On-the-fly Model Checking of Fair Non-repudiation Protocols

Guoqiang Li and Mizuhito Ogawa

Japan Advanced Institute of Science and Technology
Asahidai, Nomi, Ishikawa, 923-1292 Japan
{guoqiang, mizuhito}@jaist.ac.jp

Abstract. A fair non-repudiation protocol should guarantee, (1) when a sender sends a message to a receiver, neither the sender nor the receiver can deny having participated in this communication; (2) no principals can obtain the evidence while the other principal cannot do so. This paper extends the model in our previous work [1], and gives a sound and complete on-the-fly model checking method for fair non-repudiation protocols under the assumption of the bounded number of sessions. We also implement the method using Maude. Our experiments automatically detect flaws of several fair non-repudiation protocols.

1 Introduction

Fair non-repudiation protocols intend reliable exchange of messages in the situation that each principal can be dishonest, by trying to take advantages from other principals by aborting the communication or sending fake messages. A fair non-repudiation protocol needs to ensure two properties, non-repudiation and fairness. Non-repudiation means that when a sender sends a message to a receiver, neither the sender nor the receiver can deny participation in this communication. Fairness means no principals can obtain evidence while the other principal cannot do so. Finding flaws for these properties is more difficult than for secrecy and authentication due to misbehavior of dishonest principals.

Difficulties in verifying security properties come from various sources of infinity. For secrecy and authentication [1–5], possible reasons are,

- each principal can initiate or respond to an unbounded number of sessions;
- each principal may communicate with an unbounded number of principals;
- each intruder can produce, store, duplicate, hide, or replace an unbounded number of messages based on the messages sent in the network, following the Dolev-Yao model [6].

Fair non-repudiation protocols further introduce a new factor of infinity,

- each dishonest principal may disobey the prescription of the protocol, sending an unbounded number of messages it can generate.

This paper proposes a sound and complete on-the-fly model checking method for fair non-repudiation protocols under the restriction of the bounded number of sessions. The model checking method is based on trace analysis. There are several model checking methods applied to fairness property [7, 8], while to the best of our knowledge, this is the first model checking method applied to the non-repudiation property.

To describe non-repudiation protocols, we choose a process calculus based on a variant of Spi calculus [9]. The calculus uses environment-based communication, instead of channel-based communication, with the following features.

- The calculus excludes recursive operations, so that only finitely many sessions are represented.
- To represent an unbounded number of principals, a binder is used to represent intended destination of messages [1].
- Following the Dolev-Yao model, a deductive system, which can generate infinitely many messages, is exploited to describe abilities of intruders [4].
- Another deductive system is introduced to generate infinitely many messages that dishonest principals may produce and send [10].

A finite *parametric model* is proposed by abstracting/restricting the infinities. It is sound and complete under the restriction of a bounded number of sessions. The on-the-fly model checking on the parametric model is implemented by Maude, which successfully detects flaws of several fair non-repudiation protocols.

The rest of the paper is organized as follows. Section 2 presents the process calculus and its operational semantics. Section 3 shows how protocols and the security properties are represented. In Section 4, we introduce how to check these properties in a sound and complete parametric model. Section 5 reports the experimental results, Section 6 presents related work, and Section 7 concludes the paper.

2 Concrete model for protocol description

2.1 Process calculus and concrete trace

Assume four countable disjoint sets: $\mathcal{L}$ for labels, $\mathcal{N}$ for names, $\mathcal{B}$ for binder names and $\mathcal{V}$ for variables. Let $a, b, c, \ldots$ indicate labels, $m, n, A, B, \ldots$ indicate names, $\mathtt{m}, \mathtt{n}, \mathtt{k}, \ldots$ indicate binder names, and let $x, y, z, \ldots$ indicate variables.

Definition 1 (Messages). *Messages $M, N, L \ldots$ in a set $\mathcal{M}$ are defined iteratively as follows:*

$$\begin{aligned} pr &::= n \mid x \\ M, N, L &::= pr \mid \mathtt{m}[pr, \ldots, pr] \mid (M, N) \mid \{M\}_L \mid \mathcal{H}(M) \end{aligned}$$

A message is ground, if it does not contain any variables.

pr is ranged over a set of primary messages that cannot be further decomposed. A binder, $\mathtt{m}[pr_1, \ldots, pr_n]$ is a message that can be regarded as a special name indexed by its parameters, $pr_1, \ldots, pr_n$. One usage of binders is to represent encryption keys. For instance, binder $\mathtt{k}[A, S]$ represents a symmetric key shared with principals A and S; $+\mathtt{k}[A]$ and $-\mathtt{k}[A]$ represent A's public key and private key, respectively. (M, N) represents a pair of messages. $\{M\}_L$ is an encrypted message where M is its plain message and L is its encryption key. $\mathcal{H}(M)$ represents a one-way hash function message where M is its parameter. We say a message M is in a message N, if M is a subterm of N.

Definition 2 (Processes). *Let $\mathcal{P}$ be a countable set of processes, which are indicated by $P, Q, R, \ldots$. The syntax of processes is defined as follows:*

$$\begin{aligned} P, Q, R ::= &\ \mathbf{0} \mid \overline{a}M.P \mid a(x).P \mid [M = N]\, P \mid (\mathtt{new}\ x)P \mid (\nu n)P \mid \\ &\ let\ (x, y) = M\ in\ P \mid case\ M\ of\ \{x\}_L\ in\ P \mid P + Q \mid P \| Q \end{aligned}$$

Variables x and y are bound in $a(x).P$, $(\mathtt{new}\ x)P$, let $(x, y) = M$ in P, and case M of $\{x\}_L$ in P. Sets of free variables and bound variables in P are denoted by $f_v(P)$ and $b_v(P)$, respectively. A process P is closed if $f_v(P) = \emptyset$. A name is free in a process if it is not restricted by a restriction operator ν. Sets of free names and local names of P are denoted by $f_n(P)$ and $l_n(P)$, respectively.

Intuitively understanding,

- $\mathbf{0}$ is *Nil* process that does nothing.
- $\overline{a}M.P$ sends message M to the environment and then behaves like P.
- $a(x).P$ awaits an input message M and behaves like $P\{M/x\}$.
- If $M = N$, $[M = N]\, P$ acts as P; otherwise it will be stuck.
- $(\mathtt{new}\ x)P$ behaves like P except that x is bound in P.
- $(\nu n)P$ means that the name n is local in P. Other processes will not know n before P sends it to the environment.
- If M is a pair (N, L), *let* $(x, y) = M$ *in* P is reduced to $P\{N/x, L/y\}$; Otherwise it will be stuck.
- Process *case* M *of* $\{x\}_L$ *in* P is reduced to $P\{N/x\}$ when M it an encrypted message $\{N\}_{L'}$ that L can decrypt; Otherwise is will be stuck.
- $P + Q$ behaves like P or Q.
- $P \| Q$ means that P and Q run concurrently.

The summation $P + Q$ is introduced to our previous model [1], since we need to describe cases in which a dishonest principal in a protocol has several choices, such as aborting communication, or running a recovery stage, which makes a branching run possible.

Note that labels in the input and the output processes are not channels, but tags attached the messages of input and output actions. Furthermore, these labels are distinct from each other.

Messages that the environment can generate are started from the current finite knowledge, denoted by S ($\subseteq \mathcal{M}$), and deduced by an *environmental deductive system*. Here, we presuppose a countable set $\mathcal{E}$ ($\subseteq \mathcal{M}$), for those public names and ground binders such as each principal's name, public keys, intruders' names. For example, $I, -\texttt{k}[I'], \texttt{k}[I,S], +\texttt{k}[A] \ldots \in \mathcal{E}$. Let $\vdash$ be the least binary relation generated by the environmental deductive system in Fig. 1.

$$\frac{}{S \vdash M}\ M \in \mathcal{E}\ \ \mathit{Env} \qquad \frac{}{S \vdash M}\ M \in S\ \ \mathit{Ax}$$

$$\frac{S \vdash M\ S \vdash N}{S \vdash (M,N)}\ \mathit{Pair\_intro} \quad \frac{S \vdash (M,N)}{S \vdash M}\ \mathit{Pair\_elim1} \quad \frac{S \vdash (M,N)}{S \vdash N}\ \mathit{Pair\_elim2}$$

$$\frac{S \vdash \{M\}_{\texttt{k}[A,B]}\ S \vdash \texttt{k}[A,B]}{S \vdash M}\ \mathit{Senc\_elim} \qquad \frac{S \vdash M\ S \vdash \texttt{k}[A,B]}{S \vdash \{M\}_{\texttt{k}[A,B]}}\ \mathit{Senc\_intro}$$

$$\frac{S \vdash \{M\}_{\pm\texttt{k}[A]}\ S \vdash \mp\texttt{k}[A]}{S \vdash M}\ \mathit{Penc\_elim} \qquad \frac{S \vdash M\ S \vdash \pm\texttt{k}[A]}{S \vdash \{M\}_{\pm\texttt{k}[A]}}\ \mathit{Penc\_intro}$$

Fig. 1. Environmental deductive system

A process P that describes a dishonest principal A can send out all messages generated through $\vdash$, and can also encrypt messages with A's private key and shared key. A *P-deductive system* is defined in Fig. 6.

$$\frac{S \vdash M}{S \vdash_P M} \quad \frac{S \vdash_P M}{S \vdash_P \{M\}_{\texttt{k}[A,B]}} \quad \frac{S \vdash_P M}{S \vdash_P \{M\}_{-\texttt{k}[A]}}$$

Fig. 2. A P-deductive system

An *action* is a term of form $\overline{a}M$ or $a(M)$. It is ground if its attached message is ground. The messages in a concrete trace s, represented by $msg(s)$, are those messages in output actions of the concrete trace s. We use $s \vdash M$ to abbreviate $msg(s) \vdash M$, and $s \vdash_P M$ to abbreviate $msg(s) \vdash_P M$.

Definition 3 (Concrete trace and configuration). *A concrete trace s is a ground action string that satisfies each decomposition $s = s'.a(M).s''$ implies $s' \vdash M$, and each $s = s'.\overline{a}M.s''$ implies $s' \vdash_P M$, where P is a closed process that contains the label a. ϵ represents an empty trace. A concrete configuration is a pair $\langle s, P\rangle$, in which s is a concrete trace and P is a closed process.*

2.2 Operational Semantics

The transition relation of concrete configurations is defined by the rules in Fig. 3. Note that in rules $LCOM$ and $RCOM$, no reaction is provided between two composed processes, and both processes communicate with the environment. Furthermore, a function $\texttt{Opp}$ is defined for complemental key in decryption and encryption. Thus we have $\texttt{Opp}(+\texttt{k}[A]) = -\texttt{k}[A]$, $\texttt{Opp}(-\texttt{k}[A]) = +\texttt{k}[A]$ and $\texttt{Opp}(\texttt{k}[A,B]) = \texttt{k}[A,B]$.

$$\begin{array}{rl}
(INPUT) & \langle s, a(x).P\rangle \longrightarrow \langle s.a(M), P\{M/x\}\rangle \;\; s \vdash M \\
(OUTPUT) & \langle s, \overline{a}M.P\rangle \longrightarrow \langle s.\overline{a}M, P\rangle \\
(DEC) & \langle s, case\ \{M\}_L\ of\ \{x\}_{L'}\ in\ P\rangle \longrightarrow \langle s, P\{M/x\}\rangle \quad L' = \texttt{Opp}(L) \\
(PAIR) & \langle s, let\ (x,y) = (M,N)\ in\ P\rangle \longrightarrow \langle s, P\{M/x, N/y\}\rangle \\
(NEW) & \langle s, (\texttt{new}\ x)P\rangle \longrightarrow \langle s, P\{M/x\}\rangle \quad s \vdash_P M \\
(RESTRICTION) & \langle s, (\nu n)P\rangle \longrightarrow \langle s, P\{m/n\}\rangle \quad m \notin f_n(P) \\
(MATCH) & \langle s, [M = M]P\rangle \longrightarrow \langle s, P\rangle \\
(LSUM) & \langle s, P + Q\rangle \longrightarrow \langle s, P\rangle \\
(RSUM) & \langle s, P + Q\rangle \longrightarrow \langle s, Q\rangle \\
(LCOM) & \dfrac{\langle s, P\rangle \longrightarrow \langle s', P'\rangle}{\langle s, P \| Q\rangle \longrightarrow \langle s', P' \| Q\rangle} \\
(RCOM) & \dfrac{\langle s, Q\rangle \longrightarrow \langle s', Q'\rangle}{\langle s, P \| Q\rangle \longrightarrow \langle s', P \| Q'\rangle}
\end{array}$$

Fig. 3. Concrete transition rules

By the rule NEW, a variable bound by `new` can be instantiated to infinitely many ground messages generated by a P-deductive system. Thus an output action with fresh bound variables may send any of these messages to the environment. This leads infinity of a system. Furthermore, the rules $INPUT$ also leads a system to be infinite due to the environmental deductive system.

For convenience, we say a concrete configuration $\langle s, P\rangle$ *reaches* $\langle s', P'\rangle$, if $\langle s, P\rangle \longrightarrow^* \langle s', P'\rangle$. A concrete configuration is a *terminated configuration* if no transition rules can be applied to it. A sequence of consecutive concrete configurations is named a *path*. A concrete configuration $\langle s, P\rangle$ *generates* a concrete s', if $\langle s, P\rangle$ reaches $\langle s', P'\rangle$ for some P'.

3 Representing protocols and security properties

3.1 Representing fair non-repudiation protocols

When model checking authentication or secrecy properties, we making assumptions that the legitimate principals are honest, i.e., behave following the prescription of protocols [1]. For example, principals A and B want to have a private conversation, it is in their interests not to purposely disclose their keys and confidential messages.

Other security goals, such as non-repudiation and fairness, are rather different. We are concerned with protecting one principal against possible cheating by another. For example, a non-repudiation goal of transmission should provide the receiver with proof that the message was indeed sent by the claimed sender, even if the sender subsequently tries to deny it. Thus we cannot assume that the principal will not cheat.

To describe a case that a dishonest principal tries to cheat another principal by sending a fake message, fresh variables are used to denote the sub-message that the principal can use to deceive another principal. These variables are bound by the `new` primitive, and are later instantiated by some ground messages deduced by the dishonest principal according to the NEW transition rules.

A fair non-repudiation protocol is usually more complex than a key distributed protocol. For simplicity of representation, we use several convenient abbreviations. First, pair splitting is applied to input and decryption.

$$\begin{gathered} a(x_1, x_2).P \triangleq a(x).let\ (x_1, x_2) = x\ in\ P \\ case\ M\ of\ \{x_1, x_2\}_L\ in\ P \triangleq case\ M\ of\ \{x\}_L\ in\ let\ (x_1, x_2) = x\ in\ P \end{gathered}$$

The messages of a protocol may have more than two components. We use the following standard abbreviation to generalize the syntax of pairs to arbitrary tuples, given inductively for any $k \geq 2$.

$$(M_1, M_2, \ldots, M_k, M_{k+1}) \triangleq ((M_1, M_2, \ldots, M_k), M_{k+1})$$

Similarly, we write $let\ (x_1, x_2, \ldots, x_n) = M\ in\ P$, $a(x_1, x_2, \ldots, x_n).P$, and $case\ M\ of\ \{x_1, x_2, \ldots, x_n\}_L\ in\ P$, which can be straightforwardly translated into our standard syntax.

We will use a simplified variation of Zhou-Gollmann fair non-repudiation protocol (referred to as *simplified ZG protocol*) as a running example to illustrate how our system works. The full ZG protocol is described in [11]. Note that besides a standard flow description, a fair non-repudiation protocol also contains a description on what are evidences for participated principals.

In this protocol, A aims to send a message M to B. At the same time, A can obtain an evidence that the message was received by B, and B has an evidence that the message was sent by A. The message is transferred in two stages: an encrypted message protected by a new generated key K is first sent directly to B. After A has received evidence of receipt from B, the key K is sent via a trust third party S and both A and B can receive the evidence that K has been distributed by S. The simplified ZG protocol is described flow-by-flow as follows:

$$\begin{aligned} A \longrightarrow B: &\quad \{B, N_A, \{M\}_K\}_{-K_A} && (1) \\ B \longrightarrow A: &\quad \{A, N_A, \{M\}_K\}_{-K_B} && (2) \\ A \longrightarrow S: &\quad \{B, N_A, K\}_{-K_A} && (3) \\ S \longrightarrow A: &\quad \{A, B, N_A, K\}_{-K_S} && (4) \\ S \longrightarrow B: &\quad \{A, B, N_A, K\}_{-K_S} && (5) \end{aligned}$$

The evidence that A *sends the message* M *to* B (referred as M_1) is the pair of messages that B accepted in (1) and (5). In (1), A sends a signed message to B, and B can confirm that the intended receiver of (1) is B by decrypting it by the public key $+K_A$. In (5), B checks whether N_A in (5) coincides with that in (1). If they match, B can confirm that the TTP S has received K from A in (3). Alternatively, the evidence that B *receives the message* M *from* A (referred as M_2) is the pair of the messages that A accepted in (2) and (4).

Principal A will be represented in our calculus as follows:

$$\begin{aligned}A \triangleq\,& (\nu N_A)(\mathtt{new}\, x_1, x_2)\, \overline{a1}\{x_1, N_A, x_2\}_{-\mathtt{k}[A]}).a2(x_3).\\ & \mathit{case}\ x_3\ \mathit{of}\ \{x_4, x_5, x_6\}_{+\mathtt{k}[x_1]}\ \mathit{in}\ [x_4 = A]\,[x_5 = N_A]\,[x_6 = x_2]\,(\mathbf{0}+\\ & (\mathtt{new}\, x_7, x_8)\, \overline{a3}\{x_7, x_8\}_{-\mathtt{k}[A]}.a4(x_9).\ \mathit{case}\ x_9\ \mathit{of}\ \{x_{10}, x_{11}, x_{12}, x_{13}\}_{+\mathtt{k}[S]}\\ & \mathit{in}\ [x_{10} = A]\,[x_{11} = x_1]\,[x_{12} = N_A]\,[x_{13} = x_8].\mathbf{0})\end{aligned}$$

A cannot have the information of who he will communicate with. Thus a binder is used to denote any possible principal he communicates with, in which x_1 is bound by $(\mathtt{new}\, x_1)$. Furthermore, in the first flow, A need not send the right encrypted message, following the prescription of the protocol. On the contrary, he can send any messages, denoted by a fresh bound variable x_2. After receiving a message by a_2, A may abort the communication, or send a message. When sending a message, similarly, he still does not follow the prescription of the protocol, sending two arbitrary messages denoted by two bound variables x_7 and x_8. The latter is used to check the validation of message he has received by $a4$.

$$\begin{aligned}B \triangleq\,& b1(y_1).\mathit{case}\ y_1\ \mathit{of}\ \{y_2, y_3, y_4\}_{+\mathtt{k}[A]}\ \mathit{in}[y_2 = B]\,(\mathbf{0}+\\ & (\mathtt{new}\, y_5)\, \overline{b2}\{A, y_5\}_{-\mathtt{k}[B]}.b3(y_6).\mathit{case}\ y_6\ \mathit{of}\ \{y_7, y_8, y_9, y_{10}\}_{+\mathtt{k}[S]}\ \mathit{in}\\ & [y_7 = A]\,[y_8 = B]\,[y_9 = y_3].\mathbf{0})\end{aligned}$$

As a receiver, following our previous work [1], we will fix B's potential sender to the sender A in one session. Such an assumption is necessary when defining security properties, since otherwise the sender and the receiver we represented may have no connections with each other, and thus these properties cannot be defined between them. Similarly, after receiving a message, B may also quit the communication, or send a message, disobeying the prescription of the protocol. Thus a fresh variable y_5 is used, bound by a `new` operator.

$$S \triangleq s1(z_1).\mathit{case}\ z_1\ \mathit{of}\ \{z_2\}_{+\mathtt{k}[z_3]}\ \mathit{in}\ \overline{s2}\{z_3, z_2\}_{-\mathtt{k}[S]}.\overline{s2}\{z_3, z_2\}_{-\mathtt{k}[S]}.\mathbf{0}$$

TTP S is faithfully following the prescription of the protocol. Furthermore, simplified ZG protocol can be described as a composition among A, B and S.

$$SYS^{ZG} \triangleq A\|B\|S$$

3.2 Probing transformation

Given a process P, the *context* $P[.]$ is obtained when all occurrences of $\mathbf{0}$ in P are replaced by *holes*, $[.]$. Let $\phi(P)$ be the set of holes in $P[.]$.

Definition 4 (Probing transformation). *Given a process P that represents a protocol, a probing transformation is generated from P, by applying the following two transformations, and returns a process (named a probing process).*

- *Declaration process insertion: Let $P[.]$ be the context of P. Given a set $\psi \subseteq \phi(P)$, and a message M, $P_{\psi,M}$ is a probing process generated from P, such that holes in ψ are inserted by the same process $\overline{c}\,M.\mathbf{0}$ with a fresh label $\overline{c}$ (named declaration process), and holes in $\phi(P) - \psi$ are inserted by $\mathbf{0}$ in $P[.]$.*
- *Guardian process composition: A probing process P_g is P composed with a process $c(x).\mathbf{0}$ with a fresh label c (named guardian process), that is, $P \| c(x).\mathbf{0}$.*

For a given security property, the choice of ψ and M in a declaration process insertion are not independent to the protocol description. Currently, a probing transformation is designed for each protocol and security property one by one.

Intuitively, a probing process transformed from a protocol description is used to represent security properties. Declaration process insertion is used to show that a principal can provide some message M, used for authentication [1], non-repudiation and fairness. Guardian process composition is used to check whether a message is observable in the environment, for secrecy [1] and fairness.

3.3 Action terms

Definition 5. *Let α range over the set of actions. The set of action terms is defined as follows:*

$$T ::= \alpha \mid \neg T \mid T \wedge T \mid T \vee T$$
$$\sigma ::= T \mid T \hookleftarrow T \mid T \hookrightarrow_F T$$

Action terms inductively defined by T are *state action terms*, and those defined by σ are *path action terms.* A state action term is also a path action term.

We define two relations: the relation $\models_t$ between a concrete trace and a state action term, and $\models$ between a concrete configuration and a path action term.

- $s \models_t \alpha$: there exists a ground substitution ρ from variables to ground messages such that $\alpha\rho$ occurs in s.
- $s \models_t \neg T$: $s \not\models_t T$.
- $s \models_t T_1 \wedge T_2$: $s \models_t T_1$ and $s \models_t T_2$.
- $s \models_t T_1 \vee T_2$: $s \models_t T_1$ or $s \models_t T_2$.
- $\langle s, P\rangle \models T$: for each concrete trace s' generated by $\langle s, P\rangle$, $s' \models_t T$ holds.
- $\langle s, P\rangle \models T_1 \hookleftarrow T_2$: for each concrete trace s' generated by $\langle s, P\rangle$, if there is a ground substitution ρ such that $s' \models_t T_2\rho$, then $s' \models_t T_1\rho$, and $T_1\rho$ occurs before $T_2\rho$ in s'.
- $\langle s, P\rangle \models T_1 \hookrightarrow_F T_2$: for each concrete configuration $\langle s', P'\rangle$ reached by $\langle s, P\rangle$, if there is a ground substitution ρ such that $s' \models_t T_1\rho$, then for every path starting from $\langle s', P'\rangle$, there exists a concrete trace s'' such that $s'' \models_t T_2\rho$.

3.4 Representing non-repudiation and fairness

Non-repudiation Non-repudiation is when a sender sends some message to a receiver, neither the sender nor the receiver can deny this after participating in this communication. Usually, it concerns the following two properties [12, 13]:

- *Non-repudiation of origin (NRO)* is intended to protect against the sender's false denial of having sent the messages.
- *Non-repudiation of receipt (NRR)* is intended to protect against the receiver's false denial of having received the message.

For non-repudiation and fairness, two declaration processes, $\overline{evid_B}\, M_1.\mathbf{0}$ and $\overline{evid_A}\, M_2.\mathbf{0}$, are introduced for A and B, respectively, in a declaration process insertion. For the simplified ZG protocol, the evidence M_1 (resp. M_2) is the pair of messages in (1) and (5) (resp. (2) and (4)). In the protocol description, they are messages received at $b1$ and $b3$ (resp. $a2$ and $a4$) as y_1 and y_6 (resp. x_3 and x_9). Then the declaration process is $\overline{evid_A}(y_1, y_6).\mathbf{0}$ (resp. $\overline{evid_B}(x_3, x_9).\mathbf{0}$).

$$\begin{aligned}
A_p \triangleq & (\nu N_A)(\mathtt{new}\, x_1, x_2)\, \overline{a1}\{x_1, N_A, x_2\}_{-\mathtt{k}[A]}).a2(x_3). \\
& \mathit{case}\ x_3\ \mathit{of}\ \{x_4, x_5, x_6\}_{+\mathtt{k}[x_1]}\ \mathit{in}\ [x_4 = A]\, [x_5 = N_A]\, [x_6 = x_2]\, (\mathbf{0}+ \\
& (\mathtt{new}\, x_7, x_8)\, \overline{a3}\{x_7, x_8\}_{-\mathtt{k}[A]}.a4(x_9).\ \mathit{case}\ x_9\ \mathit{of}\ \{x_{10}, x_{11}, x_{12}, x_{13}\}_{+\mathtt{k}[S]} \\
& \mathit{in}\ [x_{10} = A]\, [x_{11} = x_1]\, [x_{12} = N_A]\, [x_{13} = x_8].\overline{\mathbf{evid_B}}(\mathbf{x_3}, \mathbf{x_9}).\mathbf{0}) \\
B_p \triangleq & b1(y_1).\mathit{case}\ y_1\ \mathit{of}\ \{y_2, y_3, y_4\}_{+\mathtt{k}[A]}\ \mathit{in}[y_2 = B]\, (\mathbf{0}+ \\
& (\mathtt{new}\, y_5)\, \overline{b2}\{A, y_5\}_{-\mathtt{k}[B]}.b3(y_6).\mathit{case}\ y_6\ \mathit{of}\ \{y_7, y_8, y_9, y_{10}\}_{+\mathtt{k}[S]}\ \mathit{in} \\
& [y_7 = A]\, [y_8 = B]\, [y_9 = y_3].\overline{\mathbf{evid_A}}(\mathbf{y_1}, \mathbf{y_6}).\mathbf{0}) \\
SYS_p^{ZG} \triangleq & A_p \| B_p \| S
\end{aligned}$$

For NRO, we guarantee that a receiver got the evidence of a sender should come after the sender sent the message. Thus $\hookleftarrow$ is used to describe the property. For NRR, we guarantee that after a sender got the evidence, his receiver will inevitably obtain the message. We will use $\hookrightarrow_F$ to characterize the property.

Characterization 1 (NRO in simplified ZG protocol) *Given the description with probing process of simplified ZG protocol, the NRO is satisfied, if*

$$\begin{aligned}
\langle \epsilon, SYS_p^{ZG} \rangle \models \overline{a1}\{B, x, y\}_{-\mathtt{k}[A]} \wedge \overline{a3}\{B, x, z\}_{-\mathtt{k}[A]} \hookleftarrow & \\
\overline{evid}_A(\{B, x, y\}_{-\mathtt{k}[A]}, \{A, B, x, z\}_{-\mathtt{k}[S]})&
\end{aligned}$$

Characterization 2 (NRR in simplified ZG protocol) *Given the description with probing process of simplified ZG protocol, the NRR is satisfied if*

$$\begin{aligned}
\langle \epsilon, SYS_p^{ZG} \rangle \models \overline{evid}_B(\{A, x, y\}_{-\mathtt{k}[B]}, \{A, B, x, z\}_{-\mathtt{k}[S]}) \hookrightarrow_F & \\
\overline{b2}\{A, x, y\}_{-\mathtt{k}[B]} \wedge \overline{s2}\{A, B, x, z\}_{-\mathtt{k}[S]}&
\end{aligned}$$

Fairness A protocol is unfair, if one principal can obtain the evidence while the other principal cannot do so. Furthermore, a receiver cannot access the message until the sender has the evidence of the receiver. To define the fairness that satisfies the above requirements, we need the following three properties [10, 13].

- Fairness for origin obtaining evidence: If a receiver has an evidence of its sender, then the sender should have an evidence of the receiver. We name it FAIRO.

- Fairness for receipt obtaining evidence: If a sender has an evidence of its receiver, then the receiver should obtain an evidence of the sender. We name it FAIRR.
- Fairness for message receipt: A receiver should not know the message until the evidence of the receiver has been provided to the sender. We name it FAIRM.

Characterization 3 (FAIRO in simplified ZG protocol) *Given the description with probing process of simplified ZG protocol, the FAIRO is satisfied if*

$$\begin{aligned}\langle \epsilon, SYS_p^{ZG}\rangle \models \overline{evid}_A(\{B,x,y\}_{-\mathtt{k}[A]}, \{A,B,x,z\}_{-\mathtt{k}[S]}) \hookrightarrow_F \\ \overline{evid}_B(\{A,x,y\}_{-\mathtt{k}[B]}, \{A,B,x,z\}_{-\mathtt{k}[S]})\end{aligned}$$

Characterization 4 (FAIRR in simplified ZG protocol) *Given the description with probing process of simplified ZG protocol, the FAIRR is satisfied, if*

$$\begin{aligned}\langle \epsilon, SYS_p^{ZG}\rangle \models \overline{evid}_B(\{A,x,y\}_{-\mathtt{k}[B]}, \{A,B,x,z\}_{-\mathtt{k}[S]}) \hookrightarrow_F \\ \overline{evid}_A(\{B,x,y\}_{-\mathtt{k}[A]}, \{A,B,x,z\}_{-\mathtt{k}[S]})\end{aligned}$$

For FAIRM, a guardian process, $check(x).\mathbf{0}$, is inserted in B_p by a guardian process composition. It is used to check whether B already knows the message.

$$\begin{aligned}B_c &\triangleq B_p \| \mathbf{check(x).0}\\ SYS_c^{ZG} &\triangleq A_p \| B_c \| S\end{aligned}$$

Characterization 5 (FAIRM in simplified ZG protocol) *Given the description with probing process of simplified ZG protocol, the FAIRM is satisfied if*

$$\langle \epsilon, SYS_c^{ZG}\rangle \models \neg\overline{check}\, z \hookleftarrow \overline{evid}_B(\{A,x,y\}_{-\mathtt{k}[B]}, \{A,B,x,z\}_{-\mathtt{k}[S]})$$

4 Parametric simulation

All messages in concrete traces generated by transition rules in Fig. 3 are guaranteed to be ground, since when a concrete trace is increased by $INPUT$ or $OUTPUT$ rules, each variable in the action added to the tail will be substituted to a ground message. This is also the reasons that a system is infinite. In this section, concrete traces and configurations are given parametric counterparts, which may contain free variables. Variables in an action will not be substituted when a parametric trace is increased.

4.1 Parametric trace and operational semantics

Definition 6 (Parametric trace and configuration). *A* parametric trace $\hat{s}$ *is a string of actions. A* parametric configuration *is a pair $\langle \hat{s}, P\rangle$, in which $\hat{s}$ is a parametric trace and P is a process.*

$$
\begin{array}{rl}
(PINPUT) & \langle \hat{s}, a(x).P\rangle \longrightarrow_p \langle \hat{s}.a(x), P\rangle \\
(POUTPUT) & \langle \hat{s}, \overline{a}M.P\rangle \longrightarrow_p \langle \hat{s}.\overline{a}M, P\rangle \\
(PDEC) & \langle \hat{s}, case\ \{M\}_L\ of\ \{x\}_{L'}\ in\ P\rangle \longrightarrow_p \langle \hat{s}\theta, P\theta\rangle \\
 & \qquad\qquad \theta = \mathtt{Mgu}(\{M\}_L, \{x\}_{\mathtt{Opp}(L')}) \\
(PPAIR) & \langle \hat{s}, let\ (x,y) = M\ in\ P\rangle \longrightarrow_p \langle \hat{s}\theta, P\theta\rangle\ \ \theta = \mathtt{Mgu}((x,y), M) \\
(PNEW) & \langle \hat{s}, (\mathtt{new}\ x)P\rangle \longrightarrow_p \langle \hat{s}, P\{y/x\}\rangle \quad y \notin f_v(P) \cup b_v(P) \\
(PRESTRICTION) & \langle \hat{s}, (\nu n)P\rangle \longrightarrow_p \langle \hat{s}, P\{m/n\}\rangle\ \ m \notin f_n(P) \\
(PMATCH) & \langle \hat{s}, [M = M']P\rangle \longrightarrow_p \langle \hat{s}\theta, P\theta\rangle \qquad \theta = \mathtt{Mgu}(M, M') \\
(PLSUM) & \langle \hat{s}, P + Q\rangle \longrightarrow_p \langle \hat{s}, P\rangle \\
(PRSUM) & \langle \hat{s}, P + Q\rangle \longrightarrow_p \langle \hat{s}, Q\rangle \\
(PLCOM) & \dfrac{\langle \hat{s}, P\rangle \longrightarrow_p \langle \hat{s}', P'\rangle}{\langle \hat{s}, P \| Q\rangle \longrightarrow_p \langle \hat{s}', P' \| Q'\rangle} \quad Q' = Q\theta \text{ if } \hat{s}' = \hat{s}\theta \text{ else } Q' = Q \\
(PRCOM) & \dfrac{\langle \hat{s}, Q\rangle \longrightarrow_p \langle \hat{s}', Q'\rangle}{\langle \hat{s}, P \| Q\rangle \longrightarrow_p \langle \hat{s}', P' \| Q'\rangle} \quad P' = P\theta \text{ if } \hat{s}' = \hat{s}\theta \text{ else } P' = P
\end{array}
$$

Fig. 4. Parametric transition rules

The transition relation of parametric configurations is given by the rules in Fig. 4. A substitution θ mapping from variables to messages is a *unifier* of M_1 and M_2 if $M_1\theta = M_2\theta$. A function $\mathtt{Mgu}(M_1, M_2)$ returns the *most general unifier* of M_1, M_2, which is a unifier θ such that any other unifier can be written as a composition of substitution $\theta\theta'$ for some θ'.

Definition 7 (Concretization and abstraction). *Given a parametric trace $\hat{s}$, if there exists a substitution ϑ that assigns each variable to a ground message, and which satisfies $s = \hat{s}\vartheta$, where s is a concrete trace, we say that s is a* concretization *of $\hat{s}$ and $\hat{s}$ is an* abstraction *of s. ϑ is named a* concretized substitution*.*

According to the definition of parametric configurations, a concrete configuration $\langle \epsilon, P\rangle$ is also a parametric configuration. We name such a configuration an *initial configuration*. We hope that from an initial configuration, each concrete trace generated by the concrete transition rules in Fig. 3, has an abstraction generated by the parametric transition rules in Fig. 4, and that each parametric trace has at least one concretization. Thus a bisimulation relation can be defined between them.

However, although each concrete trace does have an abstraction, some parametric traces may have no concretizations. Fortunately, parametric traces can still cover its concrete traces. That is, if a parametric trace has a concretization, then the concretization is a concrete trace generated by concrete transition rules. Otherwise the parametric trace cannot be instantiated to any concrete trace. Here we have the soundness and completeness theorem.

Theorem 1. *(Soundness and completeness) Let $\langle \epsilon, P\rangle$ be an initial configuration, and s be a concrete trace. $\langle \epsilon, P\rangle$ generates s, if and only if there exists $\hat{s}$, such that $\langle \epsilon, P\rangle \longrightarrow_p^* \langle \hat{s}, P'\rangle$ for some P', and s is a concretization of $\hat{s}$.*

4.2 Satisfiable normal form

Theorem 1 shows that each concrete trace generated by an initial configuration has an abstraction. However, a parametric trace may or may not have concretizations.

Example 1. Consider a naive protocol:

$$A \longrightarrow B : \{A, M\}_{K_{AB}}$$

In its parametric system, there exists a parametric trace $b1(\{A, x\}_{\mathtt{k}[A,B]})$. In its concrete system, since $\mathtt{k}[A, B]$ was not leaked in the environment, before A or B sends an encrypted message protected by $\mathtt{k}[A, B]$, B cannot accept any message encrypted by $\mathtt{k}[A, B]$. Thus, the parametric trace $b1(\{A, x\}_{\mathtt{k}[A,B]})$ has no concretizations.

We name a message like $\{A, x\}_{\mathtt{k}[A,B]}$ a *rigid message.* Intuitively, a rigid message is the pattern of a requirement of an input action. The requirement can only be satisfied by the messages generated by a proper principal. If there are no appropriate messages to satisfy the requirement, then the parametric trace has no concretizations.

A parametric trace with a rigid message needs to be substituted by unifying a rigid message to the messages in output actions of its prefix parametric traces. When model checking authentication and secrecy in our previous work [1], the above example is the only occasion that a rigid message may occur, while in this extended model, such a unification may lead a new inconsistency in the system.

Example 2. Consider a naive protocol:

$$\begin{array}{ll} A \longrightarrow B : & N_A \\ B \longrightarrow A : & \{N_B\}_{-K_B} \end{array}$$

Suppose that A may send any possible message to B, which is represented by $(\mathtt{new}\, x)\, \overline{a1}\, x$. $\overline{a1}\, x.a2\, \{y\}_{-\mathtt{k}[B]}$ is a parametric trace. In $a2$, a rigid message $\{y\}_{-\mathtt{k}[B]}$ is a requirement of A. After applying unification, the parametric trace deduces to $\overline{a1}\{y\}_{-\mathtt{k}[B]}.a2\{y\}_{-\mathtt{k}[B]}$. But as a principal A, he cannot generate any messages that satisfy the patten $\{y\}_{-\mathtt{k}[B]}$. Thus such kind of parametric message is also a rigid message, which should be further substituted by applying the similar unification.

A rigid message has the following definition.

Definition 8 (Rigid message). *Given a parametric trace $\hat{s}$, $\{N\}_L$ in M is a rigid message if*

- *M is in an input action such that $\hat{s} = \hat{s}'.a(M).\hat{s}''$, and*
  - *if L is a shared key or a private key, then $\hat{s}' \not\vdash L$ and $\hat{s}' \not\vdash \{N\}_L$;*
  - *if L is a public key, then there exists some rigid message, or at least one name or binder in N cannot be deduced by the $\hat{s}'$, and $\hat{s}' \not\vdash \{N\}_L$.*

- *M is in an output action such that $\hat{s} = \hat{s}'.\overline{a}\,M.\hat{s}''$, and*
  - *$\{N\}_L$ satisfies the above three conditions, and*
  - *L is not known by the principal that contains the label a.*

A parametric trace with a rigid message needs to be further substituted by trying to unify the rigid message to the atomic messages in output actions of its prefix parametric trace. Such unification procedures will terminate because the number of atomic messages in the output actions of its prefix parametric trace is finite. We name these messages *elementary messages*, and use $el(\hat{s})$ to represent the set of elementary messages in $\hat{s}$.

Definition 9 (Elementary messages). *Let $\mathcal{U}$ be a set of messages. $dec(\mathcal{U})$ is a minimal set that satisfies*

- *$\mathcal{U} \subseteq dec(\mathcal{U})$;*
- *If $(M, N) \in dec(\mathcal{U})$, then $M, N \in dec(\mathcal{U})$;*
- *If $\{M\}_L \in dec(\mathcal{U})$, L is ground, and $L \in dec(\mathcal{U})$, then $M, L \in dec(\mathcal{U})$;*
- *If $\{M\}_L \in dec(\mathcal{U})$, and L is not ground, then $M \in dec(\mathcal{U})$.*

Given a parametric trace $\hat{s}$, let $msg(\hat{s})$ be the set of all parametric messages in output actions of $\hat{s}$, then $el(\hat{s})$ is the set of minimal terms with respect to the subterm relation in $dec(msg(\hat{s}))$.

Given a parametric trace $\hat{s}$ and a message N, we say N is *$\hat{\rho}$-unifiable* in $\hat{s}$, if there exists $N' \in el(\hat{s})$ such that $\hat{\rho} = \mathtt{Mgu}(N, N')$.

Definition 10 (Deductive relation). *Let $\hat{s}$ be a parametric trace such that $\hat{s} = \hat{s}_1.l(M).\hat{s}_2$, in which l is an input or an output label. If there exists a rigid message N in M such that $N \notin el(\hat{s}_1)$, and N is $\hat{\rho}$-unifiable in $\hat{s}_1$, then $\hat{s} \rightsquigarrow \hat{s}\hat{\rho}$.*

For two parametric traces $\hat{s}$ and $\hat{s}'$, if $\hat{s} \rightsquigarrow^* \hat{s}'$ and there is no $\hat{s}''$ that satisfies $\hat{s}' \rightsquigarrow \hat{s}''$, we name $\hat{s}'$ the *normal form* of $\hat{s}$. The set of normal forms of $\hat{s}$ is denoted by $\mathtt{nf}_{\rightsquigarrow}(\hat{s})$.

Remark 1. Given a parametric trace $\hat{s}$, $\mathtt{nf}_{\rightsquigarrow}(\hat{s})$ is finite.

A concretization of a parametric trace $\hat{s}$ is still the concretization of $\hat{s}'$ if $\hat{s} \rightsquigarrow \hat{s}'$. Thus whether a parametric trace has concretizations is equivalent to whether there exist parametric traces in its $\mathtt{nf}_{\rightsquigarrow}(\hat{s})$ that have concretizations.

Lemma 1. *If $\hat{s}$ is a parametric trace, and s is a concretization satisfying $s = \hat{s}\vartheta$ where ϑ is a concretized substitution, then $\hat{s}$ is either a normal form, or there exists $\hat{s}'$ such that $\hat{s} \rightsquigarrow \hat{s}'$ with $\hat{s}\vartheta = \hat{s}'\vartheta$.*

Lemma 2. *Let $\hat{s}$ be a parametric trace, and let $\hat{s}'$ be a normal form in $\mathtt{nf}_{\rightsquigarrow}(\hat{s})$. $\hat{s}'$ has a concretization, if and only if, for each decomposition $\hat{s}' = \hat{s}'_1.l(M).\hat{s}'_2$ in which l is either an input label or an output label, each rigid message N in M satisfies $N \in el(\hat{s}'_1)$.*

A satisfiable normal form is a normal form of $\hat{s}$ that satisfies the requirements in Lemma 2. $\mathtt{snf}_{\leadsto}(\hat{s})$ denotes the set of *satisfiable normal forms* of $\hat{s}$. Since $\mathtt{snf}_{\leadsto}(\hat{s}) \subseteq \mathtt{nf}_{\leadsto}(\hat{s})$, $\mathtt{snf}_{\leadsto}(\hat{s})$ is finite.

Remark 2. Given a parametric trace $\hat{s}$, $\mathtt{snf}_{\leadsto}(\hat{s})$ is finite.

Thus, a parametric trace has a concretization if and only if $\mathtt{snf}_{\leadsto}(\hat{s}) \neq \emptyset$.

Lemma 3. *Let $\hat{s}$ be a parametric trace, and let s be a trace. s is a concretization of $\hat{s}$ if and only if s is a concretization of some $\hat{s}'$ with $\hat{s}' \in \mathtt{snf}_{\leadsto}(\hat{s})$.*

Theorem 2. *A parametric trace $\hat{s}$ has a concretization if and only if $\mathtt{snf}_{\leadsto}(\hat{s}) \neq \emptyset$.*

4.3 Simulation on a parametric model

In order to check non-repudiation and fairness properties under the parametric semantics, we need to simulate action terms definitions on its parametric traces.

Definition 11. *Let T be a state action term, and let $\hat{s}$ be a parametric trace that has concretizations. We say $\hat{s} \models_t T$, if for each concretization s of $\hat{s}$, $s \models_t T$.*

Definition 12. *Let σ be a path action term, and let $\langle \hat{s}, P \rangle$ be a parametric configuration, where $\hat{s}$ has concretizations. We say $\langle \hat{s}, P \rangle \models \sigma$, if for each concretization s of $\hat{s}$, where $s = \hat{s}\vartheta$, $\langle \hat{s}\vartheta, P\vartheta \rangle \models \sigma$.*

An action α is *$\hat{\rho}$-unifiable* in a parametric trace $\hat{s}$ if the parametric message in α can be unified to the message attached to the same label as α in $\hat{s}$, and $\hat{\rho}$ is the result of the unification.

Lemma 4. *Given a parametric trace $\hat{s}$,*

1. *$\hat{s} \models_t \alpha$ if and only if, α is $\hat{\rho}$-unifiable in $\hat{s}$, and for each satisfiable normal form in $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho})$ satisfying $\hat{s}\hat{\rho}\hat{\rho}'$, $\alpha\hat{\rho}\hat{\rho}'$ occurs in $\hat{s}\hat{\rho}\hat{\rho}'$.*
2. *$\hat{s} \models_t \neg\alpha$ if and only if $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho}) = \emptyset$ when α is $\hat{\rho}$-unifiable in $\hat{s}$.*
3. *For any state action term T, $\hat{s} \models_t T$ is decidable.*

Lemma 5. *Given a concrete configuration $\langle s, P \rangle$, and two state action terms T_1 and T_2, $\langle s, P \rangle \models T_1 \hookrightarrow_F T_2$, if and only if for each concrete configuration $\langle s', P' \rangle$ reached by $\langle s, P \rangle$, if there exists a ground substitution ρ such that $s' \models_t T_1\rho$, then for each terminated configuration $\langle s'', P'' \rangle$ reached by $\langle s', P' \rangle$, $T_2\rho$ occurs in s''.*

The above Lemma 5 provides a easy way to check $\hookrightarrow_F$ in parametric traces.

Theorem 3. *Given an initial configuration $\langle \epsilon, P \rangle$,*

1. *Given a state action term T, $\langle \epsilon, P \rangle \models T$, if and only if for each parametric trace $\hat{s}$ generated by $\langle \epsilon, P \rangle$, $\hat{s} \models_t T$.*

2. *Given two state action terms* T_1 *and* T_2*,* $\langle \epsilon, P\rangle \models T_2 \hookleftarrow T_1$*, if and only if for each parametric trace* $\hat{s}$ *generated by* $\langle \epsilon, P\rangle$*, if* T_1 *is* $\hat{\rho}$*-unifiable in* $\hat{s}$*, then for each satisfiable normal form in* $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho})$ *satisfying* $\hat{s}\hat{\rho}\hat{\rho}'$*,* $T_2\hat{\rho}\hat{\rho}'$ *occurs before* $T_1\hat{\rho}\hat{\rho}'$*.*
3. *Given two state action terms* T_1 *and* T_2*,* $\langle \epsilon, P\rangle \models T_1 \hookrightarrow_F T_2$*, if and only if for each parametric configuration* $\langle \hat{s}', P'\rangle$ *reached by* $\langle \epsilon, P\rangle$*, if* T_1 *is* $\hat{\rho}$*-unifiable in* $\hat{s}'$*, then for each terminated parametric configuration* $\langle \hat{s}''\hat{\rho}, P''\hat{\rho}\rangle$ *reached by* $\langle \hat{s}'\hat{\rho}, P'\hat{\rho}\rangle$*, either* $\hat{s}''\hat{\rho}$ *cannot deduce any satisfiable normal forms, or* $T_2\hat{\rho}\hat{\rho}'$ *occurs in each satisfiable normal form* $\hat{s}''\hat{\rho}\hat{\rho}'$ *in* $\mathtt{snf}_{\leadsto}(\hat{s}'\hat{\rho})$*.*

Actually, Theorem 3 implicitly shows the algorithm to check whether a system satisfies a path action term.

5 Experiments results

We implement the on-the-fly model checking method using Maude [14], since Maude can describe model generation rules by equational rewriting, instead of describing a model directly. Thus each property can be checked at the same time when a model is generated. It is named an on-the-fly model checking method.

In experiments with one session bound, the attacks for NRO, FAIRO and FAIRM of simplified ZG protocol were detected automatically (see details in Appendix D). For comparison, we also implemented the analysis for the full ZG protocol, which guarantees those three properties [1]. We also tested some protocols proposed by the International Organization for Standardization [15–17]. It is well-known that these protocols have flaws [13, 12].

The results are summarized in Fig. 5. In the figure, column "protocol spec." is the number of lines for a protocol specific part. In addition to these lines, each Maude file also contains about 400 lines for the common description of the method.

The experiments were carried out by Maude 2.2, and were performed on a Pentium 1.4 GHz, 1.5 GB memory PC, under Windows XP.

6 Related work

Trace analysis is one of the formal approaches in analyzing security protocols, both in the model checking and in theorem proving.

Gavin Lowe first used trace analysis on process calculus CSP, and implemented a model-checker FDR to discover numerous attacks [2, 18]. In his work, the intruder was represented as a recursive process. He restricted the state space to be finite by imposing upper-bounds upon messages the intruder generates, and also upon the number of principals in the network.

Many of our ideas are inspired by Michele Boreale's symbolic approach [4]. In his research, he restricted the number of principals and intruders, and represented that each principal explicitly communicates with an intruder. Our model

[1] For formal definitions of the properties of full ZG protocol, refer to [10].

protocols	property	protocol spec.	states	times(ms)	flaws
Simplified ZG protocol	NRO	50	513	3,954	detected
	NRR	50	780	3,905	secure
	FAIRO	55	770	2,961	detected
	FAIRR	55	846	3,903	secure
	FAIRM	50	4,109	45,545	detected
Full ZG protocol	NRO	50	633	7,399	secure
	FAIRO	55	788	3,394	secure
	FAIRM	60	788	3,490	secure
ISO/IEC13888-2 M2	NRO	50	1,350	7,710	detected
	FAIRO	65	1,977	6,827	detected
	FAIRR	65	2,131	7,506	secure
ISO/IEC13888-3 M-h	FAIRO	60	295	918	detected
	FAIRR	60	305	1,040	secure

Fig. 5. Experimental results

finitely represents an unlimited number of principals and intruders in the network. This is more powerful than his.

Trace analysis is also used in theorem proving. Paulson et al. first defined the traces and the security properties inductively, and proved whether a security protocol satisfies a property by Isabelle/HOL [19, 20]. The approach need not restrict the number of traces to be finite, however it cannot be fully automated.

Several other research papers also used process calculi: M. Abadi and A. Gordon developed the Spi calculus with primitives representing the cryptographic operations of encryption and decryption. They used some equivalences [9, 21] to define the security properties. Unfortunately, these equivalences are usually undecidable. Another approach based on process calculi was a static analysis based on type system [22–24]. The limitation of this method is that the intruder's model is weaker than the Dolev-Yao model, assuming that the intruder is partially trusted.

David Basin et al. proposed an On-the-fly model checking method (OFMC) [5]. They used a high-level language, HLPSL, to represent a protocol, which then translates automatically to a low-level language, IF. An intruder's messages are instantiated when necessary, which is similar to the occasion when a rigid message occurs in our model. In their work, an intruder's role is explicitly assigned, for instance, as an initiator. This is efficient, but the process needs to be performed several times to ensure that no intruders can attack a protocol in any roles. In our work, we do not explicitly define an intruder, and we have to check all situations in which intruders act in different roles at one time.

The strand space formalism [25, 26] is a framework for studying security protocol analysis. There are some similarities between our parametric trace and the strand. The difference is that in Strand Space, intruder abilities are explicitly represented as some strands.

Recently, a more general and efficient verification approach based on Horn clauses and resolution has been proposed by B. Blanchet et al. in [27, 28]. It verifies the properties in infinite sessions of a protocol with infinite principals by some approximations on both sessions and principals. The method sometimes does not terminate, as the author noted. In [28], a tag system that assigns each encrypted message a unique tag is added to the system to make each run terminate. Then the authors proved that security of a tagged protocol does not imply the security of an untagged version. Our model avoids recursive executions of protocols by restricting replication, and thus the model terminates.

The similar idea to a binder was investigated by Comon-Lundh and Cortier, in terms of logic programming. They proved that it is sufficient to consider only a bounded number of principals when verifying secrecy and authentication properties [29].

The above methods only focus on analyzing authentication or secrecy properties. These methods cannot be straightforwardly used to analyze the properties of a fair non-repudiation protocol. The following methods aim to analyzing security properties of fair non-repudiation protocols, and other kinds of fair exchange protocols.

Steve Schneider proposed a trace analysis to verify non-repudiation and fairness properties of full ZG protocol based on CSP [10]. He used a deductive system to describe a dishonest principal and *failures* of a process to define these properties. We borrow the idea of the dishonest principal description from his research.

Jianying Zhou is one of the core researchers in non-repudiation protocol [12]. He has proposed several non-repudiation protocols, and verified their correctness by SVO logic in his papers and book [11, 30, 12]. We take his definition for non-repudiation in this paper.

G. Bella and L. Paulson extended their previous Isabelle/HOL theorem proving approach for authentication property [19, 20] to the ZG protocol, and proved the correctness of its non-repudiation and fairness properties [31, 32]. The approach need not restrict the number of states to be finite, yet cannot be fully automated.

There were several studies based on game-theoretic model checking method on the fairness property. S. Kremer firstly analyzed various these protocols, and also summarized and compared many formal definitions of fairness in his thesis [7]. Recently, D. Kähler et al. proposed a more powerful AMC-model checking method for verifying the fairness property [8].

V. Shmatikov et al. analyzed fairness of two contract signing protocols based on a finite-state model checker Murφ [33]. In his model, as authors noted, model checking was limited to a bounded number of sessions, and a bounded number of messages that an intruder generates. We has released the latter bound, using a parametric abstraction on an unbounded number of messages.

7 Conclusion

This paper proposed a sound and complete on-the-fly model checking method for fair non-repudiation protocols under the restriction of the bounded number of sessions. It extended our previous work [1] to handle another infinity factor of fair non-repudiation protocols. We implemented the method using Maude. It successfully detected the flaws of several examples automatically.

Our future work will be: First, to extend the method with pushdown model checking to cover recursive processes, so that a protocol with infinitely many sessions can be analyzed. Second, to check properties of other kinds of fair exchange protocols, such as digital contract signing protocols, and certified e-mail protocols.

8 Acknowledge

The authors thank Dr.Yoshinobu Kawabe for fruitful discussion. This research is supported by the 21st Century COE "Verifiable and Evolvable e-Society" of JAIST, funded by Japanese Ministry of Education, Culture, Sports, Science and Technology.

References

1. Li, G., Ogawa, M.: On-the-fly Model Checking of Security Protocols and Its Implementation by Maude. IPSJ Transactions on Programming **48, SIG 10(PRO 33)** (2007) 50–75
2. Lowe, G.: Breaking and Fixing the Needham-Schroeder Public-key Using FDR. In: Proceedings of the 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96). Volume 1055 of Lecture Notes in Computer Science., Springer-Verlag (1996) 147–166
3. Amadio, R.M., Prasad, S.: The Game of the Name in Cryptographic Tables. In: Proceedings of the 1st International Conference on Principles and Practice of Declarative Programming (PPDP'99). Volume 1702 of Lecture Notes in Computer Science., Springer-Verlag (1999) 15–26
4. Boreale, M.: Symbolic Trace Analysis of Cryptographic Protocols. In: Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP'01). Volume 2076 of Lecture Notes in Computer Science., Springer-Verlag (2001) 667–681
5. Basin, D., Mödersheim, S., Viganò, L.: OFMC: A Symbolic Model Checker for Security Protocols. International Journal of Information Security **4(3)** (2005) 181–208
6. Dolev, D., Yao, A.C.: On the Security of Public Key Protocols. IEEE Transactions on Information Theory **29** (1983) 198–208
7. Kremer, S.: Formal Analysis of Optimistic Fair Exchange Protocols. PhD thesis, Universite Libre de Bruxelles (2003)
8. Käehler, D., Küesters, R., Truderung, T.: Infinite State AMC-Model Checking for Cryptographic Protocols. In: Proceedings of the 21th Annual IEEE Symposium on Logic in Computer Science (LICS'07). (2007)

9. Abadi, M., Gordon, A.D.: A Calculus for Cryptographic Protocols: The Spi Calculus. In: Proceedings of the 4th ACM Conference on Computer and Communications Security (CCS'97), ACM Press (1997) 36–47
10. Schneider, S.: Formal Analysis of a Non-repudiation Protocol. In: Proceedings of the 11th Computer Security Foundations Workshop (CSFW'98), IEEE Computer Society Press (1998) 54–65
11. Zhou, J., Gollmann, D.: A Fair Non-repudiation Protocol. In: Proceedings of the 17th IEEE Symposium on Security and Privacy (S&P'96), IEEE Computer Society Press (1996) 55–61
12. Zhou, J.: Non-repudiation in Electronic Commerce. Computer Security Series. Artech House (2001)
13. Kremer, S., Markowitch, O., Zhou, J.: An Intensive Survey of Fair Non-repudiation Protocols. Computer Communications **25** (2002) 1606–1621
14. Clavel, M., Durán, F., Eker, S., Lincolnand, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: Maude Manual (version 2.2). http://maude.cs.uiuc.edu/maude2-manual/ (2005)
15. ISO/IEC13888-1: Information Technology - Security Techniques - Non-repuduation - Part 1: General. (1997)
16. ISO/IEC13888-2: Information Technology - Security Techniques - Non-repuduation - Part 2: Mechanisms Using Symmetric Techniques. (1997)
17. ISO/IEC13888-3: Information Technology - Security Techniques - Non-repuduation - Part 3: Mechanisms Using Asymmetric Techniques. (1997)
18. Lowe, G.: Some New Attacks upon Security Protocols. In: Proceedings of the 9th IEEE Computer Security Foundations Workshop (CSFW'96), IEEE Computer Society Press (1996) 162–169
19. Bella, G.: Inductive Verification of Cryptographic Protocols. PhD thesis, University of Cambridge (2000)
20. Paulson, L.C.: The Inductive Approach to Verifying Cryptographic Protocols. Journal of Computer Security **6** (1998) 85–128
21. Abadi, M., Gordon, A.D.: A Bisimulation Method for Cryptographic Protocols. Nordic Journal of Computing **5** (1998) 267–303
22. Abadi, M., Blanchet, B.: Secrecy Types for Asymmetric Communication. In: Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'01). Volume 2030 of Lecture Notes in Computer Science., Springer-Verlag (2001)
23. Abadi, M.: Secrecy by Typing in Security Protocols. Journal of the ACM **46** (1999) 749–786
24. Gordon, A.D., Jeffrey, A.: Authenticity by Typing for Security Protocols. In: Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW'01), IEEE Computer Society Press (2001) 145–159
25. Guttman, J.D., Thayer, F.J.: Protocol Independence through Disjoint Encryption. In: Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW'00), IEEE Computer Society (2000) 24–34
26. Guttman, J.D., Thayer, F.J., Carlson, J.A., Herzog, J.C., Ramsdell, J.D., Sniffen, B.T.: Trust Management in Strand Spaces: a Rely-guarantee Method. In: Proceedings of the 13th European Symposium on Programming (ESOP'04). Volume 2986 of Lecture Notes in Computer Science., Springer-Verlag (2004) 325–339
27. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW'01), IEEE Computer Society (2001) 82–96

28. Blanchet, B., Podelski, A.: Verification of Cryptographic Protocols: Tagging Enforces Termination. Theoretical Computer Science **333** (2005) 67–90
29. Comon-Lundh, H., Cortier, V.: Security Properties : Two Agents are Sufficient. Science of Computer Programming **50** (2004) 51–71
30. Zhou, J., Gollmann, D.: Towards Verification of Non-repudiation Protocols. In: Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific, Springer-Verlag (1998) 370–380
31. Bella, G., Paulson, L.C.: Mechanical Proofs about a Non-repudiation Protocol. In: Proceedings of the 14th International Conference on Theorem Proving in Higher Order Logics(TPHOLs'01). Volume 2152 of Lecture Notes in Computer Science., Springer-Verlag (2001) 91–104
32. Bella, G., Paulson, L.C.: A Proof of Non-repudiation. In: Proceedings of the 9th International Workshop on Security Protocols (SPW'01). Volume 2467 of Lecture Notes in Computer Science., Springer-Verlag (2002) 119–125
33. Shmatikov, V., Mitchell, J.C.: Finite-state Analysis of Two Contract Signing Protocols. Theoretical Computer Science **283** (2002) 419–450

A A proof of Theorem 1

Theorem 1. *(Soundness and completeness) Let $\langle \epsilon, P\rangle$ be an initial configuration, and s be a concrete trace. $\langle \epsilon, P\rangle$ generates s, if and only if there exists $\hat{s}$, such that $\langle \epsilon, P\rangle \longrightarrow_p^* \langle \hat{s}, P'\rangle$ for some P', and s is a concretization of $\hat{s}$.*

Proof. "$\Rightarrow$": By an induction on the number of transitions $\longrightarrow$ and $\longrightarrow_p$, the proof is trivial in the zero-step. We assume in the n-th step the property holds. That is, for each trace s gained in the n-th $\longrightarrow$ step, there exists an $\hat{s}$ obtained by the n-th $\longrightarrow_p$ step, and $\hat{s}\vartheta = s$ holds for some substitution ϑ from variables to ground messages. Now, we perform a case analysis on the $n+1$ step:

1. Case $\langle s, 0\rangle$: Obviously.
2. Case $\langle s, a(x).P\rangle$: If $\langle s, a(x).P\rangle \longrightarrow \langle s.a(M), P\{M/x\}\rangle$, where M is a ground message, then we have $\langle \hat{s}, a(x).P'\rangle \longrightarrow_p \langle \hat{s}.a(x), P'\rangle$ and $s.a(M) = \hat{s}.a(x)(\vartheta \cup \{M/x\})$, where $P'\vartheta = P$. Thus $s.a(M)$ is a concretization of $\hat{s}.a(x)(\vartheta \cup \{M/x\})$.
3. Case $\langle s, \overline{a}M.P\rangle$: If $\langle s, \overline{a}M.P\rangle \longrightarrow \langle s.\overline{a}M, P\rangle$, then we have $\langle \hat{s}, \overline{a}M'.P'\rangle \longrightarrow_p \langle \hat{s}.\overline{a}M', P'\rangle$ and $M = M'\vartheta$, where $P'\vartheta = P$, since each variable in M' is already in the domain of ϑ. Thus $(\hat{s}.\overline{a}\,M')\vartheta = s.\overline{a}\,M$, and $s.\overline{a}\,M$ is a concretization of $\hat{s}.\overline{a}\,M'$.
4. Case $\langle s, let\ (x, y) = (M, N)\ in\ P\rangle$: We have $\langle s, let\ (x, y) = (M, N)\ in\ P\rangle \longrightarrow \langle s, P\{M/x, N/y\}\rangle$, and (M, N) is a ground message. The counterpart configuration is $\langle \hat{s}, let\ (x, y) = M'\ in\ P\rangle$, where $M'\vartheta = (M, N)$. Thus $\mathtt{Mgu}((x, y), M')$ will succeed and return a substitution θ, which satisfies $(x, y)\theta = M'\theta$. So $\langle \hat{s}, let\ (x, y) = M'\ in\ P'\rangle \longrightarrow_p \langle \hat{s}\theta, P'\theta\rangle$, where $P'\vartheta = P$. For each variable $x_1, \ldots, x_n$ both in domain θ and ϑ, we apply $\mathtt{Mgu}(\theta(x_i), \vartheta(x_i))$, which will return ground substitutions $\theta_1, \ldots, \theta_n$. Thus we have $s = (\hat{s}\theta)(\vartheta \backslash \{x_1, \ldots x_n\} \cup \theta_1, \cup \ldots \cup \theta_n)$.
5. Case $\langle s, case\ \{M\}_L\ of \{x\}_{L'}\ in\ P\rangle$: Similar to the above case.

6. Case $\langle s, [M = M]P\rangle$: If $\langle s, [M = M]P\rangle \longrightarrow \langle s, P\rangle$ and its counterpart configuration is $\langle \hat{s}, [M' = M'']P'\rangle$, where $P'\vartheta = P$, then $M'\vartheta = M''\vartheta = M$. Thus if $\theta = \texttt{Mgu}(M', M'')$, then $\theta \subseteq \vartheta$ since the θ is the most general unifier of M' and M'' and ϑ is a unifier of them. So we have $s\vartheta = (\hat{s}\theta)\vartheta$.
7. Case $\langle s, (\texttt{new}\ x)P\rangle$: Then we have $\langle s, (\texttt{new}\ x)P\rangle \longrightarrow \langle s, P\{M/x\}\rangle$ and $s \vdash_P M$. Its counterpart configuration is $\langle \hat{s}, (\texttt{new}\ x)P'\rangle$ and $s = \hat{s}(\vartheta \cup \{M/x\})$.
8. Other cases are obvious.

"$\Leftarrow$": By an induction on the number of transitions $\longrightarrow_p$ and $\longrightarrow$, the proof is trivial in the zero-step. We assume in the n-th step the property holds, that is, for each parametric trace $\hat{s}$ gained by the n-th $\longrightarrow_p$ step, if there exists a substitution ϑ from variables to ground messages, and a trace s that satisfies $s = \hat{s}\vartheta$, then s can be obtained by the n-th step of $\longrightarrow$. Now, we perform a case analysis on the $n+1$-th step:

1. Case $\langle \hat{s}, 0\rangle$: obviously.
2. Case $\langle \hat{s}, a(x).P\rangle$: If there exists a step in which $\langle \hat{s}, a(x).P\rangle \longrightarrow_p \langle \hat{s}.a(x), P\rangle$, and a ground substitution ϑ where $\hat{s}\vartheta$ is a trace, then $x\vartheta$ is a ground message which can be deduced by $s\vartheta$. So $\langle s, a(x).P'\rangle \longrightarrow \langle s.a(x\vartheta), P'\{\vartheta(x)/x\}\rangle$, where $P' = P\vartheta$.
3. Case $\langle \hat{s}, \overline{a}M.P\rangle$: If there exists a step in which $\langle \hat{s}, \overline{a}M.P\rangle \longrightarrow_p \langle \hat{s}.\overline{a}M, P\rangle$, and a ground substitution ϑ where $\hat{s}\vartheta$ is a concrete trace. So $\langle \hat{s}\vartheta, \overline{a}M\vartheta.P\vartheta\rangle \longrightarrow \langle (\hat{s}.\overline{a}M)\vartheta, P\vartheta\rangle$.
4. Case $\langle \hat{s}, let\ (x,y) = M\ in\ P\rangle$: We have $\langle \hat{s}, let\ (x,y) = M\ in\ P\rangle \longrightarrow_p \langle \hat{s}\theta, P\theta\rangle$ where $\theta = \texttt{Mgu}((x,y), M)$, and a ground substitution ϑ where $\hat{s}\theta\vartheta$ is a concrete trace. Thus $M\theta\vartheta$ is a ground pair message. Suppose it is described by (M', N'). So $\langle \hat{s}\theta\vartheta, let\ (x,y) = (M', N')\ in\ (P\theta\vartheta)\rangle \longrightarrow \langle \hat{s}\theta\vartheta, (P\theta\vartheta)\{M'/x, N'/y\}\rangle$.
5. Case $\langle \hat{s}, case\ \{x\}_L\ of M\ in\ P\rangle$: Similar to the above case.
6. Case $\langle \hat{s}, [M = M']P\rangle$: We have $\langle \hat{s}, [M = M']P\rangle \longrightarrow_p \langle \hat{s}\theta, P\theta\rangle$ where $\theta = \texttt{Mgu}(M, M')$, and a ground substitution ϑ where $\hat{s}\theta\vartheta$ is a trace. Thus $M\theta\vartheta = M'\theta\vartheta$ and both are ground messages. So we have $\langle \hat{s}\theta\vartheta, [M = M']P\theta\vartheta\rangle \longrightarrow \langle \hat{s}\theta\vartheta, P\theta\vartheta\rangle$.
7. Case $\langle \hat{s}, (\texttt{new}\ x)P\rangle$: If there exists a step in which $\langle \hat{s}, (\texttt{new}\ x)P\rangle \longrightarrow_p \langle \hat{s}, P\rangle$, and a ground substitution ϑ where $\hat{s}\vartheta$ is a concrete trace, then $x\vartheta$ is a ground message which can be P-deduced by $s\vartheta$. So $\langle s, (\texttt{new}\ x)P\rangle \longrightarrow \langle s, P\{\vartheta(x)/x\}\rangle$, where $P' = P\vartheta$.
8. Other cases are obvious.

B A proof of Theorem 2

Lemma 1. *If $\hat{s}$ is a parametric trace, and s is a concretization satisfying $s = \hat{s}\vartheta$ where ϑ is a concretized substitution, then $\hat{s}$ is either a normal form, or there exists an $\hat{s}'$ such that $\hat{s} \rightsquigarrow \hat{s}'$ with $\hat{s}\vartheta = \hat{s}'\vartheta$.*

Proof. If $\hat{s}$ is not a normal form, there exists some rigid message that is not contained in the elementary message set of its prefix. We perform case analysis on the kind of rigid messages $\{N\}_L$.

- If $\{N\}_L$ is an input rigid message in M, where $\hat{s} = \hat{s}'.a(M).\hat{s}''$. Thus $\{N\}_L \notin el(\hat{s}')$. Since $s = \hat{s}\vartheta$ and s is a trace, and thus $\hat{s}'\vartheta \vdash M\vartheta$, then $\{N\}_L\vartheta \in el(\hat{s}'\vartheta)$. By the definition of a rigid message, $L \notin el(\hat{s}')$, and thus $L\vartheta \notin el(\hat{s}')\vartheta$. Since $\{N\}_L\vartheta \in el(\hat{s}'\vartheta) = el(\hat{s}')\vartheta$, there exists $\{N'\}_L \in el(\hat{s}')$ such that $\{N\}_L\vartheta = \{N'\}_L\vartheta$. Thus $\{N\}_L$ and $\{N'\}_L$ are unifiable. Let $\hat{\rho} = \mathtt{Mgu}(\{N\}_L, \{N'\}_L)$, then $\hat{s} \rightsquigarrow \hat{s}\hat{\rho}$. Since $\{N\}_L\vartheta = \{N'\}_L\vartheta$, each corresponding variable in two messages will be assigned to the same ground message. Thus, $\hat{s}\vartheta = \hat{s}\hat{\rho}\vartheta$.
- If $\{N\}_L$ is an output rigid message in M, where $\hat{s} = \hat{s}'.\overline{a}M.\hat{s}''$. Thus L is not known by the principal P that contains the label a. Since $s = \hat{s}\vartheta$ and s is a trace, and thus $\hat{s}'\vartheta \vdash_P M\vartheta$. L is not known by P, so $\hat{s}'\vartheta \vdash M\vartheta$. Then following the similar proof for the input rigid message case, we can have $\hat{s}\vartheta = \hat{s}\hat{\rho}\vartheta$.

Lemma 2. *Let $\hat{s}$ be a parametric trace, and let $\hat{s}'$ be a normal form in $\mathtt{nf}_{\rightsquigarrow}(\hat{s})$. $\hat{s}'$ has a concretization, if and only if, for each decomposition $\hat{s}' = \hat{s}'_1.l(M).\hat{s}'_2$, where l is either an input label or an output label, each rigid message N in M satisfies $N \in el(\hat{s}'_1)$.*

Proof. "$\Rightarrow$": Prove by contradiction. Assume a normal form $\hat{s}'$ has concretizations s such that $s = \hat{s}'\vartheta$. If $\hat{s}'$ does not satisfy the first requirement, there exists at least one rigid message $\{N\}_L$ in $\hat{s}'$ that is not $\hat{\rho}$-unifiable in its prefix $\hat{s}'_1$. Thus $\{N\}_L\vartheta \notin el(\hat{s}'_1)\vartheta$. If it is an input rigid message, $\hat{s}'_1\vartheta \not\vdash L$, then $\hat{s}'_1\vartheta \not\vdash \{N\}_L\vartheta$. If it is an output rigid message, $\hat{s}'_1\vartheta \not\vdash_P \{N\}_L\vartheta$, where P is the process containing label a. This contradicts the definition of a trace.

"$\Leftarrow$": Let ϑ be an arbitrary concretized substitution that assigns each variable in $\hat{s}'$ to a name in $\mathcal{EN}$, then for each decomposition $\hat{s}'\vartheta = \hat{s}'_1\vartheta.a(M\vartheta).\hat{s}'_2\vartheta$, $\hat{s}'_1\vartheta \vdash M\vartheta$ is satisfiable. Thus $\hat{s}'\vartheta$ is a trace, and also a concretization of $\hat{s}'$.

Lemma 3. *Let $\hat{s}$ be a parametric trace, and s be a trace. s is a concretization of $\hat{s}$ if and only if s is a concretization of some $\hat{s}'$ with $\hat{s}' \in \mathtt{snf}_{\rightsquigarrow}(\hat{s})$.*

Proof. "$\Rightarrow$" If s is a concretization of $\hat{s}$, then there exists a concretized substitution ϑ with $s = \hat{s}\vartheta$. By Lemma 1 we can get either $\hat{s}$ is a normal form or $\hat{s}$ can be deduce to a parametric trace $\hat{s}'$ by $\rightsquigarrow$ such that $s = \hat{s}'\vartheta$. If $\hat{s}$ is a normal form and it has a concretization s, so $\hat{s}$ is also a satisfiable normal form according to Lemma 2 . If $\hat{s}$ is not a normal form, the number of rigid messages in $\hat{s}$ is finite, so there exists a normal form $\hat{s}''$ of $\hat{s}$ that satisfies $\hat{s}\vartheta = \hat{s}''\vartheta$ by repeatedly applying lemma 1. Since $\hat{s}'$ has the concretization s, $\hat{s}' \in \mathtt{snf}_{\rightsquigarrow}(\hat{s})$.

"$\Leftarrow$" If s is a concretization of the satisfiable normal form $\hat{s}'$ such that $\hat{s}' \in \mathtt{snf}_{\rightsquigarrow}(\hat{s})$, we have $s = \hat{s}'\vartheta$ for some concretized substitution ϑ. $\hat{s}'$ is a normal form of $\hat{s}$, so $\hat{s}' = \hat{s}\hat{\rho}$ for some $\hat{\rho}$, in which $s = \hat{s}'\vartheta = \hat{s}\hat{\rho}\vartheta$. Thus s is a concretization of $\hat{s}$.

Theorem 2. *A parametric trace $\hat{s}$ has a concretization if and only if $\mathtt{snf}_{\rightsquigarrow}(\hat{s}) \neq \emptyset$.*

The theorem is a corollary of Lemma 3.

C A proof of Theorem 3

Lemma 4. *Given a parametric trace $\hat{s}$,*

1. *$\hat{s} \models_t \alpha$ if and only if, α is $\hat{\rho}$-unifiable in $\hat{s}$, and for each satisfiable normal form in $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho})$ satisfying $\hat{s}\hat{\rho}\hat{\rho}'$, $\alpha\hat{\rho}\hat{\rho}'$ occurs in $\hat{s}\hat{\rho}\hat{\rho}'$.*
2. *$\hat{s} \models_t \neg\alpha$ if and only if $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho}) = \emptyset$ when α is $\hat{\rho}$-unifiable in $\hat{s}$.*
3. *For any state action term T, $\hat{s} \models_t T$ is decidable.*

Proof. Given a parametric trace $\hat{s}$,

1. "$\Rightarrow$": Prove by contradictions: Firstly, if α is not $\hat{\rho}$-unifiable in $\hat{s}$, then α cannot be $\hat{\rho}$-unifiable in all concretizations of $\hat{s}$. Thus given a concretization s in which α is not $\hat{\rho}$-unifiable, $\alpha\rho$ is not occurs in s for any ground substitution ρ. Thus $s \not\models_t \alpha$. So $\hat{s} \not\models_t \alpha$. Secondly, if α is $\hat{\rho}$-unifiable in $\hat{s}$, but there exists a satisfiable normal form in $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho})$ satisfying $\hat{s}\hat{\rho}\hat{\rho}'$, and $\alpha\hat{\rho}\hat{\rho}'$ does not occur in $\hat{s}\hat{\rho}\hat{\rho}'$, then for any concretization s' of $\hat{s}\hat{\rho}\hat{\rho}'$, $s' \not\models_t \alpha\hat{\rho}'$. According to Lemma 3, s' is also a concretization of $\hat{s}$ and thus $\hat{s} \not\models_t \alpha$.
   "$\Leftarrow$": If α is $\hat{\rho}$-unifiable in $\hat{s}$, and for each satisfiable normal form in $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho})$ satisfying $\hat{s}\hat{\rho}\hat{\rho}'$, $\alpha\hat{\rho}\hat{\rho}'$ occurs in $\hat{s}\hat{\rho}\hat{\rho}'$, Then for any concretization of $\hat{s}\hat{\rho}\hat{\rho}'\rho$, $\alpha\hat{\rho}\hat{\rho}'\rho$ is occurred. According to Lemma 3, each concretization of $\hat{s}$ is also a concretization of one of its satisfiable normal form. Thus for all concretization s of $\hat{s}$, $s \models_t \alpha$. So $\hat{s} \models_t \alpha$.
2. "$\Rightarrow$": Prove by contradictions: If α is $\hat{\rho}$-unifiable in $\hat{s}$, and $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho}) \neq \emptyset$, by Theorem 2, $\hat{s}\hat{\rho}$ has concretizations. We choose an arbitrary concretization s that satisfies $s = \hat{s}\hat{\rho}\vartheta$, and then $\alpha\hat{\rho}\vartheta$ occurs in s and thus $\hat{s} \not\models_t \neg\alpha$, which contradicts the assumption.
   "$\Leftarrow$": If α is $\hat{\rho}$-unifiable in $\hat{s}$ with $\mathtt{snf}_{\leadsto}(\hat{s}\hat{\rho}) = \emptyset$, by Theorem 2, $\hat{s}\hat{\rho}$ has no concretization. Since a concretization of $\hat{s}$ in which α is $\hat{\rho}$-unifiable is also a concretization of $\hat{s}\hat{\rho}$, then for each concretization s of $\hat{s}$, $s \models \neg\alpha$. Thus $\hat{s} \models_t \neg\alpha$.
3. It is easy to show that $\hat{s} \models T_1 \wedge T_2$ if and only if $\hat{s} \models T_1$ and $\hat{s} \models T_2$, and $\hat{s} \models T_1 \vee T_2$ if and only if $\hat{s} \models T_1$ or $\hat{s} \models T_2$. Furthermore, action terms satisfy De Morgan's laws. That is, $\hat{s} \models \neg(T_1 \wedge T_2)$ if and only if $\hat{s} \models \neg T_1 \vee \neg T_2$, and $\hat{s} \models \neg(T_1 \vee T_2)$ if and only if $\hat{s} \models \neg T_1 \wedge \neg T_2$. So any state action term problem can be checked on a parametric trace, Thus $\hat{s} \models_t T$ is decidable.

Lemma 5. *Given a concrete configuration $\langle s, P\rangle$, and two state action terms T_1 and T_2, $\langle s, P\rangle \models T_1 \hookrightarrow_F T_2$, if and only if for each concrete configuration $\langle s', P'\rangle$ reached by $\langle s, P\rangle$, if there exists a ground substitution ρ such that $s' \models_t T_1\rho$, then for each terminated configuration $\langle s'', P''\rangle$ reached by $\langle s', P'\rangle$, $T_2\rho$ occurs in s''.*

Proof. "$\Rightarrow$": By the definition, $\langle s, P\rangle \models T_1 \hookrightarrow_F T_2$ if for each concrete configuration $\langle s', P'\rangle$ reached by $\langle s, P\rangle$, if there is a ground substitution ρ such that $s' \models_t T_1\rho$, then for every path starting from $\langle s', P'\rangle$, there exists a concrete trace s'' such that $s'' \models_t T_2\rho$. According to the concrete transition rules in Fig. 3, if $\langle s, P\rangle$ generates a trace s', then $s' = s.s''$ for some s''. Thus, for every path starting from $\langle s', P'\rangle$, if there exists a concrete trace s'' such that $s'' \models_t T_2\rho$, then in the terminated configuration of that path $\langle s''', P'''\rangle$, $s''' \models T_2\rho$.

"$\Leftarrow$": The condition in the above lemma satisfies the definition of $\langle s, P\rangle \models T_1 \hookrightarrow_F T_2$.

Theorem 3. *Given an initial configuration $\langle \epsilon, P\rangle$,*

1. *Given a state action term T, $\langle \epsilon, P\rangle \models T$, if and only if for each parametric trace $\hat{s}$ generated by $\langle \epsilon, P\rangle$, $\hat{s} \models_t T$.*
2. *Given two state action terms T_1 and T_2, $\langle \epsilon, P\rangle \models T_2 \hookleftarrow T_1$, if and only if for each parametric trace $\hat{s}$ generated by $\langle \epsilon, P\rangle$, if T_1 is $\hat{\rho}$-unifiable in $\hat{s}$, then for each satisfiable normal form in $\mathtt{snf}_{\rightsquigarrow}(\hat{s}\hat{\rho})$ satisfying $\hat{s}\hat{\rho}\hat{\rho}'$, $T_2\hat{\rho}\hat{\rho}'$ occurs before $T_1\hat{\rho}\hat{\rho}'$.*
3. *Given two state action terms T_1 and T_2, $\langle \epsilon, P\rangle \models T_1 \hookrightarrow_F T_2$, if and only if for each parametric configuration $\langle \hat{s}', P'\rangle$ reached by $\langle \epsilon, P\rangle$, if T_1 is $\hat{\rho}$-unifiable in $\hat{s}'$, then for each terminated parametric configuration $\langle \hat{s}''\hat{\rho}, P''\hat{\rho}\rangle$ reached by $\langle \hat{s}'\hat{\rho}, P'\hat{\rho}\rangle$, either $\hat{s}''\hat{\rho}$ cannot deduce any satisfiable normal forms, or $T_2\hat{\rho}\hat{\rho}'$ occurs in each satisfiable normal form $\hat{s}''\hat{\rho}\hat{\rho}'$ in $\mathtt{snf}_{\rightsquigarrow}(\hat{s}'\hat{\rho})$.*

Proof. 1. It is a corollary of Theorem 1 and Lemma 4.

2. "$\Rightarrow$": Prove by contradictions: Given an arbitrary parametric trace $\hat{s}$ generated by $\langle \epsilon, P\rangle$, if T_1 is not $\hat{\rho}$-unifiable in $\hat{s}$, then $\hat{s} \not\models_t T_1$ according to Lemma 4. Thus for any concretization s of $\hat{s}$, $s \not\models_t T_1$. So $\langle \epsilon, P\rangle \not\models T_2 \hookleftarrow T_1$, which contradicts to our assumption. Otherwise, assume a satisfiable normal form $\hat{s}\hat{\rho}\hat{\rho}'$ in $\mathtt{snf}_{\rightsquigarrow}(\hat{s}\hat{\rho})$, and $T_2\hat{\rho}\hat{\rho}'$ does not occur before $T_1\hat{\rho}\hat{\rho}'$ in $\hat{s}\hat{\rho}\hat{\rho}'$. Let s' be a concretization of $\hat{s}\hat{\rho}\hat{\rho}'$ satisfying $s' = \hat{s}\hat{\rho}\hat{\rho}'\vartheta$. Thus $T_2\hat{\rho}\hat{\rho}'\vartheta$ does not occur before $T_1\hat{\rho}\hat{\rho}'\vartheta$ in s', that is, $s' \not\models T_2 \hookleftarrow T_1$. s' is also the concretization of $\hat{s}$. Thus $\hat{s} \not\models T_2 \hookleftarrow T_1$, which contradicts the assumption.
   "$\Leftarrow$": Given an arbitrary parametric trace $\hat{s}$ generated by $\langle \epsilon, P\rangle$, if T_1 is $\hat{\rho}$-unifiable in $\hat{s}$, and for each satisfiable normal form in $\mathtt{snf}_{\rightsquigarrow}(\hat{s}\hat{\rho})$ satisfying $\hat{s}\hat{\rho}\hat{\rho}'$, $T_2\hat{\rho}\hat{\rho}'$ occurs before $T_1\hat{\rho}\hat{\rho}'$ in $\hat{s}\hat{\rho}\hat{\rho}'$, then for each concretization satisfying $\hat{s}\hat{\rho}\hat{\rho}'\vartheta$, $\hat{s}\hat{\rho}\hat{\rho}'\vartheta \models T_2 \hookleftarrow T_1$. By Lemma 3, the concretization is also a concretization of $\hat{s}$. According to Theorem 1, any trace in $\langle \epsilon, P\rangle$ is a concretization of some counterpart parametric trace. Thus we have $\langle \epsilon, P\rangle \models T_2 \hookleftarrow T1$.
3. "$\Rightarrow$": Prove by contradiction: Given an arbitrary parametric trace $\hat{s}$ generated by $\langle \epsilon, P\rangle$, where T_1 is $\hat{\rho}$-unifiable in $\hat{s}'$, we suppose there exists a path from $\langle \hat{s}'\hat{\rho}, P'\hat{\rho}\rangle$ that will be terminated to $\langle \hat{s}''\hat{\rho}, P''\hat{\rho}\rangle$, and $T_2\hat{\rho}\hat{\rho}'$ does not occur in a satisfiable normal form $\hat{s}''\hat{\rho}\hat{\rho}'$ in $\mathtt{snf}_{\rightsquigarrow}(\hat{s}''\hat{\rho})$. Then for any ground substitution ρ, $\langle \hat{s}''\hat{\rho}\hat{\rho}'\rho, P''\hat{\rho}\hat{\rho}'\rho\rangle$ is a terminated configuration of $\langle \epsilon, P\rangle$ reached by $\langle \hat{s}'\hat{\rho}\rho, P'\hat{\rho}\rho\rangle$. $\hat{s}'\hat{\rho}\rho \models_t T_1\rho$, while $\hat{s}''\hat{\rho}\hat{\rho}'\rho \not\models_t T_2\rho$. By Lemma 5, $\langle \epsilon, P\rangle \not\models T_1 \hookrightarrow_F T_2$, which contradicts the assumption.
   "$\Leftarrow$": For each parametric trace $\hat{s}'$ generated by $\langle \epsilon, P\rangle$, if T_1 is $\hat{\rho}$-unifiable in $\hat{s}'$, then for each concretization s of $\hat{s}'$(if it has), satisfying $s = \hat{s}'\rho$, $s \models_t T_1\rho$. Furthermore, By the assumption, for each terminated parametric configuration $\langle \hat{s}''\hat{\rho}, P''\hat{\rho}\rangle$ reached by $\langle \hat{s}'\hat{\rho}, P'\hat{\rho}\rangle$, if it can reduce to some satisfiable normal form, according to Theorem 2, $\hat{s}'\hat{\rho}$ has concretization. For each satisfiable normal form, denoted by $\hat{s}''\hat{\rho}\hat{\rho}'$, $T_2\hat{\rho}\hat{\rho}'$ occurs after $T_1\hat{\rho}\hat{\rho}'$. Thus

$\langle \hat{s}''\hat{\rho}\hat{\rho}'\rho, P''\hat{\rho}\hat{\rho}'\rho\rangle$ is a terminated configuration, and $T_2\hat{\rho}\hat{\rho}'\rho$ occurs in $\hat{s}''\hat{\rho}\hat{\rho}'\rho$. According to Lemma 5, we have $\langle \epsilon, P\rangle \models T_1 \hookrightarrow_F T_2$.

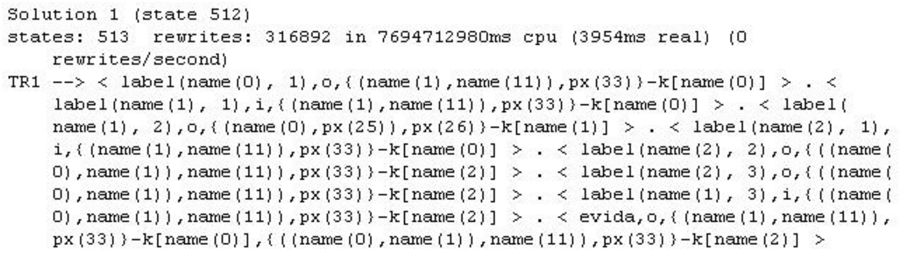
D A counterexample detected by Maude

Fig. 6 shows the snapshot of the result of NRO for simplified ZG protocol. In the counterexample, `name(0)`, `name(1)`, `name(2)`, `name(11)` denote names A, B, S, N_A, respectively, and `px(31)` denotes a variable that can substitute to any messages A generates. If before an input action, there exists an output action and two attached messages are same, it means two principals communicate by the two actions. Otherwise, it means there exists an intruder who communicates with a receiver by sending a fake message that the receiver cannot judge.

The counterexample actual represents the following attacks: A sent a message in flow (1') (x represents an arbitrary message A generates), then aborted the protocol after receiving a message by flow (2'), after that, an intruder (or B) pretended him to continue the protocol, sending the used message to S.

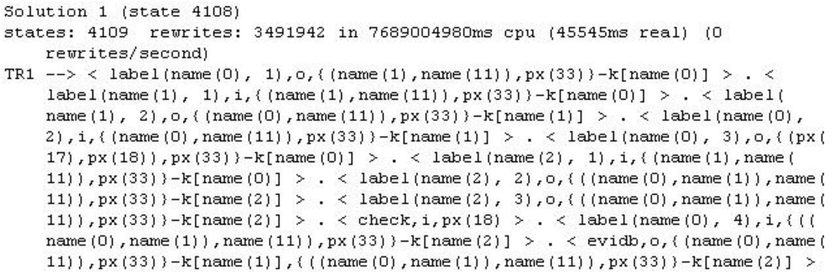
$$\begin{aligned}
A \longrightarrow B :&\quad \{B, N_A, x\}_{-K_A} && \text{(1')}\\
B \longrightarrow A :&\quad \{A, N_A, x\}_{-K_B} && \text{(2')}\\
I(A) \longrightarrow S :&\quad \{B, N_A, x\}_{-K_A} && \text{(3')}\\
S \longrightarrow I(A) :&\quad \{A, B, N_A, x\}_{-K_S} && \text{(4')}\\
S \longrightarrow B :&\quad \{A, B, N_A, x\}_{-K_S} && \text{(5')}
\end{aligned}$$

In FAIRO, the same counterexample is also detected. Similarly, a counterexample for FAIRM is also detected automatically. The result snapshot is in Fig. 7.



```
Solution 1 (state 512)
states: 513  rewrites: 316892 in 7694712980ms cpu (3954ms real) (0
    rewrites/second)
TR1 --> < label(name(0), 1),o,{(name(1),name(11)),px(33)}-k[name(0)] > . <
    label(name(1), 1),i,{(name(1),name(11)),px(33)}-k[name(0)] > . < label(
    name(1), 2),o,{(name(0),px(25)),px(26)}-k[name(1)] > . < label(name(2), 1),
    i,{(name(1),name(11)),px(33)}-k[name(0)] > . < label(name(2), 2),o,{((name(
    0),name(1)),name(11)),px(33)}-k[name(2)] > . < label(name(2), 3),o,{((name(
    0),name(1)),name(11)),px(33)}-k[name(2)] > . < label(name(1), 3),i,{((name(
    0),name(1)),name(11)),px(33)}-k[name(2)] > . < evida,o,{(name(1),name(11)),
    px(33)}-k[name(0)],{((name(0),name(1)),name(11)),px(33)}-k[name(2)] >
```

Fig. 6. Snapshot of Maude result for NRO of the simplified ZG protocol



```
Solution 1 (state 4108)
states: 4109  rewrites: 3491942 in 7689004980ms cpu (45545ms real) (0
    rewrites/second)
TR1 --> < label(name(0), 1),o,{(name(1),name(11)),px(33)}-k[name(0)] > . <
    label(name(1), 1),i,{(name(1),name(11)),px(33)}-k[name(0)] > . < label(
    name(1), 2),o,{(name(0),name(11)),px(33)}-k[name(1)] > . < label(name(0),
    2),i,{(name(0),name(11)),px(33)}-k[name(1)] > . < label(name(0), 3),o,{(px(
    17),px(18)),px(33)}-k[name(0)] > . < label(name(2), 1),i,{(name(1),name(
    11)),px(33)}-k[name(0)] > . < label(name(2), 2),o,{((name(0),name(1)),name(
    11)),px(33)}-k[name(2)] > . < label(name(2), 3),o,{((name(0),name(1)),name(
    11)),px(33)}-k[name(2)] > . < check,i,px(18) > . < label(name(0), 4),i,{((
    name(0),name(1)),name(11)),px(33)}-k[name(2)] > . < evidb,o,{(name(0),name(
    11)),px(33)}-k[name(1)],{((name(0),name(1)),name(11)),px(33)}-k[name(2)] >
```

Fig. 7. Snapshot of Maude result for FAIRM of the simplified ZG protocol