Parameterised anonymity

Jan Friso Groote and Simona Orzan

Department of Computer Science, Eindhoven University of Technology,
P.O. Box 513, NL-5600MB, Eindhoven, The Netherlands

Abstract. We introduce the notion of parameterised anonymity, to formalize the anonymity property of protocols with an arbitrary number of participants. This definition is an extension of the well known CSP anonymity formalization of Schneider and Sidiropoulos [23]. Using recently developed invariant techniques for solving parameterised boolean equation systems, we then show that the Dining Cryptographers protocol guarantees parameterised anonymity with respect to outside observers. We also argue that although the question whether a protocol guarantees parameterised anonymity is in general undecidable, there are practical subclasses where anonymity can be decided for any group of processes.

1 Introduction

Anonymity, as a security property, refers to the ability of a user to own some data or take some actions without being tracked down as the owner of that data or the originator of those actions. This property is essential in group protocols that might involve sensitive personal data, like electronic auctions, voting, anonymous broadcasts, file-sharing etc. Due to its relevance and subtle nature, anonymity has been given many definitions [1,11,12,22] and has been the subject of many theoretical studies and formal analysis work [13,19]. Protocols where anonymity is one of the aims are typically meant for large groups of users. However, formal verification of anonymity only treat (small) examples of individual protocols [17,23,24] and claims about the correctness of anonymity protocols *for any group size* are generally not made.

In this paper, we propose a parameterised possibilistic definition of anonymity based on a notion of secret choices of participants. representing actions (e.g., accessing a certain web server) or data (e.g., votes in a voting protocol). The main inspiration is the CSP definition in [23], where anonymity is formalized as the impossibility of an observer to distinguish a protocol behaviour where a participant i acts according to a choice c from a protocol behaviour where i has taken a different choice d.

We then give a formal correctness proof for Chaum's Dining Cryptographers protocol [2], arguably the most well-known example of a protocol where anonymity is the main requirement. Starting with [23], where DC has been proved correct for 3 cryptographers, various verification approaches, both process theoretical and logical, have been applied to it, e.g. [23,1,17] — but only *for concrete instances*, the maximum instance being as large as 15 cryptographers [4]. No formal proof exists so far for an arbitrary number of parties, although a mathematical argument has already been given by Chaum in the original paper. We use a recently developed theory where standard verification problems like model checking and equivalence checking are encoded as *parameterised boolean equation systems (PBES)* [10]. PBESs are usually solved by symbolic approximation and by discovering equation patterns and invariants [10,21]. In solving the PBES corresponding to the DC protocol, we make essential use of invariants.

We also formulate the *parameterised anonymity problem* for $n \geq N$ processes (NPA) as the problem of deciding whether, for every instance of a group protocol description, a different instance can be found whose observable behaviour is indistinguishable from that of the first one. We show that this is undecidable in general, but practically usable subsets exists where anonymity can be decided for groups of any size $n \geq N$.

We are aware of only one other previous anonymity proof for an arbitrary number of parties. There [13], the pi-calculus has been used for modeling and the correctness argument is based on the congruence of observational equivalence w.r.t. the parallel composition operator. This approach works essentially due to the absence of communication in the model. The matters get much more complex when communication is involved, as illustrated also in the current paper. A basic referendum protocol (much simpler than FOO) has been briefly analysed using PBESs in [21].

Decidability of the traditional security properties *secrecy* and *authentication* has been extensively studied and well understood in various models - atomic or complex keys, Dolev-Yao intruder with (un)bounded message size, (dis)allow equality tests etc. [15], and also for the multiparty case [14]. Recently, the need to answer decidability questions for other security properties like *anonymity*, *privacy*, *fairness* etc. was recognized [5] and gained interest. For the case of two-party protocols, effectiveness, fairness and balance of contract-signing is decidable [16], as well as a property related to anonymity, called *opacity* [20].

We start with preliminaries regarding the formal language used and PBESs (Section 2). Section 3 presents our anonymity formalization, Section 4 the DC correctness proof for it and Section 5 the (un)decidability results.

2 Preliminaries

A short introduction to mCRL2. mCRL2 is a process algebraic specification language with data types [8,9]. Processes are built from atomic multi-actions (e.g. $a|b|c$ is a multi-action where actions a, b and c happen simultaneously). Behaviour is combined by the process algebraic operators for sequential composition ($\cdot$), non-deterministic choice ($+$) and parallel compositions ($\|$). There are two special processes: the deadlock δ and the internal action τ.

Actions of parallel processes lead to multi-actions if they happen simultaneously. The *communication operator* Γ_C is used to let such actions communicate. E.g., if $\Gamma_{\{a|b \to c\}}$ is applied to a multi-action $a|b$, it becomes c. If data is present such communication is only possible if the processes carry the same data arguments. The *allow operator* $\nabla_V(p)$ allows only the multi-actions from V occurring in p to happen. The renaming operator $\rho_R(p)$ where R is a function from actions to actions renames the actions in p according to R. The *hiding* operator τ_I turns all occurrences of actions from the set I into the internal action τ.

There are a number of ways to tie processes up with data types. First, both processes and atomic actions can be parameterised with data elements, as in $P(x, 3)$ or $\texttt{send}(x)$. Then, $\sum_{x \in D} P(x)$ denotes alternative (possibly infinite) choice over data domain D, i.e. for any value $x_0 \in D$, the process can behave as $P(x_0)$. Finally, if b is a boolean term and p and q are processes, then the conditional construct $b \to p \diamond q$ is the process 'if b then p else q'.

PBES. A parameterised boolean equation is a fixed-point equation having as left-hand side a predicate variable with data parameters and as right-hand side a defining *predicate formula*, which is a boolean expression with quantifiers and predicate variables. Formally, a predicate formula is defined by the grammar

$$\phi ::= b \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \forall d{:}D.\ \phi \mid \exists d{:}D.\ \phi \mid X(e),$$

where b is a data term of Boolean sort $\mathbb{B}$, possibly containing data variables $d \in \mathcal{D}$. Furthermore, X (taken from some domain of variables $\mathcal{P}$) is a (sorted) predicate variable of sort D_X and e is a vector of data terms of the sort D_X. The data variables occurring in a predicate formula are taken from a set $\mathcal{D}$.

Parameterised boolean equation systems (PBES) [10] provide a fundamental framework for solving verification problems. They can encode model checking questions [10], checking of various process equivalences [3], static analysis problems [7] etc. The PBES solution is then the solution to the encoded problem. The basic PBES solving techniques are successive symbolic approximation of the system's equations and instantiation of the data parameters.

PBES invariants. In general, due to their complexity, it is not possible to solve PBESs automatically, but heuristics and interactive approaches are necessary. *Invariants* are predicates over data only (thus, no predicate variables involved), expressing a fixed relation between data variables. An invariant of a predicate variable X provides in fact an overapproximation of X's solution. In a recent extended study [21], many useful results and examples on the characterisation and use of invariants are discussed. For the current paper, we restrict those to the specific case of a PBES with one maximal fixed-point (ν) equation. An invariant for a parameterised boolean equation is defined formally as follows:

Definition 1. *Let $(\sigma X(d{:}D_X) = \phi)$ be an equation and let f be a simple predicate formula. Then f* is an invariant of *X iff*

$$f \wedge \phi \;\leftrightarrow\; (f \wedge \phi)[(f(e) \wedge X(e))/X(e)],$$

where $\leftrightarrow$ denotes boolean bi-implication and $[(f(e) \wedge X(e))/X(e)]$ denotes the substitution of all occurrences of X's instantiations $X(e)$ (for all possible data parameters e) with $f(e) \wedge X(e)$.

We make use of two results about invariants, proven in [21]. The formulations below are restricted to our current context, namely an equation system with one maximal fix point equation. The results in [21] refer to equation systems with multiple variables and to global invariants.

A sufficient condition for a simple predicate to be an invariant for an equation is to transfer its truth from the data parameters on the left-hand side to the data parameters on the right-hand side of the equation. Proposition 1 is a variation of this general principle.

Proposition 1 (from [21]). *Let $(\sigma X(d{:}D_X) = \phi)$ be an equation, with $\sigma \in \{\mu, \nu\}$ and ϕ of the form*

$$\phi(d) = \chi \wedge \bigwedge_{\star \in I} (\psi_\star \Rightarrow X(g_k(d))),$$

for I some index set, $\Rightarrow$ denoting the usual logical shortcut, χ, $\psi_\star$ simple predicate formulae and $g_\star(d)$ data terms. Moreover, let $f \in \mathcal{P}$ be a simple predicate such that, for all $\star \in I$, $f(d) \wedge \chi \wedge \psi_\star \Rightarrow f(g_\star(d))$. Then f is an invariant for the given equation.

Definition 2. *Let $\mathcal{E} \equiv (\sigma X(d{:}D_X) = \phi)$ be an equation and f an invariant for it. $\mathcal{E}$* strengthened with *f is the equation $\mathcal{E}' \equiv (\sigma X(d{:}D_X) = f \;\wedge\; \phi)$.*

Equations strengthened with well-chosen invariants are (much) easier to solve than the original equations. The main theorem in [10,21] identifies the conditions that ensure preservation of PBES solutions under strengthening with invariants. Instantiated for one equation $\mathcal{E}$, this theorem states that the solution of $\mathcal{E}$ coincides with the solution of $\mathcal{E}$ strengthened with an invariant f, on data domains satisfying f. This provides the technical base for the invariant-driven approach to equation solving. More specifically, once a good invariant is found, that characterizes the reachable data space without restricting it to uninteresting subsets, the strengthened version of an equation can be solved instead of the original one. In some specific cases, when an equation fits certain patterns, invariants can immediately lead to its solution. Such a pattern is described below.

Proposition 2 (from [21]). *Let $\mathcal{E} \equiv (\nu X(d{:}D_X) = \phi)$ be an equation. Let f be an invariant for $\mathcal{E}$ and assume ϕ has the form ($\mathsf{Q}_l \in \{\forall, \exists\}$ for any l):*

$$f \wedge \bigwedge_{\star \in I} \mathsf{Q}_1\, e^1_\star{:}E^1_\star \ldots \mathsf{Q}_{m_\star}\, e^{m_\star}_\star{:}E^{m_\star}_\star.\ (\psi_\star \Rightarrow X(g_\star(d, e^1_\star, \ldots, e^{m_\star}_\star)))$$

where, for any $\star \in I$, $\psi_\star$ is a simple predicate formula and $g_\star$ is a data term that depends only on the values of d and $e^1_i, \ldots, e^{m_i}_i$. Then X has the solution f.

The Dining Cryptographers protocol. We give here an informal description of this protocol, a metaphor for anonymous broadcast. The story goes as follows: n cryptographers have dinner together. At the end, they learn that the bill has been payed anonymously by one of them, or by the NSA (National Security Agency). They wish to find out whether the payer was NSA or not, but if the payer was one of the cryptographers, nobody should learn her identity. To achieve this, they use the following protocol: each neighbouring pair of cryptographers generates a shared bit, by flipping a coin; then each cryptographer computes the exclusive or (XOR, denoted $\oplus$) of the two random bits she shares with her neighbors. Then, if she hasn't paid, she publicly announces the result. If she was herself the payer, she announces the flipped result. Every cryptographer collects all the announcements and XORs them. The result indicates whether the payer was an insider or not - true ($\top$) means cryptographer, false ($\bot$) means NSA.

3 A parameterised formalization of anonymity

We give a formal scalable notion of anonymity, using the process algebra specification language mCRL2 and taking inspiration in existing process theoretic definitions like the one using CSP in [23]. We take the general view that anonymity for a participant means hiding parts of his behaviour or data from possible observers. We consider a *passive intruder*, who observes protocol runs but doesn't have the power to change its course.

Group protocols like the ones we're interested in can usually be written as a parallel composition of n parties and an environment process:

$$\texttt{Protocol}(x) \overset{def}{=} \tau_I \rho_R \nabla_V \Gamma_C(P(0, x_0) \| P(1, x_1) \| \cdots \| P(n{-}1, x_{n-1}) \| Q(n)) \tag{1}$$

where $x = (x_0, x_1, \ldots, x_{n-1})$ is a vector of secret choices. The parameters x_i come from a known, usually small, domain D. The processes $P(i, x_i)$ represent the behaviour of participant i and the process $Q(n)$ is some environmental process. The operator Γ_C prescribes the communications among the processes. The set C contains clauses of the form $a|b{\to}c$ expressing that actions a and b must communicate to c provided the parameters of a and b are equal. The allow operator ∇_V with V a set of multi-actions each of the form $a_1|\cdots|a_n$, shows which multi-actions are visible. All multi-actions that are not in V are blocked. The renaming operator ρ_R is used to give certain actions a different name. Finally, the operator τ_I renames actions in the set I to τ, effectively hiding them.

A typical instance of a protocol where anonymity is desired, is a voting protocol, where x_i represents i's vote. The anonymity refers to the link between x_i and i. The essence is that the intruder should not be able to conclude from his observations, that x_i must be i's secret choice. More precisely, the intruder should not be able to observe any difference in the behaviours the protocol with different values for x_i. This is formulated in the following definition.

Definition 3 (anonymity). *Let* `Protocol` *be the formal specification of a protocol, D the domain of secret choices, Restriction* : $D^* \to \{\top, \bot\}$ *a predicate on arrays of D elements,* $\sim$ *a process equivalence modeling the intruder's observing power. We say that* `Protocol` is anonymous for participant i out of n *iff*

$$\begin{array}{l} \forall x \in D^n \;\; \textit{with Restriction}(x) \\ \quad \exists v \in D^n \;\; \textit{s.t. Restriction}(v),\; v_i \neq x_i \; \textit{and} \; \texttt{Protocol}(x) \sim \texttt{Protocol}(v). \end{array}$$

The predicate *Restriction* is optional and captures possible conditions imposed by the protocol on the parameter array, describing the situations when the protocol is expected to guarantee anonymity. For instance, in the Dining Cryptographers (DC) protocol, $D = \{\top, \bot\}$ and x satisfies the condition that $x_i = \top$ for exactly one i (if nobody pays, anonymity is not guaranteed). In voting protocols, anonymity is expected only in non-unanimous votes, so the restriction there is that the parameter array should list at least two different values (votes).

The equivalence $\sim$ is a behavioural congruence (w.r.t. the operators that are used in the specification and which the intruder can use for his observations) and should be sufficiently strong, in order to ensure a sound analysis. In general, strong, branching or weak bisimulation are suitable. For standard process operators, failure and trace equivalence are also congruences.

A final remark is that there are of course different forms of anonymity. The strongest anonymity is obtained by writing $\forall$ instead of $\exists$ and transforming *Restriction*(v), $v_i \neq x_i$ into a condition in Definition 3 above. Then the intruder cannot distinguish the behaviour of this particular participant from any other's. But this is in many case (DC included) a too strong requirement. An extended study on anonymity degrees starting from a similar definition can be seen in [4].

4 A symbolic parameterised correctness proof for the Dining Cryptographers protocol

In this section, we give a formal proof that the DC protocol guarantees anonymity w.r.t. an external intruder to any participant i ($i : 0 \leq i < n$), for any number of parties $n > 1$. We consider strong bisimulation equivalence (denoted $\leftrightarrows$) as the equivalence expressing the intruder's observing power. Mostly, in the literature, trace equivalence is considered for this purpose. However, strong bisimulation is a sound choice, since whenever we prove a protocol correct according to it, it will also be correct in the trace model.

Formal model. We formalize DC as a parallel composition of n processes, each modelling the behaviour of a cryptographer. The secret choice as discussed in Section 3 is the decision to pay or not (the paying bit), represented by the Boolean values $x_i \in \{\top, \bot\}$. The characterizing condition $\mathit{Restriction}(x)$ is that x should contain exactly one value $\top$, since anonymity should hold when there is exactly one payer. A cryptographer process executes a series of actions corresponding to the three main steps of the protocol. The decision whether to pay or not is modelled by the execution of a $\mathbf{pay}(i, x_i)$ action. Flipping of the ith coin is modelled as follows: process $\mathit{Crypt}(i, x_i)$ executes a **flip** action and then shares the result with the right hand neighbour in the ring, by executing **tell** while its right hand neighbour gets to know the result of this coin flipping by executing the action **recv**. The synchronisation of these two actions results into the communication action **com**. The mCRL2 specification looks like this:

$$\begin{aligned}
\mathit{DC}(x{:}\mathit{ChoiceVector}) = \quad & \rho_{\{\forall i,d.\mathbf{flip}(i,d)\to\mathbf{flip}(i),\forall i,d.\mathbf{com}(i,d)\to\mathbf{com}(i)\}} \\
& \nabla_{\{\mathbf{flip},\mathbf{tell},\mathbf{com},\mathbf{syncbcast},\mathbf{nsa}\}} \quad \Gamma_{\{\mathbf{tell}|\mathbf{recv}\to\mathbf{com}\}} \\
& (\mathit{Crypt}(0, x_0) \| \mathit{Crypt}(1, x_1) \| \cdots \| \mathit{Crypt}(n{-}1, x_{n-1}))
\end{aligned}$$

$$\begin{aligned}
\mathit{Crypt}(i{:}\mathbb{N}, x_i{:}\mathbb{B}) = & \textstyle\sum_{\mathit{coinL}:\mathbb{B}}(\, \mathbf{flip}(i, \mathit{coinL}) \cdot \\
& \quad (\mathbf{tell}((i{+}1) \bmod n, \mathit{coinL}) \| \textstyle\sum_{\mathit{coinR}:\mathbb{B}} \mathbf{recv}(i, \mathit{coinR})) \cdot \\
& \quad \mathit{CryptAnnounce}(n, 0, i, x_i \oplus \mathit{coinR} \oplus \mathit{coinL})
\end{aligned}$$

The $\mathit{CryptAnnounce}(n, m, i, v)$ process models the third step: broadcasting the result of i's local computation (v) and computing the XOR of all broadcasted values. Since the broadcast implementation is not an actual part of the protocol and for lack of space, we do not show this subprocess here. Its visible actions are the synchronous broadcast **syncbcast** and the protocol's conclusion **nsa**. These actions have been added to the linearized version of the model. The renaming rules occurring as argument of ρ specify how much of the cryptographer's actions is visible for the intruder.

Proof idea. The proof, presented in the rest of the section, proceeds as follows:
- *linearisation*: First, the parallel composition is eliminated from the model above, by replacing it with choice and sequential composition. This is a standard operation for virtually all automatic and manual verifications in mCRL2 and can be done completely automatically. The result in our case is the linear process $\mathsf{LDC}(S, x, v, m, n)$, shown in Fig. 1. We denote LDC_i the instance of this specification for the case when i is the payer. Then proving parameterised anonymity becomes the problem of proving that for some fixed target participant i, there is another participant j such that $\mathsf{LDC}_i \leftrightarrow \mathsf{LDC}_j$. In particular, we will prove that $\mathsf{LDC}_i \leftrightarrow \mathsf{LDC}_{(i+1) \bmod n}$.
- *building a PBES*: We encode the above equivalence question as a PBES, using the translation established in [3].
- *solving the PBES*: We identify relevant invariants, and use them to prove that the solution for the PBES is $\top$ (true).
- *interpreting the solution*: This positive solution translates back to a positive answer for the equivalence $\mathsf{LDC}_i \leftrightarrow \mathsf{LDC}_{(i+1) \bmod n}$, which justifies parameterised anonymity for DC, as will be concluded in Theorem 1.

Linearisation. Eliminating the parallel composition operator from the DC process is a tool supported exercise, dictated by the mCRL2 linearisation rules [8]. The linear process

$$
\begin{array}{ll}
& \mathsf{LDC}(S,x,v,m,n) = \\
(\mathbf{a}) & \displaystyle\sum_{j\in\{0,\ldots,n-1\}} \sum_{b\in\{0,1\}} S_j{\approx}0 \rightarrow \mathbf{flip}(j){\cdot}\mathsf{LDC}(S[j\leftarrow 1], x[j\leftarrow \oplus b], v[j\leftarrow b], m, n) \\
(\mathbf{b}) & + \displaystyle\sum_{j\in\{0,\ldots,n-1\}} S_j{\approx}1 \wedge S_{j+1}{\approx}1 \rightarrow \mathbf{com}(j{+}1){\cdot}\mathsf{LDC}(S[j\leftarrow 2, j{+}1\leftarrow 3], x[j{+}1\leftarrow \oplus v_j], v, m, n) \\
(\mathbf{c}) & + \displaystyle\sum_{j\in\{0,\ldots,n-1\}} S_j{\approx}1 \wedge S_{j+1}{\approx}2 \rightarrow \mathbf{com}(j{+}1){\cdot} \\
& \qquad \mathsf{LDC}(S[j\leftarrow 2, j{+}1\leftarrow 4], x[j{+}1\leftarrow \oplus v_j], v[j{+}1\leftarrow x_{j+1}\oplus v_j], m, n) \\
(\mathbf{d}) & + \displaystyle\sum_{j\in\{0,\ldots,n-1\}} S_j{\approx}3 \wedge S_{j+1}{\approx}1 \rightarrow \mathbf{com}(j{+}1){\cdot} \\
& \qquad \mathsf{LDC}(S[j\leftarrow 4, j{+}1\leftarrow 3], x[j{+}1\leftarrow \oplus v_j], v[j\leftarrow x_j], m, n) \\
(\mathbf{e}) & + \displaystyle\sum_{j\in\{0,\ldots,n-1\}} S_j{\approx}3 \wedge S_{j+1}{\approx}2 \rightarrow \mathbf{com}(j{+}1){\cdot} \\
& \qquad \mathsf{LDC}(S[j\leftarrow 4, j{+}1\leftarrow 4], x[j{+}1\leftarrow \oplus v_j], v[j\leftarrow x_j, j{+}1\leftarrow x_{j+1}\oplus v_j], m, n) \\
(\mathbf{f}) & + \displaystyle\sum_{j\in\{0,\ldots,n-1\}} m_j{\approx}\bot \wedge \forall k. S_k{\approx}4 \rightarrow \mathbf{syncbcast}(j, x_j){\cdot}\mathsf{LDC}(S, x, v[(\forall k)k\leftarrow \oplus x_j], m[j\leftarrow \top], n) \\
(\mathbf{g}) & + \displaystyle\sum_{j\in\{0,\ldots,n-1\}} S_j{\approx}4 \wedge \forall k. m_k{\approx}\top \rightarrow \mathbf{nsa}(j, !v_j){\cdot}\mathsf{LDC}(S[j\leftarrow 5], x, v, m, n)
\end{array}
$$

Fig. 1. The linearized specification of the Dining Cryptographers protocol. The following shortcuts have been used: $\forall j$, $\forall k$ denote $\forall j{\in}\{0,\ldots,n{-}1\}$, $\forall k{\in}\{0,\ldots,n{-}1\}$; $j{+}1$ denotes $(j{+}1)$ mod n. Let LDC_i denote the protocol instance where cryptographer i is the payer: $\mathsf{LDC}_i \equiv \mathsf{LDC}\ (\overline{S}, \overline{x^i}, \overline{v}, \overline{m}, n)$

resulted, LDC, is shown in Figure 1. Its parameters are the number of cryptographers n and a few data arrays of length n, basically obtained by concatenating the local parameters of the n *Crypt* processes. For every index j, S_j (from $\mathbb{N}$) represents the current local state of process $Crypt(j, x_j)$. x_j, v_j and m_j are booleans representing j's paying bit, j's currently computed all-XOR value and a mark whether j broadcasted, respectively. Initially, the array S is 0 everywhere, while v and m are $\bot$ everywhere. We denote these default initial values by $\overline{S}$, $\overline{v}$, $\overline{m}$. Suppose i is the payer. Then the initial choice vector x is $\bot$ everywhere except for the ith position which is $\top$ (we denote this array $\overline{x^i}$).

For an array A, we write $A[k \leftarrow expr]$ to denote A after that element A_k has been assigned the expression *expr*. In particular, if the assignment involves an operation *op* on the old value of A_k, we write $A[k \leftarrow op\ expr]$. For instance, $A[k \leftarrow \top]$ denotes A updated with the assignment $A_k := \top$ and $A[k \leftarrow \oplus\top]$ denotes A updated with the assignment $A_k := A_k \oplus \top$. To keep the description readable, we also write everywhere $j+1$ instead of $(j+1)$ mod n.

The PBES. The strong bisimilarity question $\mathsf{LDC}_i \leftrightarrows \mathsf{LDC}_{i+1}$ is encoded, by applying the translation rules from [3] and several logical rewritings, to the equation $\mathcal{E}$ shown in Fig. 2. The data parameters of variable $\mathbf{E}$ represent in fact two states $\langle S, x, v, m, n\rangle$ and $\langle S', x', v', m', n'\rangle$ of the two linear specifications LDC_i and LDC_{i+1}. Intuitively, this equation enumerates all conditions that need to be satisfied for the two states to be strongly bisimilar.

$$
\begin{array}{ll}
& \nu\mathbf{E}(S,x,v,m,n,S',x',v',m',n') = \\
(\mathbf{a}) & \forall j.\forall b \in \{0,1\}.\ (S_j{\approx}0 \ \Leftrightarrow\ S'_j = 0 \ \wedge\ \underline{S_j{\approx}0} \Rightarrow \\
& \quad \mathbf{E}(S[j \leftarrow 1], x[j \leftarrow \oplus b], v[j \leftarrow b], S'[j \leftarrow 1], x'[j \leftarrow \oplus(b \oplus (j=i))], v'[j \leftarrow (b \oplus (j=i))]) \\
(\mathbf{b}) & \wedge \forall j.\ ((S_j{\approx}1 \wedge S_{j+1}{\approx}1 \ \Leftrightarrow\ S'_j{\approx}1 \wedge S'_{j+1}{\approx}1) \ \wedge\ \underline{(S_j{\approx}1 \wedge S_{j+1}{\approx}1)} \Rightarrow \\
& \qquad \mathbf{E}(S[j \leftarrow 2, j+1 \leftarrow 3], x[j+1 \leftarrow \oplus v_j], S'[j \leftarrow 2, j+1 \leftarrow 3], x'[j+1 \leftarrow \oplus v'_j])) \\
(\mathbf{c}) & \wedge \forall j.\ ((S_j{\approx}1 \wedge S_{j+1}{\approx}2 \ \Leftrightarrow\ S'_j{\approx}1 \wedge S'_{j+1}{\approx}2) \ \wedge\ \underline{(S_j{\approx}1 \wedge S_{j+1}{\approx}2)} \Rightarrow \\
& \qquad \mathbf{E}(S[j \leftarrow 2, j+1 \leftarrow 4], x[j+1 \leftarrow \oplus v_j], v[j+1 \leftarrow x_{j+1} \oplus v_j], \\
& \qquad\qquad S'[j \leftarrow 2, j+1 \leftarrow 4], x'[j+1 \leftarrow \oplus v'_j], v'[j+1 \leftarrow x'_{j+1} \oplus v'_j])) \\
(\mathbf{d}) & \wedge \forall j.\ ((S_j{\approx}3 \wedge S_{j+1}{\approx}1 \ \Leftrightarrow\ S'_j{\approx}3 \wedge S'_{j+1}{\approx}1) \wedge \underline{(S_j{\approx}3 \wedge S_{j+1}{\approx}1)} \Rightarrow \\
& \qquad \mathbf{E}(S[j \leftarrow 4, j+1 \leftarrow 3], x[j+1 \leftarrow \oplus v_j], v[j \leftarrow x_j], \\
& \qquad\qquad S'[j \leftarrow 4, j+1 \leftarrow 3], x'[j+1 \leftarrow \oplus v'_j], v'[j \leftarrow x'_{j'}])) \\
(\mathbf{e}) & \wedge \forall j.\ ((S_j{\approx}3 \wedge S_{j+1}{\approx}2 \ \Leftrightarrow\ S'_j{\approx}3 \wedge S'_{j+1}{\approx}2) \ \wedge\ \underline{(S_j{\approx}3 \wedge S_{j+1}{\approx}2)} \Rightarrow \\
& \qquad \mathbf{E}(S[j \leftarrow 4, j+1 \leftarrow 4], x[j+1 \leftarrow \oplus v_j], v[j \leftarrow x_j, j+1 \leftarrow x_{j+1} \oplus v_j], \\
& \qquad\qquad S'[j \leftarrow 4, j+1 \leftarrow 4], x'[j+1 \leftarrow \oplus v'_j], v'[j \leftarrow x'_j, j+1 \leftarrow x'_{j+1} \oplus v'_j])\ ) \\
(\mathbf{f}) & \wedge \forall j.\ ((m_j{\approx}\bot \ \wedge\ (\forall k.S_k{\approx}4) \ \Leftrightarrow\ m'_j{\approx}\bot \ \wedge\ (\forall k.S'_k{\approx}4)) \\
& \qquad \wedge\ \underline{(m_j{\approx}\bot \ \wedge\ (\forall k.S_k{\approx}4))} \Rightarrow x_j{\approx}x'_j \hfill (\alpha(j)) \\
& \qquad \wedge\ \underline{(m_j{\approx}\bot \ \wedge\ (\forall k.S_k{\approx}4))} \Rightarrow \\
& \qquad\qquad \mathbf{E}(v[(\forall k)k \leftarrow \oplus x_j], m[j \leftarrow \top], v'[(\forall k)k \leftarrow \oplus x'_j], m'[j \leftarrow \top])) \\
(\mathbf{g}) & \wedge \forall j.\ ((S_j{\approx}4 \wedge (\forall k.m_k{\approx}\top) \ \Leftrightarrow\ S'_j{\approx}4 \wedge (\forall k.m'_k{\approx}\top)) \\
& \qquad \wedge\ \underline{(S_j{\approx}4 \wedge (\forall k.m_k{\approx}\top))} \Rightarrow v_j{\approx}v'_j \hfill (\beta(j)) \\
& \qquad \wedge\ \underline{(S_j{\approx}4 \wedge (\forall k.m_k{\approx}\top))} \Rightarrow \mathbf{E}(S[j \leftarrow 5], S'[j \leftarrow 5]))
\end{array}
$$

Fig. 2. The PBES encoding the equivalence question $\mathsf{LDC}_i \leftrightarrows \mathsf{LDC}_{i+1}$. The same shortcuts as in Fig. 1 are used. Moreover, only the modified data parameters variables are shown in the parameter lists of $\mathbf{E}$ occurrences. For readability, the guards are underlined and the guard equivalences are italicized. $\approx$ is used as equality symbol for the data parameters and is assumed defined for (arrays of) $\mathbb{B}$ and $\mathbb{N}$.

The first subterm, as dictated by the encoding rules, looked in fact like this:

$$
\begin{array}{ll}
(\mathbf{a}) & \forall j.\forall b \in \{0,1\}.\ (S_j{\approx}0 \ \Leftrightarrow\ S'_j = 0 \ \wedge\ \underline{S_j{\approx}0} \Rightarrow \\
& \quad (\exists b' \in \{0,1\}.\ \ \mathbf{E}(S[j \leftarrow 1], x[j \leftarrow \oplus b], v[j \leftarrow b], S'[j \leftarrow 1], x'[j \leftarrow \oplus b'], v'[j \leftarrow b'])).
\end{array}
$$

For technical reasons, we eliminated the original existential quantification $\exists b'{:}\mathbb{B}$ by considering a concrete instantiation for b', namely $b \oplus (j = i)$, maintaining bisimulation. This transformation is sound, in the sense that any solution for the equation in Fig.2 will be a solution for the original equation (where the first subterm is given by the expression discussed above).

Solving the PBES. We start by noticing that the right-hand side of our equation fits the form of Proposition 1 and that the predicate $\iota_1(d) : (S{\approx}S' \wedge m{\approx}m' \wedge n{\approx}n')$ satisfies the condition in Proposition 1, so it is an invariant. All predicate variables occurring in this section have the same sort as $\mathbf{E}$. However, we will sometimes write only a sub-list of the parameters, in order to outline the exact parameters on which a predicate depends. d is the whole list.

Proposition 1 also immediately holds for the expression $\kappa(d) : (\forall j. S_j{\approx}0 \Rightarrow (x_j{\approx}(i{=}j)))$, which formalizes the intuition that participant i is the payer in the run specified by LDC_i. Strengthening $\mathcal{E}$ with $(\iota_1 \wedge \kappa)$ leads to a first significant simplification: the seven predicates expressing guard equivalences (shown in italics in the figure) rewrite to $\top$, because they follow from ι_1. The resulting equation $\mathcal{E}'$ has a form that still fits Proposition 1:

$$\nu\mathbf{E}(d) = (\kappa \wedge \iota_1 \wedge \forall j.\alpha(j) \wedge \forall j.\beta(j)) \ \wedge \bigwedge_{\star\in\{a\ldots g\}} \forall j.\phi_\star(d,j) \Rightarrow \mathbf{E}(g_\star(d,j)). \qquad (2)$$

The main difficulty of a parametric proof is to find the right invariants that significantly reduce the complexity of the equation, without excluding the relevant solutions. In our case, invariants that are not satisfied by the initial parameters $\langle \overline{S}, \overline{x^i}, \overline{v}, \overline{m}, n, \overline{S}, \overline{x^{i+1}}, \overline{v}, \overline{m}, n\rangle$ are not useful, because then the solution of the strengthened equation will not necessarily satisfy the original equation. Since ι_1 holds, we now need a powerful invariant relation between the rest of the parameters, (x, v, x', v'). In fact, we are intuitively aiming to properly map the states of the "actual" protocol behaviour, as modeled by the (S, x, v, m, n) parameters, to the states of the "alternative protocol" behaviour, captured by the (S', x', v', m', n') parameters. Let $\mathtt{mx}(S_k, x_k, k)$ denote the following formula:

$$((S_k{\approx}0) \wedge (k = i+1) \ \vee \ (S_k \in \{1,2\} \wedge (x_k \oplus (k = i+1))) \ \vee \ (S_k \in \{3,4\} \wedge (x_k))),$$

which can be read as the short routine "if $S_k{\approx}0$, then return $k = i+1$; else if $S_k \in \{1,2\}$ then return $(x_k \oplus (k = i+1))$; else if $S_k \in \{3,4\}$ then return x_k; else return $\perp$". Moreover, let $\mathtt{mV}(S_k, v_k, k)$ denote the following formula:

$$(S_k \in \{1,2,3\} \wedge (v_k \oplus (i = k))) \ \vee \ (S_k \in \{4,5\} \wedge (v_k)).$$

Intuitively, $\mathtt{mx}$ and $\mathtt{mV}$ are our proposed x-mapping and v-mapping, linking parameters (S, x) to x' and, respectively, (S, v) to v' (k is an index). The central piece of the correctness proof is showing that these connections are invariant. We do this in the following lemma.

Lemma 1. *The predicates $\iota_2(S, x, x') : \forall k.\ x'_k = \mathtt{mx}(S_k, x_k, k)$ and $\iota_3(S, v, v') : \forall k.\ v'_k = \mathtt{mV}(S_k, v_k, k)$ are invariants for the equation $\mathcal{E}'$.*

Proof. We will prove, for each subterm $\forall j.\phi_\star(d,j) \Rightarrow \mathbf{E}(g_\star(d,j))$ of the right-hand side of (2), that $(\iota_2(d) \wedge \iota_3(d) \wedge \phi_\star(d,j)) \Rightarrow \iota_2(g_\star(d,j)) \wedge \iota_3(g_\star(d,j))$ holds. From this we will then conclude using Proposition 1 that ι_2 and ι_3 are invariants.

The proof is more or less mechanical. For each subterm $a \ldots g$, assuming that the left-hand side of the implication holds, we rewrite the two terms in the right-hand side to $\top$. Note that a part of the left-hand side is common to all 7 subterms:

$$(\iota_2(d) \wedge \iota_3(d)) : \quad \forall k.\, x'_k = \mathtt{mx}(S_k, x_k, k) \ \wedge \ \forall k.\, v'_k = \mathtt{mV}(S_k, v_k, k) \qquad (3)$$

Let us use $\langle S[], x[], v[], S'[], x'[], v'[]\rangle$ as a shortcut for the updates suffered by d at the currently analyzed subterm. The proof obligation is, for all subterms,

$$(\iota_2(d[])) \wedge \iota_3(d[]) : \forall k.\, x'[]_k = \mathtt{mx}(S[]_k, x[]_k, k) \ \wedge \ \forall k.\, v'[]_k = \mathtt{mV}(S[]_k, v[]_k, k). \qquad (4)$$

(a) For this subterm, $\phi(d,j) \equiv (S_j{\approx}0)$ and $d[] \equiv \langle S[j \leftarrow 1], x[j \leftarrow \oplus b], v[j \leftarrow b], S'[j \leftarrow 1], x'[j \leftarrow \oplus(b \oplus (j = i))], v'[j \leftarrow (b \oplus (j = i))]\rangle$. For $k \neq j$, $d[]_k = d_k$. In particular, this means that $x'[]_k = x'_k$ and $\mathtt{mx}(S[]_k, x[]_k, k) = \mathtt{mx}(S_k, x_k, k)$, and similarly for v' and $\mathtt{mV}$. Therefore we only need to prove

$$(A)\ \ x'[]_j = \mathtt{mx}(S[]_j, x[]_j, j) \qquad\qquad (B)\ \ v'[]_j = \mathtt{mV}(S[]_j, v[]_j, j).$$

By projecting $d[]$ on j, these equalities rewrite to

$$(A)\ \ x'_j \oplus b \oplus (j = i) = \mathtt{mx}(1, x_j \oplus b, j) \qquad\qquad (B)\ \ b \oplus (j = i) = \mathtt{mV}(1, b, j).$$

We can now unfold the definitions of $\mathtt{mx}$,$\mathtt{mV}$ and obtain:

$$\begin{array}{l}(A)\ \ x'_j \oplus b \oplus (j = i) = (x_j \oplus b) \oplus (j = i + 1)\\ (B)\ \ b \oplus (j = i) = b \oplus (j = i)\end{array}$$

(B) is trivially true. $\iota_2(d)$ holds, so we can rewrite x'_j, taking into account that $S_j{\approx}0$, and obtain (A) $(j = i + 1) \oplus b \oplus (j = i) = (x_j \oplus b) \oplus (j = i + 1)$, which further rewrites to $x_j = (i = j)$. Note that $\kappa(d)$ can be used as a premise, in conjunction with any $\phi_\star$, since it stands as an independent term in the expression (2). Since in the current situation $S_j = 0$, κ ensures $x_j = (i = j)$.

(b) For this subterm, $\phi(d, j)) \equiv (S_j{\approx}1 \wedge S_{j+1}{\approx}1$ and $d[] \equiv \langle S[j \leftarrow 2, j+1 \leftarrow 3], x[j+1 \leftarrow \oplus v_j], S'[j \leftarrow 2, j+1 \leftarrow 3], x'[j+1 \leftarrow \oplus v'_j]\rangle$. For $k \notin \{j, j+1\}$, $d[]_k = d_k$. This means that $x'[]_k = x'_k$ and $\mathtt{mx}(S[]_k, x[]_k, k) = \mathtt{mx}(S_k, x_k, k)$, and similarly for v' and $\mathtt{mV}$. Therefore we only need to prove

$$\begin{array}{ll}(A1)\ \ x'[]_j = \mathtt{mx}(S[]_j, x[]_j, j) & (A2)\ \ x'[]_{j+1} = \mathtt{mx}(S[]_{j+1}, x[]_{j+1}, j+1)\\ (B1)\ \ v'[]_j = \mathtt{mV}(S[]_j, v[]_j, j) & (B2)\ \ v'[]_{j+1} = \mathtt{mV}(S[]_{j+1}, v[]_{j+1}, j+1).\end{array}$$

By projecting $d[]$ on j and $j + 1$, these rewrite to

$$\begin{array}{ll}(A1)\ \ x'_j = \mathtt{mx}(2, x_j, j) & (A2)\ \ x'_{j+1} \oplus v'_j = \mathtt{mx}(3, x_{j+1} \oplus v_j, j+1)\\ (B1)\ \ v'_j = \mathtt{mV}(2, v_j, j) & (B2)\ \ v'_{j+1} = \mathtt{mV}(3, v_{j+1}, j+1).\end{array}$$

Further, we unfold the definitions of $\mathtt{mx}$,$\mathtt{mV}$ and obtain:

$$\begin{array}{ll}(A1)\ \ x'_j = x_j \oplus (j = i + 1) & (A2)\ \ x'_{j+1} \oplus v'_j = x_{j+1} \oplus v_j\\ (B1)\ \ v'_j = v_j \oplus (j = i) & (B2)\ \ v'_{j+1} = v_{j+1} \oplus (j + 1 = i)\end{array}$$

Instantiation of $\iota_2(d)$ and $\iota_3(d)$ for j and $j+1$, while taking into account that $S_j{\approx}1 \wedge S_{j+1}{\approx}1$, leads to the truth of formulae (A1),(B1),(B2). (A2) transforms to $(x_{j+1} \oplus (j + 1 = i + 1)) \oplus (v_j \oplus (i = j)) = x_{j+1} \oplus v_j$, which obviously holds since $(j + 1 = i + 1) = (j = i)$, $x \oplus x = \perp$ and $x \oplus \perp = x$.

(c) For this subterm, $\phi(d, j) \equiv (S_j{\approx}1 \wedge S_{j+1}{\approx}2)$ and $d[] \equiv \langle S[j \leftarrow 2, j+1 \leftarrow 4], x[j+1 \leftarrow \oplus v_j], v[j+1 \leftarrow x_{j+1} \oplus v_j], S'[j \leftarrow 2, j+1 \leftarrow 4], x'[j+1 \leftarrow \oplus v'_j], v'[j+1 \leftarrow x'_{j+1} \oplus v'_j]\rangle$. An argument identical to the one at **(b)** justifies that we only need to prove

$$\begin{array}{ll}(A1)\ \ x'[]_j = \mathtt{mx}(S[]_j, x[]_j, j) & (A2)\ \ x'[]_{j+1} = \mathtt{mx}(S[]_{j+1}, x[]_{j+1}, j+1)\\ (B1)\ \ v'[]_j = \mathtt{mV}(S[]_j, v[]_j, j) & (B2)\ \ v'[]_{j+1} = \mathtt{mV}(S[]_{j+1}, v[]_{j+1}, j+1).\end{array}$$

By projecting $d[]$ on j and $j + 1$, these rewrite to

$$\begin{array}{ll}(A1)\ \ x'_j = \mathtt{mx}(2, x_j, j) & (A2)\ \ x'_{j+1} \oplus v'_j = \mathtt{mx}(4, x_{j+1} \oplus v_j, j+1)\\ (B1)\ \ v'_j = \mathtt{mV}(2, v_j, j) & (B2)\ \ x'_{j+1} \oplus v'_j = \mathtt{mV}(4, x_{j+1} \oplus v_j, j+1).\end{array}$$

Further, we apply the definitions of mx,mV and obtain:

$$\begin{array}{ll} (A1)\ x'_j = x_j \oplus (j = i+1) & (A2)\ x'_{j+1} \oplus v'_j = x_{j+1} \oplus v_j \\ (B1)\ v'_j = v_j \oplus (j = i) & (B2)\ x'_{j+1} \oplus v'_j = x_{j+1} \oplus v_j \end{array}$$

Since $S_j = 1 \wedge S_{j+1} = 2$, the instantiation of $\iota_2(d)$ and $\iota_3(d)$ for j gives exactly (A1) and (B1). By instantiating $\iota_2(d)$ for $j+1$ and $\iota_3(d)$ for j, $(A2)$ becomes $x_{j+1} \oplus (j+1 = i+1) \oplus v_j \oplus (j = i) = x_{j+1} \oplus v_j$, which evaluates to $\top$.

(d) For this subterm, $\phi(d, j) \equiv (S_j{\approx}3 \wedge S_{j+1}{\approx}1)$ and $d[] \equiv \langle S[j \leftarrow 4, j+1 \leftarrow 3], x[j+1 \leftarrow \oplus v_j], v[j \leftarrow x_j], S'[j \leftarrow 4, j+1 \leftarrow 3], x'[j+1 \leftarrow \oplus v'_j], v'[j \leftarrow x'_j]\rangle$. As with the previous subterms, we only need to prove

$$\begin{array}{ll} (A1)\ x'[]_j = \mathtt{mx}(S[]_j, x[]_j, j) & (A2)\ x'[]_{j+1} = \mathtt{mx}(S[]_{j+1}, x[]_{j+1}, j+1) \\ (B1)\ v'[]_j = \mathtt{mV}(S[]_j, v[]_j, j) & (B2)\ v'[]_{j+1} = \mathtt{mV}(S[]_{j+1}, v[]_{j+1}, j+1). \end{array}$$

By projecting $d[]$ on j and $j+1$, these rewrite to

$$\begin{array}{ll} (A1)\ x'_j = \mathtt{mx}(4, x_j, j) & (A2)\ x'_{j+1} \oplus v'_j = \mathtt{mx}(3, x_{j+1} \oplus v_j, j+1) \\ (B1)\ x'_j = \mathtt{mV}(4, x_j, j) & (B2)\ v'_{j+1} = \mathtt{mV}(3, v_{j+1}, j+1). \end{array}$$

Further, we apply the definitions of mx,mV and obtain:

$$\begin{array}{ll} (A1)\ x'_j = x_j & (A2)\ x'_{j+1} \oplus v'_j = x_{j+1} \oplus v_j \\ (B1)\ x'_j = x_j & (B2)\ v'_{j+1} = v_{j+1} \oplus (i = j+1). \end{array}$$

Since $S_j = 3 \wedge S_{j+1} = 1$, the instantiation of $\iota_2(d)$ for j is exactly formula (A1), and the instantiation of $\iota_3(d)$ for $j+1$ is exactly formula (B2). By instantiating $\iota_2(d)$ for $j+1$ and $\iota_3(d)$ for j, $(A2)$ becomes $(x_{j+1} \oplus (j+1 = i+1) \oplus v_j \oplus (i = j)) = x_{j+1} \oplus v_j$, which is true, due to $x \oplus x = \bot$ and $x \oplus 0 = x$.

(e) For this subterm, $\phi(d, j) \equiv (S_j{\approx}3 \wedge S_{j+1}{\approx}2)$ and $d[] \equiv \langle S[j \leftarrow 4, j+1 \leftarrow 4], x[j+1 \leftarrow \oplus v_j], v[j \leftarrow x_j, j+1 \leftarrow x_{j+1} \oplus v_j], S'[j \leftarrow 4, j+1 \leftarrow 4], x'[j+1 \leftarrow \oplus v'_j], v'[j \leftarrow x'_j, j+1 \leftarrow x'_{j+1} \oplus v'_j]\rangle$.

The same argument as above ensures that we only need to prove

$$\begin{array}{ll} (A1)\ x'[]_j = \mathtt{mx}(S[]_j, x[]_j, j) & (A2)\ x'[]_{j+1} = \mathtt{mx}(S[]_{j+1}, x[]_{j+1}, j+1) \\ (B1)\ v'[]_j = \mathtt{mV}(S[]_j, v[]_j, j) & (B2)\ v'[]_{j+1} = \mathtt{mV}(S[]_{j+1}, v[]_{j+1}, j+1). \end{array}$$

By projecting $d[]$ on j and $j+1$, these rewrite to

$$\begin{array}{ll} (A1)\ x'_j = \mathtt{mx}(4, x_j, j) & (A2)\ x'_{j+1} \oplus v'_j = \mathtt{mx}(4, x_{j+1} \oplus v_j, j+1) \\ (B1)\ x'_j = \mathtt{mV}(4, x_j, j) & (B2)\ x'_{j+1} \oplus v'_j = \mathtt{mV}(4, x_{j+1} \oplus v_j, j+1). \end{array}$$

Further, we apply the definitions of mx,mV and obtain:

$$\begin{array}{ll} (A1)\ x'_j = x_j & (A2)\ x'_{j+1} \oplus v'_j = x_{j+1} \oplus v_j \\ (B1)\ x'_j = x_j & (B2)\ x'_{j+1} \oplus v'_j = x_{j+1} \oplus v_j. \end{array}$$

Since $S_j = 3 \wedge S_{j+1} = 2$, the instantiation of $\iota_2(d)$ for j is (A1). By instantiating $\iota_2(d)$ for $j+1$ and $\iota_3(d)$ for j, $(A2, B2)$ becomes $(x_{j+1} \oplus (j+1 = i+1) \oplus v_j \oplus (j = i)) = x_{j+1} \oplus v_j$, which is $\top$, due to $x \oplus x = \bot$ and $x \oplus 0 = x$.

(f) Here, $\phi(d,j) \equiv (m_j \approx \bot \wedge (\forall k.S_k \approx 4))$ and $d[] \equiv \langle v[(\forall k)k \leftarrow \oplus x_j], m[j \leftarrow \top], v'[(\forall k)k \leftarrow \oplus x'_j], m'[j \leftarrow \top]\rangle$. We project $d[]$ on a random k. So, we now have to prove $x'_k = \mathtt{mx}(S_k, x_k, k)$ and $v'_k \oplus x'_j = \mathtt{mV}(S_k, v_k \oplus x_j, k)$. The first formula is true (from $\iota_2(d)$). We rewrite the second one using the definition of $\mathtt{mV}$ and $(\iota_2(d) \wedge \iota_3(d))$. The result is $v_k \oplus x_j = v_k \oplus x_j$, trivially true.

(g) For this subterm, $\phi(d,j) \equiv (S_j \approx 4 \wedge (\forall k.m_k \approx \top))$ and $d[] \equiv \langle S[j \leftarrow 5], S'[j \leftarrow 5]\rangle$. We project $d[]$ on a random k. So, we now have to prove: $x'_k = \mathtt{mx}(5, x_k, k)$ and $v'_k = \mathtt{mV}(5, v_k, k)$. They are both true, due to $\iota_2(d) \wedge \iota_3(d)$ and the fact that $\mathtt{mx}(4,x,y) = \mathtt{mx}(5,x,y)$ and $\mathtt{mV}(4,x,y) = \mathtt{mV}(5,x,y)$. □

Now we can strengthen $\mathcal{E}'$ with the invariants proved in Lemma 1 and obtain the new equation:

$$\nu\mathbf{E}(d) = \iota_2(d) \wedge \iota_3(d) \wedge \kappa(d) \wedge \iota_1(d) \wedge \forall j.\alpha(j) \wedge \forall j.\beta(j) \;\wedge \bigwedge_{\star \in \{a\ldots g\}} \forall j.\phi_\star(j) \Rightarrow \mathbf{E}(g_\star(d,j)).$$

Note that $\forall j.\alpha(j) \wedge \forall j.\beta(j)$ follows from $\iota_2(S,x,x')$ and $\iota_3(S,v,v')$. (The unfolded expression $\forall j.\alpha(j) \wedge \forall j.\beta(j)$ is $\forall j.((m_j \approx \bot \wedge (\forall k.S_k \approx 4)) \Rightarrow x_j \approx x'_j) \;\wedge\; \forall j.(S_j \approx 4 \wedge (\forall k.m_k \approx \top) \Rightarrow v_j \approx v'_j)$.) Then the equation above is equivalent to

$$\nu\mathbf{E}(d) = \iota_2(d) \wedge \iota_3(d) \wedge \kappa(d) \wedge \iota_1(d) \;\wedge \bigwedge_{\star \in \{a\ldots g\}} \forall j.\phi_\star(j) \Rightarrow \mathbf{E}(g_\star(d,j)),$$

which fits the form in the assumption of Proposition 2. Therefore, the solution is $\iota_1(d) \wedge \iota_2(d) \wedge \iota_3(d)$. Since all the used invariants ι_1, ι_2, ι_3, κ are satisfied by our instance of interest $d \equiv \langle \overline{S}, \overline{x^i}, \overline{v}, \overline{m}, n, \overline{S}, \overline{x^{i+1}}, \overline{v}, \overline{m}, n\rangle$, it follows that the solution of the original equation $\mathcal{E}$ for this instance is $\top$. We can then conclude that the solution to the encoded equivalence problem $\mathsf{LDC}_i \leftrightarrows \mathsf{LDC}_{i+1}$ is true and thus, the parameterised anonymity property holds for the DC protocol:

Theorem 1. *For any $i \geq 0$ and any $n > max(i,1)$, the protocol DC is anonymous for i out of n w.r.t. an external intruder.*

As a final remark, we note that the proof can be adapted to accommodate internal observers, i.e. for the more interesting case when a cryptographer j is the intruder. Then the mCRL2 model specifying j's view on the protocol behaviour would allow visibility of the secret bit x_j, thus the set R from ρ_R would *not* contain any renamings for actions of the form $\mathbf{flip}(j,d)$, $\mathbf{com}(j,d)$ and $\mathbf{com}(j-1,d)$ (for any d). For checking the condition $\exists v \ldots$ from Def. 3, which in the DC case is $\exists \overline{x^k} \ldots$, we need to make the distinction $j \neq i{+}1$ or $j = i{+}1$. In the first case, the proof proceeds exactly as above, since the weakening of the renaming set does not influence the validity (proof) of the used invariants. In the second case, we need to choose another k, for instance $(i-1) \bmod n$ and prove, in a very similar way, that $\mathsf{LDC}_i \leftrightarrows \mathsf{LDC}_{(i-1) \bmod n}$. Note that in both cases, i,j,k should be different, therefore the assumption $n > 2$ would be needed.

5 (Un)decidability

We now study the general question: given a multiparty protocol, can it be decided whether it guarantees anonymity to its participants, whatever their number?

(**NPA**) Given a domain D, a constant $N{\geq}1$ and a parameterised protocol $\mathtt{Protocol}$, decide whether $\mathtt{Protocol}(x)$ is anonymous for 0 with at least N processes present. In other words, decide whether for all $n{\geq}N$:

$$\begin{array}{l} \forall x{\in}D^n \text{ with } \mathit{Restriction}(x) \\ \quad \exists v{\in}D^n \text{ s.t. } \mathit{Restriction}(v),\ v_0{\neq}x_0 \text{ and } \mathtt{Protocol}(x){\sim}\mathtt{Protocol}(v). \end{array}$$

Note that using index 0 in this definition is not a loss of generality. In most protocols, the behaviours of honest parties are isomorphic and renaming schemes can reduce the anonymity question about a participant index $i \geq 0$ to NPA.

The following theorem says that for all reasonable weak equivalences (those satisfying $a{\cdot}\tau{\cdot}x = a{\cdot}x$), preserving the alphabet of a process, when sequential programs can be expressed in the specification formalism, there is in general no hope to decide anonymity.

Theorem 2. *Let $\sim$ denote any behavioural congruence refining weak trace equivalence satisfying $a{\cdot}\tau{\cdot}x = a{\cdot}x$. Then NPA is undecidable for any number $N{\geq}1$ of processes.*

Proof. We encode the question of deciding program termination as an instance of NPA. Let M be a program translated to mCRL2 (mCRL2 is sufficiently expressive for this) without visible behaviour — i.e., all its actions are τ steps. Let us construct a protocol as follows where **stop** is a visible action and $D = \{0, 1\}$:

$$\begin{array}{ll} P(i, x_i) & = i{\not\approx}0 \to \delta \ + \ i{\approx}0 \to (x_i{\approx}1 \to \tau{\cdot}M{\cdot}\mathbf{stop} + x_i{\approx}0 \to \tau{\cdot}\mathbf{stop}) \\ Q(n) & = \delta \\ \mathtt{Protocol}(x) & = P(0, x_0) \| P(1, x_1) \| \cdots \| P(n{-}1, x_{n-1}) \| Q(n) \end{array}$$

By applying the parallel composition laws, this protocol process linearizes to:

$$\mathtt{Protocol}(x) = x_0{\approx}1 \to \tau{\cdot}M{\cdot}\mathbf{stop}{\cdot}\delta \ + \ x_0{\approx}0 \to \tau{\cdot}\mathbf{stop}{\cdot}\delta. \tag{5}$$

Suppose NPA is decidable. Then we get an answer to whether this protocol is anonymous or not. NPA can be formulated as $\forall n{\geq}1\ \forall x \in \{0, 1\}^n\ \exists v \in \{0, 1\}^n\ v_0 \neq x_0 \wedge \mathtt{Protocol}(x) \sim \mathtt{Protocol}(v)$. So, according to (5), $\tau{\cdot}M{\cdot}\mathbf{stop}{\cdot}\delta \sim \tau{\cdot}\mathbf{stop}{\cdot}\delta$. Using the requirements on $\sim$, this can only be the case iff M terminates. As termination of M is undecidable, NPA is also undecidable. $\square$

Undecidability holds even for protocols without loops:

Theorem 3. *If $\sim$ denotes* strong-, weak- *or* branching bisimilarity *then NPA is undecidable, even if the protocol specification language does not contain loops (but contains the choice operator $\sum$ essentially quantifying over infinite domains).*

Proof. We reduce the problem of deciding strong bisimulation between two mCRL2 processes to NPA. Let M_1 and M_2 be two arbitrary mCRL2 processes. In a similar fashion as above, we construct a protocol that, after the linearisation of the parallel composition, looks as: $\mathtt{Protocol}(x) = x_0{\not\approx}0 \to M_1{\cdot}\delta \ + \ x_0{\approx}0 \to M_2{\cdot}\delta$. A positive answer to NPA means that $M_1{\cdot}\delta \not\sim M_2{\cdot}\delta$ and a negative answer means that $M_1{\cdot}\delta \sim M_2{\cdot}\delta$. According to [18], strong, branching and weak bisimulation are all undecidable for processes with infinite choice, hence NPA is undecidable as well. $\square$

So, unsurprisingly, parameterised anonymity is in general undecidable. However, in many cases it can still be decided by inspecting a finite collection of processes only. Let us call the communication function Γ_C *behaviour preserving* for an equivalence relation $\sim$ and processes p and q iff $\Gamma_C(p) \sim \Gamma_C(q)$ implies $p \sim q$. This is for instance the case if C is functional (i.e. $C(\alpha) = C(\alpha')$ implies $\alpha = \alpha'$) and no communication action $C(\alpha)$ occurs in p or in q.

Theorem 4. *Consider a protocol as defined (1). Assume D is finite, the set of hidden actions I is empty, renaming is effectively the identity, the set of allowed actions V does not block any actions and the communication function is behaviour preserving for $\sim$ and parallel combinations of $P(i, x_i)$ and $Q(n)$. Moreover, $\sim$ respects commutativity of the parallel operator. For any $N{\geq}1$, NPA is decidable iff it can be decided that, for any $n{\geq}N$ and $x_1, \ldots, x_{N-1}, y_1, \ldots, y_{N-1}{\in}D$,*

$$P(0, x_0)\|\cdots\|P(N{-}1, x_{N-1})\|Q(n) \sim P(0, y_0)\|\cdots\|P(N{-}1, y_{N-1})\|Q(n). \tag{6}$$

Proof. The first step is to determine whether for every x with $\mathit{Restriction}(x)$ a v can be found such that $\mathit{Restriction}(v)$ and $v_0 \neq x_0$ satisfying (6). As x and v can only attain a finite number of values, this can be done by explicit enumeration.

Suppose this fails for some x. So, for any v either $\mathit{Restriction}(v)$, $v_0 \neq x_0$ or (6) would not hold. In the last case, $\texttt{Protocol}(x){\not\sim}\texttt{Protocol}(v)$ as Γ_C is behaviour preserving. Thus NPA does not hold for $n = N$. As NPA should hold for every $n \geq N$ processes, we can conclude that NPA is invalid.

Now assume that the procedure above yields a v for every x. We find that as (6) holds, and as $\sim$ is a behavioural congruence for which $\|$ is commutative:

$$\begin{aligned}&P(0, x_0)\|\cdots\|P(N{-}1, x_{N-1})\|P(N, x_N)\|\cdots\|P(n{-}1, x_{n-1})\|Q(n) \sim\\ &\qquad P(0, v_0)\|\cdots\|P(N{-}1, v_{N-1})\|P(N, x_N)\|\cdots\|P(n{-}1, x_{n-1})\|Q(n)\end{aligned}$$

By applying the communication operator Γ_C and the ∇_V, ρ_R and τ_I operators (which effectively do nothing), all the conditions of NPA are made valid. □

The verification of the behavioural equality for the finite constellation of processes might be tricky as n is an arbitrary number. So, the verification is in a sense symbolic. Unless the behaviour of Q is very essentially dependent on n, which it rarely is, this will not pose a problem for the verification tools of mCRL2 as they are essentially symbolic manipulators.

Note furthermore that this decision procedure often does not apply because the communication operator is not behaviour preserving or the allow, renaming or hiding operators are not trivial. But as only one side of the decision procedure requires these properties, it can still be useful to determine anonymity by just investigating a finite number of processes. The procedure as sketched here is exponential in N as all vectors x must be investigated. Fortunately, in practical cases we already want to achieve anonymity for small groups of processes, so N is a fairly small number.

6 Conclusion

We gave a formal correctness proof for the Dining Cryptographers protocol with an arbitrary number of parties, using the modeling language mCRL2 and its supporting PBES theory. The model in our proof considers an external passive intruder. A very similar proof would work for single internal intruders, but more complex invariants would be needed in order to deal with coalitions of corrupted cryptographers, and this is left for future work.

Due to the fact that data plays an explicit central role in PBES equations, compact symbolic representations are possible, of, e.g., systems consisting of a number of components with similar behaviour. Finding the right invariants requires, as in other frameworks, use of intuition and protocol understanding. However, proving that the proposed predicates are invariants is a mechanical exercise, as well as the application of those invariants to simplifying and eventually solving the target PBES. This makes the PBES framework a comfortable and powerful formalism for such complex correctness proofs.

We showed that the parameterised anonymity problem is undecidable. However, under some restrictions, decidability is possible based on the investigation of a small subgroup of processes. An open challenge is to settle parameterised anonymity for classes of protocols with the help of cutoff theorems as these exist for the parameterised model checking problem(PMCP), e.g. [6].

References

1. M. Bhargava and C. Palamidessi. Probabilistic anonymity. In *Proc. CONCUR'05*, LNCS 3653, pages 171–185, 2005.
2. D. Chaum. The dining cryptographers problem: unconditional sender and receiver untraceability. *Journal of Cryptology*, 1:65–75, 1988.
3. T. Chen, B. Ploeger, J. van de Pol, and T.A.C. Willemse. Equivalence checking for infinite systems using parameterized boolean equation systems. In *Proc. CONCUR'07*, LNCS 4703, pages 120–135, 2007.
4. T. Chothia, S.M. Orzan, J. Pang, and M. Torabi Dashti. A framework for automatically checking anonymity with μCRL. In *Proc. TGC'06*, LNCS 4661, pages 301–318, 2007.
5. H. Comon and V. Shmatikov. Is it possible to decide whether a cryptographic protocol is secure or not? *J. Telecomm. and Inf. Tech.*, 4:3–13, 2002.
6. E. Allen Emerson and Kedar S. Namjoshi. On reasoning about rings. *Int. J. Found. Comput. Sci.*, 14(4):527–550, 2003.
7. M.M. Gallardo, C. Joubert, and P. Merino. Implementing influence analysis using parameterised boolean equation systems. In *Proc. ISOLA'06*, pages 416–424, 2006.
8. J.F. Groote, A.H.J. Mathijssen, M.A. Reniers, Y.S. Usenko, and M.J. van Weerdenburg. The formal specification language mCRL2. In *MMOSS*, Dagstuhl Seminar Proceedings 06351, 2007.
9. J.F. Groote and M.A. Reniers. Algebraic process verification. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 1151–1208. North-Holland, 2001.
10. J.F. Groote and T.A.C. Willemse. Parameterised boolean equation systems. *Theor. Comput. Sci*, 343(3):332–369, 2005.
11. J.Y. Halpern and K.R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, pages 483–514, 2005.
12. D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: A modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.
13. S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *Proc. ESOP'05)*, LNCS 3444, pages 186–200, 2005.
14. R. Küsters. On the decidability of cryptographic protocols with open-ended data structures. *International Journal of Information Security*, 4(1-2):49–70, 2005.
15. R. Küsters and T. Wilke. Automata-based analysis of recursive cryptographic protocols. In *Proc. STACS'04*, LNCS 2996, pages 382–393, 2004.
16. D. Khler, R. Küsters, and T. Wilke. Deciding properties of contract-signing protocols. In *Proc. STACS'05*, LNCS 3404, pages 158–169, 2005.
17. A. Lomuscio and F. Raimondi. MCMAS: A model-checker for multi-agent systems. In *Proc. TACAS'06*, LNCS 3920, pages 450–454, 2006.
18. B. Luttik. On the expressiveness of choice quantification. *Ann. Pure Appl. Logic*, 121(1):39–87, 2003.
19. S. Mauw, J. Verschuren, and E.P. de Vink. A formalization of anonymity and onion routing. In *Proc. ESORICS'04*, LNCS 3193, pages 109–124, 2004.
20. L. Mazaré. Decidability of opacity with non-atomic keys. In *Proc. FAST'04*, pages 71–84, 2004.
21. S.M. Orzan and T.A.C. Willemse. Invariants for parameterised boolean equation systems. In *Proc. CONCUR'08*, LNCS, 2008. To appear.
22. A. Pfitzmann and M. Hansen. Anonymity, unobservability, and pseudonymity: A proposal for terminology, draft v0.23, August 2005.
23. S. Schneider and A. Sidiropoulos. CSP and anonymity. In *Proc. ESORICS'96*, LNCS 1146, 1996.
24. V. Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3/4):355–377, 2004.