Behavioral Equivalence of Security-Oriented Interactive Systems

Guanjun LIU†, Member and Changjun JIANG††, Nonmember

SUMMARY In the classical computation theory, the language of a system features the computational behavior of the system but it does not distinguish the determinism and nondeterminism of actions. However, Milner found that the determinism and nondeterminism affect the interactional behavior of interactive systems and thus the notion of language does not features the interactional behavior. Therefore, Milner proposed the notion of (weak) bisimulation to solve this problem. With the development of internet, more and more interactive systems occur in the world, such as electronic trading system [4]. Due to the openness of the internet, more and more interactive systems occur in the world, such as electronic trading system. Security is one of the most important topics for these systems. We find that different security policies can also affect the interactional behavior of a system, which exactly is the reason why a good policy can strengthen the security. In other words, two interactive systems with different security policies are not of an equivalent behavior although their functions (or business processes) are identical. However, the classic (weak) bisimulation theory draws an opposite conclusion that their behaviors are equivalent. The notion of (weak) bisimulation is not suitable for these security-oriented interactive systems since it does not consider a security policy. This paper proposes the concept of secure bisimulation in order to solve the above problem.

key words: interactive systems, labelled petri nets, labelled transition systems, bisimulation, security

1. Introduction

With the development of the internet, an increasing number of distributed/concurrent systems have occurred, e.g., electronic trading system [4]. Due to the openness of the internet and the flaws/bugs of these systems, they are often attacked by Trojans or Hackers [11]. Therefore, security is one of the most important topics for most of these systems. For the security of these systems, there are many measures to take, such as data encryption and access control. These measures play an irre replaceable role in safeguarding data/information privacy [18].

Besides those classic measures, some other policies are also taken in order to increase the security of a system when the system is designed and developed. For example, when a user transfers money via an internet-bank system, the bank will sends a verification message, and the terminal device of receiving the message is generally separated from the terminal device by which the user logs in the system. These policies play an important role in preventing online fraud/hacking. In this paper, these policies are called behavior security policies in order to distinguish (in presentation) from those classical security measures (e.g., encryption). Later we will see that these policies can affect the interacting behavior between a system and its users, which is exactly the reason why we call them as behavior security policies. This paper focuses on behavior security policies.

As we all know, those classic security measures (such as encryption) have strict mathematical theories (such as cryptography) to support them. To the best of our knowledge, however, there has not been a mathematical method to support these behavior security policies. In other words, how to mathematically/formally declare that one behavior security policy can make a system to be securer than another one? and how to mathematically/formally define the (non-)equivalence of two systems that have the same behavior/function but take different behavior security policies? When we try to use the notions of language-equivalence and bisimulation-equivalence [7], [10] to answer these questions, we find them unsuitable.

An interactive system has some interfaces used to interact with its user. When the system is in some states of using some interfaces, some external and unobservable attacks (e.g., tampering with some data) can happen. Just because the user/system cannot observe/sense these attacks, their interacting behavior is affected. Therefore, we partition the states of an interactive system into two parts: secure ones and insecure ones. Although a good security policy cannot avoid all attacks in insecure states, it can prevent some key attacks and thus effectively avoid a negative outcome since a negative outcome generally needs multiple steps of attacks. Latter we will see some examples.

In this paper, we propose the concept of secure bisimulation for security-oriented interactive systems. We first use a labelled Petri nets to model an interactive system. Next, we partition its states into two parts: one is secure and another is insecure. Furthermore, over the set of insecure states, we create a propositional formula to characterize a security policy. Finally, we formally define the behavioral equivalence of two interactive systems.

The remainder of this paper is organized as follows: Section 2 uses two motivated examples to show the problem we find. Section 3 recalls the concepts of bisimulation and weak bisimulation. Section 4 defines secure bisimulation over labelled Petri nets. Section 5 concludes this paper.
2. Motivated Examples

A central question focused on by the automata theory is: when do two automata have equivalent behavior? The theory defines that two automata are equivalent if they recognize the same language [7]. Later, Milner found that this classical notion of equivalence is not suitable for some applications. That is to say, two automata have equivalent behavior according to the classical notion of equivalence, but that is not the fact. The famous example given by him is about a tea/coffee vending machine. He thought that those classic automata models like Turing machines, register machines, and the lambda calculus are concerned with computational behavior but not interactional behavior. A basic action of interactive systems is to communicate across an interface with a handshake. Determinism or nondeterminism of handshake protocol affects a system’s behavior while the classical notion of equivalence of automaton does not consider the difference between determinism and nondeterminism. Therefore, he put forward the notion of bisimulation [9], [10]. Roughly speaking, two states are bisimilar if they have the same outgoing actions and the corresponding succeeding states are also bisimilar; two automata are bisimilar iff their starting states are bisimilar. If two automata are bisimilar, then they recognize the same language, but not vice versa. It must be mentioned that Milner also defined weak bisimulation (also called observational equivalence) that is a generalization of bisimulation. Weak bisimulation is used to reflect that two systems with different internal structure (and hence with different internal behavior) have the same external behavior, i.e., external observers consider them to be equivalent.

Just as we said in Sect. 1, a number of interactive systems have occurred with the development of the internet and security must be considered by most of them. When we try to apply bisimulation to these security-oriented interactive systems, we find that this notion is not suitable for some requirements. That is to say, two such systems are thought of as being (observationally) equivalent by the notion of (weak) bisimulation, but that is not the fact. We further find that different behavior security policies can affect a system’s behavior, which is also the reason why a good behavior security policy can enhance the security of the system. Now we consider the following two examples.

2.1 Example 1

Figures 1 (a) and (b) show an internet-bank transfer system.

The two sketched systems both implement the same functions described as follows:
First, a user logs in her/his internet-bank account by using her/his PC. This login process is modeled by transitions $t_{1,1}$–$t_{1,4}$ in Fig. 1 (a) or $t_{2,1}$–$t_{2,4}$ in (b). Then the user submits a request of bank transfer (transition $t_{1,5}$ in (a), or $t_{2,5}$ in (b)). The submitted information is packaged in the PC and then is transported from the PC to the bank system. The process of packaging the information is modelled by place $p_{1,9}$ in (a) or $p_{2,9}$ in (b). Firing transition $t_{1,6}$ in (a) or $t_{2,6}$ in (b) represents that the packaged information is sent from the PC to the bank system. After receiving the request ($t_{1,7}$ in (a) or $t_{2,7}$ in (b)), the bank system checks if the request is legal. If it is illegal, the bank system sends ($t_{1,8}$ in (a) or $t_{2,8}$ in (b)) a message of canceling this request. If it is legal, then the bank system sends ($t_{1,9}$ in (a) or $t_{2,9}$ in (b)) a verification message. When the user receives ($t_{1,10}$ in (a) or $t_{2,10}$ in (b)) the message of canceling the request, (s)he returns back in order to refill a request. When the user receives ($t_{1,11}$ in (a) or $t_{2,11}$ in (b)) the verification message, (s)he will decide if (s)he agrees this transfer or not. Place $p_{1,16}$ in (a) or $p_{2,16}$ in (b) represents the (local) state which means that the PC has received the verification message but not yet shown to the user. Firing transition $t_{1,12}$ in (a) or $t_{2,12}$ in (b) means that the message is shown to the user. Transition $t_{1,13}$ in (a) or $t_{2,13}$ in (b) means that the user disagrees this transfer, while transition $t_{1,14}$ in (a) or $t_{2,14}$ in (b) means agreement.

The two systems are identical in functions/behaviour. The distinction between them is that the system in Fig. 1 (a) sends the verification message to the PC itself while the one in (b) sends the verification message to another independent device (e.g., the user’s cell phone). In other words, Fig. 1 (a) means that the devices of logging in the bank transfer system and receiving the verification message are the same but (b) means that the two devices are separated. The bisimulation equivalence theory also asserts that the two systems are equivalent. However, are they actually equivalent? The answer is NO. In fact banks usually use the system in (b) rather than the one in (a) because they believe that the former is much secure than the latter. Now we demonstrate that they have no equivalent behavior.

A Trojan program is implanted into the user’s PC. The Trojan program can tamper with the data in her/his transfer request during the period of packaging the request information. The Trojan can also tamper with the verification message before the message is shown to the user. After the user submits a request of transferring money from Account 1 to Account 2, the Trojan changes this request into the one of transferring money from Account 1 to Account 3. Thus, the tampered request is sent to the bank system. The bank system finds that Accounts 1 and 3 are both legal and then sends a verification message in which the bank tells the user that money will be transferred into Account 3 and gives the user a verification code. After the PC receives the verification information, the Trojan changes Account 3 in this verification message into Account 2 and then shows the modified message to the user. Finally, the user finds that all are cor-
Fig. 1 Two LPNs modeling a bank transfer system: (a) logging in the system and receiving the verification information are operated in the same terminal device, or (b) in two different ones.

rect and then inputs the verification code in order to agree on this transfer. Unfortunately, the money is actually transferred into Account 3 instead of Account 2. If banks use the system in Fig. 1 (b), then it is almost impossible that a Trojan program is simultaneously implanted into both the PC and the cell phone of a user. Therefore, the user can find problem and then terminal this transfer in advance. Therefore, the two systems are not equivalent: they produce different results for the same input. Different behavior security policies can yield different interacting behavior between the bank and its user, which is also the reason why the policy in Fig. 1 (b) can increase the security. Therefore, the bisimulation theory is not suitable for this case.

2.2 Example 2

When some smartphone systems are accessed, it needs to verify not only the input password but also the user’s behavioral features of inputting the password (such as the pressure of clicking the keyboard, the writing trajectory, and so on) [14], [17], [19]. This kind of policy of double verifications is securer than that of single verification. Figure 2 (a) shows the process of verifying the user’s identity by the single verification policy. At the initial state, a user inputs a password (this action is represented by $i\text{ppw}$). Then, the system verifies if the password is legal or not. If the password is illegal (this unobservable action is represented by $i\hat{\text{lgpw}}$), then the system returns to the initial state ($r$ represents this action). If the password is legal ($\hat{\text{lgpw}}$), then the user may access the system ($\text{ac}$). Figure 2 (b) shows the process of executing the double verifications. When the password is legal, the system continues to verify whether the captured behavioral feature agrees with the legal user’s. If they are unanimous ($\hat{\text{lfgf}}$), then the system is opened ($\text{ac}$), or else ($\hat{\text{ilgf}}$), it also returns to the initial state. Notice that an action with a hat (e.g., $\hat{\text{lfgf}}$) means that it is unobservable, i.e., it is equal to $\epsilon$ in [10]. The unobservable action $\hat{\text{clf}}$ in Fig. 2 (b) means that the system captures the user’s behavioral features.

The two systems are observationally equivalent (weakly bisimilar) according to the weak bisimulation theory [10]. However, even from the aspect of the observers (e.g., a legal user and an illegal user), the two systems are not observationally equivalent because 1) for the system in Fig. 2 (b), the legal user can log in it after (s)he inputs the correct password, but the illegal one cannot log in it after (s)he inputs the correct password; and 2) for the system
in Fig. 2(a), the legal user and the illegal one both log in it if they input the correct password, but neither of them can log in if they input an incorrect password. Therefore, observers do not observe the same behaviors for the same inputs. Therefore, the weak bisimulation theory is not suitable for this case. Obviously, a behavior security policy can affect the interacting behavior between a system and its user. By our common sense, we know that the policy used in Fig. 2(b) leads to a secure system than that used in Fig. 2(a).

In what follows, we formally define the behavioral (non-)equivalence for these security-oriented interactive systems on the base of (weak) bisimulation.

3. Bisimulation and Weak Bisimulation

We first review labelled transition system (LTS), bisimulation, and weak bisimulation\[10\] in order to understand our main work. Bisimulation and weak bisimulation can be defined over LTS.

Definition 1 (Labelled transition system): An LTS is a 3-tuple $\Gamma = (Q, Act, Tr)$, where

1. $Q$ is a set of states;
2. $Act$ is a set of actions including the unobservable action $\epsilon$; and
3. $Tr \subseteq (Q \times Act \times Q)$ is a set of transitions.

In general, transition $(q, a, q')$ is denoted as $q \xrightarrow{a} q'$. It is denoted as $q \xrightarrow{} q'$ for short if $a = \epsilon$. An LTS may be depicted by a digraph in which a state is represented by a circle/ellipse and a transition is represented by an arc. For instance, Figs. 3 and 4 show four LTSs (Notice: an action with a cap means that it is unobservable).

Definition 2 (Bisimulation): Let $(Q_i, Act_i, Tr_i)$ be two LTSs such that there is no transition with $\epsilon$, where $i \in \{1, 2\}$. $B \subseteq (Q_1 \times Q_2) \cup (Q_2 \times Q_1)$ is a bisimulation if

1. $B$ is a symmetric binary relation; and
2. If $(q, r) \in B$ and $q \xrightarrow{a} q'$, then there exists $r'$ such that $r \xrightarrow{a} r'$ and $(q', r') \in B$.

Given two states $q$ and $r$ in $Q_1 \cup Q_2$, $q$ is bisimilar to $r$, written as $q \sim r$, if there is a bisimulation $B$ such that $(q, r) \in B$. Obviously, for the two LPNs in Fig. 3 the binary relation $\{(M_{1,1}, M_{2,1}), (M_{2,2}, M_{1,2})\}_{i = 0 \ldots 13}$ is a bisimulation. Therefore, $M_{1,0} \sim M_{2,0}$. In an LTS $(Q, Act, Tr)$, if $q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_n$, then it is denoted as $q_1 \xrightarrow{+} q_n$. We establish $\sim$ as reflexive, i.e., $q \sim q$. If $q_1 \xrightarrow{+} q_0 \xrightarrow{a} q_{n+1}$, then it is denoted as $q_1 \xrightarrow{a} q_{n+1}$.

Definition 3 (Weak bisimulation): Let $(Q_i, Act_i, Tr_i)$ be two LTSs, where $i \in \{1, 2\}$. $B \subseteq (Q_1 \times Q_2) \cup (Q_2 \times Q_1)$ is a weak bisimulation (or observational equivalence) if

1. $B$ is a symmetric binary relation; and
2. If $(q, r) \in B$ and $q \rightarrow q'$, then there exists $r'$ such that $r \xrightarrow{a} r'$ and $(q', r') \in B$; and
3. If $(q, r) \in B$ and $q \xrightarrow{a} q'$, then there exists $r'$ such that $r \xrightarrow{a} r'$ and $(q', r') \in B$.

Given two states $q$ and $r$ in $Q_1 \cup Q_2$, $q$ is weakly bisimilar to $r$, written as $q \xrightarrow{\approx} r$, if there is a weak bisimulation $B$ such that $(q, r) \in B$. It can be easily checked that $\{(M_{1,0},
Fig. 3 Two LTSs generated by LPNs in Figs. 1 (a) and (b), respectively.

Fig. 4 Two LTSs generated by LPNs in Figs. 2 (a) and (b), respectively.
Definition 4 (Labelled Petri net): An LPN is a 6-tuple \( \Sigma = (P, T, F, M_0, E, \lambda) \), where

1. \( P \) is a finite set of places and \( T \) is a finite set of transitions such that \( P \cap T = \emptyset \);
2. \( F \subseteq (P \times T) \cup (T \times P) \) is a set of arcs;
3. \( M_0: P \rightarrow \{0, 1, 2, \ldots\} \) is the initial marking;
4. \( E \) is a set of actions including the unobservable action \( e \); and
5. \( \lambda: T \rightarrow E \) is a label function.

Denote \( \preceq = \{ p \in P \mid (p, t) \in F \} \) and \( \preceq^* = \{ p \in P \mid (t, p) \in F \} \) as the pre- and post-sets of transition \( t \), respectively.

Let \( M: P \rightarrow \{0, 1, 2, \ldots\} \) be a marking of an LPN and \( t \) be a transition. Then, \( t \) is enabled at \( M \) if \( \forall p \in \preceq: M(p) > 0 \). This is denoted as \( M(t) \). Firing an enabled transition \( t \) at \( M \) yields a new marking \( M' \) such that \( \forall p \in P: M'(p) = M(p) - 1 \) if \( p \in \preceq \setminus \preceq^* \), \( M'(p) = M(p) + 1 \) if \( p \in \preceq^* \setminus \preceq \), or \( M'(p) = M(p) \) otherwise. This is denoted as \( M[\{t\}] \).

Definition 5 (Reachable marking set): The set \( R(M_0) \) of all reachable markings of an LPN \( \Sigma = (P, T, F, M_0, E, \lambda) \) is recursively defined as follows:

1. \( M_0 \in R(M_0) \).
2. For each \( M \in R(M_0) \), if \( t \in T \) such that \( M(t) > 0 \), then \( M' \in R(M_0) \).

For instance, Figs. 1 and 2 show four LPNs. An action with a cap in Fig. 2 means that it is unobservable. Based on the reachable marking set of an LPN, we can construct an LTS as the interleaving semantics of the LPN.

Definition 6 (LTS generated by LPN): LTS \( \Gamma = (Q, \text{Act}, T_r) \) is generated by LPN \( \Sigma = (P, T, F, M_0, E, \lambda) \) if

1. \( Q = R(M_0) \);
2. \( \text{Act} = E \); and
3. \( M \xrightarrow{a} M' \) in \( T_r \) if \( \exists t \in T: M(t)M' \land \lambda(t) = a \).

For example, Figs. 3 (a) and (b) are the LTSs generated by the LPNs in Figs. 1 (a) and (b), respectively. The following conclusion is obvious:

Proposition 1: The LTS generated by an LPN is unique.

An LPN can model the normal interaction between a system and its users. As shown in the previous examples, however, some unobservable/illegal events can affect these normal interactions since an interactive system always has some interfaces used to interact with its users and some unobservable/illegal events can happen in these interfaces. In other words, this is an inherent defect for interactive systems. Fortunately, some security policies can be taken such that some bad results can finally be avoided even though some unobservable/illegal events take place in some interfaces. However, a problem is that an LPN does not formally model the related behavior security policy. In this paper, we take the following two measures to solve this problem:

- We partition the set of places of an LPN into two parts: one is secure and another is insecure. For example, \( p_{1,9}, p_{1,16}, p_{2,9}, \) and \( p_{2,16} \) in Fig. 1 are insecure because the data in these places can be tampered with. For another example, \( p_{1,2} \) and \( p_{2,2} \) in Figs. 2 (a) and (b) are insecure since a legal user and an illegal one both can make the system to reach the related states. Therefore, an LPN can be denoted as \( \Sigma = (P_S \cup P_U, T, F, M_0, E, \lambda) \) where \( P_S \) is the set of secure places and \( P_U \) is the set of insecure places. This measure can reflect the fact that insecure states are the key points at which some unobservable events can happen to affect a normal behavior. However, it cannot characterize a security policy. Hence, the second measure is taken.

- We create a propositional formula over the set of insecure places to represent the behaviors prevented/avoided by a security policy. For example, the policy of the system in Fig. 1 (b), we use the propositional formula \( \neg(p_{2,9} \land p_{2,16}) \) to represent that this secure policy guarantees that the data in \( p_{2,9} \) and \( p_{2,16} \) cannot be tampered with by the same trojan program, while for the policy of the system in Fig. 1 (a) the propositional formula \( p_{1,9} \land p_{1,16} \) represents that the data in \( p_{1,9} \) and \( p_{1,16} \) can be tampered with by the same trojan program under this policy.

Based on the above two measures, we define a labelled Petri net with a security policy (LPNSP) as follows:

Definition 7 (Labelled Petri net with a security policy): An LPNSP is a 7-tuple \( \Sigma_v = (P_S \cup P_U, T, F, M_0, E, \lambda, \nu) \) such that

1. \( \Sigma = (P_S \cup P_U, T, F, M_0, E, \lambda) \) is an LPN such that...
\[ P_S \cap P_U = \emptyset; \text{ and} \]

2. \( \nu \) is a propositional formula over \( P_U \).

The above measures do not change the rules of enabling and firing a transition, but we must consider how to define the insecure/secure states and propositional formula for the LTS \( \Gamma = (Q, \Act, \Tr) \) generated by an LPNSP \( \Sigma_r = (P_S \cup P_U, T, F, M_0, E, \lambda, \nu) \).

- First, we divide the states of the generated LTS into two parts \( Q_S \) (the set of secure states) and \( Q_U \) (the set of insecure states) by the following rule: a state \( M \in Q \) is insecure if there exists a place \( p \in P_U \) such that \( M(p) > 0 \).

- Second, given an LPNSP \( \Sigma_r = (P_S \cup P_U, T, F, M_0, E, \lambda, \nu) \) and the LTS \( \Gamma = (Q_S \cup Q_U, \Act, \Tr) \) generated by \( \Sigma \), we yield a propositional formula for \( \Gamma \) by the following rule: for each \( p \in \nu \), if \( p \) occurs in \( M_1, \ldots, M_k \), then \( p \) in \( \nu \) is replaced by \( M_1 \wedge \cdots \wedge M_k \). Then the final formula is denoted as \( \mu \).

- Finally, we obtain a labelled transition system with a security policy (LTSSP, denoted as \( \Gamma_\mu = (Q_S \cup Q_U, \Act, \Tr, \mu) \)) that is generated by an LPNSP \( \Sigma_i = (P_S \cup P_U, T, F, M_0, E, \lambda, \nu) \) via the above two steps.

Now we consider the examples in Sects. 2 and 3.

Figure 1 (a) (resp. Fig. 1 (b)) is an LPNSP in which \( p_{1,9} \) and \( p_{1,16} \) (resp. \( p_{2,9} \) and \( p_{2,16} \)) are insecure places and the propositional formula over them is \( p_{1,9} \wedge p_{1,16} \) (resp. \( \neg(p_{2,9} \wedge p_{2,16}) \)). Figure 3 (a) (resp. Fig. 3 (b)) is the LTSSP generated by the LPNSP in Fig. 1 (a) (resp. Fig. 1 (b)), in which \( M_{1,5} \) and \( M_{1,10} \) (resp. \( M_{2,5} \) and \( M_{2,10} \)) are insecure states and the propositional formula is \( M_{1,5} \wedge M_{1,10} \) (resp. \( \neg(M_{2,5} \wedge M_{2,10}) \)).

Figure 2 (a) is an LPNSP in which \( p_{1,2} \) is insecure. We use the propositional formula \( p_{1,2} \) to represent this security policy. If the value of proposition \( p_{1,2} \) is TRUE, this means that this state is led to by the legal user, otherwise it is done by an illegal one. Similarly, Fig. 2 (b) is an LPNSP in which \( p_{2,2} \) is insecure and the propositional formula is \( p_{2,2} \). Therefore, for the LTS in Fig. 4 (a), \( M_{1,1} \) is the insecure state and the propositional formula is \( M_{1,1} \); for the LTS in Fig. 4 (b), \( M_{2,1} \) is the insecure states and the propositional formula is \( M_{2,1} \).

Obviously, the LTSSP generated by an LPNSP is still unique. In what follows, we define secure bisimulation based on LTSSP.

**Definition 8** (Secure bisimulation of two LTSSPs): Let \( \Gamma_{\mu_i} = (Q_{\Sigma_i}, Q_{U_i}, \Act_i, \Tr_i, \mu_i) \) be two LTSSPs, where \( i \in \{1, 2\} \). \( B \subseteq ((Q_{\Sigma_i} \cup Q_{U_i}) \times (Q_{\Sigma_i} \cup Q_{U_i})) \cup ((Q_{\Sigma_i} \cup Q_{U_i}) \times (Q_{\Sigma_i} \cup Q_{U_i})) \) is a secure bisimulation if

1. \( B \) is a weak bisimulation;
2. if \((q, r) \in B \) and \( q \in Q_U \), then \( r \in Q_U \); and
3. \( \mu_1 \leftrightarrow \mu_2 \) is a tautology after each state in \( \mu_1 \) is replaced by a bisimilar state in \( \mu_2 \).

Given two states \( q \) and \( r \) in \( Q_{\Sigma_i} \cup Q_{\Sigma_i} \cup Q_{U_i} \cup Q_{U_i} \), \( q \) is securely bisimilar to \( r \), written as \( q \overset{B}{\cong} r \), if there is a secure bisimulation \( B \) such that \((q, r) \in B \).

By secure bisimulation, we know that for the two LTSSPs in Fig. 3, there is no secure bisimulation such that \( M_{1,0} \overset{B}{\cong} M_{2,0} \). Although there is a bisimulation \( \{(M_{1,i}, M_{2,i}), (M_{2,j}, M_{1,j}) | i = 0 \ldots 13 \} \) satisfying the first two requirements of Definition 8, the propositional formula \( (M_{1,5} \wedge M_{1,10}) \leftrightarrow (\neg(M_{2,5} \wedge M_{2,10})) \) is not a tautology after propositional variable \( M_{1,5} \) (resp. \( M_{1,10} \)) is replaced by \( M_{2,5} \) (resp. \( M_{2,10} \)).

Similarly, for the two LTSSPs in Fig. 4, there is no secure bisimulation such that \( M_{1,0} \overset{B}{\cong} M_{2,0} \). Although there is a weak bisimulation (see the description following Definition 3) fulfilling the first and third requirements of Definition 8, it does not fulfill the second one, i.e., \( M_{1,1} \) is weakly bisimilar to \( M_{2,2} \), \( M_{1,1} \) is insecure, but \( M_{2,2} \) is secure.

**Definition 9** (Secure bisimulation of two LPNSPs): Let \( \Sigma_{\nu_i} = (P_S \cup P_U, T_i, F_i, M_0, E_i, \lambda_i, \nu_i), i \in \{1, 2\} \), be two LPNSPs, and \( \Gamma_{\mu_i} = (Q_{\Sigma_i} \cup Q_{U_i}, \Act_i, \Tr_i, \mu_i) \) be their LTSSPs. \( \Sigma_{\nu_1} \) is securely bisimilar to \( \Sigma_{\nu_2} \), denoted as \( \Sigma_{\nu_1} \overset{B}{\cong} \Sigma_{\nu_2} \), if \( M_0 \overset{B}{\cong} M_0 \) over LTSs \( \Gamma_{\mu_1} \) and \( \Gamma_{\mu_2} \).

Because \( M_0 \overset{B}{\cong} M_0 \) implies \( M_0 \overset{B}{\cong} M_0 \), we have that \( \Sigma_1 \overset{B}{\cong} \Sigma_2 \) implies \( \Sigma_2 \overset{B}{\cong} \Sigma_1 \), i.e., \( \overset{B}{\cong} \) satisfies the symmetry. Therefore, it is suitable that we use “bisimilar” in the above definition.

Here, secure bisimulation, instead of (weak) bisimulation, is used as the criterion of equivalence of two security-oriented interactive systems. By this criterion we know that the two systems in Fig. 1 and the two systems in Fig. 2 are both nonequivalent, but they are considered to be equivalent according to (weak) bisimulation. Therefore, the notion of secure bisimulation is suitable for security-oriented interactive systems.

Obviously, \( \overset{B}{\cong} \) is reflexive, symmetric and transitive. Therefore, we draw the following conclusion:

**Proposition 2:** \( \overset{B}{\cong} \) is an equivalence relation.

If two LTSSPs both have finite states, then the set of binary relations over these states is finite. That is to say, we can always decide whether there is a secure bisimulation for two finite LTSSPs. Unfortunately, the problem of deciding whether there is a secure bisimulation for any two infinite LTSSPs is undecidable since the problem of weak bisimulation is undecidable for infinite LTSs [6].

### 5. Conclusion

The language recognized by an automaton is the set of all action sequences (i.e., words) recognized by the automaton. The behavior equivalence is a means of specifying how a designed system behaves. In other words, the designed system is held to be correct if its actual behavior is equivalent to its specified behavior.

Starting with two real applications, this paper proposes the question on how to utilize the bisimulation theory to distinguish two systems that have the same functions/behaviors but different securities. Based on the classic bisimulation theory, we propose the concept of secure bisimulation that
can answer the above question. Compared with its previous version [8], this paper uses a propositional formula to reflect the impact on the interactional behavior made by a security policy. This paper encourages one to re-think of the behaviors of security-oriented interactive systems: how to enhance the security of a system by designing some policies that affect the behaviors between the system and its user/environment. One future work is to develop an algorithm to decide if two finite LTSSPs are securely bisimilar.

The (weak) bisimulation over LPNs defined in this paper is the same with the interleaving-bisimulation defined in [1]. The secure bisimulation is defined on the basis of this interleaving semantic. In fact, other kinds of definitions of bisimulations such as step-bisimulation and process-bisimulation were proposed for LPNs to reflect their behavior equivalency in more details [2], [5], [12], [13], [20]. Therefore, another future work may redefine the secure bisimulation on the basis of these complex ones and explore their properties.

More and more applications such as online shopping/payment are carried out through the mobile terminal devices [3], [17], [19]. For example, the devices of logging in an electronic bank and receiving the verification messages are both the user’s cell-phone. It is challenging how to ensure these systems’ security and credibility. Another future work is to consider the security policy for the systems running in the mobile terminal devices.

Acknowledgments

The authors would like to thank the referees of both the conference version and the current version of this paper for their constructive comments. This paper is supported in part by the National Natural Science Foundation of China (Grant Nos. 61572360 and 91218301) and in part by Shanghai Education Development Foundation and Shanghai Municipal Education Commission (Shuguang Program).

References


Guanjun Liu received the Ph.D. degree from Tongji University, Shanghai, China, in 2011. He was a postdoctoral research fellow at Singapore University of Technology and Design, Singapore from 2011 to 2013. From 2013 to 2014 the Alexander von Humboldt Foundation of Germany supported him to do research at Humboldt University of Berlin. He is currently an associate professor at Tongji University of China. He has published 40+ papers. His research interests include formal methods, model checking, Petri net theory, and workflow.

Changjun Jiang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Science, Beijing, China, in 1995. He is currently a professor in Donghua University, Shanghai, China. He is also the head of the Key Laboratory of Ministry of Education for Embedded Systems and Service computing, Tongji University, Shanghai, China. He has published 300+ papers. His current research interests include concurrency theory, Petri nets, formal verification of software, intelligent transportation systems, and service-oriented computing.