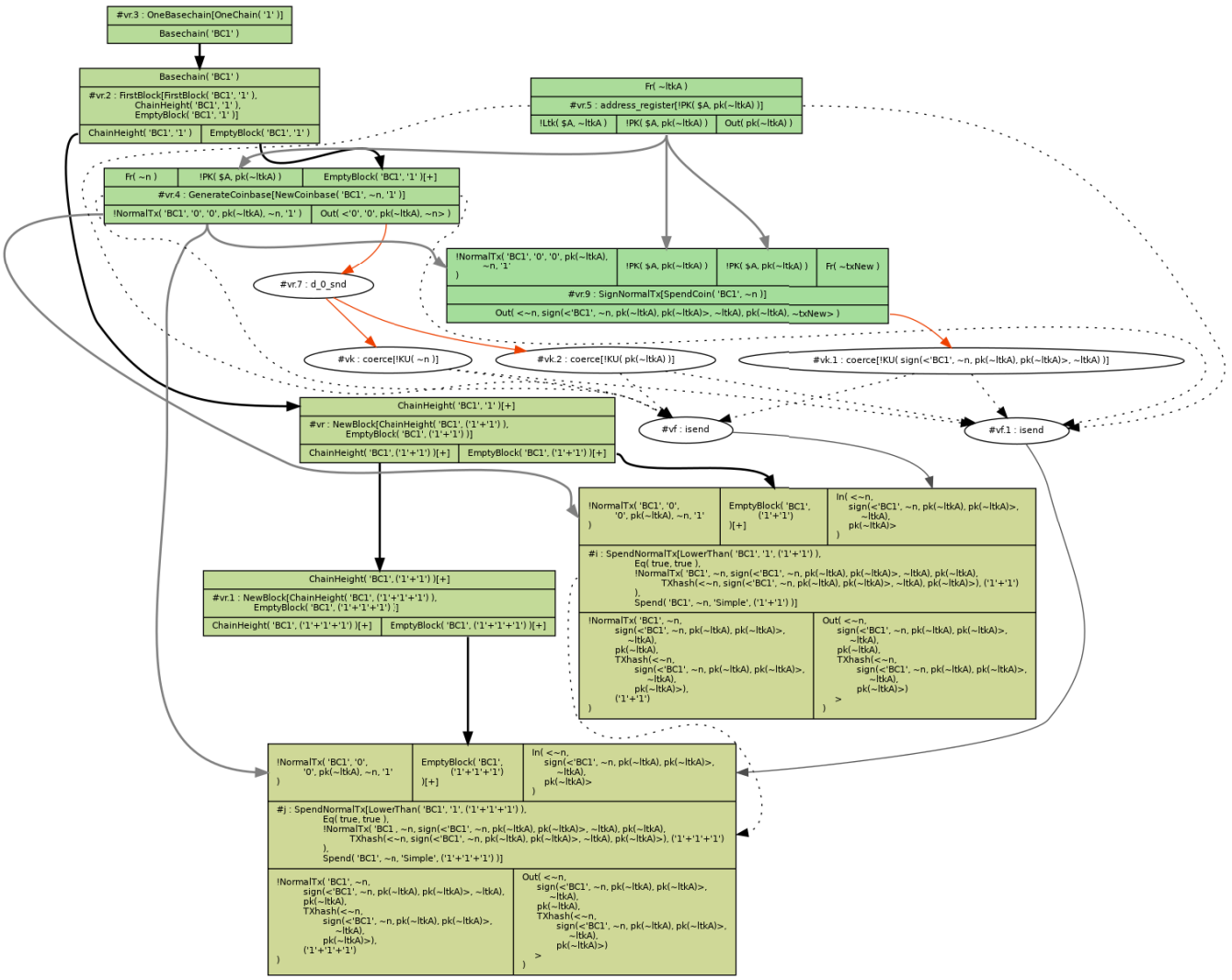
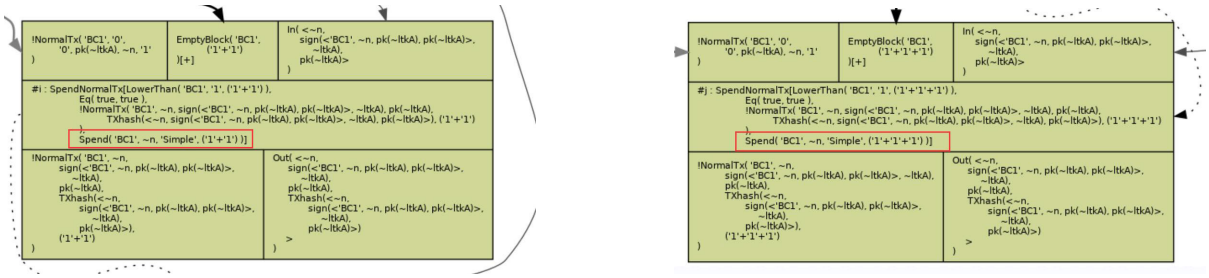


FIGURE 14. **Proof of lemma** `No_Double_Spending`.

definition of our model to specify the protocols with different features.

Here we introduce the main idea of formalizing the Coinswap protocol based on our proposed model as an example of extending our model. When formalizing Coinswap, the characteristic of hash lock and time lock are required to be captured additionally. We specify the hash lock with the local macro `e=<pk(x),hash(i)>`. It means that `e` equals the



(a) The first Spend (b) The second Spend

FIGURE 15. **The conflicted actions detected in the proof violate the lemma** `No_Double_Spending`.

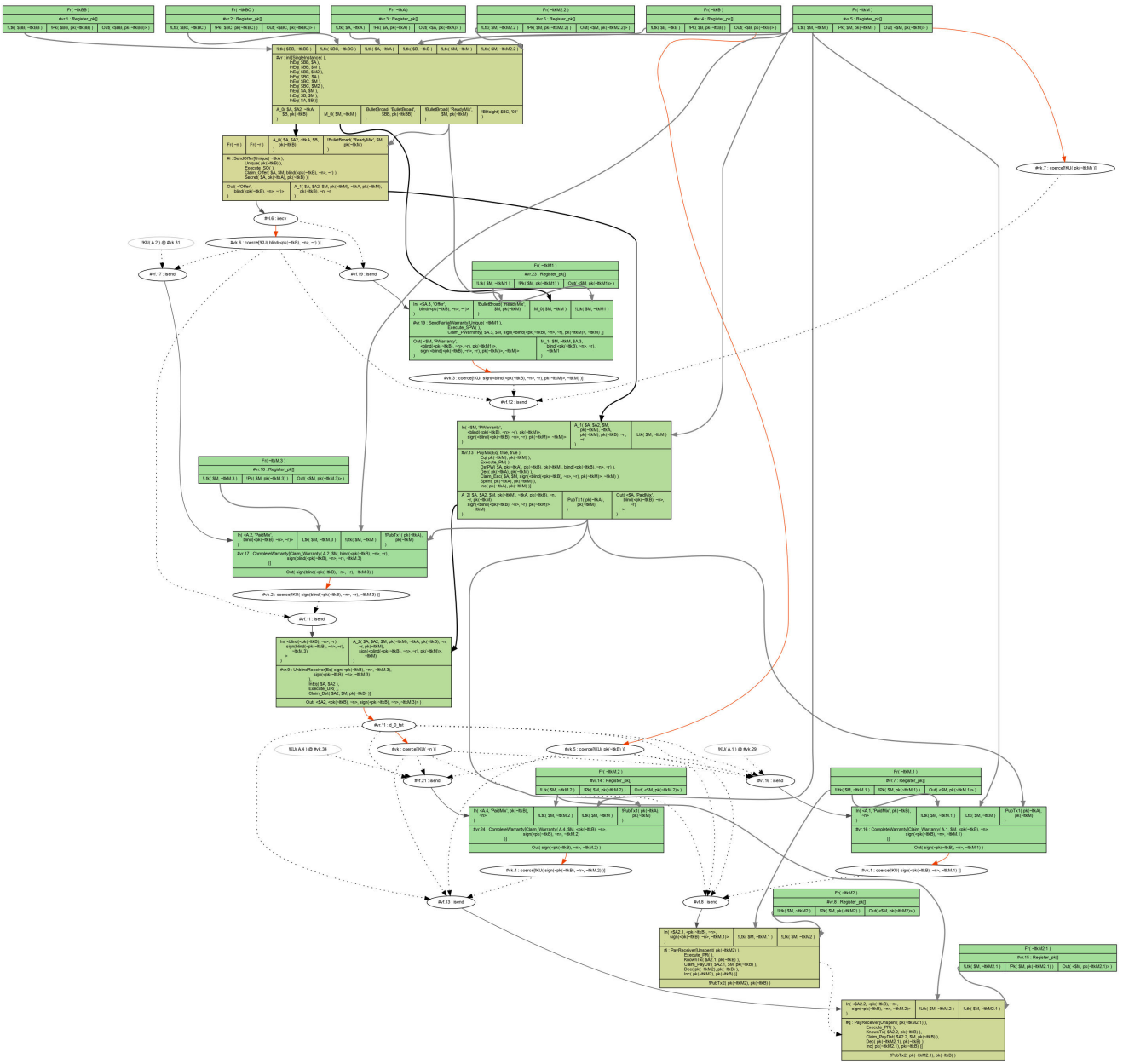
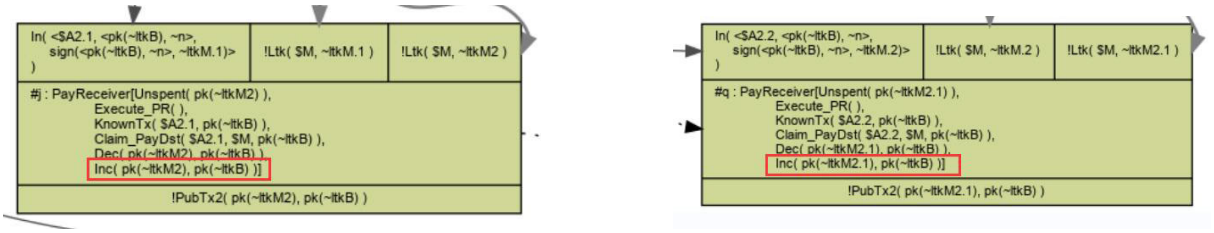


FIGURE 16. **Proof of** `Consistency_SR`.

public key of coin receiver appended with the hash value of a secret value, and `s` should be the signature of the coin receiver appended with the hash preimage. We specify the time lock with the local macro `e=<pk(x1),h+j,pk(x)>`.



(a) The first Inc (b) The second Inc

FIGURE 17. **The conflicted actions detected in the proof violate the lemma** `Consistency_SR`.

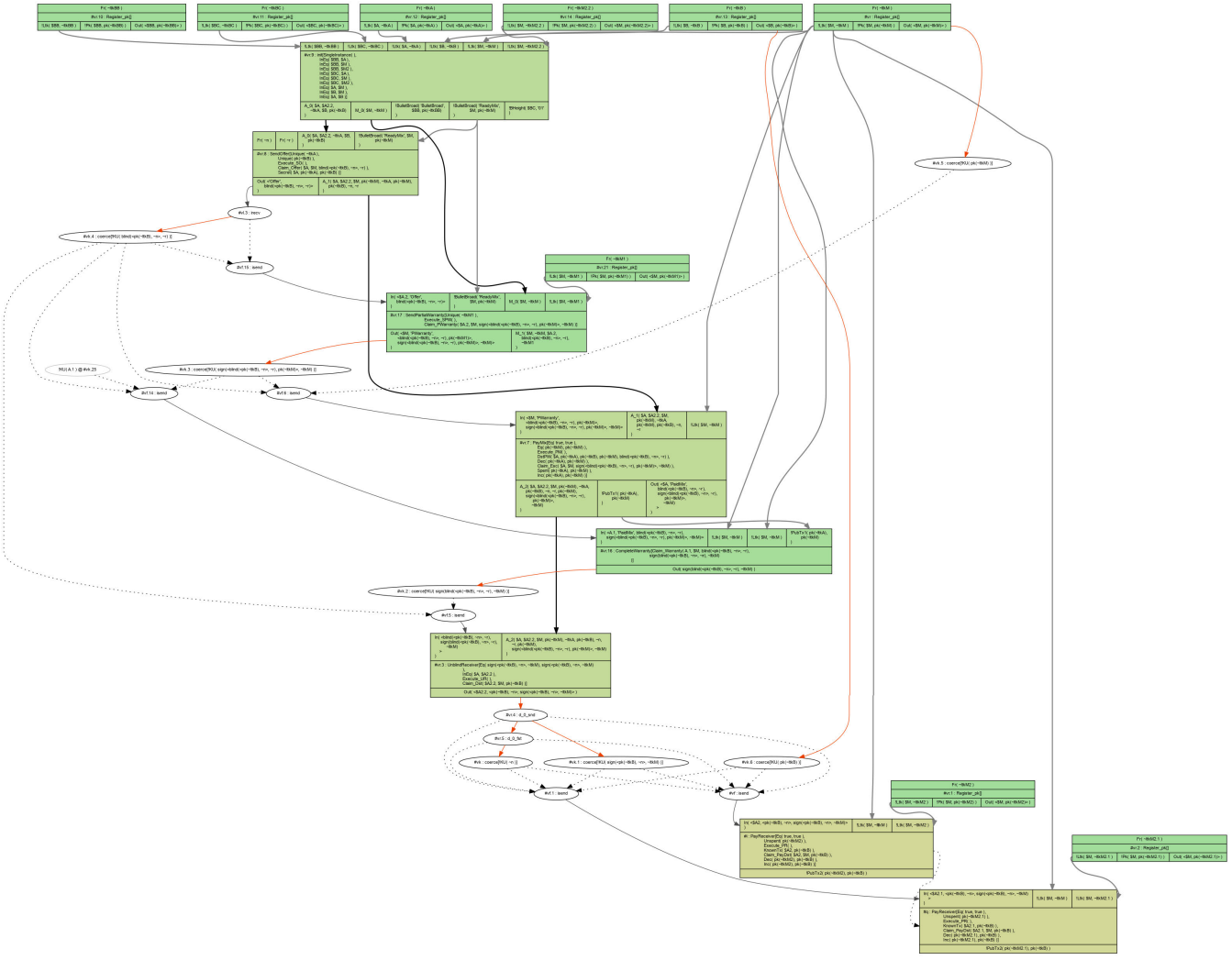
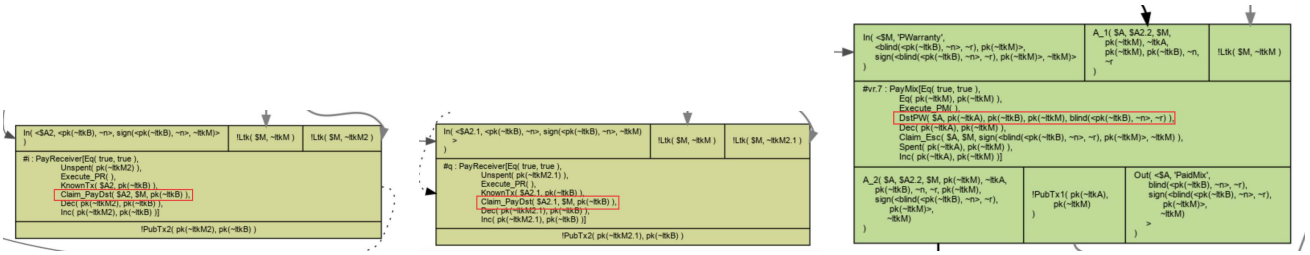


FIGURE 18. **Proof of** `Consistency_Fund`.

It means that `e` equals the public key of coin refunder appended with a certain block height and the public key of the coin receiver, and `s`. The transaction can be unlocked if the coin receiver provides its signature and the chain height is no higher than $(h+j)^{th}$ blocks, or the coin refunder provides its signature and the chain height is higher than $(h+j)^{th}$ blocks. Similarly, we specify the local macro `e=<pk(x1),h+j,h(i),pk(x)>`, which means that `e` is the hash lock combined with the time lock. We also add the definition and the restrictions on the action `Claim_lock` and `Claim_unlock` and remove the definition and the restrictions on the Blindcoin warranty-relevant actions in our model.

The security properties and the encoded model lemmas of Coinswap can refer to the ones in our model. Here we take the formalization of Coinswap authentication for



(a) The first Claim_PayDst (b) The second Claim_PayDst (c) DstPW

FIGURE 19. **The conflicted actions detected in the proof violate the lemma** `Consistency_Fund`.

example. We guard the insecure states of CoinSwap that violate the *Non-injective Agreement* with three lemmas `Agreement_nia_Lock`, `Agreement_nia_Offer`, and `Agreement_nia_ Unlock`. We guard against the CoinSwap attacks violating the *Injective Agreement* with three lemmas `Agreement_ia_Lock`, `Agreement_ia_ Offer` and `Agreement_ia_ Unlock` in the model. As an example, the encoded model lemmas `Agreement_nia_ Lock` and `Agreement_ia_Lock` are shown below.

```
lemma Agreement_nia_Lock:
"All a b ts #i.
   Claim_commit_agr(a,b,<'CR','CS',<'Lock',ts>>)@i
 ==> (Ex #j.
   Claim_running_agr(b,a,<'CR','CS',<'Lock',ts>>)@j) "
lemma Agreement_ia_Lock:
"All a b ts #i.
   Claim_commit_agr(a,b,<'CR','CS',<'Lock',ts>>)@i
 ==> (Ex #j.
   Claim_running_agr(b,a,<'CR','CS',<'Lock',ts>>)@j
   & j<i
   & not (Ex a2 b2 #i2.
   Claim_commit_agr(a2,b2,<'CR','CS',<'Lock',ts>>)@i2
   & not (#i2 = #i))) "
```

VIII. CONCLUSION

This work proposes an automated-verifiable symbolic model for the Blindcoin protocol. It is a comprehensive model that captures the structural transactions, growing blockchain, and mixing services to detect attacks in real-work security scenarios. We overcome the challenges of formalizing and automatically verifying Bitcoin-compatible mixing protocol with a large number of interactions. We formalize the required security properties with fine-grained definitions and encode them into Blindcoin security lemmas, including authentication, accountability, anonymity, and balance consistency. Blindcoin state transitions are deliberately modeled to capture the semantics of multiple layers of Bitcoin and the features of Blindcoin. Our formal model is tool-friendly which can be reasoned automatically and our verification considers different security scenarios. We detect the Blindcoin vulnerabilities and provide the fix suggestions. Furthermore, the proposed model is generic and easy to be extended to verify other Bitcoin-compatible mixing protocols.

In the future, we will enlarge the set of model elements to capture other kinds of blockchain-compatible mixing protocols, not limited to the Blindcoin protocol. We are also trying to achieve automatic modeling, which means that the applied Blindcoin model elements can be selected automatically based on the informal protocol description.

APPENDIX
PROOFS OF VIOLATED LEMMAS AND THE DETECTED INSECURE STATES

See Figs. 6–19.

REFERENCES

[1] M. Touloupou, M. Themistocleous, E. Iosif, and K. Christodoulou, "A systematic literature review toward a blockchain benchmarking framework," *IEEE Access*, vol. 10, pp. 70630–70644, 2022.
[2] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117134–117151, 2019.
[3] M. N. M. Bhutta, A. A. Khwaja, A. Nadeem, H. F. Ahmad, M. K. Khan, M. A. Hanif, H. Song, M. Alshamari, and Y. Cao, "A survey on blockchain technology: Evolution, architecture and security," *IEEE Access*, vol. 9, pp. 61048–61073, 2021.
[4] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang, "Detecting mixing services via mining bitcoin transaction network with hybrid motifs," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 4, pp. 2237–2249, Apr. 2022.
[5] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, "A survey on privacy protection in blockchain system," *J. Netw. Comput. Appl.*, vol. 126, pp. 45–58, Jan. 2019.
[6] A. Biryukov and S. Tikhomirov, "Deanonymization and linkability of cryptocurrency transactions based on network analysis," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Jun. 2019, pp. 172–184.
[7] D. Ermilov, M. Panov, and Y. Yanovich, "Automatic Bitcoin address clustering," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 461–466.
[8] M. C. K. Khalilov and A. Levi, "A survey on anonymity and privacy in Bitcoin-like digital cash systems," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2543–2585, 3rd Quart., 2018.
[9] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for Bitcoin with accountable mixes," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2014, pp. 486–504.
[10] J. Pakki, Y. Shoshitaishvili, R. Wang, T. Bao, and A. Doupé, "Everything you ever wanted to know about Bitcoin mixers (but were afraid to ask)," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2021, pp. 117–146.
[11] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for Bitcoin," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2015, pp. 112–126.
[12] L. Wu, Y. Hu, Y. Zhou, H. Wang, X. Luo, Z. Wang, F. Zhang, and K. Ren, "Towards understanding and demystifying Bitcoin mixing services," in *Proc. Web Conf.*, Apr. 2021, pp. 33–44.

[13] Y. Xiong, C. Su, W. Huang, F. Miao, W. Wang, and H. Ouyang, "SmartVerif: Push the limit of automation capability of verifying security protocols by dynamic strategies," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 253–270.
[14] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 2015, pp. 281–310.
[15] N. Atzei, M. Bartoletti, S. Lande, and R. Zunino, "A formal model of Bitcoin transactions," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2018, pp. 541–560.
[16] B. Xiang-Lin, X. Yan, H. Wen-Chao, C. Kai-Jie, W. Wan-Sen, M. Zhao-Yi, X. Xiao-Feng, and F. Xian-Jin, "Detection of the computational power stealing attack in Bitcoin protocols based on SmartVerif," *Acta Electonica Sinica*, vol. 49, no. 12, p. 2390, 2021.
[17] P. Modesti, S. F. Shahandashti, P. McCorry, and F. Hao, "Formal modelling and security analysis of Bitcoin's payment protocol," *Comput. Secur.*, vol. 107, Aug. 2021, Art. no. 102279.
[18] Y. Hirai, "Defining the ethereum virtual machine for interactive theorem provers," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2017, pp. 520–535.
[19] S. Amani, M. Bégel, M. Bortin, and M. Staples, "Towards verifying ethereum smart contract bytecode in isabelle/HOL," in *Proc. 7th ACM SIGPLAN Int. Conf. Certified Programs Proofs*, Jan. 2018, pp. 66–77.
[20] Z. Yang, H. Lei, and W. Qian, "A hybrid formal verification system in Coq for ensuring the reliability and security of ethereum-based service smart contracts," *IEEE Access*, vol. 8, pp. 21411–21436, 2020.
[21] W. Nam and H. Kil, "Formal verification of blockchain smart contracts via ATL model checking," *IEEE Access*, vol. 10, pp. 8151–8162, 2022.
[22] M. Arapinis, A. Gkaniatsou, D. Karakostas, and A. Kiayias, "A formal treatment of hardware wallets," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2019, pp. 426–445.
[23] P. Das, S. Faust, and J. Loss, "A formal treatment of deterministic wallets," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 651–668.
[24] B. Blanchet, V. Cheval, and V. Cortier, "ProVerif with lemmas, induction, fast subsumption, and much more," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 1–19.
[25] H. Feng, H. Li, X. Pan, and Z. Zhao, "A formal analysis of the FIDO UAF protocol," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–15.
[26] K. Bhargavan, I. Boureanu, A. Delignat-Lavaud, P.-A. Fouque, and C. Onete, "A formal treatment of accountable proxying over TLS," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 799–816.
[27] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5G authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1383–1396.
[28] N. Kobeissi, G. Nicolas, and K. Bhargavan, "Noise explorer: Fully automated modeling and verification for arbitrary noise protocols," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Jun. 2019, pp. 356–370.
[29] V. Cortier, D. Galindo, and M. Turuani, "A formal analysis of the neuchatel e-Voting protocol," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Apr. 2018, pp. 430–442.
[30] J. Dreier, A. Kassem, and P. Lafourcade, "Formal analysis of E-cash protocols," in *Proc. 12th Int. Conf. Secur. Cryptogr.*, vol. 4, 2015, pp. 65–75.
[31] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," in *Proc. Int. Conf. Comput. Aided Verification*. Berlin, Germany: Springer, 2013, pp. 696–701.
[32] J. Dreier, L. Hirschi, S. Radomirovic, and R. Sasse, "Automated unbounded verification of stateful cryptographic protocols with exclusive OR," in *Proc. IEEE 31st Comput. Secur. Found. Symp. (CSF)*, Jul. 2018, pp. 359–373.
[33] M. Saad, A. Anwar, S. Ravi, and D. Mohaisen, "Revisiting Nakamoto consensus in asynchronous networks: A comprehensive analysis of Bitcoin safety and ChainQuality," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 988–1005.
[34] S. Zhang and J.-H. Lee, "Double-spending with a Sybil attack in the Bitcoin decentralized network," *IEEE Trans. Ind. Informat.*, vol. 15, no. 10, pp. 5715–5722, Oct. 2019.
[35] G. Lowe, "A hierarchy of authentication specifications," in *Proc. 10th Comput. Secur. Found. Workshop*, 1997, pp. 31–43.
[36] R. K. Üsters, T. Truderung, and A. Vogt, "Accountability: Definition and relationship to verifiability," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 526–535.

XIANGLIN BAO received the B.S. degree in information security from Anhui University, China, in 2017, and the M.S. degree in computer application technology from the University of Science and Technology, China, in 2020.

She is currently a Teacher with the School of Computer and Information, Anhui Polytechnic University, China. Her research interests include formal method, blockchain technology, and information security.

XIAOFENG XU (Member, IEEE) received the B.S. and Ph.D. degrees in computer science and technology from the Nanjing University of Science and Technology, China, in 2014 and 2020, respectively.

He is currently a Lecturer with the School of Computer and Information, Anhui Polytechnic University, China. His research interests include digital image processing, machine learning, deep learning, computer vision, and information security.

PING ZHANG (Member, IEEE) received the B.S. degree in mathematics from Anhui Normal University, China, in 2004, the M.S. degree in statistics from Southeast University, China, in 2007, and the Ph.D. degree from the School of Information Science and Engineering, Southeast University, China, in 2014.

He is currently an Associate Professor with the School of Computer and Information, Anhui Polytechnic University, China. His research interests include sensor network localization and blockchain.

TAO LIU received the B.S. degree in computer science and education from Anhui Normal University, China, in 1997, and the M.S. degree in computer application technology from the Hefei University of Technology, China, in 2004.

She is currently a Professor with the School of Computer and Information, Anhui Polytechnic University, China. Her research interests include information security, software engineering, and blockchain.