Anonymity is the property of keeping secret the identity of the user performing a certain action. The need for anonymity may arise in a wide range of situations, like electronic voting, Web browsing, and so on. In order to verify the anonymity of security protocols, a framework for formal analysis of anonymous communication protocols is proposed. In this framework, we define operational semantics for security protocols using a labelled transition system; the transition relation is defined by the transition rules, which include create rule, send rule, and receive rule. In addition, the formal description of intruder model in this framework is given. The proposed intruder model specifies the capabilities of the intruder and is weaker than the Dolev–Yao model. Moreover, the concepts of mapping and trace equivalence are proposed; the sender anonymity is formally defined. To illustrate the applicability of proposed framework, we explore the use of probabilistic model checking tool PRISM to analyze the sender anonymity of Crowds protocol. The experimental results show the relationship between sender anonymity and the number of nodes, path reformulations, and forwarding probability, which provides a good way about how to protect the sender anonymity of anonymous communication protocols.

1. Introduction

The need for anonymity arises in a wide range of domains, such as Web browsing, electronic voting, and so on [1, 2]. These activities can be observed and recorded in the network by intruders; therefore, protecting sensitive information is a crucial goal. It is possible that the private data belonging to users will be not properly used and analyzed, resulting in the leakage of all kinds of information, so the study of anonymity in network communication system is an important work.

Anonymous communication protocols can be used in network communications to preserve the identity of users; protocols for ensuring anonymity often use random mechanisms. For example, the Dinning Cryptographers, Crowds, and Onion Routing have been proposed [3]. A systematic framework for evaluating two-factor authentication schemes has been proposed; it is composed of a well-refined criteria set including user anonymity [4]. In addition, some practical solutions to protect identity information and privacy for Industrial IoT and wireless sensor networks have been designed [5, 6]. It can be seen that protecting user anonymity and preventing privacy leakage is a very important job. However, before the widespread deployment of these protocols, its correctness and security requirements must be verified. Formal analysis of security protocols is a well-established field, which is an effective and powerful method to assure the correctness of security protocols. Model checking and theorem proving techniques have been widely used to analyze authentication and secrecy; security protocols are usually modeled at a highly abstract level and the underlying cryptographic primitives are treated as “black-box” to simplify the protocol’s model. However, formal analysis of other properties such as anonymity is still in its infancy.

Most of research work focuses on the authentication and secrecy of security protocols, with anonymity playing a significant role in many protocols receiving less attention from researchers. Formal methods are widely used in formal analysis and verification of security protocols, which can be
used to analyze whether a protocol meets the expected security properties [7]. In Section 2, we introduce related work about formal analysis and verification of anonymous communication protocols, including logic reasoning, process algebra, theorem proving, and universally composable framework. These methods can only qualitatively verify the anonymity property, but cannot give quantitative verification results. Considering the probability of message forwarding in the anonymous communication protocol, the above methods are not in line with the actual situation. Aiming at the defects of the above methods, this paper proposes a framework for formal analysis of anonymous communication protocol, in which the formal definitions of intruder model and anonymity are given. At the same time, in order to verify the feasibility of the framework, Crowds protocol is formally verified using probabilistic model checking as an example to illustrate.

In a nutshell, our contribution of this paper is twofold: a framework for formal analysis of anonymous protocols and its application in Crowds protocol. We briefly discuss the novelties of our work below:

1. This paper provides a formal framework for formal analysis of anonymous communication protocols. In this framework, firstly, we formally define some common concepts in anonymous communication protocols, including message, event, trace, system, and initial configuration of the protocol; then, we define operational semantics for security protocols using a labelled transition system; the transition relation is defined by the transition rules, including create rule, send rule, and receive rule. The formal description of intruder model is given, the proposed intruder model is based on four assumptions and is weaker than Dolev–Yao model, which is determined by the characteristics of anonymity. Moreover, two rules for the growth of intruder’s knowledge set with the execution of the protocol are introduced; the concepts of mapping and trace equivalence are proposed, and the sender anonymity is formally defined.

2. To illustrate the applicability of our framework, we analyze the sender anonymity of Crowds protocol. In this paper, the protocol is modeled as the discrete-time Markov chain, the sender anonymity is expressed by PCTL formulas, and the PRISM model checker is used to analyze the sender anonymity. Based on the experimental results, three conclusions can be drawn: (1) With the increase of the number of nodes, the probability of the sender being observed decreases, which indicates that the sender anonymity of Crowds protocol increases with the increase of the protocol scale. Therefore, increasing the number of nodes in Crowds protocol is an effective way to improve sender anonymity. (2) As the number of paths from the same sender increases, the sender anonymity of Crowds protocol decreases. Therefore, we can reduce the number of paths to improve sender anonymity. (3) With the increase of forwarding probability $P_f$, the probability of the sender being observed decreases, and the sender anonymity is improved. The sender can better hide his own identity information by increasing forwarding probability $P_f$, thus improving the sender anonymity.

The rest of this paper is organized as follows. Section 2 discusses related work on formal analysis and verification of anonymous communication protocols. In Section 3, we introduce a framework for formal analysis of anonymous communication protocols in detail. In Section 4, we take Crowds protocol as an example to verify our framework. In Section 5, the framework proposed in this paper is compared with other methods in detail, which highlights the advantages of our method. In Section 6, we conclude the paper and discuss the future work.

2. Related Work

Various formal definitions and frameworks for analyzing anonymity have been developed. In this section, we summarize existing papers which are mostly related to our work. These methods can be classified into logic reasoning, process algebra, model checking, theorem proving, and universally composable (the abbreviation is UC) framework.

Garcia et al. [8] have come up with the epistemic logic to formalizing anonymity, a formal framework for the analysis of information hiding properties of anonymous communication protocols in terms of epistemic logic was proposed, and a series of formal specifications of different anonymity properties were defined in epistemic logic. This research method can accurately describe the properties of a protocol. However, reasoning security properties rely on manual proof, and without automatic tool support, which is easy to make mistakes [9].

Electronic voting protocols belong to anonymous communication protocols, which are widely used in network voting, so it is particularly important to ensure the anonymity of this type of protocol. The formal analysis of anonymous communication protocols by process algebra was proposed by some researchers. There have been several attempts at formal analysis of FOO protocol in the Dolev–Yao model [10–12]. These analyses assume perfectly anonymous channels. Gergei et al. [13] analyze the FOO electronic voting protocol in the provable security model using the technique of Computationally Complete Symbolic Attacker (CCSA). Unlike the Dolev–Yao analyses of the protocol, this method assumes neither perfect cryptography nor existence of perfectly anonymous channels. The analysis reveals new attacks on vote privacy, including an attack that arises due to the inadequacy of the blindness property of blind signatures.

Chothia et al. [14] present a powerful and flexible method for automatically checking anonymity in a probabilistic general-purpose process algebraic verification toolset named µCRL tools; to illustrate the flexibility of the method, they test the Dining Cryptographers problem and the FOO 92 voting protocol. Shmatikov [15] has analyzed the
anonymity properties of Crowds by using probabilistic model checking; how probabilistic model checking can be applied to the analysis of security properties based on discrete probabilities is demonstrated. Américo et al. [16] use probabilistic model checking to obtain channels modeling two popular anonymity protocols: the Dining Cryptographers and Crowds. Probabilistic model checking can provide qualitative analysis of protocol anonymity for protocols. However, the defect of this method has mostly focused on the model checking approach on systems with fixed configurations, and the model construction is complex, difficult to understand, and prone to state explosion. Unfortunately, formally stating information hiding properties is quite difficult. The current research work considers only one form of anonymity, and other types of anonymity are not taken into account, such as untraceability or relationship anonymity.

Theorem proving is an effective method when dealing with general systems of infinite state space. The notion of provable anonymity theory was built in Isabelle/HOL to achieve a mechanical framework for the analysis of anonymity protocols [17, 18]. Its feasibility is illustrated through two case studies of the Crowds and Onion Routing protocols. In order to make the strand space model satisfy the special needs of anonymity analysis, the anonymity formalization framework based on strand space model was extended [19–21]. In addition, a method for formalizing anonymity based on protocol composition logic (PCL) is proposed, which extends the application of PCL [22]. Formalizing anonymity based on theorem proving is difficult to construct proof strategy, which is a tedious task.

The UC framework combines abstract cryptography operation with security definition in practical cryptography. In [23], based on UC framework, a black-box model for anonymity communication protocols is proposed. In [24], a time-sensitive UC framework named TUC is proposed. This framework integrates the concept of time in the asynchronous communication model and strictly proves the anonymity provided by the protocol in the case that there are time-sensitive attackers who can launch timed attack. The formal verification of anonymous communication protocol based on UC framework lacks the support of automation tools, which is also a tedious task.

In order to more clearly explain the preliminary work, the related work has been classified and listed in the form of tables. We sorted out the listed references from four aspects: method, research object, automatic proof or manual proof, and characteristic (see Table 1).

3. A Framework for Formal Analysis of Anonymous Communication Protocols

In this section, a formal framework for formal analysis of anonymous communication protocols is presented. Firstly, some formal system notations are introduced. Then, the formal definitions of intruder model and sender anonymity are given in detail.

3.1. Formal System Notation. Security protocols describe the process of message interaction between protocol participants; a session is a single run of the protocol. Participants of the session are called agents. The environment in which the communication between sender and receiver is unsecure. Before formal analysis and verification of security protocols, we need to formally introduce the formal system notation.

Definition 1 (Agent). Agents send and receive message. There are three types of agents in anonymous communication protocols: the server, the honest nodes, and bad nodes. The type of agent is formally defined as follows:

\[
\text{Agent} := \text{server} | \text{honest nodes} | \text{bad nodes.}
\] (1)

The agent represents the someone who performs the protocol role, which can be denoted as \( \hat{X} \), and \( X \) represents the corresponding instance of \( \hat{X} \) in the process of protocol execution. Honest nodes strictly follow the protocol specification, and bad nodes will conspire with intruders to disclose the sender’s identity information.

Definition 2 (Term). The term is the data involved in protocol execution, which mainly includes variants, protocol roles, fresh numbers, composite messages, and ciphertext messages. These symbols are defined as follows:

\[
\text{Term} := \text{Var} | \text{Role} | \text{Fresh} | (\text{Term}, \text{Term})
\] \( \cdot \) | \( \text{Term} \}_{\text{Term}} F (\text{Term}),
\] (2)

where \( F \) denotes a function with parameters, for example, encryption and decryption.

Definition 3 (Event). The event is the action of the protocol participants in the execution of the protocol. In anonymous communication protocols, we mainly focus on encryption, decryption, sending, receiving, and matching operations. It is defined as follows:

\[
\text{event} := \text{enc} (m, K) | \text{dec} (m, K^{-1}) | \text{send} (m)
\] \( \cdot \) | \( \text{receive} (m) \cdot \text{match} (m, m_2).
\] (3)

Here \( K \) and \( K^{-1} \) denote the public key and private key of an honest node.

Definition 4 (Transition Relationship). The transition relationship describes the state change relationship of the protocol participants after the event is executed. For example, the state \( \sigma_i \) of agent \( X \) is transformed into state \( \sigma_{i+1} \) after the execution of event \( a \), which can be formally represented as \( \sigma_i \longrightarrow a \sigma_{i+1}. \)

Definition 5 (Trace). The trace is communication alternation sequence generated by different times of protocol operation. The formal description is as follows:

The trace \( \Pi = \sigma_0 a_1 \sigma_1 \sigma_2 \cdots a_n \sigma_n \) is a bounded alternating sequence between state \( \sigma_{i-1} \) and action \( a_i \), for all \( 0 < i \leq n \), the \( \sigma_{i-1} \longrightarrow^{a_i} \sigma_i \) holds, \( |\Pi| \) represents the length of trace \( \Pi. \)
**Definition 6** (Run). The run means each a single execution of a role. To be more precise, from a trace standpoint, the run is a complete execution process of the protocol; the formal description is as follows:

\[ \text{run} := (\Pi = \sigma_0 \sigma_1 \sigma_2 \ldots \sigma_n \land (|\Pi| = \text{length}(\rho))). \]  

(4)

Here \text{length}(\rho) denotes the length of a protocol trace when the protocol \rho is executed.

**Definition 7** (Instantiation). A role term is transformed into a run term by applying an instantiation from the set \text{Inst}, which is defined as

\[ \text{RID} \times (\text{Role} \mapsto \text{Agent}) \times (\text{Var} \mapsto \text{Run Term}). \]  

(5)

Here \text{RID} denotes the identifier of \text{Run} and \text{RunTerm} denotes the instantiation of a term.

**Definition 8** (RunSet). \text{RunSet} is the set of all run when a protocol is executed many times.

\[ \text{Run Set} := \bigcup_{i=1}^{n} \text{run}_i. \]  

(6)

Here \( n \) denotes the number of protocols runs.

**Definition 9** (System). The system is the parallel of agents and intruder activities:

\[ \text{System} := \bigwedge\top X X \in \text{Agent} \parallel \text{Intruder}. \]  

(7)

Here \( \parallel \) denotes for parallel and \( \bigwedge\top \) denotes the instantiations of agent X.

**Definition 10** (Initial configuration of the protocol). The initial configuration of the protocol refers to the initial message set held by agent X before the protocol starts to run, mainly including the public key of other agents, the number of agents and the IP address of the agent.

\[ \Psi := \text{Agent.pubkey} \cdot \text{Agent.num} \cdot \text{Agent.adress} \ldots. \]  

(8)

**Definition 11** (Message). Message transformed between protocol principals has the form

\[ \text{Message} := \text{sender} \times \text{receiver} \times \text{term}. \]  

(9)

Here sender \( \in \text{Agent} \), receiver \( \in \text{Agent} \), and term \( \in \text{Term} \).

### 3.2. The Formal Model of Anonymous Communication Protocols

For the purposes of analyzing anonymous communication protocols, we must build the formal model of anonymous communication protocols. The state of the protocol system is modeled as a tuple \(<s, \text{system}>\), where \( s \) is the path composed of input and output actions of protocol participants and the \text{system} is composed of some principals. Therefore, the message knowledge set of the current environment is recorded. The protocol state is represented as \( C_0, C_1, \ldots \). When the protocol has not started running, \( s \) is an empty string \( \varepsilon \). \( <s, \text{agents}> \) indicates the initial state of the protocol, which is represented as \( C_0 \).

When the protocol is executed, the protocol principals send and receive messages, and the state of the protocol will change accordingly with the action. The transition relationship of protocol state is described by operation

---

**Table 1: A History of Formal Analysis of Anonymous Communication Protocols.**

| Research work                  | Method         | Research object     | Automatic proof or manual proof | Characteristic                        |
|-------------------------------|----------------|---------------------|---------------------------------|---------------------------------------|
| Garcia et al. (2005) [8]      | Logic reasoning| General protocols   | Manual proof (epistemic logic)   | Properties can be stated directly as formulas |
| Gelernter et al. (2013) [9]    |                |                     |                                 |                                       |
| Murat et al. (2012) [10]       |                |                     |                                 |                                       |
| Chadha et al. (2016) [11]      |                |                     |                                 |                                       |
| Cortier et al. (2017) [12]     |                |                     |                                 |                                       |
| Gergei et al. (2018) [13]      |                |                     |                                 |                                       |
| Chothia et al. (2006) [14]     |                |                     |                                 |                                       |
| Shmatikov (2004) [15]         |                |                     |                                 |                                       |
| Américo et al. (2017) [16]     |                |                     |                                 |                                       |
| Li et al. (2015) [17]         |                |                     |                                 |                                       |
| Tarek et al. (2015) [18]      |                |                     |                                 |                                       |
| Li et al. (2013) [19]         |                |                     |                                 |                                       |
| Lu et al. (2008) [20]         |                |                     |                                 |                                       |
| Dong et al. (2011) [21]       |                |                     |                                 |                                       |
| Feng et al. (2015) [22]       |                |                     |                                 |                                       |
| Feigenbaum et al. (2012) [23]  |                |                     |                                 |                                       |
| Backes et al. (2014) [24]     |                |                     |                                 |                                       |

- **Method**: Manual proof (epistemic logic)
- **Research object**: General protocols
- **Automatic proof or manual proof**: Manual proof (PCL)
- **Characteristic**: Depend on computational complexity

---
semantics rules. Before operation semantics rules are introduced, the definition about LTS (Labelled Transition System) is given as follows.

An LTS is a four-tuple \((S, L, \to, s_0)\), where \(S\) is the set of states; \(L\) is the set of labels; \(\to: S \times L \times S\) is a transition relation; and \(s_0 \in S\) is the initial state.

We are now able to define our operational semantics for security protocols using a labelled transition system. The transition relation is defined by the transition rules. The three rules have a similar structure, the upside of the rule described the current state and knowledge set of protocol principals, the system will transition from the current state to the next state after the action is executed, which is represented by the underside of the rule.

\[
\begin{align*}
R \in \text{do } m(P), \Sigma_i, \sigma_i & \rightarrow \text{create}(R) \sigma_j \quad \text{[create]}, \\
\text{update}(\Sigma_i, \Sigma_j), \sigma_j & \triangleq \Sigma i \cup \text{run of } (P, R), \\
\text{send}(m) \sigma_i & \rightarrow m \in \text{Message}, \Sigma_i, \sigma_i \rightarrow \Sigma_j, \sigma_j \triangleq \Sigma i \cup m \quad \text{[send]}, \\
\text{receive} & \quad \text{match}(B, m) \rightarrow m \in \text{Message}, \Sigma_i, \sigma_i \rightarrow \Sigma_j, \sigma_j \triangleq \Sigma i \cup \text{match}(B, m) \quad \text{[receive]}.
\end{align*}
\]

The create rule expresses that a new run from the set of possible runs \(\text{runsof}(P, R)\) can be created. The send rule states that if a run executes a send event, the sent message is added to the \(\Sigma_j\); meanwhile, the adversary learns the sent message and the run progresses to the next step. The receive rule states the message specified in the receive event pattern should match any of the message from the buffer.

The function \(\text{update}: \Sigma_i \times \Sigma_j \rightarrow \text{Boolean}\) is defined as:

\[
\text{procedure } \text{update}(\Sigma_i, \Sigma_j) \\
\text{if } (\sigma_i \rightarrow \text{event} \sigma_j) \\
\quad \text{the system state } \Sigma_i \text{ transition to state } \Sigma_j; \\
\quad \text{return } \text{True}; \\
\text{else.} \\
\quad \text{return } \text{False};
\]

Messages from the channel are received by agents if they match a certain pattern, specified in the receive event. The typed matching predicate \(\text{Match}: \text{Buffer} \times \text{Message} \rightarrow \text{Message}\) is defined as follows:

\[
\text{procedure } \text{match}(\text{buf, message}) \\
\text{Foreach(buf }\in\text{ Buffer}) \\
\quad \text{if(buf }=\text{ message)} \\
\quad \quad \text{the message added to the state } \Sigma_j; \\
\quad \text{return } \Sigma_j; \\
\text{else.} \\
\quad \text{return False;}
\]

### 3.3 Intruder Model

Some practical schemes in unreliable communication environment were proposed, and the security of the proposed scheme based on the defined intruder model was analyzed. For example, four important assumptions about the intruder model in the multi factor authentication scheme have been summarized [25]. Specifically, the intruder completely manipulates the public channel (e.g., eavesdrop, delete, insert, modify, or block any transcripts). In addition, the intruder can steal the victim’s identity information to launch an attack such as reply attack, side-channel attack, and so on. The threats of passive eavesdropping and active attacks have been considered, and the capabilities of the intruder have been discussed [26–28].

We discuss anonymity property based on the observation of the intruder. In this section, we explain the intruder model for intruder’s ability in this section. Intruder model is an important part of formal model for anonymous communication protocols, which specifies the capabilities of the intruder. The purpose of attacking anonymous communication protocols is to distinguish the identity of sender and receiver. For example, an intruder observes a Web request from an anonymous network and wants to be able to identify the originator of the request. Another purpose of the intruder is to link different anonymous paths together to determine whether these paths are established by the same initiator, which can be performed by observing the communication mode of data.

In the Dolev–Yao model, the intruder has full control of the communication channel. The intruder can intercept, eavesdrop, forward, substitute, or replay any sent message. While intercepting messages, the intruder receives the message transmitted on the public channel and constructs new messages based on the received messages. In the anonymous communication environment, the intruder is interested in which session is happening, who sends and receives messages, and how the communication mode works. Following the intruder model defined in [25–28] and combined with the characteristics of anonymous communication protocols, the intruder model for anonymous communication protocols has been formally defined in this framework, which is summarized as follows:

1. The intruder can observe and record some part of the network traffic data in the anonymous communication network, but not all of the traffic data. In other words, the intruder cannot launch a global attack.
2. The intruder can join the anonymous communication system and become the relay node.
(3) The intruder can control some nodes in the anonymous network.
(4) The intruder is able to generate new messages based on observations or collected messages. In other words, the intruder has the capabilities to learn new terms.

The intruder model follows the strong encryption assumption of Dolev–Yao model; the cryptographic algorithm used in the protocol is secure and unbreakable, without considering the security of the cryptographic algorithm itself. It should be noted that the intruder is passive in the sense that he observes all network traffic, but does not actively modify the message or inject new message. The capabilities of the intruder in our intruder model are weaker than those in the Dolev–Yao model; the intruder is passive in the sense that he observes all network traffic, but he does not actively modify the message. This is determined by the basic characteristic of anonymity, which prevents the initiator’s identity from being observed and analyzed by the intruder. The intruder model can be illustrated in Figure 1. The intruder can not only run its own node, but also control other nodes, which are represented by grey nodes. The white nodes represent honest nodes, which will not disclose their own identity information and the previous and next nodes identity information.

\[ m \in \text{Message}, \sum_i \sigma_i \xrightarrow{\text{eavesdrop}(m)} \sigma_j \]
\[ \text{update}(\sum_i, \sum_j), \sum_j \triangleq \sum_i \cup \rho(\hat{X}_i \xrightarrow{m} \hat{X}_j) \]

The function \(\text{update}(\sum_i, \sum_j)\) has been explained in Section 3.2 Whenever a message is sent to receiver, the intruder will decrypt the message (if he can) and get the information. We represent this decryption as the function \(\text{analyze} : \text{Message} \rightarrow S_T\), where \(S_T\) is the set of terms.

**procedure** analyze(m)
if (m ∈ Agent)
    return {m};
if (m is [t1, t2])
    return analyze(t1) ∪ analyze(t2);
if (m is \([t_1], \&\& \text{intruder has } t_2^{-1}\))
    return analyze(t1);
if (m is \(\mathcal{F}(t), \&\& \text{intruder has } \mathcal{F}^{-1}\))
    return analyze(t);

In a nutshell, if the intruder has the corresponding encryption key or decryption key, then the intruder can compose and decompose pair term. For a detailed description of the capabilities of the intruder, the five inference rules were given in this chapter. In addition, the intruder has the action of eavesdropping and injection. As stated in the eavesdrop rule, the intruder can learn the message during transmission. The inject rule states that the injection of any message inferable from the intruder knowledge into the input buffer. The operational semantics of eavesdrop rule and inject rule have been formally described.

3.4. Formalization of Sender Anonymity. Anonymity of an entity means that the entity is not identifiable within a set of entities, anonymity is a high-level security property, and there are various notions of user anonymity. According to whether anonymous communication protocol provides anonymous protection for sender, receiver, and send-receive relationship, anonymity can be divided into sender anonymity, recipient anonymity, and relationship anonymity. Sender anonymity means that the intruder cannot sufficiently identify the sender in a set of potential senders, recipient anonymity means that the intruder cannot sufficiently identify the recipient in a set of potential recipients, and relationship anonymity means that the intruder cannot sufficiently identify that a sender (in a set of potential senders) and a recipient (in a set of potential recipients) are communicating.

The anonymity to be analyzed in this paper refers to the sender anonymity, our proposed framework only analyzes the sender anonymity of the anonymous communication protocols, and the formal definition of sender anonymity is
introduced in this framework. We formalize anonymity from the attacker’s perspective; if it receives an encrypted message without its decryption key, then the message looks like a random bit stream, which is indistinguishable from other random bit streams. Based on this idea, first of all, the concept of mapping equivalence was proposed, which reveals the ability of entities to distinguish between two messages. Then, the message equivalence and trace equivalence were defined, which means the sequences of events match each other in two traces. Meanwhile, every message in events is equivalent to the message the corresponding event has. Finally, based on the above defined concepts, we formally define the sender anonymity; that is, for all entities in the anonymity set, if any receiver satisfy trace equivalence, we say that the protocol satisfies the sender anonymity.

Definition 12 (Mapping). If an agent’s understanding of two messages is consistent, that is, the result of parsing two messages by agent X is equivalent, then two messages are considered to be parsing equivalent for the agent X. This behavior is called mapping, which is represented by predicate $\Omega$. If two messages look similar in the view of the intruder, this is called mapping equivalence. Some mapping rules are as follows:

- **R1**: $\Omega(m) = \Omega(m')$
- **R2**: $\Omega([t_1]k_1) = \Omega([t_2]k_2)$
- **R3**: $\Omega(t_1, t_2) = \Omega(t_1', t_2')$

R1 indicates that when $m$ and $m'$ are plaintext messages, such as random number, key, agent identity, and so on, if $m$ and $m'$ are two plaintext messages of the same type, the agent cannot distinguish them.

R2 indicates that when $m$ and $m'$ are ciphertext messages, $m = [t_1]k_1$, $m' = [t_2]k_2$, if the agent does not know $k_1$ and $k_2$, or cannot distinguish $t_1$ and $t_2$, then $m$ and $m'$ are not encrypted information constructed by the agent itself, so $m$ and $m'$ cannot be distinguished.

R3 indicates that when $m$ and $m'$ are composite messages, $m = [t_1, t_2]$, $m' = [t_1', t_2']$, if the agent cannot distinguish $t_1$ and $t_1'$, $t_2$ and $t_2'$, then $m$ and $m'$ cannot be distinguished.

Definition 13 (Message equivalence). If messages $m$ and $m'$ satisfied one of the mapping rules, $m$ and $m'$ cannot be distinguished by agent X, which is denoted as $m \approx_{\text{AgentX}} m'$.

Definition 14 (Extremum). It is used to describe the starting point and terminal point of a protocol trace. For an anonymous communication protocol, there are some messages used to hide the real communication agent (sender and receiver) in the protocol, so the extremum is used to describe the real sender and receiver. For example, a protocol trace $Tr$ contains sender S and receiver R, which can be denoted as $\mathcal{E}(Tr) = <S, R>$.

Definition 15 (Trace equivalence). Supposing $\hat{X}$ and $\hat{Y}$ are different execution instances of the same role in the protocol, and $Tr_X$ and $Tr_Y$ are corresponding protocol trace, $Tr_X^i$ denotes the $i$th event in protocol trace $Tr_X$, and predicate $Msg(Tr_X^i)$ denotes the message in event $Tr_X^i$. If the following three conditions are satisfied, we call the trace equivalence is true. (1) The sender $S$ and receiver $R$ of trace $Tr_X$ and $Tr_Y$ are the same. (2) The event sequence in $Tr_X$ and $Tr_Y$ is matching. (3) The message in event $Tr_X^i$ and $Tr_Y^i$ is equivalence. The formal expression of the above conditions is as follows:
\[ \mathcal{H}(Tr_X) = \mathcal{H}(Tr_Y) \land \text{Match}(Tr_X) \]
\[ = \text{Match}(Tr_Y) \land Tr_X \cdot \text{Msg}(Tr_X^i) = \text{Agent} \cdot Tr_Y \cdot \text{Msg}(Tr_Y^i) \]
\[ \iff Tr_X = \text{Agent} \cdot Tr_Y \]

**Definition 16 (Sender anonymity).** For any two agents \( \hat{X} \) and \( \hat{Y} \) in the sender anonymous set \( \mathcal{U} \), if the receiver agent satisfies the \( Tr_X \approx \text{Agent} \cdot Tr_Y \), we consider that the sender anonymity is satisfied.

Although anonymity and untraceability are similar to some extent and can hide the identity information of the entity, there are still differences between them.

Anonymity means that there is no way to identify a user uniquely from any other individual. The protocol preserves anonymity to the entities against the attacker if this attacker cannot distinguish the messages that come from these entities. From the view of the attacker, if it receives an encrypted message for which it does not have the decryption key, this message just looks like a random bit string. Anonymity is defined as the unrecognizability of a communication entity in an anonymous set with the same characteristics. The set of all possible entities in the system is an anonymous set that cannot be distinguished by attacker, which is the basis of anonymity. As shown in Figure 2(a), the users in the anonymous system are \{A, B, C, X, Y, Z\}, where comprised users C and Z are controlled by the attacker, the sender anonymous set is \{A, B\}, and the receiver anonymous set is \{X, Y\}. The attacker cannot distinguish whether the message \( m \) is sent by user A or B, and whether the message \( m \) is received by user X or Y. Therefore, the communication system realizes anonymity.

Untraceability means that nobody is able to trace back user actions to gain any information even related to user pseudonym or real identity. As shown in Figure 2(b), for a sender anonymous set \{A, B\}, the attacker cannot judge whether the two messages are sent by the same sender, and the probability that the message is sent by the same sender is 1/2. Assuming that two messages are sent by different senders, the attacker cannot associate the sender and receiver of a specific message; that is, he cannot distinguish between \( \{A \rightarrow Y, B \rightarrow X\} \) and \( \{A \rightarrow Y, B \rightarrow X\} \).

**4. Case Study**

To illustrate the applicability of our framework, we analyze the sender anonymity of Crowds protocol. Firstly, we give a brief introduction of the Crowds protocol; then, the construction of formal model for Crowds protocol and formal verification based on our framework are presented in detail. This case study shows how our framework is capable of verifying the sender anonymity of anonymous communication protocols.

**4.1. The Overview of Crowds Protocol.** The Crowds protocol proposed by Reiter and Rubin aims to ensure anonymous Web browsing by routing communication randomly with a group of similar users [29]. The sender’s identity can be hidden and not be detected by intruders even if he acts as a global eavesdropper, who can monitor and analyze the information on the channel to identify the real sender of the message. The Crowds protocol works in the following way:

1. The sender selects a Crowds member randomly and forwards the message which is encrypted by the corresponding pairwise key to the selected node. (2) The selected node flips a biased coin. We introduce a system parameter \( P_0 \) which represents the forwarding probability of the selected node. With probability \( P_0 \), it selects a Crowds member randomly (possible itself) as the next node in the path and forwards the message to it, encrypted with the appropriate pairwise key. With probability \( 1-P_0 \), it delivers the message directly to the destination. The next node repeats the above steps again.

The working principle of the Crowds protocol can be illustrated in Figure 3.

**4.2. Formal Verification of Crowds Protocol.** In this section, based on our proposed framework, the protocol is modeled as the asynchronous composition of a set of named communicating processes, which model the honest principals and the intruder. The PRISM model checker is used to verify the formal model with different system parameters. PRISM is a probabilistic model checker, a formal verification software tool for the modeling and analysis of systems that exhibit probabilistic behavior [30]. This in turn allows quantitative statements to be made about the system’s behavior, expressed as probabilities. We observe the state space of the protocol’s model under different number of nodes and paths originated from the same sender; detailed information is shown in Table 2. We run our experiments on a PC equipped with an intel i5-10201U 2.11Ghz processor and 8 GB RAM running Ubuntu 18.04 operating system; the probabilistic model checker tool is PRISM 4.6.

From Table 2, we conclude that with the growth of the number of nodes and paths, the required state space of the system increases sharply. If it continues to increase, the model cannot be verified. The model checking is facing a serious problem of state explosion. It is the future work to apply some effective optimization strategies to reduce the state space of the model.

Next, we discuss the sender anonymity of Crowds protocol. We assume that the intruder can associate several paths from the same sender. In the intruder’s view, if a node looks like the sender than other nodes, he has reason to think that the node is the real initiator of the protocol. We want to analyze the probability of the intruder guessing the sender. We use \( k_t \) to represent the number of times a node is observed by an intruder. According to the formal definition of anonymity in the framework, if observation time of a node is more than that of other nodes, then we think that the node is...
Figure 3: The working principle of the Crowds protocol.

Table 2: The change of state space with different node number and path number.

| Crowds protocol  | States | Transitions | The path number |
|------------------|--------|-------------|-----------------|
| 10 honest nodes  | 11437  | 22087       | 213331          |
| 15 honest nodes  | 32767  | 76417       | 1101091         |
| 20 honest nodes  | 70997  | 193597      | 584329          |
most likely to be the initiator of the protocol. We define the following event:

\[ E_{det} = K_0 > K_j, \quad \forall j \neq 0. \]  

(13)

What we need to do is to analyze the probability of the event, which is expressed by a PCTL formula as follows:

\[ P(E_{det}) = [\text{true} \ U \ runNumber = 0 \& \text{observe}[0] > \text{observe}[1] \& \ldots \& \text{observe}[0] > \text{observe}[N]] \]

In the previous chapter, we mentioned that two global parameters affect sender anonymity: the number of nodes \( N \) and the number of bad nodes \( C \) in Crowds protocol. We let the number of nodes increase, but at the same time ensure that the proportion of bad nodes \( C/N \) is fixed; the forwarding probability is fixed at 0.8, and we observe how the sender anonymity changes; the verification result is shown in Figure 4.

From Figure 4, we can draw two conclusions: (1) With the increase of the number of nodes, when the proportion of bad nodes remains at 1/6, the probability of the sender being observed decreases, which indicates that the sender anonymity of Crowds protocol increases with the increase of the system scale. Therefore, increasing the number of nodes Crowds protocol is an effective way to improve sender anonymity. (2) As the number of paths from the same sender observed by intruder increases, the sender anonymity of Crowds protocol decreases. Therefore, we can reduce the number of paths to improve sender anonymity.

In addition, we continue to analyze the changes of sender anonymity under different forwarding probability \( P_f \). Assuming that the initial value of \( P_f \) is 0.5, the relationship between sender anonymity and forwarding probability is analyzed by gradually increasing the value of \( P_f \) to 0.5. The verification result is shown in Figure 5.

From Figure 5, we can conclude that with the increase of forwarding probability \( P_f \), the probability of the sender being observed decreases; that is to say, the sender anonymity is improved. This is because the increase of the forwarding probability \( P_f \) means the increase of the path length, and the sender can better hide his own identity information, thus improving the sender anonymity.

5. Comparison with Other Methods

Compared with other frameworks, such as logic reasoning, process algebra, model checking, theorem proving, and UC framework, the framework proposed in this paper has the following advantages.

Logic reasoning grows out of a desire to form a mathematical model of reasoning, which could be used to express the logical relationship between stated concepts, and to generate new “true” statements by the application of rules to existing statements. Propositions are proved using these rules, from facts that are already known and basic axioms that are assumed to be true. In a word, the main advantage of logic reasoning is that even fairly complex information hiding properties can be stated directly as formulas in the logic. However, logic reasoning depends on a high-level abstraction of a security protocol and may allow flaws that exist in the lower-level protocol implementation to pass undetected. In addition, logic reasoning relies on manual proof, which is nontrivial and requires expert knowledge of the chosen logic. The framework proposed in this paper does not require a high-level abstraction of a security protocol; based on our proposed framework, we only need to use formal language to model the anonymous communication protocol and the anonymity. By inputting the model and properties into the model checker PRISM, we can automatically verify whether the anonymity of an anonymous communication protocol is satisfied, which avoids the error-prone defects caused by manual proof in the logical reasoning method.

Process algebra provides a mathematical notation for describing communicating processes; it models the system in terms of processes, which can synchronize and interact with the environment. Besides, it provides several semantic models to analyse the behaviour of processes and systems. Process algebra has strict syntax and semantics and can use verification tools such as FDR to automatically verify anonymity. However, process algebra does not consider an
active attacker with the ability to inject or block messages. The formal description of intruder model is given in our framework; the proposed intruder model is based on four assumptions and is weaker than Dolev–Yao intruder model, which is determined by the characteristic of anonymity. Moreover, two rules (eavesdrop rule and inject rule) for the growth of intruders knowledge set with the execution of protocol are given.

The framework we proposed in this paper belongs to the category of model checking; the inherent state explosion problem of model detection is inevitable, which is a challenge we have to face. The formalization of anonymous communication protocols, intruder model, and sender anonymity were proposed in our framework. Based on this framework, we can build a formal model of anonymous communication protocol and analyse whether the sender anonymity of the protocol is satisfied.

Theorem proving is a very useful approach when dealing with general systems of infinite state spaces. The formal specification of anonymous communication protocols and security properties are submitted to the specific formal system, such as strand space and protocol composition logic, then whether the security properties on the formal model are satisfied is proved. The proof of security properties mostly depends on manual proof; although there are some general-purpose theorem provers, such as Isabelle/HOL and Coq, which can build a mechanical proof method for the analysis of anonymity protocols, it is still a cumbersome and time-consuming work. In addition, how to construct a soundness and completeness protocol proof system is a problem that must be solved. Based on our proposed framework, when verifying anonymous communication protocols, we only need to input the protocol model and property specification into the tool to get the verification results. Although the problem of state explosion will be encountered in the verification process, it is acceptable to a certain extent. And we do not need to spend too much time and energy on constructing a protocol proof system.

The UC framework is one of the computational complexity models, which can be used to analyse and ensure the compositional security of cryptographic protocols. The security of a protocol is based on the probability and computational complexity of the adversary successful attack, the proof of security protocols must be completed manually. In our proposed framework, the message is expressed by formal expression; cryptographic operation is an abstract function in symbolic expression space. The security of a protocol is modeled by formal expression. The proof of security protocols are based on symbolic reasoning, which is easy to automate.

6. Conclusion

The main contribution of this paper is to provide a framework for formal analysis anonymous communication protocols. This framework gives some formal system notations for security protocols, and then we define operational semantics for security protocols using a labelled transition system; the transition relation is defined by the transition rules including create rule, send rule, and receive rule. In addition, the formal description of intruder model is given; the ability of intruder and the rule for the growth of intruder knowledge set with protocol execution are described in detail. Finally, the formal definition of sender anonymity is given. To illustrate the applicability of our framework, we analyzed the sender anonymity of Crowds protocol. We reveal the relationship between sender anonymity and the number of nodes in protocol, path reformations, and forwarding probability, which provides a good way about how to protect the sender anonymity of anonymous communication protocols.

There are several extensions of our present work. An obvious one is applying some effective optimization strategies to reduce the state space of the model. What is more, how to extend the framework and integrate the properties of receiver anonymity, untraceability, and relationship anonymity into the framework is a worthy research topic.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants nos. 61562026and61962020); Academic and Technical Leaders of Major Disciplines in Jiangxi Province (Grant no. 20172BCB22015); and Special Fund Project for Postgraduate Innovation in Jiangxi Province (Grant no. YC2020-B1141).

References

[1] L. M. Wang, X. L. Ni, and H. Zhao, “Survey of network-layer anonymous communication protocols,” Chinese Journal of Network and Information Security, vol. 4, no. 1, pp. 11–26, 2020.
[2] H. Zhao, L. M. Wang, T. H. Shen, H. Lei, and X. L. Ni, “Survey on anonymity metrics in communication network,” Journal of Software, vol. 32, no. 1, pp. 218–245, 2021.
[3] F. Shirazi, M. Simeonovski, M. R. Asghar, M. Backes, and C. Diaz, “A survey on routing in anonymous communication protocols,” ACM Computing Surveys, vol. 51, no. 3, pp. 1–39, 2019.
[4] D. Wang and P. Wang, “Two birds with one stone: two-factor authentication with security beyond conventional bound,” IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 4, p. 1, 2016.
[5] D. Wang and P. Wang, “On the anonymity of two-factor authentication schemes for wireless sensor networks: attacks, principle and solutions,” Computer Networks, vol. 73, pp. 41–57, 2014.
[6] X. Y. Zhang, X. F. Chen, J. K. Liu, and Y. Xiang, “DeepPAR and DeepDPA: privacy-preserving and asynchronous deep
learning for Industrial IoT,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2081–2090, 2020.

[7] M. Bugliesi, S. Calzavara, and R. Focardi, “Formal methods for web security,” *Journal of Logical and Algebraic Methods in Programming*, vol. 87, no. 1, pp. 110–126, 2017.

[8] F. D. Garcia, I. Hasuo, W. Pieters, and P. Rossum, “Provable anonymity,” in *Proceedings of the 2005 ACM Workshop on Formal Methods in Security Engineering*, pp. 63–72, Alexandria, Virginia, USA, November 2005.

[9] N. Gelernter and A. Herzberg, “On the limits of provable anonymity,” in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society* (WPES’13), pp. 225–236, New York, NY, USA, 2013.

[10] M. Murat, J. Heather, and S. Schneider, “Verifying anonymity in voting systems using CSP,” *Formal Aspects of Computing*, vol. 26, no. 1, pp. 63–98, 2012.

[11] R. Chadha, V. Cheval, C. Ciobaca, and S. Kremer, “Automated verification of equivalence properties of cryptographic protocols,” *ACM Transactions on Computational Logic*, vol. 17, no. 4, pp. 1–32, 2016.

[12] V. Cortier, C. Dragan, F. Dupressoir, B. Schmidt, and P.-Y. Strub, “Machine-checked proofs of privacy for electronic voting protocols,” in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 993–1008, San Jose, CA, USA, May 2017.

[13] B. Gergei, R. Chadha, and A. K. Eeralla, *Formal Analysis of Vote Privacy Using Computationally Complete Symbolic Attackers*, IACR Cryptol. ePrint Arch, 2018.

[14] T. Chothia, S. Orzan, J. Pang, and M. T. Dashti, “A framework for automatically checking anonymity with µCRL,” in *Proceedings of the 2nd International Conference on Trustworthy Global Computing* (TGC’06), pp. 301–318, Springer-Verlag, Berlin, Heidelberg, November 2006.

[15] V. Shmatikov, “Probabilistic analysis of an anonymity system,” *Journal of Computer Security*, vol. 12, no. 3-4, pp. 355–377, 2004.

[16] A. Américo, A. Vaz, M. S. Alvim, S. V. A. Campos, and A. Mclver, “Formal analysis of the information leakage of the DC-Nets and Crowds anonymity protocols,” in *Proceedings of the Brazilian Symposium on Formal Methods*, pp. 142–158, Recife, Brazil, November, 2017.

[17] Y. J. Li and J. Pang, “Formalizing provable anonymity in Isabelle/HOL,” *Formal Aspects of Computing*, vol. 27, no. 2, pp. 255–282, 2015.

[18] T. Mhamdi, O. Hasam, and S. Tahar, “Evaluation of anonymity and confidentiality protocols using theorem proving,” *Formal Methods in System Design*, vol. 47, no. 3, pp. 265–286, 2015.

[19] Y. J. Li and J. Pang, “A strand space approach to provable anonymity,” in *Proceedings of the 2nd International Workshop of Formal Techniques for Safety-Critical Systems*, FTSCS 2013, vol. 419, pp. 71–87, New Zealand, October 2013.

[20] Z. Lu and J. Z. Luo, “Formal analysis of anonymity based on strand space model,” in *Proceedings of the First IEEE International Conference on Ubi-media Computing*, pp. 75–81, Lanzhou, China, July 2008.

[21] X. W. Dong, W. S. Niu, and J. F. Ma, “Improvement of the anonymity formalization based on strand space model,” *Journal on Communications*, vol. 32, no. 6, pp. 124–131, 2011.

[22] T. Feng, S. N. Han, X. Guo, and D. L. Ma, “A new method of formalizing anonymity based on protocol composition logic,” *Security and Communication Networks*, vol. 8, no. 6, pp. 1132–1140, 2015.

[23] J. Feigenbaum, A. Johnson, and P. Syverson, “Probabilistic analysis of onion routing in a black-box model,” *ACM Transactions on Information and System Security*, vol. 15, no. 3, pp. 1–28, 2012.

[24] M. Backes, P. Manoharan, and E. Mohammadi, “TUC: time-sensitive and modular analysis of anonymous communications,” in *Proceedings of the 27th IEEE Computer Security Foundations Symposium*, pp. 383–397, Vienna, Austria, July 2014.

[25] D. Wang, X. Z. Zhang, Z. J. Zhang, and P. Wang, “Understanding security failures of multi-factor authentication schemes for multi-server environments,” *Computers & Security*, vol. 88, Article ID 101619, 2020.

[26] S. Jarecki, H. Krawczyk, and J. Y. Xu, “OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks,” in *Proceedings of the 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques EUROCRYPT*, May 2018.

[27] Q. Wang, D. Wang, C. Cheng, and D. He, “Quantum2FA: efficient quantum-resistant two-factor authentication scheme for mobile devices,” *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2022.

[28] G. C. Zhao, Q. Jiang, X. H. Huang, X. Ma, Y. Tian, and J. Ma, “Secure and useable handshake based pairing for wrist-worn smart devices on different users,” *Mobile Networks and Applications*, vol. 26, no. 6, pp. 2407–2422, 2021.

[29] M. K. Reiter and A. D. Rubin, “Crowds: anonymity for Web transactions,” *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[30] M. Kwiatkowska, G. Norman, and D. Parker, “Prism 4.0: verification of probabilistic Real-time systems,” in *Proceedings of the Computer Aided Verification*, vol. 6806, pp. 585–591, Snowbird, UT, USA, July 2011.