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 evidence while the other principals cannot do so. This paper extends the model in our previous work [12], and gives a sound and complete on-the-fly model checking method for fair non-repudiation protocols under the assumption of a 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 a reliable exchange of messages in the situation that each principal can be dishonest, who tries to take advantage of 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 principals cannot do so. Difficulties in verifying these security properties come from various factors of infinity,

- 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 [7].
- 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 a bounded number of sessions. This method is based on trace analysis. 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 [1]. 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 [12].
- Following the Dolev-Yao model, a deductive system, which can generate infinitely many messages, is exploited to describe abilities of intruders [3].
- Another deductive system is introduced to generate infinitely many messages that dishonest principals may produce and send [17].

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.

Due to the lack of space, we omit proofs of lemmas and theorems; these can be found in the extended version [13].

2 Concrete model for protocol description

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.

A binder, $\mathtt{m}[pr_1, \ldots, pr_n]$ is a message that can be regarded as a special name indexed by its parameters. One usage of binders is to denote encryption keys. For instance, $+\mathtt{k}[A]$ and $-\mathtt{k}[A]$ represent A's public key and private key, respectively.

Definition 2 (Processes). *Processes in a set $\mathcal{P}$ are defined as follows:*

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

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.

A new process, summation $P+Q$ that behaves like P or Q, intends a dishonest principal that has several choices, such as aborting communication, or running a recovery stage.

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, and intruders' names. The environmental deductive system is shown in Fig. 1.

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

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

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

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

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. 2.

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

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.*

The transition relation of concrete configurations is defined by the rules listed in Fig. 3. Two symmetric forms, $(RSUM)$ of $(LSUM)$, and $(RCOM)$ of $(LCOM)$ are omitted from the figure. Furthermore, a function $\mathtt{Opp}$ is defined for complemental key in decryption and encryption. Thus we have $\mathtt{Opp}(+\mathtt{k}[A]) = -\mathtt{k}[A]$, $\mathtt{Opp}(-\mathtt{k}[A]) = +\mathtt{k}[A]$ and $\mathtt{Opp}(\mathtt{k}[A,B]) = \mathtt{k}[A,B]$.

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

Fig. 3. Concrete transition rules

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 protocols

For simplicity of representation, we use several convenient abbreviations. Pair splitting is applied to input and decryption.

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

Similarly, we write $let\ (x_1, x_2, \ldots, x_n) = M\ in\ P$, $a(x_1, x_2, \ldots, x_n).P$, and $case\ M\ of\ \{x_1, x_2, \ldots, x_n\}_L\ in\ P$ for tuples of messages.

We will use a simplified variation of Zhou-Gollmann non-repudiation protocol to illustrate how our system works. The full ZG protocol is proposed in [19]. Note that besides a standard flow description, a fair non-repudiation protocol also contains a description on what are evidences for participated principals.

$$A \longrightarrow B: \quad \{B, N_A, \{M\}_K\}_{-K_A} \tag{1}$$
$$B \longrightarrow A: \quad \{A, N_A, \{M\}_K\}_{-K_B} \tag{2}$$
$$A \longrightarrow S: \quad \{B, N_A, K\}_{-K_A} \tag{3}$$
$$S \longrightarrow A: \quad \{A, B, N_A, K\}_{-K_S} \tag{4}$$
$$S \longrightarrow B: \quad \{A, B, N_A, K\}_{-K_S} \tag{5}$$

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).

Fresh variables are used to denote the sub-messages that the principal can use to deceive another principal. These variables are bound by the `new` primitive. After receiving messages, each principal may abort the communication. Thus a summation " $+$ " is used to represent nondeterministic choices of a principal.

$$
\begin{aligned}
A \triangleq\ & (\nu N_A)(\mathtt{new}\, x_1, x_2)\, \overline{a1}\{x_1, N_A, x_2\}_{-\mathtt{k}[A]}.a2(x_3).\\
& \mathit{case}\ x_3\ \mathit{of}\ \{x_4, x_5, x_6\}_{+\mathtt{k}[x_1]}\ \mathit{in}\ [x_4 = A]\,[x_5 = N_A]\,[x_6 = x_2]\,(\mathbf{0}+\\
& (\mathtt{new}\, x_7, x_8)\, \overline{a3}\{x_7, x_8\}_{-\mathtt{k}[A]}.a4(x_9).\ \mathit{case}\ x_9\ \mathit{of}\ \{x_{10}, x_{11}, x_{12}, x_{13}\}_{+\mathtt{k}[S]}\\
& \mathit{in}\ [x_{10} = A]\,[x_{11} = x_1]\,[x_{12} = N_A]\,[x_{13} = x_8].\mathbf{0})\\
B \triangleq\ & b1(y_1).\mathit{case}\ y_1\ \mathit{of}\ \{y_2, y_3, y_4\}_{+\mathtt{k}[A]}\ \mathit{in}[y_2 = B]\,(\mathbf{0}+\\
& (\mathtt{new}\, y_5)\, \overline{b2}\{A, y_5\}_{-\mathtt{k}[B]}.b3(y_6).\mathit{case}\ y_6\ \mathit{of}\ \{y_7, y_8, y_9, y_{10}\}_{+\mathtt{k}[S]}\ \mathit{in}\\
& [y_7 = A]\,[y_8 = B]\,[y_9 = y_3].\mathbf{0})\\
S \triangleq\ & s1(z_1).\mathit{case}\ z_1\ \mathit{of}\ \{z_2\}_{+\mathtt{k}[z_3]}\ \mathit{in}\ \overline{s2}\{z_3, z_2\}_{-\mathtt{k}[S]}.\overline{s2}\{z_3, z_2\}_{-\mathtt{k}[S]}.\mathbf{0}\\
SYS^{ZG} \triangleq\ & A \| B \| S
\end{aligned}
$$

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 formed of P composed with a process $c(x).\mathbf{0}$ with a fresh label c (named guardian process), that is, $P \| c(x).\mathbf{0}$.*

Intuitively, declaration process insertion is used to show that a principal can provide a message M at the end of the session. Guardian process composition is used to check whether a message is observable in the environment.

3.3 Action terms

Definition 5. *Let α range over the set of actions. Action terms are defined as follows:*

$$
\begin{aligned}
T &::= \alpha \mid \neg T \mid T \wedge T \mid T \vee T\\
\sigma &::= T \mid T \hookleftarrow T \mid T \hookrightarrow_F T
\end{aligned}
$$

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 security properties

For the simplified ZG protocol, evidences M_1 and M_2 in Section 3.1 correspond to the two non-repudiation properties [21, 9], respectively.

- *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.

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. $\overline{evid}_B(x_3, x_9).\mathbf{0}$).

Each process may have several action paths, since it may contain the summation $+$. The probing transformation replaces $\mathbf{0}$ reachable by paths containing both $b1$ and $b3$ (resp. $a2$ and $a4$) with the declaration process.

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

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

Fig. 4. Parametric transition rules

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

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

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

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

4 Parametric simulation

All messages in concrete traces generated by transition rules in Fig. 3 are ground. In this section, parametric traces, in which irrelevant messages to a protocol execution are replaced with free variables, are presented.

4.1 Parametric model

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.*

The transition relation of parametric configurations [3] is given by the rules listed in Fig. 4. Two symmetric forms ($PRSUM$) of ($PLSUM$), and ($PRCOM$) of ($PLCOM$) are omitted from the figure. A function $\mathtt{Mgu}(M_1, M_2)$ returns the *most general unifier* of M_1 and M_2.

Definition 7 (Concretization and abstraction). *Given a parametric trace $\hat{s}$, if there exists a substitution ϑ that assigns each parametric 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 it an *initial configuration*. From an initial configuration, each concrete trace has an abstraction generated by parametric transition rules. On the other hand, if a parametric trace has a concretization, then the concretization is generated by concrete transition rules. Otherwise the parametric trace cannot be instantiated to any concrete trace.

Theorem 1. *(Soundness and completeness) Let $\langle \epsilon, P \rangle$ be an initial configuration, and let 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 sends a message $\{A, M\}_{K_{AB}}$ to B. There exists a parametric trace $b1(\{A, x\}_{\mathtt{k}[A,B]})$. Since $\mathtt{k}[A, B]$ was not leaked to 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*. 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 satisfying the requirement, the parametric trace has no concretizations.

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

- *M is included 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 a rigid message, or at least one name or binder in N, which cannot be deduced by the $\hat{s}'$, and $\hat{s}' \not\vdash \{N\}_L$.*
- *M is included 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 $\hat{s}$.

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 9 (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 $\mathtt{nf}_{\rightsquigarrow}(\hat{s})$. "A parametric trace has concretizations" is equivalent to there exists a parametric trace in its $\mathtt{nf}_{\rightsquigarrow}(\hat{s})$ that has concretizations.

Lemma 1. *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 1. $\mathtt{snf}_{\rightsquigarrow}(\hat{s})$ denotes the set of *satisfiable normal forms* of $\hat{s}$.

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

4.3 Simulation on a parametric model

Definition 10. *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 11. *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 2. *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 $\mathtt{snf}_{\rightsquigarrow}(\hat{s}\hat{\rho})$ 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 $\mathtt{snf}_{\rightsquigarrow}(\hat{s}\hat{\rho}) = \emptyset$ when α is $\hat{\rho}$-unifiable in $\hat{s}$.*
3. *For any state action term T, $\hat{s} \models_t T$ is decidable.*

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 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 $\mathtt{snf}_{\rightsquigarrow}(\hat{s}'\hat{\rho})$.*

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

5 Experimental results

We implement the on-the-fly model checking method using Maude [6], 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.

Due to the space limitation, we have only explained the non-repudiation property. Fairness for fair non-repudiation protocols [17, 9] that is classified into FAIRO, FAIRR, and FAIRM, is presented in the extended version [13].

In experiments with one session bound, the attacks for NRO, FAIRO and FAIRM of simplified ZG protocol were detected automatically. 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 ISO [8].

The results are summarized in Fig. 5, in which the 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.

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

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

6 Related work

Gavin Lowe first used trace analysis on process calculus CSP, and implemented a model-checker FDR to discover numerous attacks [14, 15]. In his work, the intruder was represented as a recursive process. He restricted the state space to

[1] For formal definitions of the properties of full ZG protocol, refer to [17].

be finite by imposing upper-bounds upon messages generated by intruders, and also upon the number of principals in the network.

Many of our ideas are inspired by Michele Boreale's symbolic approach [3]. In his research, he restricted the number of principals and intruders, and represented that each principal explicitly communicates with an intruder. Our model finitely describes an unlimited number of principals and intruders in the network.

David Basin et al. proposed an on-the-fly model checking method (OFMC) [4]. In their work, an intruder's messages are instantiated only when necessary, known as *lazy intruder*, which is similar to the use of a rigid message in our model. Unlike our method, an intruder's role is explicitly assigned. This is efficient, but the process needs to be performed several times to ensure that no intruders can attack a protocol in any roles.

Schneider proposed a trace analysis to prove non-repudiation and fairness properties of the ZG protocol based on CSP [17]. 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 et al. proposed several non-repudiation protocols, and proved their correctness by SVO logic in their papers and book [19–21]. We use their definition for non-repudiation in this paper.

G. Bella and L. Paulson extended their previous Isabelle/HOL theorem proving approach for authentication property [2, 16] to the ZG protocol, and proved the correctness of its non-repudiation and fairness properties [5]. 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 several protocols, and also summarized and compared many formal definitions of fairness in his thesis [11]. Recently, D. Kähler et al. proposed a more powerful AMC-model checking method for verifying the fairness property [10].

V. Shmatikov et al. analyzed fairness of two contract signing protocols based on a finite-state model checker Murφ [18]. His model was limited to a bounded number of sessions and principals, and bounded number of messages that an intruder generates. We have released the bounds for principals and messages, using a parametric abstraction on an unlimited 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 a bounded number of sessions. It extended our previous work [12] to handle all infinity factors 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 for 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.

Acknowledgements The authors thank Dr. Yoshinobu Kawabe for discussions. 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. Abadi, M., Gordon, A. D.: A Calculus for Cryptographic Protocols: The Spi Calculus. In: Proceedings of the CCS'97. ACM Press (1997) 36–47
2. Bella, G.: Inductive Verification of Cryptographic Protocols. PhD thesis, University of Cambridge (2000)
3. Boreale, M.: Symbolic Trace Analysis of Cryptographic Protocols. In: Proceedings of the ICALP'01. LNCS 2076, Springer-Verlag (2001) 667–681
4. Basin, D., Mödersheim, S., Viganò, L.: OFMC: A Symbolic Model Checker for Security Protocols. International J. of Information Security **4(3)** (2005) 181–208
5. Bella, G., Paulson, L. C.: A Proof of Non-repudiation. In: Proceedings of the SPW'01. LNCS 2467, Springer-Verlag (2002) 119–125
6. Clavel, M., Durán, F., Eker, S., Lincolnand, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: Maude Manual. http://maude.cs.uiuc.edu/maude2-manual/ (2005)
7. Dolev, D., Yao, A. C.: On the Security of Public Key Protocols. IEEE Trans. on Information Theory **29** (1983) 198–208
8. ISO/IEC13888: Information Technology - Security Techniques - Non-repuduation - Part 1–Part 3. (1997)
9. Kremer, S., Markowitch, O., Zhou, J.: An Intensive Survey of Fair Non-repudiation Protocols. Computer Communications **25** (2002) 1606–1621
10. Käehler, D., Küesters, R., Truderung, T.: Infinite State AMC-Model Checking for Cryptographic Protocols. In: Proceedings of the LICS'07. (2007)
11. Kremer, S.: Formal Analysis of Optimistic Fair Exchange Protocols. PhD thesis, Universite Libre de Bruxelles (2003)
12. Li, G., Ogawa, M.: On-the-fly Model Checking of Security Protocols and Its Implementation by Maude. IPSJ Trans. on Programming **48**, SIG 10, (2007) 50–75
13. Li, G., Ogawa, M.: On-the-fly Model Checking of Fair Non-repudiation Protocols (extended version). http://www.jaist.ac.jp/~guoqiang/Fullnon.pdf (2007)
14. Lowe, G.: Breaking and Fixing the Needham-Schroeder Public-key Using FDR. In: Proceedings of the TACAS'96. LNCS 1055, Springer-Verlag (1996) 147–166
15. Lowe, G.: Some New Attacks upon Security Protocols. In: Proceedings of the CSFW'96. IEEE Computer Society Press (1996) 162–169
16. Paulson, L.C.: The Inductive Approach to Verifying Cryptographic Protocols. J. of Computer Security **6** (1998) 85–128
17. Schneider, S.: Formal Analysis of a Non-repudiation Protocol. In: Proceedings of the CSFW'98. IEEE Computer Society Press (1998) 54–65
18. Shmatikov, V., Mitchell, J. C.: Finite-state Analysis of Two Contract Signing Protocols. Theoretical Computer Science **283** (2002) 419–450
19. Zhou, J., Gollmann, D.: A Fair Non-repudiation Protocol. In: Proceedings of the S&P'96. IEEE Computer Society Press (1996) 55–61
20. 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
21. Zhou, J.: Non-repudiation in Electronic Commerce. Artech House (2001)