Verification of randomized security protocols

Rohit Chadha
University of Missouri
Columbia, MO, USA
Email: chadhar@missouri.edu

A. Prasad Sistla
University of Illinois
Chicago, IL USA
Email: sistla@uic.edu

Mahesh Viswanathan
University of Illinois
Urbana-Champaign, IL, USA
Email: vmahesh@illinois.edu

***Abstract*—We consider the problem of verifying the security of finitely many sessions of a protocol that tosses coins in addition to standard cryptographic primitives against a Dolev-Yao adversary. Two properties are investigated here — *secrecy*, which asks if no adversary interacting with a protocol P can determine a secret sec with probability $> 1 - p$; and *indistinguishability*, which asks if the probability observing any sequence $\overline{o}$ in P_1 is the same as that of observing $\overline{o}$ in P_2, under the same adversary. Both secrecy and indistinguishability are known to be coNP-complete for non-randomized protocols. In contrast, we show that, for randomized protocols, secrecy and indistinguishability are both decidable in coNEXPTIME. We also prove a matching lower bound for the secrecy problem by reducing the non-satisfiability problem of monadic first order logic without equality.**

1. Introduction

Randomization is used in security protocols to achieve security guarantees such as anonymity (see for example dining cryptographers [11], Crowds [32] and onion routing [27]) and voter privacy in electronic voting (see for example [34]). Automated techniques to formally analyze security protocols are necessary because the design of such protocols is subtle and error-prone. While there has been a lot of work on understanding the computational difficulty of verifying security protocols that do not employ randomness, very little is known when it comes to randomized security protocols; this paper aims to address this void. We consider the problem of verifying security for randomized protocols with a bounded number of sessions. The reason for considering only bounded number of sessions is because it is well known that secrecy is undecidable for non-randomized protocols, in general [25], [23], [17].

The usefulness of a formal analysis crucially depends on a judicious delineation of the powers of an adversary. Given the success of the "Dolev-Yao" adversary model and perfect cryptography in non-probabilistic protocols, we make similar assumptions here. Thus protocol participants send and receive messages over public channel controlled by an omnipotent adversary who tosses coins, reads all messages sent on the channel, and can inject new messages into the network as well. The adversarial behavior can depend on the entire communication transcript, but the adversary cannot decrypt messages whose key (s)he is not aware of. However, in the presence of coin tossing steps taken by the protocol, new subtleties need to be accounted for. As many researchers have recently observed [12], [7], [26], [10], [9], it is essential that the protocol coin tosses remain *private to the protocol participants*. In the absence of such guarantees, the analysis can reveal "flaws" where none exist (see examples in [12], [7], [26], [10], [9]).

In order to faithfully model private coin tosses, we follow [5] and our protocol semantics is described using (infinite-branching) partially observable Markov decision processes (POMDPs). POMDPs are often used as models of randomized, nondeterministic systems where some systems states are *indistinguishable* to a scheduler/adversary, who resolves the process nondeterminism. Indistinguishability among states is captured by an equivalence relation on states, and the assumption that the scheduler/adversary only observes the equivalence class of states, and not the actual states, during a computation. In our formalism, each state of the POMDP consists of the state of the protocol principals (that doesn't include the result of coin tosses) and the *frame*, the state of the adversary (i.e., the messages received by the adversary). The nondeterminism models the actions of an adversary who chooses both the next message and its recipient. For equivalence of states, we use static equivalence [2]. The adversary must take the same action in two executions with the same *view*. The view of an execution is the sequence of the equivalence classes of the states and the adversary actions in the execution. Our notion of indistinguishability of views coincides with the trace-indistinguishability of applied-pi calculus processes [2].

Our Contributions. In this paper we establish the complexity of checking two properties of randomized protocols, namely, *secrecy*, and *indistinguishability*. The secrecy problem asks, given a bounded number of sessions of a randomized protocol, a secret name sec, and probability threshold p, is it the case that no adversary learns the secret sec with probability $> (1 - p)$? Our first result is that this secrecy problem is decidable in **coNEXPTIME** for randomized protocols.

We outline the ideas behind establishing this decidability result. Recall that the secrecy problem for *non-probabilistic*

protocols with bounded sessions is **coNP**-complete [33], [23]. This result for non-probabilistic protocols is established by observing that in the "smallest" attack, any message sent by the adversary is either a subterm of a message received by the adversary (i.e., sent by a protocol participant), or is constructed by a composition of such subterms. Moreover, if the adversary ever composes subterms of messages (s)he sends, then the constructed message *must* match (ie, unify with) a non-variable subterm of the protocol of the protocol description. This results in observing that the sizes of messages (when encoded as dags), sent in a smallest attack, are linear in the protocol size and gives an **NP** algorithm to prove insecurity. Unfortunately, these observations no longer hold in the case of randomized protocols. In Example 5.1 on Page 10, we show that there are randomized protocols for which the adversary must send exponential-sized messages in order to break secrecy.

While the strong guarantees of message terseness that hold for non-probabilistic protocols are no longer true, we demonstrate that weaker properties do hold. We show that in the smallest attack, the adversary constructs a new composed message only if in an "equivalent" trace (from the perspective of the adversary's view), the corresponding composed message matches a subterm of the protocol description. This gives us an exponential upper bound on message sizes in the smallest attack, yielding an **NEXPTIME** algorithm to demonstrate insecurity.

Next we prove that this upper bound is optimal, i.e., we show that the secrecy problem is **coNEXPTIME**-hard. We establish this result by reducing the non-satisfiability problem for monadic first order logic without equality, which is known to be **coNEXPTIME**-complete [30]. There are three key ideas that play a role in establishing this result. The first is an observation due to Rusinowitch and Turuani [33] that shows how the satisfiability of propositional logic can be reduced to protocol insecurity. The second observation is that randomization can be used to simulate quantifier alternation, with the adversary making existential choices, and the protocol making universal choices by probabilistic steps. The last ingredient needed is the ability of a randomized protocol to "examine" the contents of an exponential sized message, which underlies the ideas in Example 5.1 on Page 10.

We also consider the problem of checking indistinguishability of two randomized protocols. We say that two protocols P and P' are indistinguishable if for each adversary $\mathcal{A}$ and view $\overline{o}$, the sum of probability of executions of P under $\mathcal{A}$ with the view $\overline{o}$ is the same as sum of probability of executions of P' under $\mathcal{A}$ with the view $\overline{o}$. We show that the indistinguishability problem for randomized protocols is decidable in **coNEXPTIME** for randomized protocols. This is achieved once again by bounding the size of the recipes in a bounded attack. We observe that the protocols in our formalism are *simple*; a protocol is said to be *simple* if there is no principal-level nondeterminism. As a consequence, our notion of indistinguishability coincides with the notion of trace-equivalence for *simple* non-probabilistic protocols [19]. This is because in a simple, non-probabilistic protocol, for each view $\overline{o}$, there can be only one execution with the view $\overline{o}$.

Our last observation is a fixed-parameter complexity result for the secrecy and the indistinguishability problems: if we fix the number of coin tosses but not the number of protocol steps or the size of protocol terms, the secrecy and indistinguishability problems are **coNP**-complete. This generalizes the results for checking secrecy of non-probabilistic security protocols, and for checking indistinguishability of *simple* non-probabilistic protocols [19], which are protocols that fix the number of coin tosses to 0.

The rest of paper is organized as follows. The syntax of the protocols is presented in Section 2 and the semantics in Section 3. The upper bounds for complexity results are presented in Section 4 and lower bounds in Section 5. We conclude in Section 6.

Related Work. For (non-randomized) security protocols, secrecy was shown to be undecidable in [25], [23], [17], although decidability can be obtained for certain subclasses. For bounded number of sessions, secrecy was shown to be **coNP**-complete in [33], [23].

For indistinguishability (equivalence) properties of (non-randomized) security protocols, undecidability is shown in [29]. For bounded number of sessions, decidability is established in [29], [4], [21], [35], [19], [13], [14], [15]. While [29], [35], [13], [14] allow only a limited set of cryptographic primitives (symmetric and asymmetric encryption and cryptographic hash), [21], [4], [35], [19], [15] allow cryptographic primitives modeled as subterm convergent rewrite systems [1]. Furthermore, if the protocols are *determinate* then the decision problem is also **coNP**-complete. Determinate protocols are a generalization of simple protocols described above. [14] is the only decision procedure to consider negative tests, i.e., else branches and non-determinate processes. The complexity of the decision problem for non-determinate protocols remain open.

There are a number of tools that check for security in the non-randomized setting; examples include, Maude-NPA [24], ProverIf [6], AVISPA [3], APTE [13], [14], Scyther [20], Tamarin [31] and AKiSs [8].

2. Protocol syntax

We assume the reader is familiar with probability spaces. The set of all discrete measures on a set S will be denoted by $\mathsf{Dist}(S)$. The *support* of a measure $\mu \in \mathsf{Dist}(S)$ is the set $\{s \in S \mid \mu(s) > 0\}$. We will assume that for each $\mu \in \mathsf{Dist}(S)$ the support of μ is finite. As usual, the set of finite sequences over A will be denoted by A^*.

2.1. Terms, substitutions and frames

Terms. We assume a countable set Pub of *public names*, and a disjoint countable set Prv of *private names* that will model secret nonces and secret keys that a protocol participant may use. We assume that there is a countable subset $\mathsf{PrivKeys} \subset \mathsf{Prv}$ which will model private asymmetric

encryption keys. We assume that there is a countable set $\mathsf{PublicKeys} \subset \mathsf{Pub}$ which will model public asymmetric encryption keys. Furthermore, we assume that there is a bijection $^{-1} : \mathsf{PublicKeys} \rightarrow \mathsf{PrivKeys}$ which maps public keys to their corresponding private decryption keys. We use $\mathsf{pk}, \mathsf{pk}_1, \ldots$ to range over public encryption keys. For a public encryption key pk, the corresponding private key will be denoted by pk^{-1}. We also assume that the sets $\mathsf{Prv} \setminus \mathsf{PrivKeys}$ and $\mathsf{Pub} \setminus \mathsf{PublicKeys}$ are countably infinite.

We assume two sets of function symbols; a set $\mathcal{F}_c$ of *constructor* function symbols and a set $\mathcal{F}_d$ of *destructor* function symbols. The set $\mathcal{F}_c$ consists of binary function symbols $\mathsf{senc}, \mathsf{aenc}$ and the pairing function $[\cdot,\cdot]$. The set $\mathcal{F}_d$ consists of binary function symbols $\mathsf{sdec}, \mathsf{adec}$ and unary function symbols proj_1 and proj_2. $\mathsf{senc}/\mathsf{sdec}$ will model symmetric encryption/decryption and $\mathsf{aenc}/\mathsf{adec}$ will model asymmetric encryption/decryption. We assume a countable set of protocol variables $\mathcal{X}$ and a disjoint countable set $\mathcal{X}_w$ of frame variables. $\mathcal{X}$ will be used as variables in the protocol description, while $\mathcal{X}_w$ will be used to refer to messages received by the adversary. We assume that there is a fixed enumeration $\{w_1, w_2, \ldots, \}$ of variables in $\mathcal{X}_w$.

Given a set of names $N \subseteq \mathsf{Pub} \cup \mathsf{Prv}$, a set of function symbols $F \subseteq \mathcal{F}_c \cup \mathcal{F}_d$ and a set of variables $X \subseteq \mathcal{X} \cup \mathcal{X}_w$, the set of terms built using F, N and X is defined as usual. The subterms of a term t are defined as usual. We will say that a term t is well-formed if for each term t_1, t_2, whenever $\mathsf{aenc}(t_1, t_2)$ is a subterm of t then $t_1 \in \mathsf{PublicKeys}$. The set of well-formed terms built using F, N and X will be denoted by $\mathcal{T}(F, N, X)$. As is the case in [33], for the rest of the paper, we will assume that all terms are well-formed [1]. We use $Sub(t)$ to denote the set of subterms of t and $Sub(T)$ to denote the set of subterms of a set of terms T. We will write $t \sqsubseteq t'$ if t is a subterm of t' and $t \sqsubset t'$ if $t \sqsubseteq t'$ and $t \neq t'$. The set of variables occurring in a term is denoted by $vars(t)$. The set of names occurring in a term will be denoted by $names(t)$. A ground term is one that has no variables appearing in it. We identify the following abbreviations for sets of terms:

- The set of all terms $\mathcal{T}(\mathcal{F}_c \cup \mathcal{F}_d, \mathsf{Prv} \cup \mathsf{Pub}, \mathcal{X} \cup \mathcal{X}_w)$ will be denoted by $Terms$.
- The set $\mathcal{T}(\mathcal{F}_c, \mathsf{Pub} \cup \mathsf{Prv}, \mathcal{X})$ of *constructor terms* will be denoted by $CTerms$.
- The set $\mathcal{T}(\mathcal{F}_c \cup \mathcal{F}_d, \mathsf{Pub}, \mathcal{X}_w)$ of *recipes* will be denoted by $Recipes$.

Any term t can be viewed as a node-labeled and ordered finite tree, with nodes labeled by either a function symbol, or a name, or a variable. Leaves are labeled by either a name or a variable. Internal nodes are labeled by function symbols, with the number of children being determined by the arity of the function symbol. The edges from an internal node to its children are assumed to be numbered sequentially from left to right starting from one; these numbers are considered as the labels of the edges. Now, the position of each node of t can be uniquely represented by the sequence of the labels of the edges on the path from the root to the node, with the root node being represented by the empty sequence. We let $t|p$ denote the subterm of t represented by the subtree of t rooted at position p. We also let $positions(t)$ denote the set of positions of nodes of t. By the height of a term, denoted $height(t)$, we mean the height of the tree representing t; the height of a tree with one node being taken to be 0. Size of a term t is defined to be the size of the tree representing t, *i.e.*, the number of nodes in the tree representing t.

Any term t can also be represented as a node-labeled directed acyclic graph (dag) $dag(t)$ [33], [1]. Each node of t is labeled by either a function symbol or a name or a variable. If a node is labeled by a name or a variable then its out-degree is 0. If it is labeled by a function symbol f then its out-degree is the arity of the function symbol f, and its outgoing edges are assumed to be numbered sequentially from 1 to the arity of the function symbol f. Every node n of $dag(t)$ represents a sub-term of t. If the node is labeled by a name (variable respectively) then it represents this name (variable respectively). If n is labeled by a function symbol f of arity n then it represents the term $f(t_1, \ldots, t_n)$ where t_i is the term represented by the node n_i such that the edge from n to n_i is numbered i. Furthermore, we assume that the distinct nodes of $dag(t)$ represent different subterms of t. The dag-size of t is assumed to be the number of vertices of $dag(t)$ and is the number of distinct subterms of t.

We assume that cryptographic operations are modeled via the means of a convergent rewriting system $\mathcal{R}$ on $Terms$. The set $\mathcal{R}$ consists of

$$\begin{array}{lcl} \mathsf{adec}(\mathsf{pk}^{-1}, \mathsf{aenc}(\mathsf{pk}, y)) & \rightarrow & y \\ \mathsf{sdec}(x, \mathsf{senc}(x, y)) & \rightarrow & y \\ \mathsf{proj}_1([x, y]) & \rightarrow & x \\ \mathsf{proj}_2([x, y]) & \rightarrow & y. \end{array}$$

For $t \in Terms$, $\mathsf{nf}(t)$ denotes the normal form obtained by rewriting t using $\mathcal{R}$. We write $t_1 =_{\mathcal{R}} t_2$ if $\mathsf{nf}(t_1) = \mathsf{nf}(t_2)$. We also say that a term t is valid, denoted $valid(t)$, if $\mathsf{nf}(u) \in CTerms$ for any subterm u of t.

Substitutions. A substitution σ is a function that maps variables to terms. The set $dom(\sigma) = \{x \in \mathcal{X} \cup \mathcal{X}_w \,|\, \sigma(x) \neq x\}$ is said to be the *domain* of the substitution σ. For the rest of the paper, each substitution will have a finite domain. A substitution σ with domain $\{x_1, ..., x_k\}$ will be denoted as $\{x_1 \mapsto t_1, ..., x_k \mapsto t_k\}$ if $\sigma(x_i) = t_i$. The set $\{t_1, .., t_k\}$ will be denoted by $ran(\sigma)$. A substitution σ is said to be *ground* if every term in $ran(\sigma)$ is ground and *valid* if every term in $ran(\sigma)$ is valid. A substitution with empty domain will be denoted as $\emptyset$. A substitution can be extended to terms in the usual way. We write $t\sigma$ for the term obtained by applying the substitution σ to the term t.

Two terms t_1, t_2 are said to be *unifiable* if there exists a substitution σ such that $t_1\sigma = t_2\sigma$; here σ is said to be the *unifier* of t_1 and t_2. We write $mgu(t_1, t_2)$ for the most general unifier for t_1, t_2. For the rest of the paper, we assume

1. This definition of well-formed terms does not restrict the adversary's power. Protocol specifications in our formalism assume that each (sent or received) message encrypted using asymmetric encryption is encrypted with a public key. An attacker sending a message that will be received by a protocol role can always use the expected public key.

that substitutions are in normal form, i.e., for each $x \in dom(\sigma)$, $\mathsf{nf}(\sigma(x)) = \sigma(x)$. Given two substitutions σ_1 and σ_2 such that $dom(\sigma_1) \cap dom(\sigma_2) = \emptyset$, we write $\sigma_1 \cup \sigma_2$ as the unique substitution whose domain is $dom(\sigma_1) \cup dom(\sigma_2)$ and $\sigma(x) = \sigma_1(x)$ if $x \in dom(\sigma_1)$ and $\sigma(x) = \sigma_2(x)$ if $x \in dom(\sigma_2)$.

Frames. Intuitively, a frame represents the sequence of messages obtained by the adversary. Formally, a frame φ is a ground and valid substitution such that $dom(\varphi) \subseteq \mathcal{X}_w$ and $dom(\varphi) = \{w_1, w_2, \ldots, w_{|dom(\varphi)|}\}$. The set of frames will be denoted by $Frames$. $\varphi(w_i)$ denotes the ith message received by the adversary. For a frame φ with $dom(\varphi) = \{w_1, w_2, \ldots, w_n\}$ and a ground term t in normal form, $\varphi \uplus t$ will denote the frame φ' such that $|dom(\varphi')| = n + 1$ and

$$\varphi'(w_i) = \begin{cases} \varphi(w_i) & \text{if } i \leq n \\ t & \text{otherwise} \end{cases}.$$

Intuitively, a term t is deducible if the adversary can compute it using the messages it has received and public names. A *valid ground term* t in normal form is deducible from a frame φ with a recipe $r \in Recipes$, denoted $\varphi \vdash^r t$ if $vars(r) \subseteq dom(\varphi)$, $valid(r\varphi)$ and $\mathsf{nf}(r\varphi) = t$.

Intuitively, two frames are considered statically equivalent if the adversary cannot distinguish them. An adversary tries to distinguish two frames by performing tests. There are two kinds of tests, one for validity of recipes and the second for equality of recipes. Formally, two frames φ_1, φ_2 are said to be *statically equivalent*, denoted $\varphi_1 \sim \varphi_2$, if

- $dom(\varphi_1) = dom(\varphi_2)$,
- For each recipe r with $vars(r) \subseteq dom(\varphi_1)$, $valid(r\varphi_1)$ iff $valid(r\varphi_2)$, and
- For each pair of recipes r, r' such that $vars(r), vars(r') \subseteq dom(\varphi_1)$, $valid(r\varphi_1)$, $valid(r'\varphi_1)$, $r\varphi_1 =_{\mathcal{R}} r'\varphi_1$ iff $r\varphi_2 =_{\mathcal{R}} r'\varphi_2$.

Our definition is inspired from [2] and follows more recent definitions, for example [16].

Example 2.1. Let k be a private name and let $\mathbf{0}$ and $\mathbf{1}$ be two distinct public names. Consider the two frames $\varphi = \{w_1 \mapsto \mathsf{senc}(k, \mathbf{0})\}$ and $\varphi' = \{w_1 \mapsto \mathsf{senc}(k, \mathbf{1})\}$. These two frames are statically equivalent even though the adversary has different terms. Intuitively, this is because the adversary cannot decrypt the ciphertext. On the other hand, the two frames $\varphi_1 = \{w_1 \mapsto \mathsf{senc}(k, \mathbf{0}), w_2 \mapsto k\}$ and $\varphi_1' = \{w_1 \mapsto \mathsf{senc}(k, \mathbf{1}), w_2 \mapsto k\}$ are not statically equivalent as $\mathsf{sdec}(w_2, w_1)\,\varphi_1 =_{\mathcal{R}} \mathbf{0} =_{\mathcal{R}} \mathbf{0}\,\varphi_1$ but $\mathsf{sdec}(w_2, w_1)\,\varphi_1' =_{\mathcal{R}} \mathbf{1} \neq_{\mathcal{R}} \mathbf{0}\,\varphi_1'$.

2.2. Protocols

We define protocols as a finite set of communicating roles. A role models the actions of one participant in one instance of the protocol and all communication is assumed to be mediated by the adversary. Each role performs a finite number of actions. In each action of the protocol participant, the participant receives a message, tosses coins and sends out a message. We will assume that each protocol role itself is deterministic. Before giving the formal definition, we give an example of a protocol that will be used to illustrate the protocol syntax and semantics.

Example 2.2. We model a simple electronic voting protocol in our formalism. The electronic voting protocol will have 2 voters A and B with public keys $\mathsf{pk_A}$ and $\mathsf{pk_B}$ respectively and an election authority (EA) with public key $\mathsf{pk_{EA}}$. The two voters will choose amongst two candidates who will be modeled as public names $\mathbf{0}$ and $\mathbf{1}$ in the set $\mathsf{Pub} \setminus \mathsf{PublicKeys}$. The protocol will proceed as follows. Initially, the election authority will generate two private tokens $\mathsf{tk_A}$ and $\mathsf{tk_B}$ and send them to A and B encrypted under their respective public keys. These tokens will be used by the voters as proofs of their eligibility. A voter after receiving its token will send its choice to the EA along with its proof of eligibility encrypted under $\mathsf{pk_{EA}}$. The EA, after receiving both the votes and checking their validity, tosses a fair coin. If tails turns up then it outputs A's vote first and then $B's$ vote. The order is reversed if head turns up. The security property that we are interested is that the protocol must respect *privacy* of votes, i.e., an adversary should not be able to deduce from the results how each voter voted.

Defintion 2.3. A protocol role R is a tuple $(S, s_0, \prec, \Delta)$ where

- S is a finite set of states,
- s_0 is the initial state,
- $\prec$ is a strict partial order on S,
- $\Delta \subseteq S \times CTerms \times \mathsf{Dist}(S \times CTerms)$ is a finite set of transitions such that
  - If $(s, t, \mu) \in \Delta$ then $s \prec s'$ whenever $\exists t'.\, \mu((s', t')) > 0$.
  - If (s, t, μ) and (s, t_1, μ_1) are distinct elements of Δ then t and t_1 are not unifiable.
  - If $(s, t, \mu) \in \Delta$ then the support of μ is finite.

Remark: The condition that if (s, t, μ) and (s, t_1, μ_1) are elements of Δ then t and t_1 are not unifiable ensures that a role is deterministic.

Notation: For a transition $tr = (s, t, \mu) \in \Delta$ we denote t as $lht(tr)$ (left-hand term of tr). If the support of μ is $(s_1, t_1), \ldots, (s_n, t_n)$ and $p_i = \mu((s_i, t_i))$ for each $i = 1, \ldots, n$, then we write tr as

$$s : t \Rightarrow [s_1 : t_1]^{p_1} \oplus \cdots \oplus [s_n : t_n]^{p_n}.$$

If the $p_i = \frac{1}{n}$ for each i, we will ignore the superscripts and just write

$$s : t \Rightarrow [s_1 : t_1] \oplus \cdots \oplus [s_n : t_n].$$

We use $vars(tr)$ for $vars(t) \bigcup \cup_{1 \leq i \leq n} vars(t_i)$. For the role R, we write $vars(R) = \cup_{tr \in \Delta}\ vars(tr)$. Similarly, we use $terms(tr) = \{t, t_1, \ldots, t_n\}$ and $terms(R) = \cup_{tr \in \Delta}\ terms(tr)$. We take $span(tr) = n$ and $span(R) = \max_{tr \in \Delta}\ span(tr)$. The *size* of tr is the sum of the sizes of

each term $t, t_i \in terms(tr)$ and the sizes of the numerator and denominator of each p_i written in binary.

Example 2.4. We show how EA in Example 2.2 can be specified as a role R_{EA}. R_{EA} will have 3 states $s_0 \prec s_w \prec s_f$. s_0 is the initial state of the protocol, s_w is the state in which EA has sent the eligibility token and is waiting for the votes, and s_f is the final state of the protocol. There are two protocol transitions:

$$s_0 : ok \Rightarrow [s_w : [\mathsf{aenc}(\mathsf{pk_A}, \mathsf{tk_A}), \mathsf{aenc}(\mathsf{pk_B}, \mathsf{tk_B})]]$$
$$s_w : [\mathsf{aenc}(\mathsf{pk_{EA}}, [\mathsf{tk_A}, [x, z_1]]), \mathsf{aenc}(\mathsf{pk_{EA}}, [\mathsf{tk_B}, [y, z_2]]] \Rightarrow [s_f : [x, y]] \oplus [s_f : [y, x]]$$

Here ok is a public name, $\mathsf{tk_A}$, $\mathsf{tk_B}$ are private names and x, z_1, y, z_2 are variables. Intuitively, x and y are the votes of A and B respectively, and z_1 and z_2 are the nonces used by A and B when sending their votes to the EA. The importance of these nonces will be explained later.

Defintion 2.5. A protocol P with n roles is a tuple $(\varphi_0, R_1, \ldots, R_n)$ such that φ_0 is a frame and $R_1, \ldots, R_n$ are roles such that $vars(R_i) \cap vars(R_j) = \emptyset$ for $i \neq j$.

In a protocol $P = (\varphi_0, R_1, \ldots, R_n)$, φ_0 models the initial knowledge the adversary has and $R_1, \ldots, R_n$ model the different roles executing the protocol P.

Example 2.6. Consider the voting protocol given in Example 2.2. We have already described the role R_{EA} in Example 2.4. Now for a voter $v \in \{A, B\}$ and a choice $vote \in \{\mathbf{0}, \mathbf{1}\}$, $R_v(vote)$ will denote the role in which v votes for choice $vote$. We now describe the role $R^A_{\mathbf{0}}$ when A votes for candidate $\mathbf{0}$. A has two states $s^A_0 \prec s^A_1$. Intuitively s^A_0 is the initial state where A is waiting for the token from EA and s^A_1 is the final state for A. There is only one transition:

$$s^A_0 : \mathsf{aenc}(\mathsf{pk_A}, tk) \Rightarrow [s^A_1 : \mathsf{aenc}(\mathsf{pk_{EA}}, [tk, [\mathbf{0}, n_A]]].$$

Here tk is a variable and n_A is a private name used by A. We can similarly define the roles $R^A_{\mathbf{1}}, R^B_{\mathbf{0}}$ and $R^B_{\mathbf{1}}$. We assume that the attacker has a private/public key pair, $\mathsf{sk_O}/\mathsf{pk_O}$. The scenario when A votes for v_A and B votes for v_B can be described as a protocol $P_{v_A, v_B} = (\{w_1 \mapsto \mathsf{sk_O}\}, R_{EA}, R^A_{v_A}, R^B_{v_B})$ with three roles and the initial frame containing the knowledge $\mathsf{sk_O}$.

Remark: In order to model multiple sessions of the same protocol, we will model a principal's action in one session as a separate role.

3. Protocol semantics

The semantics of a protocol will be defined using Partially Observable Markov Decision Processes (POMDP)s. We begin by recalling some standard notions from Markov Chains and POMDPs.

3.1. Discrete-time Markov Chains (DTMCs)

A DTMC is used to model systems which exhibit probabilistic behavior. Formally, a DTMC is a tuple $\mathcal{M} = (Z, z_s, \Delta)$ where Z is a countable set of *states*, z_s the *initial* state and $\Delta : Z \hookrightarrow \mathsf{Dist}(Z)$ is a (partial) function which is called the *transition function* and which maps Z to a (discrete) probability distribution over Z. We say that $enabled(z)$ is true iff $\Delta(z)$ is defined. Informally, the process modeled by $\mathcal{M}$ evolves as follows. The process starts in the state z_s. After i execution steps, if the process is in the state z, the process moves to state z' at execution step $(i+1)$ with probability $\Delta(z)(z')$.

An execution ρ of $\mathcal{M}$ of length m is a sequence $z_0 \to z_1 \to \cdots \to z_m$ such that $z_0 = z_s$ and for each $i \geq 0$, $\Delta(z_i)(z_{i+1}) > 0$. For an execution ρ, as given above, we say $last(\rho) = z_m$. The *measure* of the execution ρ is said to be the product $\prod_{i=0}^{m-1} \Delta(z_i)(z_{i+1})$. The set of all executions of $\mathcal{M}$ will be denoted by $\mathsf{Exec}(\mathcal{M})$. For each Markov chain $\mathcal{M}$ that we will construct, there is a bound $L_{\mathcal{M}}$ such that the length of each execution in $\mathsf{Exec}(\mathcal{M})$ is $\leq L_{\mathcal{M}}$.

An execution ρ_1 is said to be a *one-step extension* of the execution $\rho = z_0 \to z_1 \to z_2 \cdots \to z_m$ if there exists z_{m+1} such that $\rho_1 = z_0 \to z_1 \to z_2 \cdots \to z_m \to z_{m+1}$. In this case, we say that ρ_1 extends ρ by z_{m+1}. We will say that an execution ρ is *maximal* if ρ does not have any one-step extension. Notice that the set of all executions of $\mathcal{M}$ forms a labeled tree $\mathcal{T}_{\mathcal{M}}$ whose root node is z_s. Each node in this tree is an execution of $\mathcal{M}$. For the tree $\mathcal{T}_{\mathcal{M}}$, we take the label of a node ρ to be $last(\rho)$.

3.2. Partially Observable Markov Decision Processes (POMDPs)

POMDPs are used to model processes which exhibit both probabilistic and non-deterministic behavior, where the states of the system are only partially observable. Formally, a POMDP is a tuple $\mathcal{M} = (Z, z_s, Act, \Delta, \mathcal{O}, Obs)$ where Z is a countable set of *states*, $z_s \in Z$ is the *initial* state, Act is a (countable) set of *actions*, $\Delta : Z \times Act \hookrightarrow Dist(Z)$ is a partial function called the *probabilistic transition relation*, $\mathcal{O}$ is a set of *observations* and $Obs : Z \to \mathcal{O}$ is a function called the *observation function* that maps each state to an observation. As a matter of notation, we will write $z \xrightarrow{\alpha} \mu$ whenever $\Delta(z, \alpha) = \mu$. Furthermore, we say that $enabled(z, \alpha)$ is true iff $\Delta(z, \alpha)$ is defined. A POMDP is like a Markov Chain except that at each state z, there is a choice amongst several possible probabilistic transitions. The choice of which probabilistic transition to *trigger* is resolved by an *adversary*. Informally, the process modeled by $\mathcal{M}$ evolves as follows. The process starts in the state z_s. After i execution steps, if the process is in the state z, then the adversary chooses an action α with probability p_α such that $z \xrightarrow{\alpha} \mu$ and the process moves to state z' at the $(i+1)$-st execution step with probability $\mu(z')p_\alpha$.

An *execution* ρ of length m in the POMDP $\mathcal{M}$ is a finite sequence $z_0 \xrightarrow{\alpha_0} z_1 \cdots \xrightarrow{\alpha_{m-1}} z_m$ such that $z_0 = z_s$,

and for each $i \geq 0$, there exists μ_{i+1} with $z_i \xrightarrow{\alpha_i} \mu_{i+1}$ and $\mu_{i+1}(z_{i+1}) > 0$; the length of an execution will be denoted as $length(\rho)$. The set of all finite executions of $\mathcal{M}$ will be denoted by $\mathsf{Exec}(\mathcal{M})$. For $\rho = z_0 \xrightarrow{\alpha_0} z_1 \xrightarrow{\alpha_1} z_2 \cdots \xrightarrow{\alpha_{m-1}} z_m$, we write $last(\rho) = z_m$ and $last\_action(\rho) = \alpha_{m-1}$ if $m > 0$. An execution ρ_1 is said to be a *one-step extension* of the execution $\rho = z_0 \xrightarrow{\alpha_0} z_1 \xrightarrow{\alpha_1} z_2 \cdots \xrightarrow{\alpha_{m-1}} z_m$ if there exists α_m and z_{m+1} such that $\rho_1 = z_0 \xrightarrow{\alpha_0} z_1 \xrightarrow{\alpha_1} z_2 \cdots \xrightarrow{\alpha_{m-1}} z_m \xrightarrow{\alpha_m} z_{m+1}$. In this case, we say that ρ_1 extends ρ by (α_m, z_{m+1}). In the rest of the paper we only consider POMDPs $\mathcal{M}$ with the property that there is a bound $L_{\mathcal{M}}$ such that all executions in $\mathsf{Exec}(\mathcal{M})$ have length $\leq L_{\mathcal{M}}$.

Given an execution $\rho = z_0 \xrightarrow{\alpha_0} z_1 \xrightarrow{\alpha_1} z_2 \cdots \xrightarrow{\alpha_{m-1}} z_m$, the *view* of ρ, written $view(\rho)$, is the sequence $Obs(z_0)\alpha_0 Obs(z_1)\alpha_1 \cdots \alpha_{m-1} Obs(z_m)$. Let $\mathcal{V}$ be the set $(\mathcal{O} \cdot Act)^* \cdot \mathcal{O}$. Observe that $view(\rho)$ is an element of $\mathcal{V}$. As discussed above, the choice of which transition to take in an execution is resolved by an adversary. Formally, an *adversary* is a function $\mathcal{A} : \mathcal{V} \to \mathsf{Dist}(Act)$. An adversary $\mathcal{A}$ resolves all non-determinism and the resulting behavior can be described by a DTMC $\mathcal{M}^{\mathcal{A}} = (\mathsf{Exec}(\mathcal{M}) \cup (Act \times \{\diamond\}), z_s, \Delta^{\mathcal{A}})$ where $\diamond$ is a special symbol. $\Delta^{\mathcal{A}}((\alpha, \diamond))$ is undefined for each $\alpha \in Act$. For each $\rho \in \mathsf{Exec}(\mathcal{M})$, $\Delta^{\mathcal{A}}(\rho)$ is the unique discrete distribution that satisfies the following:

- For each $\rho_1 \in \mathsf{Exec}(\mathcal{M})$, $z \in Z$, $\alpha \in Act$ such that $z = last(\rho_1)$ and ρ_1 extends ρ by (α, z), $\Delta^{\mathcal{A}}(\rho)(\rho_1) = \mathcal{A}(view(\rho))(\alpha) \cdot \Delta(last(\rho), \alpha)(z)$,
- $\Delta^{\mathcal{A}}(\rho)((\alpha, \diamond)) = \mathcal{A}(view(\rho))(\alpha)$ if and only if $enabled(last(\rho), \alpha)$ is false.

Observe that an execution $\kappa = \zeta_0 \to \zeta_1 \to \ldots \to \zeta_m \in \mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$ is such that $\zeta_i \in \mathsf{Exec}(\mathcal{M})$ for each $i < m$. ζ_m is either an element in $\mathsf{Exec}(\mathcal{M})$ or a state $(\alpha, \diamond)$ for some $\alpha \in Act$. Furthermore κ is maximal if and only if ζ_m is $(\alpha, \diamond)$ for some $\alpha \in Act$.

State-Based Safety properties. Given a POMDP $\mathcal{M} = (Z, z_s, Act, \Delta, \mathcal{O}, Obs)$, a set $\Psi \subseteq Z$ is said to be a *state-based safety property*. An execution $\rho = z_0 \xrightarrow{\alpha_0} z_1 \xrightarrow{\alpha_1} z_2 \cdots \xrightarrow{\alpha_{m-1}} z_m$ of $\mathcal{M}$ is said to satisfy Ψ if $z_j \in \Psi$ for all $0 \leq j \leq m$. An execution $\kappa = \zeta_0 \to \zeta_1 \to \ldots \to \zeta_n \in \mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$ is said to satisfy Ψ, denoted $\kappa \models \Psi$, if whenever $\zeta_i \in \mathsf{Exec}(\mathcal{M})$ then ζ_i satisfies Ψ [2]. We say $\mathcal{M}$ satisfies Ψ with probability $\geq p$ against the adversary $\mathcal{A}$ (written $\mathcal{M}^{\mathcal{A}} \models_p \Psi$) if the sum of the measures of executions in the set $\{\kappa \in (\mathsf{Exec}(\mathcal{M}^{\mathcal{A}})) \mid \kappa \text{ is maximal and } \kappa \models \Psi\}$ in the DTMC $\mathcal{M}^{\mathcal{A}}$ is $\geq p$. We say that $\mathcal{M}$ satisfies Ψ with probability $\geq p$ (written $\mathcal{M} \models_p \Psi$) if for all adversaries $\mathcal{A}$, $\mathcal{M}^{\mathcal{A}} \models_p \Psi$.

Indistinguishability. Recall that $\kappa = \zeta_0 \to \zeta_1 \to \ldots \to \zeta_m \in \mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$ is such that $\zeta_i \in \mathsf{Exec}(\mathcal{M})$ for each $i < m$. We will write that $view(\kappa)$ is $view(\zeta_m)$ if $\zeta_m \in \mathsf{Exec}(\mathcal{M})$ and $view(\zeta_{m-1}) \cdot \zeta_m$ otherwise. Given a POMDP $\mathcal{M} = (Z, z_s, Act, \Delta, \mathcal{O}, Obs)$, a sequence $\overline{o} \in \mathcal{V} \cup (\mathcal{V} \cdot (Act \times \{\diamond\}))$ and an adversary $\mathcal{A}$, the probability of $\mathcal{A}$ observing $\overline{o}$, written $pr_{\mathcal{M}}(\overline{o}, \mathcal{A})$, is the sum of the measures of executions in the set $\{\kappa \in (\mathsf{Exec}(\mathcal{M}^{\mathcal{A}})) \mid view(\kappa) = \overline{o}\}$.

Given two POMDPs $\mathcal{M} = (Z, z_s, Act, \Delta, \mathcal{O}, Obs)$ and $\mathcal{M}' = (Z', z'_s, Act, \Delta', \mathcal{O}, Obs')$ with the same set of actions and observations, we say that $\mathcal{M}$ and $\mathcal{M}'$ are distinguishable by an adversary $\mathcal{A}$ if there is an $\overline{o} \in \mathcal{V} \cup (\mathcal{V} \cdot (Act \times \{\diamond\}))$ such that $pr_{\mathcal{M}}(\overline{o}, \mathcal{A}) \neq pr_{\mathcal{M}'}(\overline{o}, \mathcal{A})$. We say that $\mathcal{M}$ and $\mathcal{M}'$ are indistinguishable if they cannot be distinguished by any attacker. The following proposition states that it suffices to consider only $\overline{o} \in \mathcal{V}$ for indistinguishability (see Appendix B for the proof).

Proposition 3.1. Let $\mathcal{M} = (Z, z_s, Act, \Delta, \mathcal{O}, Obs)$, $\mathcal{M}' = (Z', z'_s, Act, \Delta', \mathcal{O}, Obs')$ and $\mathcal{A}$ be an adversary. If there is a $\overline{o} \in \mathcal{V} \cdot (Act \times \{\diamond\})$ such that $pr_{\mathcal{M}}(\overline{o}, \mathcal{A}) \neq pr_{\mathcal{M}'}(\overline{o}, \mathcal{A})$ then there is a $\overline{o}_1 \in \mathcal{V}$ such that $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}) \neq pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A})$.

Pure Adversaries. An adversary $\mathcal{A} : \mathcal{V} \to \mathsf{Dist}(Act)$ is said to be pure if for each $\overline{o} \in \mathcal{V}$, $\mathcal{A}(\overline{o})(\alpha)$ is non-zero for exactly one $\alpha \in Act$. It turns out that it suffices to consider only pure adversaries for safety and indistinguishability properties. Intuitively, an adversary does not gain by tossing coins as it can always choose to do the worst-possible action.

Proposition 3.2. If $\mathcal{M}^{\mathcal{A}} \not\models_p \Psi$ then there is a pure adversary $\mathcal{A}_0$ such that $\mathcal{M}^{\mathcal{A}_0} \not\models_p \Psi$. If $\mathcal{M}$ and $\mathcal{M}'$ are distinguishable then they are distinguishable by a pure adversary.

Remark: For a POMDP $\mathcal{M}$ and adversary $\mathcal{A}$ let $\mathcal{M}^{\mathcal{A}}_{alive}$ be the DTMC obtained from $\mathcal{M}^{\mathcal{A}}$ by removing the set of states $Act \times \{\diamond\}$. Observe that if $\mathcal{A}$ is a pure adversary then for any execution ρ, the transition probability of going from ρ to a state $(\alpha, \diamond)$ in $\mathcal{M}^{\mathcal{A}}$ is either 0 and 1. An execution κ in $\mathsf{Exec}(\mathcal{M}^{\mathcal{A}}_{alive})$ is maximal iff the execution $\kappa \cdot (\alpha, \diamond)$ is maximal in $\mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$ where $\alpha = \mathcal{A}(view(last(\kappa)))$. These facts imply that it suffices to consider $\mathcal{M}^{\mathcal{A}}_{alive}$ for pure adversaries $\mathcal{A}$ when considering the probability of satisfying a safety property. Similarly, Proposition 3.1 implies that it suffices to consider $\mathcal{M}^{\mathcal{A}}_{alive}$ and $(\mathcal{M}')^{\mathcal{A}}_{alive}$ when distinguishing $\mathcal{M}$ and $\mathcal{M}'$.

For the rest of the paper, we will assume that all of our adversaries are pure and we will confuse $\mathcal{M}^{\mathcal{A}}$ by $\mathcal{M}^{\mathcal{A}}_{alive}$. Note that a pure adversary $\mathcal{A}$ can also be uniquely identified by the function $\mathcal{A}_{pure}$ from $\mathcal{V}$ to Act such that for each $\overline{o}$, $\mathcal{A}_{pure}(\overline{o}) = \alpha$ where α is the action such that $\mathcal{A}(\overline{o})(\alpha) = 1$. Thus, in the rest of the paper, we will consider a pure adversary to be a function from $\mathcal{V}$ to Act. Also observe that as defined above, the measure of an execution $\kappa \in \mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$ is exactly the measure of the execution $last(\kappa) \in \mathsf{Exec}(\mathcal{M})$ where the measure of the execution $\rho = z_0 \xrightarrow{\alpha_0} z_1 \xrightarrow{\alpha_1} z_2 \cdots \xrightarrow{\alpha_{m-1}} z_m$ is said to be the product $\prod_{i=0}^{m-1} \Delta(z_i, \alpha_i)(z_{i+1})$.

3.3. Semantics

The semantics of the protocol P is given in terms of a POMDP $\mathcal{M}_P$. A state of the POMDP $\mathcal{M}_P$ consists of

2. This definition means that we are treating $(\alpha, \diamond)$ as safe iff α is safe for our purposes.

a state s_i of each role, a ground substitution σ that maps variables in the protocol to ground terms, the frame φ that models the messages that have been received by the adversary, the message received from the adversary before the current state was entered, and the message sent out by the protocol role just before transitioning to the current state. The actions of the POMDP $\mathcal{M}_P$ model the actions of the adversary. In an action, the adversary chooses which role will move next and also constructs (using its frame) the message that is sent to that role. The role accepts a message from the adversary only if the message is valid and matches one of the left hand sides of one of its rules.

Defintion 3.3. The semantics of the protocol $P = (\varphi_0, R_1, R_2, \dots R_n)$ with s_i^0 as the initial state of R_i is given in terms of a POMDP $\mathcal{M}_P = (Z, z_s, Act, \Delta, Frames/\sim, Obs)$ where

- $Z = \{((s_1, s_2, \dots s_n), \sigma, \varphi, \ell) \mid s_i \text{ is a state of } R_i, \sigma$ a ground substitution, φ a frame and ℓ is an element in the set $\{\star\} \cup (CTerms \times CTerms)\}$.
- $Act = \{1, 2, \dots, n\} \times Recipes$.
- $z_s = ((s_1^0, s_2^0, \dots, s_n^0), \emptyset, \varphi_0, \star)$.
- For a state $z = ((s_1, s_2, \dots s_n), \sigma, \varphi, \ell) \in Z$ and an action $\alpha = (i, r)$, the transition $\Delta(z, (i, r))$ is defined iff there is a transition tr given as

$$s_i : t_i \Rightarrow [s_1' : t_1']^{p_1} \oplus \dots \oplus [s_n' : t_n']^{p_n}$$

  of R_i and a ground term t such that
  - $vars(tr) \setminus vars(lht(tr)) \subseteq dom(\sigma)$,
  - $\varphi \vdash^r t$ and
  - t is valid and unifiable with $t_i\sigma$.

  If defined, let tr be the unique transition as given above (uniqueness follows from the fact that R_i is deterministic) and t be the term as given above. Let $\sigma' = \sigma \cup mgu(t, t_i\sigma)$.
  $\Delta(z, \alpha)$ is the unique probability distribution such that for a state $z' = ((s_1, s_2, \dots, s_{i-1}, s_j', s_{i+1}, \dots s_n), \sigma', \varphi \uplus \mathsf{nf}(t_j'\sigma'), (t_i, t_j'))$, (where $[s_j' : t_j']$ is one of the options on the right hand side of tr)
  $\Delta(z, \alpha)(z') = p_j$ (where p_j is the probability for option $[s_j' : t_j']$ in tr).
- The set of observations is $Frames/\sim$, the set of equivalence classes of the set of frames under static equivalence.
- $Obs(((s_1, s_2, \dots s_n), \sigma, \varphi, \ell)) = \varphi/\sim$.

Remark: Since all our substitutions are in normal form and t is a valid ground term, it follows that σ' is also a valid substitution in the above definition.

We now define the secrecy and indistinguishability decision problems. For a protocol P with n roles, the size of P is defined to be the sum of sizes of each term in the initial frame, the number of states in each role and the sizes of each transition tr of P.

Defintion 3.4. Given a protocol P, a name $\mathsf{sec} \in \mathsf{Prv}$ and a state $z = ((s_1, s_2, \dots s_n), \sigma, \varphi, \ell)$ of $\mathcal{M}_P$, consider the state-based secrecy property $\mathsf{Secret}(\mathsf{sec})$ defined as follows: $z \models \mathsf{Secret}(\mathsf{sec})$ iff there is no recipe r such that $\varphi \vdash^r \mathsf{sec}$.
Given a rational threshold p, a protocol P, a name $\mathsf{sec} \in \mathsf{Prv}$, P keeps sec secret with probability at least p (written $P \models_p \mathsf{Secret}(\mathsf{sec})$) if $\mathcal{M}_P \models_p \mathsf{Secret}(\mathsf{sec})$.
The *secrecy* problem is, given a protocol P, a secret $\mathsf{sec} \in \mathsf{Prv}$ and a probability value p, determine if $\mathcal{M}_P \models_p \mathsf{Secret}(\mathsf{sec})$.

Defintion 3.5. Given protocols P and P' with n roles each, P and P' are said to be indistinguishable if the POMDPs $\mathcal{M}_P$ and $\mathcal{M}_{P'}$ are indistinguishable.
The *indistinguishability* problem is, given two protocol P and P', determine if P and P' are indistinguishable.

Example 3.6. We continue Example 2.6 on Page 5. Privacy of votes can be modeled as an indistinguishability property [22] as follows. The protocol satisfies privacy of votes if the adversary cannot distinguish the scenario in which A votes for $\mathbf{0}$ and B votes for $\mathbf{1}$ from the scenario in which A votes for $\mathbf{1}$ and B votes for $\mathbf{0}$, i.e., $P(\mathbf{0}, \mathbf{1})$ is indistinguishable from $P(\mathbf{1}, \mathbf{0})$. This property is true for the electronic voting protocol in Example 2.6. Privacy can also be modeled as secrecy property (see Appendix A). We highlight two things.
First, observe that if the result of coin tosses was made public then $P(\mathbf{0}, \mathbf{1})$ will be distinguishable from $P(\mathbf{1}, \mathbf{0})$. Essentially, if the coin toss had resulted in tails then the attacker will distinguish the two protocols by test whether the first vote output is $\mathbf{0}$. This test will succeed in the protocol $P(\mathbf{0}, \mathbf{1})$ and fail in the protocol $P(\mathbf{1}, \mathbf{0})$, revealing A's vote.
Second, attaching secret names to votes protects vote privacy even if the eligibility tokens are made public after the protocol finishes. For example, if the adversary learns $\mathsf{tk_A}$ and votes did not have nonces then the adversary will be able to check whether $\mathsf{aenc}(\mathsf{pk_A}, [\mathsf{tk_A}, \mathbf{0}])$ matches the vote sent by A to reveal the vote of A.

3.4. Simple adversaries

In order to establish the decidability results, we show that if a protocol is not secure then there is an attack of bounded size. The bounded attack that we construct is going to be a *simple* attack, which is an adaptation of the notion of simple attacks in non-randomized protocols [18] to our formalism. In order to define a simple attack, we first define simple executions. An execution is simple if for each term t, the adversary always uses the same recipe whenever it has to construct t.

Defintion 3.7. Given a protocol P and an *execution* $\rho = (g_0, \sigma_0, \varphi_0, \ell_0) \xrightarrow{(i_0, r_0)} (g_1, \sigma_1, \varphi_1, \ell_1) \cdots \xrightarrow{(i_{m-1}, r_{m-1})} (g_m, \sigma_m, \varphi_m, \ell_m)$ of the POMDP $\mathcal{M}_P$, we say that ρ is *simple* if for any two recipes r and r' such that r is a subterm of r_i for some $0 \leq i \leq m-1$ and r' is a subterm of r_j for some $0 \leq j \leq m-1$, $r\varphi_i =_{\mathcal{R}} r'\varphi_j \Rightarrow r = r'$.

An adversary $\mathcal{A}$ is simple if every reachable state of the DTMC $\mathcal{M}_P^{\mathcal{A}}$ is a simple execution.

Defintion 3.8. An adversary $\mathcal{A}$ for a protocol P is said to be a *simple* adversary if for each execution $\rho_1\rho_2\ldots\rho_m$ of the DTMC $\mathcal{M}_P^{\mathcal{A}}$, ρ_i is simple for each $0 \leq i \leq m$.

We have the following:

Proposition 3.9. For a protocol P, a name $\mathsf{sec} \in \mathsf{Prv}$ and a number p, if there is an adversary $\mathcal{A}$ such that $\mathcal{M}_P^{\mathcal{A}} \not\models_p \mathsf{Secret}(\mathsf{sec})$ then there is simple adversary $\mathcal{B}$ such that $\mathcal{M}_P^{\mathcal{B}} \not\models_p \mathsf{Secret}(\mathsf{sec})$.
Furthermore, if P and P' are two protocols such that P and P' are distinguishable then there is an adversary $\mathcal{A}$, simple for both P and P', such that the POMDPs $\mathcal{M}_P$ and $\mathcal{M}_{P'}$ are distinguishable by $\mathcal{A}$.

4. Decidability proof

We now establish that the secrecy and indistinguishability problems are decidable in **coNEXPTIME**. We start by showing that the *secrecy* problem is decidable in **coNEXPTIME** by proving that the *non-secrecy* problem is in **NEXPTIME**; the *non-secrecy* problem is the one where one needs to determine that P *does not* keep the secret sec with probability at least p. This is done showing the *boundedness* property: if P does not keep the secret with probability at least p then there is a bound b, exponential in the size of the protocol, and a simple adversary $\mathcal{A}$ such that $\mathcal{M}_P^{\mathcal{A}} \not\models_p \mathsf{sec}$ and $\mathcal{A}$ only uses recipes whose dag-size is bounded by b.

Proof strategy. We recall the strategy used in [33] to prove that the *non-secrecy* problem in non-randomized protocols is decidable in **NP**. The proof proceeds by bounding the size of the smallest attack, if one exists. In case of a non-randomized protocol P, the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ for an adversary $\mathcal{A}$ has a single branch. Thus, there is at most one value assigned to each variable x in $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$. Let σ denote the substitution that maps each variable to the value assigned to it. We say that t is a subterm of P if it occurs either in the initial frame or the lefthand side or the right hand side of a transition or if it is a public name. The proof in [33] relies on the key observation that in the smallest attack, for each variable x there is a $t \in CTerms \setminus \mathcal{X}$ which is a subterm of P such that $\sigma(x) = t\sigma$. As the number of subterms of a protocol P is polynomial, this implies that the messages sent by the adversary and the recipes used by the adversary to construct those messages have a polynomial dag-size. Thus, the whole attack can be represented by the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ which is polynomial in the size of the protocol. The **NP** procedure then guesses this attack.

A natural extension of this strategy for randomized protocols would be to show that the value of a variable x at some node of $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ is the same as the value of a non-variable subterm t of P at some other node (note that values of variables and terms can be different in different branches). We show that this observation fails for randomized protocols due to the fact that the adversary has to use the same recipe in executions that have the same view.

Example 4.1. Consider a protocol P which has one role R. The role R itself has 4 states s_0, s_1, s_2, s_3 and the following 3 transitions.

$$s_0 : m_0 \Rightarrow [s_1 : \mathsf{senc}(\mathsf{key}, m_1)] \oplus [s_2 : \mathsf{senc}(\mathsf{key}, m_2)]$$
$$s_1 : [\mathsf{senc}(\mathsf{key}, x), \mathsf{senc}(\mathsf{key}, m_1)] \Rightarrow [s_3 : \mathrm{sec}]$$
$$s_2 : z \Rightarrow [s_3 : \mathrm{sec}].$$

Here m_0, m_1 and m_2 are pairwise distinct public names and $\mathsf{key}, \mathrm{sec}$ are private names. There is a (unique) attack on P that reveals the secret sec with probability 1. The attack initially sends the public name m_0. As a result, the role R may land in two different states each with probability $\frac{1}{2}$. Note that these two states are indistinguishable. Hence the adversary has to use the same recipe in both branches to send the next message. The adversary uses the recipe $[w_1, w_1]$ to send the next message. The secret sec is now revealed with probability 1, x gets the value m_1 in one branch and z gets the value $[\mathsf{senc}(\mathsf{key}, m_2), \mathsf{senc}(\mathsf{key}, m_2)]$ in the other branch. Observe that the value of the variable z is not the value of a non-variable subterm of P.

In our strategy for randomized protocols detailed below, we show that there is a simple attack such that if the adversary uses a recipe r to send a message in an execution ρ then for any $r_1 \sqsubset r$ such that r_1 is not a public name or a variable, there must be an execution ρ' in which the term deduced by r_1 matches the value of a term $t \in CTerms \setminus \mathcal{X}$ of the protocol in some branch of the attack tree. This is achieved by showing that if there is a recipe r_1 which does not satisfy this property then we can replace r_1 with a fresh public name new and get a new attack. A combinatorial argument is then used to establish that this implies that the dag-size of r must be exponentially bounded.

Boundedness Result. We first define the notion of a recipe size of an adversary $\mathcal{A}$.

Defintion 4.2. The *recipe-size* of a recipe r, denoted $rsize(r)$, is the dag-size of r. Given a protocol P and adversary $\mathcal{A}$ for P, we say that a recipe r is *used* by $\mathcal{A}$ if there is an execution ρ of the DTMC $\mathcal{M}_P^{\mathcal{A}}$ such that $last\_action(last(\rho)) = (i, r)$ for some role name i. The *recipe-size* of $\mathcal{A}$, $rsize(\mathcal{A})$ is defined to be the quantity $\max\{rsize(r) \mid r \text{ is used by } \mathcal{A}\}$.

Theorem 4.3. Let $P = (\varphi_0, R_1, R_2, \ldots R_n)$ and let $\mathsf{sec} \in \mathsf{Prv}$. Let λ be the total number of transitions in the protocol. Let γ be the total number of transitions in the protocol whose span is > 1. Let $\eta = \max\{span(R_i) | 1 \leq i \leq n\}$. Let θ be the cardinality of the set $subterms(P) = Sub(ran(\varphi)) \cup (\cup_{1\leq i\leq n} Sub(terms(R_i)))$. If $P \not\models_p \mathsf{Secret}(\mathsf{sec})$ then there is a simple adversary $\mathcal{A}$ such that $\mathcal{M}_P^{\mathcal{A}} \not\models_p \mathsf{Secret}(\mathsf{sec})$ and such that $rsize(\mathcal{A}) \leq 3\theta\lambda\eta^{2^\gamma}$.

Proof: Recall that the POMDP $\mathcal{M}_P$ denotes the semantics of the protocol P. For a given adversary $\mathcal{A}$, the behavior of the protocol P with respect to the adversary is given by the DTMC $\mathcal{M}_P^{\mathcal{A}}$. The actions of the adversary $\mathcal{A}$ are pairs of the form (i, r) where $1 \leq i \leq n$ is a role

name and r is a recipe. The nodes/states of the DTMC are executions of $\mathcal{M}_P$ and its transitions are one-step extensions obtained by using the action specified by the $\mathcal{A}$. Every transition advances the state of a role. So a transition cannot be triggered twice in the same execution. As a consequence, the DTMC is a finite tree whose height is bounded by, λ, the total number of transitions in the protocol. Any path in the tree has at most γ nodes who have more than one child. Furthermore, each such node has at most η children where η be the maximum of spans of each role R_i. Thus, the total number of branches in the tree is bounded by η^γ and hence the size of the tree is bounded by $\lambda\eta^\gamma$.

Assume now that P does not keep sec secret with probability at least p. Let $\mathcal{M}_P = (Z, z_s, Act, \Delta, \mathcal{O}, Obs)$ where its components $Z, z_s, Act, \Delta, \mathcal{O}, Obs$ are as given in Section 2. This means that there is a simple adversary $\mathcal{A}$ such that $\mathcal{M}_P^{\mathcal{A}} \not\models_p \mathsf{sec}$. Now consider the DTMC $\mathcal{M}_P^{\mathcal{A}}$. Recall that each state of $\mathcal{M}_P^{\mathcal{A}}$ is a simple execution ρ of $\mathcal{M}_P$ which is a finite sequence $z_0 \xrightarrow{\alpha_0} z_1 \cdots \xrightarrow{\alpha_{m-1}} z_m$ such that $z_0 = z_s$, and for each $j \geq 0$, $z_j \xrightarrow{\alpha_j} \mu_j$ and $\mu_j(z_{j+1}) > 0$; here, $\alpha_j = \mathcal{A}(view(\rho_j))$ where $\rho_j = z_0 \xrightarrow{\alpha_1} z_1 \cdots \xrightarrow{\alpha_{j-1}} z_j$ and $\mu_j = \Delta(z_j, \alpha_j)$. We view $\mathcal{M}_P^{\mathcal{A}}$ as a finite tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ where each node is uniquely labeled by a simple execution of $\mathcal{M}_P$. For a node n labeled by ρ, as given above, its children are all labeled one step extensions $z_0 \xrightarrow{\alpha_0} z_1 \cdots \xrightarrow{\alpha_m} z_{m+1}$ of ρ such that $\alpha_m = \mathcal{A}(view(\rho))$. Observe that $\rho_m = \rho$. Furthermore, we will say $prev(\rho)$ is the execution ρ_{m-1}. Observe that the execution $prev(\rho)$ is the label of the parent of n. Also, recall that, for ρ as given above, $last(\rho)$ is defined as the state z_m.

For the execution ρ as given above, let $\alpha_i = (k_i, r_i)$ for each $0 \leq i \leq m-1$. We say that $Recipes(\rho) = \{r_i \mid 0 \leq i \leq m-1\}$. We will say that a recipe r *originates* at ρ if $r \sqsubseteq r_{m-1}$ and $r \not\sqsubseteq r_i$ for any $0 \leq i < m-1$. Observe that if r originates at ρ then it does not originate at a state that labels a proper ancestor or proper descendant of n. Furthermore, if r originates at ρ then it also originates at any execution ρ' such that $view(prev(\rho')) = view(prev(\rho))$ and ρ' labels some node in the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$.

Consider the execution ρ labeling node n as above and let r be a recipe originating at ρ. Given a term $t \in subterms(P) \setminus \mathcal{X}$ and executions ρ^1, ρ^2 such that ρ^1 and ρ^2 label some nodes in $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$, $view(prev(\rho)) = view(\rho^1)$, $enabled(\rho^1, \alpha_{m-1})$ and $last(\rho^j) = (g^j, \sigma^j, \varphi^j, (t^j_{in}, t^j_{out}))$ for $j = 1, 2$, we say that the recipe r is a (ρ^1, t, ρ^2)-*match* if r is not a public name or recipe variable, ie, $r \notin \mathsf{Pub} \cup \mathcal{X}_w$ and one of the following hold:

1) ρ^2 is the initial state of $\mathcal{M}_P^{\mathcal{A}}$, t is a subterm of $ran(\varphi_0)$ and $\mathsf{nf}(r\varphi^1) = \mathsf{nf}(t\sigma^2)$. In this case, $\mathsf{nf}(t\sigma^2) = t$ as t is a subterm of initial frame.
2) ρ^2 is not the initial state of $\mathcal{M}_P^{\mathcal{A}}$, t is a subterm of either t^2_{in} or t^2_{out} and $\mathsf{nf}(r\varphi^1) = \mathsf{nf}(t\sigma^2)$.

We say that the execution ρ is an $\mathcal{A}$-match if for every recipe r originating at ρ, either $r \in \mathsf{Pub} \cup \mathcal{X}_w$ or there is a term t and executions ρ^1, ρ^2 such that r is a (ρ^1, t, ρ^2)-match. We say that a simple adversary $\mathcal{A}$ is *small* if every execution ρ labeling a node in $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$, is an $\mathcal{A}$-match. The following is proved in Appendix D.1.

Lemma 4.4. If a simple adversary $\mathcal{A}$ is small then $rsize(\mathcal{A}) \leq 3\theta\lambda\eta^{2\gamma}$.

Essentially, the proof of the Lemma constructs a *one-to-one* function that maps a recipe r used in the protocol to a triple (ρ^1, t, ρ^2) such that r is (ρ^1, t, ρ^2)-match. A straightforward counting argument is then used to count the number of elements in the co-domain of this function. Thus, the Theorem will follow if we can establish that there is a small adversary.

Lemma 4.5. For a protocol P, a name $\mathsf{sec} \in \mathsf{Prv}$ and a number p, if there is a simple adversary $\mathcal{A}$ such that $\mathcal{M}_P^{\mathcal{A}} \not\models_p \mathsf{Secret}(\mathsf{sec})$ then there is a small adversary $\mathcal{B}$ such that $\mathcal{M}_P^{\mathcal{B}} \not\models_p \mathsf{Secret}(\mathsf{sec})$.

Proof: For recipes r, r_1, r_2, let $r[r_1 \mapsto r_2]$ be the recipe obtained by replacing every occurrence of the recipe r_1 as a subterm in r by r_2. For a substitution σ and terms $t, t_2 \in CTerms$, let $\sigma[t \mapsto t_2]$ be the substitution obtained by replacing every occurrence of t as subterm in $ran(\sigma)$ by t_2.

Let $\mathcal{A}$ be a simple adversary which is not small. Consider the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$. For a node n of this tree labeled ρ, we define $nsize$ of n to be 0 if n is the root and the size of the recipe r such that $last\_action(\rho) = (k, r)$ for some k otherwise. The $nsize$ of the tree will be defined as the sum of sizes of all nodes.

Since $\mathcal{A}$ is not small, there must be a non-atomic recipe r_0, an execution ρ_0 labeling a node n_0 in the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ such that r_0 originates at ρ_0 and is not a (ρ, t, ρ')-match for any ρ, ρ', t. Fix $\mathsf{r}_0, \mathsf{n}_0$ and ρ_0. Let ρ^1 be $prev(\rho_0)$ and let $\mathcal{S}$ be the set $\{\rho' \mid view(\rho') = view(\rho^1)$ and ρ' labels a node of $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}\}$. Let $\mathcal{S} = \{\rho^1, \rho^2, \ldots, \rho^N\}$. Let $Nodes(\mathcal{S})$ be the sets of nodes labeled by an execution in $\mathcal{S}$. For $1 \leq j \leq N$, let $last(\rho^j) = (g^j, \sigma^j, \varphi^j, \ell^j)$ and $t^j = \mathsf{nf}(\mathsf{r}_0\varphi^j)$. Let $\mathsf{new} \in \mathsf{Pub} \setminus \mathsf{PublicKeys}$ be a fresh public name that does not occur anywhere in the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$.

Now, we will construct a new tree $\mathcal{T}^{\mathsf{new}}$ by relabeling the nodes of the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ as follows. Consider a node n of the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ and let this be labeled by the execution ρ. The new label of the n in $\mathcal{T}^{\mathsf{new}}$ will be denoted as ρ^{new} and is constructed as follows.

- If n is not a descendant of a node in the set $Nodes(\mathcal{S})$ then $\rho^{\mathsf{new}} = \rho$.
- Otherwise let ρ be the execution $(z_0 \xrightarrow{\alpha_0} z_1 \cdots \xrightarrow{\alpha_{m-1}} z_m)$ and let ρ extend an execution $\rho^j \in \mathcal{S}$. For each $0 < i \leq m$, let $z_i = (g_i, \sigma_i, \varphi_i, \ell_i)$ and let $\alpha_{i-1} = (k_{i-1}, r_{i-1})$. ρ^{new} is the execution $(z_0^{\mathsf{new}} \xrightarrow{\alpha_0^{\mathsf{new}}} z_1^{\mathsf{new}} \cdots \xrightarrow{\alpha_{m-1}^{\mathsf{new}}} z_m^{\mathsf{new}})$ where
  - $\alpha_i^{\mathsf{new}} = \alpha_i$ for each $i < length(\rho^j)$ and is $(k_i, r_i[\mathsf{r}_0 \mapsto \mathsf{new}])$ otherwise.
  - $z_i^{\mathsf{new}} = z_i$ for each $i \leq length(\rho^j)$ and is $(g_i, \sigma_i[t^j \mapsto \mathsf{new}], \varphi_i[t^j \mapsto \mathsf{new}], \ell_i)$ otherwise.

We have the following claim which is proved in the Appendix D.2.

Claim 1. For each execution ρ labeling a node of $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$, ρ^{new} is a simple execution of the POMDP $\mathcal{M}_P$. The measure of the execution ρ^{new} is the measure of the execution ρ. Furthermore, if ρ is either an execution in $\mathcal{S}$ or a descendant of an execution in $\rho^j \in \mathcal{S}$ then for each recipe r, if $enabled(\rho^{\mathsf{new}}, (k, r[\mathsf{r}_0 \mapsto \mathsf{new}]))$ then $enabled(\rho, (k, r))$ also. Furthermore, $\rho \not\models \mathsf{Secret}(\mathsf{sec})$ iff $\rho^{\mathsf{new}} \not\models \mathsf{Secret}(\mathsf{sec})$.

We also have the following claim which says that two distinguishable executions continue to remain distinguishable in the new tree. The claim is proved in the Appendix D.

Claim 2. If ρ_1 and ρ_2 are two executions labeling nodes of $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ such that $view(\rho_1) \neq view(\rho_2)$ then $view(\rho_1^{\mathsf{new}}) \neq view(\rho_2^{\mathsf{new}})$.

Consider the partial function $\mathcal{A}^{\mathsf{new}}$ that maps an execution $prev(\rho^{\mathsf{new}})$ in $\mathcal{T}^{\mathsf{new}}$ to the action $last\_action(\rho^{\mathsf{new}})$. The above two claims imply that $\mathcal{A}^{\mathsf{new}}$ can be extended to the set of all executions such that $\mathcal{A}^{\mathsf{new}}$ is a simple adversary, the attack tree of $\mathcal{A}^{\mathsf{new}}$ is $\mathcal{T}^{\mathsf{new}}$ and $\mathcal{M}_P^{\mathcal{A}^{\mathsf{new}}} \not\models_p \mathsf{Secret}(\mathsf{sec})$. Observe that the $nsize$ of the tree $\mathcal{T}^{\mathsf{new}}$ is strictly less than the $nsize$ of the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$. Now, if $\mathcal{A}^{\mathsf{new}}$ is small then we are done. Otherwise, we can construct another simple adversary $\mathcal{A}'$ such that the size of the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}'}}$ is strictly less than the size of the tree $\mathcal{T}^{\mathsf{new}}$. Since the size of the tree is $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ is finite, repeating this process eventually leads to a small adversary. □

This completes the proof of the Theorem. □

Using similar ideas, we can show that we need to only consider bounded size recipes for indistinguishability (see Appendix E for the proof).

Theorem 4.6. Let P and P' be two protocols with n roles. Let λ be the total number of transitions in P and P' and γ be the total number of transitions in P and P' whose span is > 1. Let η be the maximum span of the transitions of P and P' and θ be the total number of subterms of P and P'. If P and P' are distinguishable then there is a simple adversary $\mathcal{A}$ such that $\mathcal{M}_P$ and $\mathcal{M}_{P'}$ are distinguishable by $\mathcal{A}$ and such that $rsize(\mathcal{A}) \leq 12\theta\lambda\eta^{2\gamma}$.

Complexity results. Thanks to the boundedness property and the fact that static equivalence and deducibility can be checked in polynomial time [1], [15], we get that the secrecy and indistinguishability problems are in **coNEXPTIME** for randomized protocols. Essentially, the **coNEXPTIME** decision procedure guesses a simple attack of bounded size, and checks that this attack violates secrecy/indistiguishability. The proof of the following theorem can be found in Appendix F.

Theorem 4.7. $Secrecy$ and $Indistinguishability$ problems are in **coNEXPTIME**.

We say that a protocol P has *bounded randomness* k if there are at most k transitions of P whose span is greater than one. For each constant k, we define $Secrecy(k)$ to be the decision problem that given a protocol P with bounded randomness k, secret sec and rational number p checks if $P \models_p \mathsf{Secret}(\mathsf{sec})$. Similarly, $Indistinguishability(k)$ problem is the decision problem that given two protocols P and P' with bounded randomness k each checks if P and P' are indistinguishable. Observe that for each constant k, the number of branches in an attack tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ for a protocol P with bounded randomness k is bounded by a polynomial η^k and hence the number of nodes of $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ is polynomial in the size of P. Theorems 4.3 and 4.3 then imply that the adversary only need to use recipes of polynomial size. This yields the following:

Corollary 4.8. For each constant k, $Secrecy(k)$ and $Indistinguishability(k)$ are in **coNP**.

5. Lower Bound for Secrecy

One of the key ideas used in establishing the complexity of non-probabilistic protocols, is the observation that if there is an attack that breaches security, then there is one where the adversary is only required to send messages whose size is bounded by a polynomial in the size of the protocol. However, this observation no longer holds for randomized protocols. We illustrate this through the next example.

Example 5.1. Consider a protocol P with one role $R = (S, s_0, \prec, \delta)$, with the states $S = \{s_0, s_1, s_2, s_\top\}$ and ordering $s_0 \prec s_1 \prec s_2 \prec s_\top$. We assume that the initial frame φ_0 is $\emptyset$. We assume that $k_1, k_2 \in \mathsf{Prv}$ are secret keys unknown to the adversary, and $\mathsf{sec} \in \mathsf{Prv}$ is the secret message that the adversary is trying to discover. We assume that $\mathbf{0}, \mathbf{1} \in \mathsf{Pub}$ are public names. The transitions in δ are as follows.

$$\begin{aligned}
s_0 : [y_0, y_1] \;\Rightarrow\; & [s_1 : \mathsf{senc}(k_1, [y_0, \mathbf{0}])] \oplus \\
& [s_1 : \mathsf{senc}(k_1, [y_1, \mathbf{1}])] \\
s_1 : \mathsf{senc}(k_1, [[x_0^1, x_1^1], g]) \;\Rightarrow\; & \\
& [s_2 : \mathsf{senc}(k_2, [x_0^1, [g, \mathbf{0}]])] \oplus \\
& [s_2 : \mathsf{senc}(k_2, [x_1^1, [g, \mathbf{1}]])] \\
s_2 : \mathsf{senc}(k_2, [z, z]) \;\Rightarrow\; & [s_\top : \mathsf{sec}]
\end{aligned}$$

In order for the adversary to get the secret message sec with probability $> \frac{3}{4}$ (which will imply that $P \not\models_{\frac{1}{4}} \mathsf{Secret}(\mathsf{sec})$), in the first step, the adversary has to send the message

$$[[[\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{1}]], [[\mathbf{1}, \mathbf{0}], [\mathbf{1}, \mathbf{1}]]].$$

In this case the adversary succeeds with probability 1. Notice that this message is nothing but a full binary tree of height 2, where each leaf contains a different pair of the public names $\mathbf{0}, \mathbf{1}$. Thus, the size of this message (even in a dag representation) is the size of the binary tree.

The above example can be generalized to force the adversary to send a full binary tree of height linear in the size of the protocol. Once again the protocol has only one role $R = (S, s_0, \prec, \delta)$, where $S = \{s_0, s_1, \ldots s_n, s_\top\}$ with the ordering $s_i \prec s_j$ if $i < j$

and $s_i \prec s_\top$ for all $i \in \{0, 1, \ldots n\}$. We assume $k_1, \ldots k_n \in \mathsf{Prv}$ are secret keys, and transitions are given as follows.

$$s_0 : [y_0, y_1] \Rightarrow [s_1 : \mathsf{senc}(k_1, [y_0, \mathbf{0}])] \oplus [s_1 : \mathsf{senc}(k_1, [y_1, \mathbf{1}])]$$
$$s_i : \mathsf{senc}(k_i, [[x_0^i, x_1^i], g]) \Rightarrow [s_{i+1} : \mathsf{senc}(k_{i+1}, [x_0^i, [g, \mathbf{0}]])] \oplus [s_{i+1} : \mathsf{senc}(k_{i+1}, [x_1^i, [g, \mathbf{1}]])] \text{ for } 1 \leq i < n$$
$$s_n : \mathsf{senc}(k_n, [z, z]) \Rightarrow [s_\top : \mathsf{sec}]$$

The secret sec is revealed with probability 1 if and only if the adversary sends, as the first message, a full binary tree of height n, where each leaf has different strings (over $\mathbf{0}, \mathbf{1}$). The size of this message as a dag is 2^n, which is exponential in the size of the protocol.

Before concluding this example, we would like to observe that even though the protocol (as described above) keeps its coin tosses private, this is not essential to ensure the need for exponential sized messages. The same initial message would need to be sent even if we considered a modified protocol where in each step the protocol reveals the result of its coin toss in plaintext to the adversary.

We will show that the problem of checking if the security of a randomized protocol can be breached with probability greater than some threshold p, is **NEXPTIME**-hard. The hardness proof is shown by reducing the satisfiability problem of first-order logic with monadic predicates [30]. Before presenting the reduction, we first introduce the logic.

Defintion 5.2. A *monadic* signature τ is a set of relation symbols of arity one. First order formulas over a monadic signature τ is given by the grammar

$$\varphi ::= Rx \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x.\varphi \mid \forall x.\varphi$$

where R is a relation symbol in τ and x is a first-order variable.

Notice that $=$ is not allowed in this logic and there are no constant symbols. The satisfiability problem for this logic is known to be **NEXPTIME**-complete [30] [3]. We will reduce the satisfiability problem of this logic to the non-secrecy problem as follows.

Theorem 5.3. The *secrecy* problem is **coNEXPTIME**-hard.

Proof: It suffices to show that given a protocol P, secret $\mathsf{sec} \in \mathsf{Prv}$, $p > 0$, the problem of checking if $P \not\models_p \mathsf{Secret}(\mathsf{sec})$ is **NEXPTIME**-hard.

There are 3 key ideas that play a role in establishing this result. The first is an observation due to Rusinowitch and Turuani [33] that shows how the satisfiability of propositional logic can be reduced to protocol insecurity. Given a 3CNF formula φ, Rusinowitch-Turuani construct a protocol that proceeds in two phases. First, the adversary guesses a satisfying truth assignment, and sends this to the protocol. The protocol ensures the commitment of the adversary to this assignment for the rest of the execution by encrypting the assignment using a key that is secret. After this phase, the adversary demonstrates the satisfaction of each clause by revealing a relevant portion of the assignment that (s)he committed to in the first step. If all clauses are satisfied, the secret is revealed.

The second observation is that randomization can be used to simulate quantifier alternation, with the adversary making existential choices, and the protocol making universal choices by probabilistic steps. Before presenting the **NEXPTIME**-hardness result, we show a **PSPACE**-hardness result that shows how these two observations can be combined to reduce QSAT to protocol insecurity of randomized protocols. Consider a QBF $\varphi = Q_1 x_1.\, Q_2 x_2 \ldots Q_n x_n.\, \psi$, where $Q_i \in \{\exists, \forall\}$, x_i is a propositional variable and $\psi = \wedge_{j=1}^{m} C_j$ is a 3CNF formula with clauses C_j. The satisfiability of φ can be reduced to the insecurity of a protocol P with only one role R. We assume that $\top, \bot$ are public names in Pub standing for *true* and *false*, respectively. Keys $\{k_1, \ldots k_n\}$ (one for each propositional variable) are secret names in Prv. For each variable x_i, the protocol P has a state g_i and for each clause C_j, P has a state c_j. P has one additional state $s_\top$. P proceeds in phases. First, a truth assignment for the variables is chosen. The value for variable x_i is picked in state g_i; if x_i is existentially quantified then the adversary picks the value for x_i, and if x_i is universally quantified then the protocol picks the value for x_i. The transition rules capturing these steps are as follows (with the understanding that $g_{n+1} = c_1$).

$$g_i : t \Rightarrow [g_{i+1} : \mathsf{senc}(k_i, t)] \text{ if } Q_i = \exists$$
$$g_i : \top \Rightarrow [g_{i+1} : [\mathsf{senc}(k_i, \bot), \bot]] \oplus [g_{i+1} : [senc(k_i, \top), \top]] \text{ if } Q_i = \forall$$

Notice, that when the value of a universally quantified variable is picked, the result of the coin toss is made public so that the adversary's choice for future existential variables may depend on this value. Also, the variables are chosen in the order in which they are quantified in φ by the use of states g_i. After all variables have been assigned truth values, the clauses in ψ are evaluated in sequence, using states $\{c_1, \ldots c_m\}$. In each state c_j, the adversary sends a (encrypted) truth value of a variable that makes one of the literals in C_j true. The rules capturing these steps are as follows (with the understanding that $c_{m+1} = s_T$).

$$c_j : \mathsf{senc}(k_i, \bot) \Rightarrow [c_{j+1} : \top] \quad \text{if } \neg x_i \in C_j$$
$$c_j : \mathsf{senc}(k_i, \top) \Rightarrow [c_{j+1} : \top] \quad \text{if } x_i \in C_j$$

Observe that the only way the protocol can reach state c_m is if all clause $C_1 \ldots C_{m-1}$ are satisfied by the truth assignment picked initially. In the state c_m, if the adversary sends a truth value that satisfies a literal in C_m, the protocol moves to state r ("reveal"). Here the secret message sec is revealed, moving to the terminal state $s_\top$.

$$r : \top \Rightarrow [s_\top : \mathsf{sec}]$$

3. Strictly speaking, [30] shows that the satisfiability problem is **NE**-complete, where $\mathbf{NE} = \cup_c \mathbf{NTIME}(2^{cn})$. From the results of [28], it also follows that satisfiability is NEXP-complete.

TABLE 1. COMPLEXITY OF VERIFYING SECURITY PROTOCOLS. WE ONLY CONSIDER SIMPLE PROTOCOLS. THERE IS NO DIFFERENCE IN COMPLEXITY FOR PUBLIC AND PRIVATE COIN TOSSES WHEN THE NUMBER OF COIN TOSSES ARE FIXED.

	Non-randomized Protocols	Coin tosses are private	Coin tosses are public	Fixed number of coin tosses
Secrecy	**coNP**-complete ([33], [23])	**coNEXPTIME**-complete	**coNEXPTIME**-complete	**coNP**-complete
Indistinguishability	**coNP**-complete ([19])	**coNEXPTIME**	**coNP**-complete	**coNP**-complete

Observe that, under the assumption that the keys $\{k_i\}_i$ are not known to the adversary, φ is satisfiable iff sec is obtained with probability 1. If φ is not satisfiable then the protocol keeps sec secret with probability at least $\frac{1}{2^{n_1}}$ where n_1 is the number of universal quantifications.

The last ingredient needed in order to prove **NEXPTIME**-hardness, is the ability of a randomized protocol to "examine" the contents of an exponential sized message. This is accomplished using ideas in the protocol described in Example 5.1. Details of how we combine all these ideas to reduce the satisfaction problem of monadic first-order formulas to the insecurity problem of randomized protocols is given in Appendix G. One particular feature of the proof is that all coin tosses in the reduction are public. Thus, the **NEXPTIME**-hardness result holds even for checking the insecurity of randomized protocols where all coin tosses are public. □

6. Conclusions

We have shown that the problem of checking secrecy of randomized security protocols is **coNEXPTIME**-complete, and the problem of checking indistinguishability of two randomized security protocols is in **coNEXPTIME**. The hardness results for secrecy hold even if we assume that the results of the coin tosses are public. In contrast, it can be shown that for public coin tosses, the indistinguishability problem is **coNP**-complete. Our results also show that, when the number of coin tosses are bounded, i.e., the number of probabilistic transitions are bounded, both secrecy and indistinguishability are **coNP**-complete. Our results are summarized in Table 1. The model of protocols we considered, allow the cryptographic primitives corresponding to symmetric as well as asymmetric encryption and decryption. It can easily be shown that our results extend to the case when digital signatures and hash functions are also allowed in the protocols. Obtaining tight bounds for the indistinguishability problem needs to be explored as part of future work. The decidability and complexity of the above problems also needs to be investigated for the case when we allow conditional statements in the protocols and for other cryptographic primitives that can be modeled as subterm convergent rewrite systems.

Acknowledgements. We thanks the anonymous reviewers for their useful comments and suggestions. Rohit Chadha was partially supported by NSF CNS 1314338 and NSF CNS 1553548. A. Prasad Sistla was partially supported by NSF CCF 1319754, NSF CCF 1564296 and NSF CNS 1314485. Mahesh Viswanathan was partially supported by NSF CNS 1314485.

References

[1] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM Symp. on Principles of Programming Languages (POPL'01)*, pages 104–115, 2001.

[3] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, Cuéllar J, Paul Hankes Drielsma, P. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *17th International Conference on Computer Aided Verification (CAV'05)*, Lecture Notes in Computer Science, pages 281–285. Springer, 2005.

[4] M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.

[5] M. Bauer, R. Chadha, and M. Viswanathan. Composing protocols with randomized actions. In *Principles of Security and Trust (POST 2016)*, pages 189–210, 2016.

[6] B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *20th Symposium on Logic in Computer Science (LICS'05)*, pages 331–340. IEEE Comp. Soc. Press, 2005.

[7] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, P. Pereira, and R. Segala. Task-Structured Probabilistic I/O Automata. In *Workshop on Discrete Event Systems*, 2006.

[8] R. Chadha, V. Cheval, S. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. *ACM Trans. Comput. Log.*, 17(4):23:1–23:32, 2016.

[9] R. Chadha, A.P. Sistla, and M. Viswanathan. Model checking concurrent programs with nondeterminism and randomization. In *the International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 364–375, 2010.

[10] K. Chatzikokolakis and C. Palamidessi. Making Random Choices Invisible to the Scheduler. *Information and Computation*, 208(6):694–715, 2010.

[11] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.

[12] L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. PhD thesis, Radboud University of Nijmegen, 2006.

[13] V. Cheval, H. Comon-Lundh, and S. Delaune. Automating security analysis: symbolic equivalence of constraint systems. In *Proc. 5th International Joint Conference on Automated Reasoning (IJCAR'10)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 412–426. Springer-Verlag, 2010.

[14] V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, Chicago, Illinois, USA, October 2011. ACM Press. To appear.

[15] Y. Chevalier and M. Rusinowitch. Decidability of equivalence of symbolic derivations. *Journal of Automated Reasoning*, 48(2):263–292, 2012.

[16] R. Chrétien, V. Cortier, and S. Delaune. Decidability of trace equivalence for protocols with nonces. In *Proceedings of the 28th IEEE Computer Security Foundations Symposium (CSF'15)*, pages 170–184. IEEE Computer Society Press, July 2015.

[17] H. Comon-Lundh and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science*, 331(1):143–214, February 2005.

[18] H. Comon-Lundh, V. Cortier, and E. Zălinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Transactions on Computational Logic*, 11(2), January 2010.

[19] V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proc. 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276. IEEE Comp. Soc. Press, 2009.

[20] C. J. F. Cremers. The Scyther tool: Verification, falsification, and analysis of security protocols. In *20th International Conference on Computer Aided Verification*, pages 414–418, 2008.

[21] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 2009. To appear.

[22] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

[23] N. A. Durgin, Patrick Lincoln, J. C. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[24] S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V*, volume 5705 of *Lecture Notes in Computer Science*, pages 1–50. Springer, 2009.

[25] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *24th Symposium on Foundations of Computer Science (FOCS'83)*, pages 34–39. IEEE Comp. Soc. Press, 1983.

[26] F.D. Garcia, P. van Rossum, and A. Sokolova. Probabilistic Anonymity and Admissible Schedulers. *CoRR*, abs/0706.1019, 2007.

[27] D. M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Onion routing. *Commun. ACM*, 42(2):39–41, 1999.

[28] S. Homer. Structural properties of complete problems for exponential time. In Lane Hemaspaandra and Alan Selman, editors, *Complexity Theory Retrospective II*, pages 135–154. Springer-Verlag, 1997.

[29] H. Hüttel. Deciding framed bisimilarity. In *Proc. 4th International Workshop on Verification of Infinite-State Systems (INFINITY'02)*, pages 1–20, 2002.

[30] H.R. Lewis. Complexity results for classes of quantification formulas. *Journal of Computer and System Sciences*, 21:317–353, 2001.

[31] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *25th International Conference on Computer Aided Verification*, pages 696–701, 2013.

[32] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.

[33] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Comp. Soc. Press, 2001.

[34] P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. Prêt à voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, 2009.

[35] A. Tiu and J. Dawson. Automating open bisimulation checking for the spi-calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 307–321. IEEE Comp. Soc. Press, 2010.

Appendix A.
Privacy of votes as secrecy property

We modeled privacy of votes as an indistinguishability property in Example 2.6. The same property can modeled as secrecy. We explain the modeling informally here. Essentially, in the modeling, the two voters A and B first toss a fair coin each to decide which candidate. After EA publishes the votes, the adversary sends a message to A which is its guess of the candidate chosen by A. If the guess is correct then A outputs a secret sec. Now, privacy of votes can be modeled as secrecy of the secret sec. Observe that if both A and B choose the same candidate (which happens with probability $\frac{1}{2}$), the attacker knows how A voted and hence can get sec. If A and B both choose different candidates then thanks to the mixing of the votes by EA, the best the attacker can do is to guess randomly. Thus, the protocol will keep the votes private if sec is kept secret with probability at least $\frac{1}{4}$. The modeling of privacy as secrecy allows us to model and reason about deviations from ideal behavior. If the coin tosses used by the protocol are biased then the protocol will only be able to guarantee probable privacy, namely, that sec is kept secret with probability > 0.

Appendix B.
Proof of Proposition 3.1

Fix $\overline{o} \in \mathcal{V} \cdot (Act \times \{\Diamond\})$ such that $pr_{\mathcal{M}}(\overline{o}, \mathcal{A}) \neq pr_{\mathcal{M}'}(\overline{o}, \mathcal{A})$. Let $\overline{o} = \overline{o}_0 \cdot (\alpha, \Diamond)$. It can be shown that

$$pr_{\mathcal{M}}(\overline{o}_0, \mathcal{A})\mathcal{A}(\overline{o}_0)(\alpha) = pr_{\mathcal{M}}(\overline{o}, \mathcal{A}) + \sum_{o \in \mathcal{O}} pr_{\mathcal{M}}(\overline{o}_0 \cdot \alpha \cdot o, \mathcal{A})$$

and

$$pr_{\mathcal{M}'}(\overline{o}_0, \mathcal{A})\mathcal{A}(\overline{o}_0)(\alpha) = pr_{\mathcal{M}'}(\overline{o}, \mathcal{A}) + \sum_{o \in \mathcal{O}} pr_{\mathcal{M}'}(\overline{o}_0 \cdot \alpha \cdot o, \mathcal{A}).$$

As $pr_{\mathcal{M}}(\overline{o}, \mathcal{A}) \neq pr_{\mathcal{M}'}(\overline{o}, \mathcal{A})$, the above two equations imply that either $pr_{\mathcal{M}}(\overline{o}_0, \mathcal{A}) \neq pr_{\mathcal{M}'}(\overline{o}_0, \mathcal{A})$ or there is an $o \in \mathcal{O}$ such that $pr_{\mathcal{M}}(\overline{o}_0 \cdot \alpha \cdot o, \mathcal{A}) \neq pr_{\mathcal{M}'}(\overline{o}_0 \cdot \alpha \cdot o, \mathcal{A})$.

Appendix C.
Proof of proposition 3.2

Let $\mathcal{M} = (Z, z_s, Act, \Delta, \mathcal{O}, Obs)$ be a POMDP and $\Psi \subseteq Z$ a safety property.. Fix an adversary $\mathcal{A} : \mathcal{V} \to \mathsf{Dist}(Act)$. We will say that $\overline{o} \in \mathcal{V}$ is *$\mathcal{A}$-interesting for $\mathcal{M}$* if $pr_{\mathcal{M}}(\overline{o}, \mathcal{A}) > 0$ and the set $\{\alpha \mid \mathcal{A}(\overline{o})(\alpha > 0)\}$ has at least two elements. Let $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}) = \{\overline{o} \mid \overline{o} \text{ is } \mathcal{A}\text{-interesting for } \mathcal{M}\}$. Note that $\mathsf{Interesting}(\mathcal{M}, \mathcal{A})$ is finite. For an execution $\kappa \in \mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$, we will write $pr_{\mathcal{M}}(\kappa, \mathcal{A})$ to be the measure of the execution κ. If κ is not an execution but a sequence of states of $\mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$ then we will write $pr_{\mathcal{M}}(\kappa, \mathcal{A}) = 0$. For a set $W \subseteq \mathsf{Exec}(\mathcal{M}^{\mathcal{A}})$, we will write $pr_{\mathcal{M}}(W, \mathcal{A})$ for $\sum_{\kappa \in \mathsf{W}} pr_{\mathcal{M}}(\kappa, \mathcal{A})$.

We first consider consider safety properties. It suffices to show that the following claim:

Claim 3. For each adversary $\mathcal{A}$, if $\mathcal{M}^{\mathcal{A}} \not\models_p \Psi$ then there is an $\mathcal{A}'$ such that $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}') = \emptyset$ and $\mathcal{M}^{\mathcal{A}'} \not\models_p \Psi$.

Proof: The proof is by induction on the cardinality of the set $\mathsf{Interesting}(\mathcal{M}, \mathcal{A})$. The claim is trivially true if the cardinality is 0. Now, suppose that the claim is true for all adversaries $\mathcal{A}$ such that $\mathsf{Interesting}(\mathcal{M}, \mathcal{A})$ has at most k elements.

Now, consider an adversary $\mathcal{A}_0$ such that $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}_0)$ has $k+1$ elements and such that $\mathcal{M}^{\mathcal{A}_0} \not\models_p \Psi$. Fix $\overline{o} \in \mathsf{Interesting}(\mathcal{M}, \mathcal{A}_0)$. Fix an enumeration $\alpha_1, \alpha_2, \ldots, \alpha_m$ of actions α such $\mathcal{A}_0(\overline{o})(\alpha) > 0$. For each $1 \leq i \leq m$, let $p_i = \mathcal{A}_0(\overline{o})(\alpha_i)$ and $\mathcal{A}_i : \mathcal{V} \to \mathsf{Dist}(Act)$ to be the adversary such that $\mathcal{A}_i(\overline{o})(\alpha_i) = 1$ and $\mathcal{A}_i(\overline{o}') = \mathcal{A}(\overline{o}')$ for $\overline{o}' \neq \overline{o}$. It is easy to see $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}_i)$ has at most k elements for each $1 \leq i \leq m$.

Now, let W be the set $\{\kappa \mid \kappa$ is a maximal execution of $\mathcal{M}^{\mathcal{A}}$ and $\kappa \not\models \Psi\}$. Now, we have that

$$pr_{\mathcal{M}}(\mathsf{W}, \mathcal{A}) > 1 - p.$$

Let $\mathsf{W}_{\overline{o}}$ be the set $\{\kappa \in \mathsf{W} \mid \overline{o}$ is a prefix of $view(\kappa)\}$.

It is easy to see that

$$pr_{\mathcal{M}}(\mathsf{W}, \mathcal{A}_0) = pr_{\mathcal{M}}(\mathsf{W} \backslash \mathsf{W}_{\overline{o}}, \mathcal{A}_0) + \sum_{1 \leq i \leq m} p_i \cdot pr_{\mathcal{M}}(\mathsf{W}_{\overline{o}}, \mathcal{A}_i))$$

and that

$$pr_{\mathcal{M}}(\mathsf{W} \setminus \mathsf{W}_{\overline{o}}, \mathcal{A}_0) = pr_{\mathcal{M}}(\mathsf{W} \setminus \mathsf{W}_{\overline{o}}, \mathcal{A}_i)$$

for each $1 \leq i \leq m$. Now, since $\sum_{1 \leq i \leq m} p_i = 1$, the above two observations imply that

$$\begin{aligned} pr_{\mathcal{M}}(\mathsf{W}, \mathcal{A}_0) &= \sum_{1 \leq i \leq m} p_i \cdot (pr_{\mathcal{M}}(\mathsf{W} \setminus \mathsf{W}_{\overline{o}}, \mathcal{A}_0) \\ &\qquad + pr_{\mathcal{M}}(\mathsf{W}_{\overline{o}}, \mathcal{A}_i)) \\ &= \sum_{1 \leq i \leq m} p_i \cdot (pr_{\mathcal{M}}(\mathsf{W} \setminus \mathsf{W}_{\overline{o}}, \mathcal{A}_i) \\ &\qquad + pr_{\mathcal{M}}(\mathsf{W}_{\overline{o}}, \mathcal{A}_i)) \\ &= \sum_{1 \leq i \leq m} p_i \cdot pr_{\mathcal{M}}(\mathsf{W}, \mathcal{A}_i). \end{aligned}$$

The above equation implies that there must an i such that $prob_{\mathcal{M}}(\mathsf{W}, \mathcal{A}_i) \geq pr_{\mathcal{M}}(\mathsf{W}, \mathcal{A}_0) > 1 - p$. This implies that there must be an i such that $\mathcal{M}^{\mathcal{A}_i} \not\models_p \Psi$. The claim follows. □

The proof for indistinguishability properties is similar.

Claim 4. If $\mathcal{M}$ and $\mathcal{M}'$ are POMDPs with the same set of actions and observations, then $\mathcal{M}$ and $\mathcal{M}'$ are distinguishable by an adversary $\mathcal{A}$, then an $\mathcal{A}'$ such that $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}') = \mathsf{Interesting}(\mathcal{M}', \mathcal{A}') = \emptyset$ and $\mathcal{M}$ and $\mathcal{M}'$ are distinguishable by an adversary $\mathcal{A}'$.

Proof: The proof is by induction on the cardinality of the set $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}) \cup \mathsf{Interesting}(\mathcal{M}', \mathcal{A})$. The claim is trivially true if the cardinality is 0. Now, suppose that the claim is true for all adversaries $\mathcal{A}$ such that $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}) \cup \mathsf{Interesting}(\mathcal{M}', \mathcal{A})$ has at most k elements.

Now, consider an adversary $\mathcal{A}_0$ such that $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}_0) \cup \mathsf{Interesting}(\mathcal{M}', \mathcal{A})$ has $k+1$ elements and $\mathcal{A}_0$ distinguishes $\mathcal{M}$ and $\mathcal{M}'$. Fix $\overline{o} \in \mathsf{Interesting}(\mathcal{M}, \mathcal{A}_0) \cup \mathsf{Interesting}(\mathcal{M}', \mathcal{A})$. Fix $\overline{o}_1 \in \mathcal{V}$ such that $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_0) \neq pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_0)$. We haver two possible cases.

1) The first case is that $\overline{o}$ is a strict prefix of $\overline{o}_1$. Thus, $\overline{o}_1 = \overline{o}_1 \alpha \overline{o}_0$ for some $\alpha \in Act$ and $\overline{o}_0 \in \mathcal{V}$. As either $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_0) \neq 0$ or $pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_0) \neq 0$, it means that $\mathcal{A}_0(\overline{o})(\alpha) \neq 0$. Consider the adversary $\mathcal{A}_\alpha$ such that $\mathcal{A}_\alpha(\overline{o})(\alpha) = 1$ and $\mathcal{A}_\alpha(\overline{o}_2) = \mathcal{A}_0(\overline{o}_2)$ for each $\overline{o}_2 \neq \overline{o}$. Clearly, $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}_0) \cup \mathsf{Interesting}(\mathcal{M}', \mathcal{A})$ has at most k elements. It is easy to see that $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_0) = \mathcal{A}_0(\overline{o})(\alpha) \cdot pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_\alpha)$ and that $pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_0) = \mathcal{A}_0(\overline{o})(\alpha) \cdot pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_\alpha)$. Since $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_0) \neq pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_0)$, we get that $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_\alpha) \neq pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_\alpha)$. The claim follows in this case.
2) The second case is when $\overline{o}$ is not a strict prefix of $\overline{o}_1$. Fix α such that $\mathcal{A}_0(\overline{o})(\alpha) > 0$. Consider the adversary $\mathcal{A}_\alpha$ such that $\mathcal{A}_\alpha(\overline{o})(\alpha) = 1$ and $\mathcal{A}_\alpha(\overline{o}_2) = \mathcal{A}_0(\overline{o}_2)$ for each $\overline{o}_2 \neq \overline{o}$. Clearly, $\mathsf{Interesting}(\mathcal{M}, \mathcal{A}_0) \cup \mathsf{Interesting}(\mathcal{M}', \mathcal{A})$ has at most k elements. It is easy to see that $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_\alpha) = pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_0)$ and that $pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_\alpha) = pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_0)$. Since $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_0) \neq pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_0)$, we get that $pr_{\mathcal{M}}(\overline{o}_1, \mathcal{A}_\alpha) \neq pr_{\mathcal{M}'}(\overline{o}_1, \mathcal{A}_\alpha)$. The claim follows in this case also.

□

Appendix D. Proof of Theorem 4.3

D.1. Proof of Lemma 4.4

Let n be a node of the tree $\mathcal{T}_{\mathcal{M}^{\mathcal{A}}_P}$. Let $\rho = z_0 \xrightarrow{\alpha_0} z_1 \cdots \xrightarrow{\alpha_{m-1}} z_m$ be the label of n and let $\alpha_i = (k_i, r_i)$ for each $0 \leq i < m$. It suffices to show that $rsize(r_{m-1}) \leq 3\theta\eta^{2\gamma}\lambda$. In order to establish the latter, it suffices to show that the cardinality of the set $SR(r_{m-1}) = \{r^1 \mid r^1 \notin \mathsf{Pub} \cup \mathcal{X}_w, r^1 \sqsubseteq r_{m-1}\}$ is less than or equal to $\theta\eta^{2\gamma}\lambda$. This is because every node in the dag-representation of r_{m-1} has at most 2 outgoing edges. If $SR(r_{m-1}) \leq \theta\eta^{2\gamma}\lambda$ then there are at most $\theta\eta^{2\gamma}\lambda$ nodes which have outgoing edges, and thus, the total number of nodes is bounded by $3\theta\eta^{2\gamma}\lambda$.

Pick $r \in SR(r_{m-1})$. Observe that either r originates in ρ or in an execution labeling some ancestor of n. Let ρ_r be ρ if r originates in ρ else ρ_r is the execution labeling the ancestor of n such that r originates in ρ_r. Since $\mathcal{A}$ is a small attack, there must be executions ρ^1, ρ^2 and a term t such that r is a (ρ^1, t, ρ^2)-match. Observe that $view(prev(\rho_r)) = view(\rho^1)$. Let $Nodes$ be the set of nodes of the tree $\mathcal{T}_{\mathcal{M}^{\mathcal{A}}_P}$. Thus, we can define a function $map : SR(r_{m-1}) \to Nodes \times terms(P) \times Nodes$ such that if $map(r) = (\mathsf{n}_1, t, \mathsf{n}_2)$ then r is a (ρ^1, t, ρ^2)-match where ρ^i

is the execution labeling the node n_i for $i \in \{1,2\}$. As observed above, there are at most $\eta^\gamma\lambda$ nodes in the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$. Thus, the cardinality of the set $Nodes \times terms(P) \times Nodes$ is less than or equal to $\theta\eta^{2\gamma}\lambda$. Thus, the result will follow if we can show that map is a one-to-one function.

We now establish that map is a one-to-one function. We proceed by contradiction. Fix $r, r' \in SR(r_{m-1})$ such that $r \neq r'$ and $map(r) = map(r') = (\mathsf{n}_1, t, \mathsf{n}_2)$. Let $\rho_r, \rho_{r'}$ be the executions in which r and r' originate as given above. For $i \in \{1,2\}$, let ρ^i be the execution labeling the node n_i. Either $\rho_r = \rho_{r'}$ or one of $\rho_r, \rho_{r'}$ labels a node which is an ancestor of a node labeling the other. In particular, it means that either $\rho_r = \rho_{r'}$ or $view(prev(\rho_r)) \neq view(prev(\rho_{r'}))$. In the latter case, we immediately get a contradiction as $map(r) = map(r')$ implies that $view(prev(\rho_r)) = view(\rho^1) = view(prev(\rho_{r'}))$ by definition of map. Hence it must be the case that $\rho_r = \rho_{r'}$. Let $last(\rho^i) = (g^i, \sigma^i, \varphi^i, \ell^i)$ for $i \in \{1,2\}$. By definition of map, we have that $\mathsf{nf}(r\varphi^1) = \mathsf{nf}(t\sigma^2)$ and $\mathsf{nf}(r'\varphi^1) = \mathsf{nf}(t\sigma^2)$. Hence $\mathsf{nf}(r\varphi^1) = \mathsf{nf}(r'\varphi^1)$. Since $\mathcal{A}$ is a simple adversary, we have $r = r'$. This contradicts our assumption that $r \neq r'$.

D.2. Proof of Lemma 4.5

We say that a recipe r is a *constructive* recipe if the root of r is a constructor function symbol. We say that r is a *destructive* recipe if the root of r is a destructor function symbol. We say that the recipe r is (r_1, φ)-simple for a frame φ and a recipe r_1 if $vars(r), vars(r_1) \subseteq dom(\varphi)$, $valid(r\varphi), valid(r_1\varphi)$, and for every recipe $r_2 \sqsubseteq r$ and recipe $r_3 \sqsubseteq r_1$ such that $r_2\varphi =_{\mathcal{R}} r_3\varphi$, we have that $r_2 = r_3$. Observe that if r is (r_1, φ)-simple then so is every subterm of r. We say that the recipe r is φ-simple for a frame φ if r is (r, φ)-simple. When the frame φ is clear from the context, we will abuse terminology and say that "recipe r is simple" to mean "recipe r is φ-simple". Observe that every recipe used by a simple adversary is simple and that every subterm of a simple recipe is simple. We say that a term t is away from a public name $\mathsf{new} \in \mathsf{Pub} \setminus \mathsf{PublicKeys}$ if $\mathsf{new} \notin Sub(t)$. We say that a substitution σ is away from a public name new if $\mathsf{new} \notin Sub(ran(\sigma))$.

We first show that if a recipe is simple and destructive, then it computes a subterm of the frame.

Proposition D.1. Given a destructive recipe r, a frame φ such that r is φ-simple, $\mathsf{nf}(r\varphi) \in Sub(ran(\varphi))$.

Proof: Proof is by induction on the height of r. The proposition is trivially true for a recipe of height 0 as r cannot be destructive. Assume now that the proposition is true for every φ-recipe r' such that the height of r' is $\leq k$. Now fix a recipe r of height $k+1$. The root of r must be in the set $\{\mathsf{proj}_1, \mathsf{proj}_2, \mathsf{sdec}, \mathsf{adec}\}$. We show that $\mathsf{nf}(r\varphi)$ must be a subterm of $ran(\varphi)$ for the case that the root of r is sdec. The other cases are similar. Assume that $r = \mathsf{sdec}(r_1, r_2)$. As r such that $valid(r\varphi)$, it follows that r_2 is either a variable $w_j \in dom(\varphi)$ or is a destructive recipe or is a encrypted recipe $\mathsf{senc}(r_1', r_2')$. If r_2 is a variable $w_j \in dom(\varphi)$, the proposition follows immediately from definition. If r_2 is a destructive recipe, the proposition follows from inductive hypothesis. If r_2 is $\mathsf{senc}(r_1', r_2')$ then as $valid(r\varphi)$, $r\varphi =_{\mathcal{R}} r_2'\varphi$. This contradicts the fact that r is φ-simple. The proposition follows in this case as well. □

The above proposition immediately implies that if r is a φ-simple recipe and t is a subterm of $\mathsf{nf}(r\varphi)$ then either t is a subterm of $ran(\varphi)$ or t is constructed explicitly by r using a constructive recipe.

Proposition D.2. Given a recipe r, a frame φ such that r is φ-simple, and a term t such that $t \notin Sub(ran(\varphi))$. If $\mathsf{nf}(r\varphi)$ contains t as a subterm then there is a constructive $r' \sqsubseteq r$ such that $\mathsf{nf}(r'\varphi) = t$.

If a recipe r is simple then a constructive recipe is never destructed in r. This intuition is captured in the following proposition.

Proposition D.3. Let r, r' be recipes and φ be a frame such that $valid(r\varphi)$, r' is a constructive recipe and r is (r', φ)-simple. Let new be a public name such that r, r', φ are away from new. Let $t = \mathsf{nf}(r\varphi)$ and $t' = \mathsf{nf}(r'\varphi)$. Then

- $valid(r^{\mathsf{new}}\varphi^{\mathsf{new}})$, and
- $\mathsf{nf}(r^{\mathsf{new}}\varphi^{\mathsf{new}}) = t^{\mathsf{new}}$ where

$r^{\mathsf{new}} = r[r' \mapsto \mathsf{new}], t^{\mathsf{new}} = t[t' \mapsto \mathsf{new}]$ and $\varphi^{\mathsf{new}} = \varphi[t' \mapsto \mathsf{new}]$.

Proof: Proof is by induction on r. The proposition is trivially true if r is a public name. If r is a variable $w_j \in dom(\varphi)$ then w_j is not the recipe r' as the latter is a constructive recipe. Hence the proposition is trivially true in this case also.

Now, suppose r is constructive. Thus, the root of r is a function symbol in the set $\{[\cdot,\cdot], \mathsf{aenc}, \mathsf{senc}\}$. We consider the case when the root of r is the function symbol $[\cdot,\cdot]$. The other cases are similar. Thus, $r = [r_1, r_2]$ for some r_1, r_2. Now, either $r = r'$ or $r \neq r'$. In the former case, the proposition follows trivially since $r^{\mathsf{new}} = \mathsf{new}$ and $t^{\mathsf{new}} = \mathsf{new}$. In the latter case, the proposition is immediate from induction hypothesis.

Now, suppose r is a destructive. Thus, the root of r is a function symbol in the set $\{\mathsf{proj}_1, \mathsf{proj}_2, \mathsf{sdec}, \mathsf{aenc}\}$. We consider the case when the root of r is the function symbol sdec. The other cases are similar. Thus, $r = \mathsf{sdec}(r_1, r_2)$ for some r_1, r_2. Now, r cannot be the recipe r' as r' is a constructive recipe. Thus $r^{\mathsf{new}} = \mathsf{sdec}(r_1^{\mathsf{new}}, r_2^{\mathsf{new}})$. By induction hypothesis we have that $valid(r_1^{\mathsf{new}}\varphi^{\mathsf{new}})$ and $valid(r_2^{\mathsf{new}}\varphi^{\mathsf{new}})$. Thus, we will have that $valid(r^{\mathsf{new}}\varphi^{\mathsf{new}})$ if we can show that the term $\mathsf{sdec}(r_1^{\mathsf{new}}, r_2^{\mathsf{new}})\varphi^{\mathsf{new}} \in CTerms$. Observe also that since $valid(r\varphi)$, r_2 must be a variable or a destructive recipe or an encryption recipe $\mathsf{senc}(r_3, r_4)$ for some r_3, r_4.

If r_2 is a variable w_j or a destructive recipe, then $\mathsf{nf}(r_2\varphi) = \mathsf{senc}(t_1, t_2)$, $\mathsf{nf}(r_1\varphi) = t_1$ and $\mathsf{nf}(r\varphi) = t_2$ for some $t_1, t_2 \in CTerms$. Furthermore, as r is (r', φ)-simple and r' is constructive, it follows that $r_2 \neq r'$ and that $\mathsf{nf}(r_2\varphi) \neq t'$. Thus, by inductive hypothesis,

$\mathsf{nf}(r_2^{\mathsf{new}}\varphi^{\mathsf{new}}) = \mathsf{senc}(t_1^{\mathsf{new}}, t_2^{\mathsf{new}})$ where $t_i^{\mathsf{new}} = t_i[t' \mapsto \mathsf{new}]$ for each $i = 1, 2$. By induction hypothesis, we have that $\mathsf{nf}(r_1^{\mathsf{new}}\varphi^{\mathsf{new}}) = t_1^{\mathsf{new}}$. Since t_1 is a constructor term, so is t_1^{new}. Thus, the proposition follows in this case.

Now, suppose that r_2 is $\mathsf{senc}(r_3, r_4)$ for some r_3, r_4. As $valid(r\varphi)$, $\mathsf{nf}(r\varphi) = t_4, \mathsf{nf}(r_1\varphi) = \mathsf{nf}(r_3\varphi) = t_1$ and $\mathsf{nf}(r_4\varphi) = t_4$ for some $t_1, t_4 \in CTerms$. Thus, we have that $r\varphi =_{\mathcal{R}} r_4\varphi$ and hence r_4 cannot be a subterm of r' since r is (r', φ)-simple. Thus $r_2 \neq r'$, and by induction hypothesis $valid(\mathsf{nf}(r_2^{\mathsf{new}}\varphi^{\mathsf{new}}))$, $\mathsf{nf}(r_2^{\mathsf{new}}\varphi^{\mathsf{new}}) = \mathsf{senc}(t_1^{\mathsf{new}}, t_4^{\mathsf{new}})$. We also have that $valid(r_1^{\mathsf{new}}\varphi^{\mathsf{new}})$ and $\mathsf{nf}(r_1^{\mathsf{new}}\varphi^{\mathsf{new}}) = t_1^{\mathsf{new}}$ by induction hypothesis. As r is different from r', we get that $r^{\mathsf{new}} = \mathsf{sdec}(r_1^{\mathsf{new}}, r_2^{\mathsf{new}})$. Thus, $r^{\mathsf{new}}\varphi^{\mathsf{new}} = t_4^{\mathsf{new}}$. As t_4 is a constructor term, so is t_4^{new}. The proposition follows. □
The following proposition can also be proved using induction.

Proposition D.4. Let r, r' be recipes and φ be a frame such that $vars(r), vars(r') \subseteq dom(\varphi), valid(r'\varphi)$ and r', φ are away from new. Let $\varphi^{\mathsf{new}} = \varphi[\mathsf{nf}(r'\varphi) \mapsto \mathsf{new}]$. If $valid(r\varphi^{\mathsf{new}})$ and $\mathsf{nf}(r\varphi^{\mathsf{new}}) = t$ then $valid(r[\mathsf{new} \mapsto r']\varphi)$ and $\mathsf{nf}(r[\mathsf{new} \mapsto r']\varphi) = t[\mathsf{new} \mapsto \mathsf{nf}(r'\varphi)]$.

Proof of Claim 1. Now going back to the proof of the lemma, let ρ be an execution which is a descendant of some execution ρ^j in the set $\mathcal{S}$. Let ρ be the execution $z_0 \xrightarrow{\alpha_0} z_1 \cdots \xrightarrow{\alpha_{m-1}} z_m$ where each $1 \leq i \leq m$, $z_i = (g_i, \sigma_i, \varphi_i, \ell_i)$ and $\alpha_i = (k_i, r_i)$. Let $m_0 = length(\rho^j)$. Recall that $t^j = \mathsf{nf}(\mathsf{r}_0\varphi_{m_0})$. We first show that t^j does not occur in ρ^j.

Proposition D.5. For each $0 \leq i \leq m_0$, $t^j \not\sqsubseteq ran(\sigma_i) \cup ran(\varphi_i)$.

Proof: The proof is by induction on i. Since the recipe r_0 originating at ρ_0 is not a (ρ', t, ρ'')-match for any ρ', t, ρ'', it follows that t^j is not a subterm of any term in the initial frame φ_0. Thus, the proposition is true for $i = 0$.

Now, assume that the proposition has been proved for all $i \leq i'$. Now, we first claim that $t^j \not\sqsubseteq \mathsf{nf}(r_i\varphi_i)$ for each $i \leq i'$. Indeed, if $t^j \sqsubseteq \mathsf{nf}(r_i\varphi_i)$ then thanks to Proposition D.2 above and induction hypothesis, it follows that there is a subrecipe $r' \sqsubseteq r_i$ such that $t^j = \mathsf{nf}(r'\varphi_i)$. Since ρ is simple, this means that $r' = \mathsf{r}_0$. But this contradicts the fact that r_0 originates in the execution ρ_0. Therefore $t^j \not\sqsubseteq \mathsf{nf}(r_i\varphi_i)$.

Now, let $\ell_{i'+1} = (t_{in}, t_{out})$. By definition of the semantics, we have that $\mathsf{nf}(r_i\varphi_i) = t_{in}\sigma_{i'+1}$. Since t^j is not a subterm of $\mathsf{nf}(r_i\varphi_i)$, we get immediately that $ran(\sigma_{i'+1})$ cannot contain t^j as a subterm. Also, $\varphi_{i'+1} = \varphi_i \uplus t_{out}\sigma_{i'+1}$. By induction hypothesis, we have that t^j is not a subterm of $ran(\varphi_i)$. Observe also that we have already shown that t^j is not a subterm of $ran(\sigma_{i'+1})$. Therefore if t^j is a subterm of $t_{out}\sigma_{i'+1}$ then it must be the case that there is a subterm t' of t_{out} such that t' is not a variable and $t^j = t'\sigma_{i'+1}$. Now this contradicts the fact that r_0 is not a (ρ', t, ρ'')-match for any ρ', t, ρ''. Thus, $t^j \not\sqsubseteq ran(\varphi_{i'+1})$ as well. The proposition follows. □

We get as a consequence of Proposition D.5 that r_0 is a constructive recipe.

Corollary D.6. r_0 is a constructive recipe.

Proof: Observe that Proposition D.5 implies that $t^j = \mathsf{nf}(\mathsf{r}_0\varphi_{m_0})$ is not a subterm of φ_{m_0}. Proposition D.1 immediately implies that r_0 cannot be a destructor recipe or a subterm of a destructive recipe. Thus, r_0 must be a constructive recipe. □

For, $1 \leq i \leq m$, let $\rho[0:i]$ be the execution $z_0 \xrightarrow{\alpha_0} z_1 \cdots \xrightarrow{\alpha_{i-1}} z_i$ and $\rho^{\mathsf{new}}[0:i]$ be $z_0^{\mathsf{new}} \xrightarrow{\alpha_0^{\mathsf{new}}} z_1^{\mathsf{new}} \cdots \xrightarrow{\alpha_{i-1}^{\mathsf{new}}} z_i^{\mathsf{new}}$.

Proposition D.7. For each $0 < i \leq m$, $\alpha_{i-1}^{\mathsf{new}}$ is enabled in z_{i-1}^{new}, $\rho^{\mathsf{new}}[0:i]$ is a simple execution of $\mathcal{M}_P$ and the measure of the execution ρ^{new} is the same as the measure of execution ρ. Furthermore, $\rho[0:i] \not\models \mathsf{Secret}(\mathsf{sec})$ iff $\rho^{\mathsf{new}}[0:i] \not\models \mathsf{Secret}(\mathsf{sec})$.

Proof: We first show that $\rho^{\mathsf{new}}[0:i]$ is an execution of $\mathcal{M}_P$ and the measure of the execution ρ^{new} is the same as the measure of execution ρ. It suffices to show that $z_{i-1}^{\mathsf{new}} \xrightarrow{\alpha_{i-1}^{\mathsf{new}}} \mu_i^{\mathsf{new}}$ and $\mu_i^{\mathsf{new}}(z_i^{\mathsf{new}}) = \mu_i(z_i)$ where μ_i is such that $z_{i-1} \xrightarrow{\alpha_{i-1}} \mu_i$.

Fix $m_0 < i \leq m$ and let $\ell_i = (t_{in}, t_{out})$. Note that r_0 is a constructive recipe. Since ρ is a simple execution, r_{i-1} is also $(\mathsf{r}_0, \varphi_{i-1})$-simple. Proposition D.3 implies that $valid(r_{i-1}^{\mathsf{new}}\varphi_{i-1}^{\mathsf{new}})$. Furthermore, if $t_{i-1} = \mathsf{nf}(r_{i-1}\varphi_{i-1})$ then $r_{i-1}^{\mathsf{new}}\varphi_{i-1}^{\mathsf{new}} = t_{i-1}[t^j \mapsto \mathsf{new}]$.

Now, by definition, we also have that $r_{i-1}\varphi_{i-1} = t_{in}\sigma_i$. Consider the term t_{in} as a tree. Let $Vpositions(t_{in}) \subseteq positions(t_{in})$ be the set $\{p \mid (t_{in}|p) \in \mathcal{X}\}$. Observe that the tree representation of $t_{in}\sigma_i$ is the obtained from t_{in} by replacing the variable x at position $p \in Vpositions(t_{in})$ by $\sigma_i(x)$. Now, it is easy to see that if t^j is a subterm of $t_{in}\sigma_i$ then t_j cannot be a term at a position $position(t_{in}) \setminus Vpositions(t_{in})$ as r_0 is not a (ρ'', t, ρ'')-match for any ρ'', t, ρ''. Thus, it is easy to see that $(t_{in}\sigma_i)[t^j \mapsto \mathsf{new}] = t_{in}(\sigma_i[t^j \mapsto \mathsf{new}]) = t_{in}\sigma_i^{\mathsf{new}}$. Thus, $r_{i-1}^{\mathsf{new}}\varphi_{i-1}^{\mathsf{new}} = t_{in}\sigma_i^{\mathsf{new}}$. Hence, $\alpha_{i-1}^{\mathsf{new}}$ is enabled in z_{i-1}^{new}.

As in the case of t_{in} above, we can show that $(t_{out}\sigma_i)[t^j \mapsto \mathsf{new}] = t_{out}(\sigma_i[t^j \mapsto \mathsf{new}])$. Thus $\varphi_{i-1}^{\mathsf{new}} \uplus t_{out}(\sigma_i[t^j \mapsto \mathsf{new}]\varphi_{i-1}^{\mathsf{new}}) = \varphi_i^{\mathsf{new}}$. Observe that this implies that $z_{i-1}^{\mathsf{new}} \xrightarrow{\alpha_{i-1}^{\mathsf{new}}} \mu_i^{\mathsf{new}}$.

If $\rho[0:i] \not\models \mathsf{Secret}(\mathsf{sec})$ then there is a recipe r such that $r\varphi_i \vdash \mathsf{sec}$. Without loss of generality, we can assume that r is away from new and that r is $(\mathsf{r}_0, \varphi_i)$-simple. Proposition D.3 implies that $r^{\mathsf{new}}\varphi_i^{\mathsf{new}} \vdash \mathsf{sec}$.

If $\rho^{\mathsf{new}}[0:i] \not\models \mathsf{Secret}(\mathsf{sec})$ then there is a recipe r such that $r\varphi_i^{\mathsf{new}} \vdash \mathsf{sec}$. Using Proposition D.4, we have that $valid(r[\mathsf{new} \mapsto \mathsf{r}_0]\varphi_i)$ and that $\mathsf{nf}(r[\mathsf{new} \mapsto \mathsf{r}_0]\varphi_i) = \mathsf{nf}(r\varphi_i^{\mathsf{new}})[\mathsf{new} \mapsto t^j] = \mathsf{sec}$.

The fact that $\rho^{\mathsf{new}}[0:i]$ is simple can be easily shown from the fact that $\mathcal{A}$ is simple. The proposition follows. This completes the proof of Claim 1. □

Proof of Claim 2. It suffices to consider ρ_1 and ρ_2 such that ρ_1 and ρ_2 label nodes that are descendants of nodes labeling ρ^{j_1} and ρ^{j_2} respectively for some $\rho^{j_1}, \rho^{j_2} \in \mathcal{S}$. We proceed by contradiction.

Fix $\rho_1, \rho_2, \rho^{j_1}, \rho^{j_2}$ which contradict the claim. So, we have that $view(\rho_1) \neq view(\rho_2)$ but $view(\rho_1^{\mathsf{new}}) =$

$view(\rho_2^{\mathsf{new}})$. Thus, the lengths of $\rho_1, \rho_2, \rho_1^{\mathsf{new}}, \rho_2^{\mathsf{new}}$ must be equal. Let $last(\rho_i) = (g_i, \sigma_i, \varphi_i, \ell_i)$ for $i \in \{1,2\}$ and let $last(\rho_i^{\mathsf{new}}) = (g_i, \sigma_i^{\mathsf{new}}, \varphi_i^{\mathsf{new}}, \ell_i)$. Let $last(\rho^{j_i}) = (g^i, \sigma^i, \varphi^i, \ell^i)$ for $i \in \{1,2\}$. Recall that $t^{j_i} = \mathsf{nf}(\mathsf{r}_0\varphi^{j_i})$ for $i \in \{1,2\}$. For the rest of the proof, for each recipe r, by r^{new} we mean the recipe $r[\mathsf{r}_0 \to \mathsf{new}]$.

Claim 5. For each recipe r, r_{sub} and $i \in \{1,2\}$ such that r is away from new and $\mathsf{r}_{sub} \sqsubseteq \mathsf{r}_0$ if $valid(r\varphi_i)$ and $r\varphi_i =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_i$ then $valid(r\varphi_{3-i})$ and $r\varphi_{3-i} =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_{3-i}$.

Proof: The proof is by induction on the height of the recipe r. The base case when r is a public name is immediate from definition.

If r is a variable w then r cannot be r_0 as r_0 is a constructive recipe. Thus we have that $r^{\mathsf{new}} = w$ and hence $valid(r^{\mathsf{new}}\varphi_i^{\mathsf{new}})$ for each $i \in \{1,2\}$. Fix i and $\mathsf{r}_{sub} \sqsubseteq \mathsf{r}_0$. Assume that $valid(r\varphi_i)$ and $r\varphi_i =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_i$. Let $t = w\varphi_i = r\varphi_i = \mathsf{r}_{sub}\varphi_i$. By definition, we have that $r^{\mathsf{new}}\varphi_i^{\mathsf{new}} = w\varphi_i^{\mathsf{new}} = t[t^{j_i} \mapsto \mathsf{new}]$. We also have r_{sub} is $(\mathsf{r}_0, \varphi_i)$-simple. Hence by Proposition D.3, $\mathsf{r}_{sub}^{\mathsf{new}}\varphi_i^{\mathsf{new}} = t[t^{j_i} \mapsto \mathsf{new}]$. Thus, we get that $r^{\mathsf{new}}\varphi_i^{\mathsf{new}} = \mathsf{r}_{sub}^{\mathsf{new}}\varphi_i^{\mathsf{new}}$. Thanks to the assumption that $view(\rho_1^{\mathsf{new}}) = view(\rho_2^{\mathsf{new}})$ we get that $r^{\mathsf{new}}\varphi_{3-i}^{\mathsf{new}} =_{\mathcal{R}} \mathsf{r}_{sub}^{\mathsf{new}}\varphi_{3-i}^{\mathsf{new}}$ also. Now, $\mathsf{r}_{sub}^{\mathsf{new}}$ is new (if $\mathsf{r}_{sub} = \mathsf{r}_0$) or r_{sub} (if $\mathsf{r}_{sub} \sqsubset \mathsf{r}_0$). In the latter case, r_0 not occur as a subterm of r_0. Thus, by Proposition D.4, we have that $r\varphi_{3-i} =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_{3-i}$.

Now, assume that for each r' such that height of r' is $\leq j$, each r_{sub} such that $\mathsf{r}_{sub} \sqsubseteq \mathsf{r}_0$ and for each $i \in \{1,2\}$, $valid(r'\varphi_i), r'\varphi_i =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_i \Rightarrow valid(r'\varphi_{3-i}), r'\varphi_{3-i} =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_{3-i}$.

Now consider a recipe r whose height is $j+1$. The root of the tree representing r must be in the set $\{[\cdot,\cdot], \mathsf{senc}, \mathsf{aenc}, \mathsf{proj}_1, \mathsf{proj}_2, \mathsf{sdec}, \mathsf{adec}\}$. We show that the inductive step holds when the root is $[\cdot,\cdot]$ or sdec. The other cases are similar.

1) Assume that $r = [r_1, r_2]$. Let P_1 be the set of positions p of r_1 different from the root such that there is a recipe $\mathsf{r}_{sub} \sqsubseteq \mathsf{r}_0$ and either $valid((r_1|p)\varphi_1), (r_1|p)\varphi_1 =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_1$ or $valid((r_2|p)\varphi_2), (r_2|p)\varphi_2 =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_2$. Now let P_{fix} be the subset of positions of P_1 such that if $p \in P_{\text{fix}}$ then no strict prefix of p is in P_1. Let rfix_1 be the recipe obtained from r_1 by replacing the term $r_1|p$ by r_{sub}^p for each position $p \in P_{\text{fix}}$ where $\mathsf{r}_{sub}^p \sqsubseteq \mathsf{r}_0$ is the (unique) recipe such that either $(r_1|p)\varphi_1 =_{\mathcal{R}} \mathsf{r}_{sub}^p\varphi_1$ or $(r_2|p)\varphi_2 =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_2$. The uniqueness of r_{sub}^p follows from induction hypothesis.
   Thanks to induction hypothesis, for each $p \in P_{\text{fix}}$, if $r' = r_1|p$ then $valid(r'\varphi_1), r'\varphi_1 =_{\mathcal{R}} \mathsf{r}_{sub}^p\varphi_1, valid(r'\varphi_2)$ and $r'\varphi_2 =_{\mathcal{R}} \mathsf{r}_{sub}^p\varphi_2$. Therefore, for each $i \in \{1,2\}$ $valid(\mathsf{rfix}_1\varphi_i) \Leftrightarrow valid(r_1\varphi_i)$ and $\mathsf{rfix}_1\varphi_i =_{\mathcal{R}} r_1\varphi_i \Leftrightarrow \mathsf{rfix}_1\varphi_{3-i} =_{\mathcal{R}} r_1\varphi_{3-i}$. We can define rfix_2 similarly. Let $\mathsf{rfix} = [\mathsf{rfix}_1, \mathsf{rfix}_2]$. By construction, we have that for each $i \in \{1,2\}$, $valid(\mathsf{rfix}\,\varphi_i) \Leftrightarrow valid(r\varphi_i)$ and $\mathsf{rfix}\,\varphi_i =_{\mathcal{R}} r\varphi_i \Leftrightarrow \mathsf{rfix}\,\varphi_{3-i} =_{\mathcal{R}} r\varphi_{3-i}$.
   Fix i. Assume that $valid(r\,\varphi_i)$ and $r\,\varphi_i =_{\mathcal{R}} \mathsf{r}_{sub}\,\varphi$ for some $rsub \sqsubseteq \mathsf{r}_0$. Then we have $valid(\mathsf{rfix}\,\varphi_i)$ and $\mathsf{rfix}\,\varphi_i =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_i$. Observe that the construction of $\mathsf{rfix}_1, \mathsf{rfix}_2$ also imply that $\mathsf{rfix}_1, \mathsf{rfix}_2$ are $(\mathsf{r}_0, \varphi_i)$-simple. We have two cases. Either $\mathsf{r}_{sub} \sqsubset \mathsf{r}_0$ or $\mathsf{r}_{sub} = \mathsf{r}_0$.
   First assume that $\mathsf{r}_{sub} \sqsubset \mathsf{r}_0$. If $t_1 = \mathsf{nf}(r_1\varphi_i)$ and $t_2 = \mathsf{nf}(r_2\varphi_i)$, it follows that $\mathsf{nf}(\mathsf{rfix}\,\varphi_i) = [t_1, t_2] = \mathsf{nf}(\mathsf{r}_{sub}\varphi_i)$. Note that as r_{sub} only contains frame variables in $dom(\varphi^{j_i})$, we get that $[t_1, t_2] = \mathsf{nf}(\mathsf{r}_{sub}\varphi^{j_i})$. Now, we already established in the proof of Claim 1 that t^{j_i} is not a subterm of $ran(\varphi^{j_i})$ (see Proposition D.5). Thanks to Proposition D.2, the only way that t^{j_i} is a subterm of t_1 or t_2 is if r_{sub} has a sub-recipe r'_{sub} such that $\mathsf{r}'_{sub}\varphi_i = t^{j_i}$. This contradicts the fact that r_0 is a simple recipe. It follows that $t^{j_1} \not\sqsubseteq t_1, t_2$.
   Thus, $t_1[t^{j_1} \mapsto \mathsf{new}] = t_1$ and $t_2[t^{j_2} \mapsto \mathsf{new}] = t_2$. Observe that $\mathsf{rfix} \neq \mathsf{r}_0$ and hence $\mathsf{rfix}^{\mathsf{new}} = [\mathsf{rfix}_1^{\mathsf{new}}, \mathsf{rfix}_2^{\mathsf{new}}]$. Thus, $\mathsf{rfix}^{\mathsf{new}}\varphi_i^{\mathsf{new}} = [\mathsf{rfix}_1^{\mathsf{new}}\varphi_i^{\mathsf{new}}, \mathsf{rfix}_2^{\mathsf{new}}\varphi_i^{\mathsf{new}}]$. Since rfix_1 and rfix_2 are $(\mathsf{r}_0, \varphi_i)$-simple, we get by Proposition D.3 that $\mathsf{nf}(\mathsf{rfix}_1^{\mathsf{new}}\varphi_i^{\mathsf{new}}) = t_1[t^{j_i} \mapsto \mathsf{new}] = t_1$ and $\mathsf{nf}(\mathsf{rfix}_2^{\mathsf{new}}\varphi_i^{\mathsf{new}}) = t_2[t^{j_i} \mapsto \mathsf{new}] = t_2$. Thus, $\mathsf{nf}(\mathsf{rfix}^{\mathsf{new}}\varphi_i^{\mathsf{new}}) = [t_1, t_2] = \mathsf{nf}(\mathsf{r}_{sub}\varphi_i)$. Since r_{sub} only contains frame variables in $dom(\varphi^{j_1})$, it follows that $\mathsf{nf}(\mathsf{rfix}^{\mathsf{new}}\varphi_i^{\mathsf{new}}) = \mathsf{nf}(\mathsf{r}_{sub}\varphi_i^{\mathsf{new}})$. As, we have that $view(\rho_1^{\mathsf{new}}) = view(\rho_2^{\mathsf{new}})$, we get that $valid(\mathsf{rfix}^{\mathsf{new}}\varphi_{3-i}^{\mathsf{new}})$ and $\mathsf{nf}(\mathsf{rfix}^{\mathsf{new}}\varphi_{3-i}^{\mathsf{new}}) = \mathsf{nf}(\mathsf{r}_{sub}\varphi_{3-i}^{\mathsf{new}})$. The proposition now follows from Proposition D.4.
   Now assume that $\mathsf{r}_{sub} = \mathsf{r}_0$. Now, as r_0 is a constructive recipe and $\mathsf{nf}(\mathsf{r}_0\varphi_i) = [\mathsf{nf}(r_1\varphi_i), \mathsf{nf}(r_2\varphi_i)]$, it follows that $\mathsf{r}_0 = [r_0^1, r_0^2]$ for some recipes r_0^1, r_0^2. Furthermore, $\mathsf{nf}(r_1\varphi_i) = \mathsf{nf}(r_0^1\varphi_i)$ and $\mathsf{nf}(r_2\varphi_i) = \mathsf{nf}(r_0^2\varphi_i)$. Observe that r_0^1, r_0^2 are subrecipes of r_0. Hence, the proposition immediately follows from induction hypothesis in this case.
2) Now assume that $r = \mathsf{sdec}(r_1, r_2)$. Now, as in the case of pairing above, construct $\mathsf{rfix}, \mathsf{rfix}_1, \mathsf{rfix}_2$. Fix i and $\mathsf{r}_{sub} \sqsubseteq \mathsf{r}_0$. Suppose now that $valid(r\varphi_i)$ and $r\varphi_i =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_i$. Thanks to the construction of rfix, we have that $valid(\mathsf{rfix}\,\varphi_i)$ and $\mathsf{rfix}\,\varphi_i =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_i$. Thanks to the construction of rfix, it suffices to show that $valid(\mathsf{rfix}\,\varphi_{3-i})$ and $\mathsf{rfix}\,\varphi_{3-i} =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_{3-i}$. Observe also that $\mathsf{rfix}_1, \mathsf{rfix}_2$ are (r_0, φ_i)-simple recipes.
   Observe that as $valid(\mathsf{rfix}\,\varphi_i)$, rfix_2 can be either a constructor recipe or a frame variable or a destructive recipe.
   If rfix_2 is a constructive recipe, it must be the case $\mathsf{rfix}_2 = \mathsf{senc}(r_3, r_4)$ where r_3 and r_4 are recipes such that $\mathsf{rfix}_1\varphi_i =_{\mathcal{R}} r_3\varphi_i$ and $\mathsf{rfix}\varphi_i =_{\mathcal{R}} r_4\varphi_i =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_i$. By construction of rfix_2, r_4 must be r_{sub}. Furthermore, as $\mathsf{rfix}_1\varphi_i = r_3\varphi_i$ and $\mathsf{rfix}_1, r_3, \mathsf{r}_{sub}$ are (r_0, φ_i)-simple recipes, we get using Proposition D.3 that

$valid(\mathsf{rfix}_1^{\mathsf{new}}\varphi_i^{\mathsf{new}}), valid(r_3^{\mathsf{new}}\varphi_i^{\mathsf{new}}), valid(\mathsf{r}_{sub}^{\mathsf{new}}\varphi_i^{\mathsf{new}})$ and that $\mathsf{rfix}_1^{\mathsf{new}}\varphi_i^{\mathsf{new}} = r_3^{\mathsf{new}}\varphi_i^{\mathsf{new}}$. Let $r_{final} = \mathsf{sdec}(\mathsf{rfix}_1^{\mathsf{new}}, \mathsf{senc}(r_3^{\mathsf{new}}, \mathsf{r}_{sub}^{\mathsf{new}}))$. We have that $valid(r_{final}\varphi_i^{\mathsf{new}})$ and that $\mathsf{nf}(r_{final}\varphi_i^{\mathsf{new}}) = \mathsf{nf}(\mathsf{r}_{sub}^{\mathsf{new}}\varphi_i^{\mathsf{new}})$. Since $view(\varphi_1^{\mathsf{new}}) = view(\varphi_2^{\mathsf{new}})$, we get that $valid(r_{final}\varphi_{3-i}^{\mathsf{new}})$ and that $\mathsf{nf}(r_{final}\varphi_{3-i}^{\mathsf{new}}) = \mathsf{nf}(\mathsf{r}_{sub}^{\mathsf{new}}\varphi_{3-i}^{\mathsf{new}})$. Proposition D.4 implies that $valid(\mathsf{rfix}\varphi_{3-i})$ and that $\mathsf{nf}(\mathsf{rfix}\varphi_{3-i} = \mathsf{r}_{sub}\varphi_{3-i})$. If rfix_2 is a frame variable or a destructive recipe then rfix_2 cannot be r_0 as r_0 is a constructive recipe. By construction of rfix_2, we have that $\mathsf{nf}(\mathsf{rfix}_2\varphi_i) \neq t^{j_i}$. Now as $valid(\mathsf{rfix}\varphi_i)$, we get that $\mathsf{nf}(\mathsf{rfix}_1\varphi_i) = t_1$, $\mathsf{nf}(\mathsf{rfix}_2\varphi_i) = \mathsf{senc}(t_1, t_2)$ and that $\mathsf{rfix}\varphi_i = \mathsf{r}_{sub}\varphi_i = t_2$ for some $t_1, t_2 \in CTerms$. As, $\mathsf{rfix}_1, \mathsf{rfix}_2$ are (r_0, φ_i)-simple recipes, we get by Proposition D.3 that $\mathsf{nf}(\mathsf{rfix}_1^{\mathsf{new}}\varphi_i^{\mathsf{new}}) = t_1[t^{j_i} \mapsto \mathsf{new}]$ and that $\mathsf{nf}(\mathsf{rfix}_2^{\mathsf{new}}\varphi_i^{\mathsf{new}}) = \mathsf{senc}(t_1[t^{j_i} \mapsto \mathsf{new}], t_2[t^{j_i} \mapsto \mathsf{new}])$. From this it is easy to see that the recipe $valid(\mathsf{sdec}(\mathsf{rfix}_1^{\mathsf{new}}, \mathsf{rfix}_2^{\mathsf{new}})\varphi_i^{\mathsf{new}})$ and that $\mathsf{nf}(\mathsf{sdec}(\mathsf{rfix}_1^{\mathsf{new}}, \mathsf{rfix}_2^{\mathsf{new}})\varphi_i^{\mathsf{new}}) = t_2[t^{j_i} \mapsto \mathsf{new}])$. Now, thanks to Proposition D.3 again, we have that $valid(\mathsf{r}_{sub}^{\mathsf{new}}\varphi_i^{\mathsf{new}})$ and that $\mathsf{nf}(\mathsf{r}_{sub}^{\mathsf{new}}\varphi_i^{\mathsf{new}}) = t_2[t^{j_i} \mapsto \mathsf{new}] = \mathsf{nf}(\mathsf{sdec}(\mathsf{rfix}_1^{\mathsf{new}}, \mathsf{rfix}_2^{\mathsf{new}})\varphi_i^{\mathsf{new}})$. Now we can conclude that $valid(\mathsf{rfix}\varphi_{3-i})$ and that $\mathsf{nf}(\mathsf{rfix}\varphi_{3-i}) = \mathsf{nf}(\mathsf{r}_{sub}\varphi_{3-i})$ as in the above case. Therefore, $\mathsf{nf}(r\varphi_{3-i}) = \mathsf{nf}(\mathsf{r}_{sub}\varphi_{3-i})$, and the proposition follows.

□

Claim 6. For any $i \in \{1, 2\}$ and recipe r away from new, if $valid(r\varphi_)$ then $valid(r\varphi_{3-i})$.

Proof: First consider the case when r is $(\mathsf{r}_0, \varphi_i)$-simple. Now, r_0 is a constructive recipe. Thanks to Proposition D.3, we have that $valid((r[\mathsf{r}_0 \mapsto \mathsf{new}])\varphi_i^{\mathsf{new}})$. Since $view(\rho_1^{\mathsf{new}}) = view(\rho_2^{\mathsf{new}})$, we get that $valid((r[\mathsf{r}_0 \mapsto \mathsf{new}])\varphi_{3-i}^{\mathsf{new}})$ as well. Proposition D.4 implies that $valid(r\varphi_{3-i})$.

Now suppose r is not $(\mathsf{r}_0, \varphi_i)$-simple. Let P_1 be the set of positions p of r such that there is a recipe $\mathsf{r}_{sub} \sqsubseteq \mathsf{r}_0$ and either $valid((r|p)\varphi_1), (r|p)\varphi_1 =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_1$ or $valid((r|p)\varphi_2), (r|p)\varphi_2 =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_2$. Let P_{fix} be the largest subset of positions of P_1 such that if $p \in P_{\text{fix}}$ then no strict prefix of p is in P_1. Let rfix be the recipe obtained from r by replacing the term $r|p$ by r_{sub}^p for each position $p \in P_{\text{fix}}$ where $\mathsf{r}_{sub}^p \sqsubseteq \mathsf{r}_0$ is the (unique) recipe such that either $(r_1|p)\varphi_1 =_{\mathcal{R}} \mathsf{r}_{sub}^p\varphi_1$ or $(r_2|p)\varphi_2 =_{\mathcal{R}} \mathsf{r}_{sub}\varphi_2$. The uniqueness follows from the fact that r_0 is simple and Claim 5. It is easy to see by Claim 5 that rfix is $(\mathsf{r}_0, \varphi_i)$-simple and that $valid(r\varphi_j)$ iff $valid(\mathsf{rfix}\varphi_j)$ for $j \in \{1, 2\}$. The above case implies that $valid(\mathsf{rfix}\varphi_{3-i})$ if $valid(r\varphi_i)$. Hence, $valid(r\varphi_{3-i})$ as well. □

Claim 7. For any $i \in \{1, 2\}$ and recipes r, r' away from new such that $valid(r\varphi_i), valid(r'\varphi_i)$, if $r\varphi_i =_{\mathcal{R}} r'\varphi_i$ then $r\varphi_{3-i} =_{\mathcal{R}} r'\varphi_{3-i}$ also.

Proof: As in the proof of Claim 6, it suffices to consider the case when r, r' are $(\mathsf{r}_0, \varphi_j)$-simple for $j \in \{1, 2\}$. Now, r_0 is a constructive recipe. Thanks to Proposition D.3, we have that $(r[\mathsf{r}_0 \mapsto \mathsf{new}])\varphi_i^{\mathsf{new}} =_{\mathcal{R}} (r'[\mathsf{r}_0 \mapsto \mathsf{new}])\varphi_i^{\mathsf{new}}$. Since $view(\rho_1^{\mathsf{new}}) = view(\rho_2^{\mathsf{new}})$, we get that $(r[\mathsf{r}_0 \mapsto \mathsf{new}])\varphi_{3-i}^{\mathsf{new}} =_{\mathcal{R}} (r'[\mathsf{r}_0 \mapsto \mathsf{new}])\varphi_{3-i}^{\mathsf{new}}$. Proposition D.4 implies that $r\varphi_{3-i} = r'\varphi_{3-i}$. □

Claim 6 and Claim 7 contradict the fact that $view(\rho_1) \neq view(\rho_2)$. This completes the proof of the claim.

Appendix E. Proof of Theorem 4.6

The proof of the theorem is similar to the proof of Theorem 4.3. Let $\mathcal{A}$ be a simple adversary that distinguishes $\mathcal{M}_P$ and $\mathcal{M}_{P'}$. As in the case of the proof of Theorem 4.3, we can view the DTMCs $\mathcal{M}_P^{\mathcal{A}}$ and $\mathcal{M}_{P'}^{\mathcal{A}}$ as trees $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ and $\mathcal{T}_{\mathcal{M}_{P'}^{\mathcal{A}}}$ respectively. The nodes of $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ and $\mathcal{T}_{\mathcal{M}_{P'}^{\mathcal{A}}}$ are labeled by executions of $\mathcal{M}_P$ and $\mathcal{M}_{P'}$ respectively. Observe that the total number of nodes in either of these two trees is $\leq \lambda\eta^{\gamma}$.

As in the case of proof of Theorem 4.3, we can again define the notion of a recipe r originating at an execution labeling a node of $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ or $\mathcal{T}_{\mathcal{M}_{P'}^{\mathcal{A}}}$. Analogous to the notion of a match defined in the proof of Theorem 4.3, we can also define the notion of a small adversary. Consider a recipe r originating at execution ρ. Given a term $t \in subterms(P, P') \setminus \mathcal{X}$ and executions ρ^1, ρ^2 such that ρ^1 and ρ^2 label some nodes in $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ or in $\mathcal{T}_{\mathcal{M}_{P'}^{\mathcal{A}}}$, $view(prev(\rho)) = view(\rho^1)$, $enabled(\rho^1, \alpha_{m-1})$ and $last(\rho^j) = (g^j, \sigma^j, \varphi^j, (t_{in}^j, t_{out}^j))$ for $j = 1, 2$, we say that the recipe r is a (ρ^1, t, ρ^2)-*match* if r is not a public name or recipe variable, ie, $r \notin \mathsf{Pub} \cup \mathcal{X}_w$ and one of the following hold:

1) ρ^2 is the initial state of $\mathcal{M}_P^{\mathcal{A}}$ or of $\mathcal{M}_{P'}^{\mathcal{A}}$, t is a subterm of $ran(\varphi_0)$ and $\mathsf{nf}(r\varphi^1) = \mathsf{nf}(t\sigma^2)$.
2) ρ^2 is not the initial state of $\mathcal{M}_P^{\mathcal{A}}$ or of $\mathcal{M}_{P'}^{\mathcal{A}}$, t is a subterm of either t_{in}^2 or t_{out}^2 and $\mathsf{nf}(r\varphi^1) = \mathsf{nf}(t\sigma^2)$.

We say that the execution ρ is an $\mathcal{A}$-match if for every recipe r originating at ρ, either $r \in \mathsf{Pub} \cup \mathcal{X}_w$ or there is a term t and executions ρ^1, ρ^2 such that r is a (ρ^1, t, ρ^2)-match. We say that a simple adversary $\mathcal{A}$ is *small* if every execution ρ labeling a node in $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ or a node in $\mathcal{T}_{\mathcal{M}_{P'}^{\mathcal{A}}}$, is an $\mathcal{A}$-match.

An argument similar to the proof of Lemma 4.4 then shows that if a simple adversary $\mathcal{A}$ is small then $rsize(\mathcal{A}) \leq 12\theta\lambda\eta^{2\gamma}$. A factor of 4 arises because the total number of nodes in the two trees is at most $2\lambda\eta^{\gamma}$.

We proceed once again as in the proof of Theorem 4.3 by showing that if P and P' are distinguishable by an adversary $\mathcal{A}$ then they are distinguishable by a small adversary as follows.

Fix a simple adversary $\mathcal{A}$ that distinguishes P and P'. As in the proof of Lemma 4.5, if $\mathcal{A}$ is not small then we can construct a new adversary $\mathcal{A}_{small}$ by iteratively replacing recipes that witness the fact $\mathcal{A}$ is not small by fresh public names. The main obligation that we have to show is that $\mathcal{A}_{small}$ distinguishes P and P'.

Let $\mathcal{E}$ be the set of executions of the DTMCs $\mathcal{M}_P^{\mathcal{A}}$ and $\mathcal{M}_{P'}^{\mathcal{A}}$. Let $\mathcal{E}_{small}$ be the set of executions of the DTMCs $\mathcal{M}_P^{\mathcal{A}_{small}}$ and $\mathcal{M}_{P'}^{\mathcal{A}_{small}}$. The proof of Lemma 4.5 implies that there is a bijection $sm : \mathcal{E} \to \mathcal{E}_{small}$ such that

1) The measure of execution $\kappa \in \mathcal{E}$ is exactly the measure of the execution $sm(\kappa)$.
2) If $view(last(\kappa)) \neq view(last(\kappa')$ for $\kappa, \kappa' \in \mathcal{E}$ then $view(last(sm(\kappa))) \neq view(last(sm(\kappa')))$.

As $\mathcal{A}$ distinguishes P and P' there must be a sequence $\overline{o} \in (Frames/\sim)^* \times (\{1, 2, \ldots, n\} \times Recipes)$ such that $pr_{\mathcal{M}_P}(\overline{o}, \mathcal{A}) \neq pr_{\mathcal{M}_{P'}}(\overline{o}, \mathcal{A})$. Fix $\overline{o}$. Let $\mathcal{E}_{\overline{o}}$ be the set $\{\kappa \mid view(last(\kappa)) = \overline{o}\}$. Let $\overline{O}_{small}$ be the set $\{\overline{o}_{sm} \mid \overline{o}_{sm} = view(last(sm(\kappa)))$ for some $\kappa \in \mathcal{E}_{\overline{o}}\}$. The above two conditions imply that

$$pr_{\mathcal{M}_P}(\overline{o}, \mathcal{A}) = \sum_{\overline{o}_{sm} \in \overline{O}_{small}} pr_{\mathcal{M}_P}(\overline{o}_{sm}, \mathcal{A}_{small})$$

and

$$pr_{\mathcal{M}_{P'}}(\overline{o}, \mathcal{A}) = \sum_{\overline{o}_{sm} \in \overline{O}_{small}} pr_{\mathcal{M}_{P'}}(\overline{o}_{sm}, \mathcal{A}_{small}).$$

Since $pr_{\mathcal{M}_P}(\overline{o}, \mathcal{A}) \neq pr_{\mathcal{M}_{P'}}(\overline{o}, \mathcal{A})$, we get immediately that there must be an $\overline{o}_{sm} \in \overline{O}_{small}$ such that

$$pr_{\mathcal{M}_P}(\overline{o}_{sm}, \mathcal{A}_{small}) \neq pr_{\mathcal{M}_{P'}}(\overline{o}_{sm}, \mathcal{A}_{small}).$$

Thus, $\mathcal{A}_{small}$ also distinguishes P and P'. The theorem follows.

Appendix F. Proof of Theorem 4.7

We first consider the secrecy problem and show that the non-secrecy problem can be decided in **NEXPTIME**. The **NEXPTIME** time procedure for deciding the non-secrecy problem non-deterministically constructs the tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ which corresponds to a simple adversary $\mathcal{A}$ such that $rsize(\mathcal{A}) \leq 3\theta\lambda\eta^{2\gamma}$ where θ, η, γ, and λ are as defined in the statement of Theorem 4.3. The tree $\mathcal{T}_{\mathcal{M}_P^{\mathcal{A}}}$ is constructed incrementally. Initially all children of the root are guessed by guessing the initial recipe used by the adversary, the role the first message is sent to and the transition of the role that is triggered. The recipe is guessed in its dag form and is of size $\leq 3\theta\lambda\eta^{2\gamma}$. The procedure checks that the recipe chosen is valid, message computed by the recipe chosen unifies with the left-hand size of the transition chosen and computes the unifier. These steps can be performed in time polynomial in the size of the recipe. Then the children of the root are computed, making sure that the terms in the labels of the children are also kept in dag form. We also label the edge from the root to its child by the probability of making that transition. The procedure then guesses the children of each of nodes at level 1 in a similar fashion except that the adversary has to perform some additional tests. These tests are done to make sure that if the adversary uses different recipes for two executions at level 1 then these two executions are distinguishable. In order to check if two executions are distinguishable, the procedure checks that the last states in the executions of these two nodes are distinguishable. The latter is achieved by guessing the test which distinguishes the frames at these states. The results of [1], [15] imply that this test must be of size polynomial in the size of the frames. The subsequent levels of the tree are guessed incrementally in a similar fashion. The resulting tree has most $\lambda\eta^{\gamma}$ nodes. It is easy to show that the dag-size of terms in each state is exponential in the size of the protocol. After guessing the tree, the procedure marks the leaf nodes such that the secret sec is revealed in the last state of executions labeling these nodes. Finally, the procedure checks that the measure of the paths from the root to the marked nodes is $> p$.

Now, indistinguishability of two protocols can be shown to be in **coNEXPTIME** along similar lines using Theorem 4.6.

Appendix G. Proof of Theorem 5.3

Before proving the hardness result, we state the precise pair of results about the monadic logic in Definition 5.2 that we will exploit in our proof.

Theorem G.1 (Theorem 5.1 in [30]). Given a monadic formula φ, φ is satisfiable if and only if φ has a model of size $\leq 2^k$, where k is the number of monadic relation symbols appearing in φ.

Theorem G.2 (Theorem 4.1 in [30]). Let φ be a monadic formula of the form

$$\exists z.F_1(z) \wedge \forall y \exists x.F_2(y, x) \wedge \forall y_1 \forall y_2.F_3(y_1, y_2),$$

where F_1, F_2, and F_3 are quantifier-free 3-CNF formulas. The problem of checking if φ is satisfiable is **NEXPTIME**-hard.

We are now ready to prove Theorem 5.3. Given a monadic formula φ, we will construct (in polynomial time) a protocol P such that the secret is revealed in P with probability 1 iff φ is satisfiable. First, recall that from Theorem G.2, we can assume that φ is of the form $\exists z.F_1(z) \wedge \forall y \exists x.F_2(y, x) \wedge \forall y_1 \forall y_2.F_3(y_1, y_2)$, where F_1, F_2, and F_3 are quantifier-free 3-CNF formulas. Second, we know that if φ is satisfiable, it has a model of size at most 2^k, where k is the number of predicate symbols appearing in φ (Theorem G.1). In fact, without loss of generality, we can take the model to have size exactly 2^k. This is because we can construct a formula $T(\varphi)$ with one additional monadic predicate inUniv such that $T(\varphi)$ has a model of size 2^k iff φ is satisfiable. In $T(\varphi)$, inUniv determines if an element of the universe is a "valid" element of the universe. The formula $T(\varphi)$ is constructed inductively, with $T(\exists x.\psi) = \exists x.\mathsf{inUniv}(x) \wedge T(\psi)$ and $T(\forall x.\psi) = \forall x.\mathsf{inUniv}(x) \rightarrow T(\psi)$. Notice, $T(\varphi)$ has the same pattern of quantifiers as φ, and hence is of the same "form" as φ.

The protocol P will have one role R and proceeds as follows. In the first step, the adversary guesses a model for the formula φ and sends this to the protocol. In order to ensure that the adversary cannot change this model at a later step, this model is encrypted by the protocol using a secret key. This very much like what happens in the reduction from 3SAT [33] or QSAT. From Theorem G.1 and the observations in the previous paragraph, we can assume that the size of the model constructed by the adversary is 2^k where k is the number of predicate symbols in φ; we will assume that the k predicate symbols in φ are $\{P_0, \ldots P_{k-1}\}$. The protocol assumes that the model sent by the adversary is encoded as a full binary tree of height k, with leaves of the tree corresponding to the elements of the universe. Each leaf in the tree is labeled by a tuple of size k of the form $[b_1, [b_2, \cdots [b_{k-1}, b_k]]]$, where $b_i \in \{\top, \bot\}$ encodes if the ith predicate is true at this element. Thus, the first step in the protocol is a rule as follows.

$$s_0 : m \Rightarrow [g_{z0} : \mathsf{senc}(M, m)]$$

Here s_0 is the initial state, M is a secret key, and g_{z0} is the state in the next phase of the protocol where the variable z (see format of φ) is assigned an element of the universe.

The next phase of the protocol is one where the variables in the formula are instantiated to a specific element in the model. Like in the QSAT case, existential variables are chosen by the adversary, and universal variables are picked stochastically by the protocol. Picking the value of a variable takes k steps of the protocol, where in each step you walk through the binary tree representation of the model until a leaf (i.e., an element of the universe) is reached. After this, the protocol takes another k steps to send to the adversary the truth of each predicate at the chosen element, in an encrypted form; encryption is again used so that the adversary does not cheat during the formula evaluation phase. Here we show the rules that are used for $\forall y \exists x$; the other variables are handled similarly. Here g_{y0} is the state where the first bit of y is picked, and the keys $\{y^i \mid i \leq 2k\} \cup \{x^i \mid i \leq 2k\} \cup \{P_i(y) \mid i \leq k\} \cup \{P_i(x) \mid i \leq k\} \subseteq \mathsf{Prv}$ are assumed to be unknown to the adversary. We begin with the rules for picking y.

$$\begin{aligned}
g_{y0} &: \mathsf{senc}(M, [u_0^0, u_1^0]) \Rightarrow [g_{y1} : [\mathsf{senc}(y^1, u_0^0), \mathbf{0}]] \oplus \\
&\qquad [g_{y1} : [\mathsf{senc}(y^1, u_1^0), \mathbf{1}]] \\
g_{yi} &: \mathsf{senc}(y^i, [u_0^i, u_1^i]) \Rightarrow \\
&\qquad [g_{y(i+1)} : [\mathsf{senc}(y^{i+1}, u_0^{i+1}), \mathbf{0}]] \oplus \\
&\qquad [g_{y(i+1)} : [\mathsf{senc}(y^{i+1}, u_1^{i+1}), \mathbf{1}]] \text{ for } 1 \leq i < k \\
g_{yj} &: \mathsf{senc}(y^j, [t_j^y, r_j^y]) \Rightarrow \\
&\qquad [g_{y(j+1)} : [\mathsf{senc}(P_{j-k}(y), t_j^y), \mathsf{senc}(y^j, r_j^y)]] \\
&\qquad\qquad \text{for } k \leq j < 2k-2 \\
g_{y(2k-2)} &: \mathsf{senc}(y^{2k-2}, [t_{2k-2}^y, t_{2k-1}^y]) \Rightarrow \\
&\qquad [g_{x0} : [\mathsf{senc}(P_{k-2}(y), t_{2k-2}^y), \mathsf{senc}(P_{k-1}(y), t_{2k-1}^y)]]
\end{aligned}$$

Notice that the coin tosses in the first k steps (choosing the bits of y) are made public. Also at the end, the adversary has messages of the form $\mathsf{senc}(P_i(y), b_i)$ where b_i is the truth of predicate P_i at the chosen y in the model. The steps for $\exists x$ are similar, except that the adversary chooses the bits of x.

$$\begin{aligned}
g_{x0} &: [\mathsf{senc}(M, [v_0^0, v_1^0]), \mathbf{0}] \Rightarrow [g_{x1} : \mathsf{senc}(x^1, v_0^0)] \\
g_{x0} &: [\mathsf{senc}(M, [v_0^0, v_1^0]), \mathbf{1}] \Rightarrow [g_{x1} : \mathsf{senc}(x^1, v_1^0)] \\
g_{xi} &: [\mathsf{senc}(x^i, [v_0^i, v_1^i]), \mathbf{0}] \Rightarrow [g_{x(i+1)} : \mathsf{senc}(x^{i+1}, v_0^i)] \\
&\qquad \text{for } 1 \leq i < k \\
g_{xi} &: [\mathsf{senc}(x^i, [v_0^i, v_1^i]), \mathbf{1}] \Rightarrow [g_{x(i+1)} : \mathsf{senc}(x^{i+1}, v_1^i)] \\
&\qquad \text{for } 1 \leq i < k \\
g_{xj} &: \mathsf{senc}(x^j, [t_j^x, r_j^x]) \Rightarrow [g_{x(j+1)} : [\mathsf{senc}(P_{j-k}(x), t_j^x), \\
&\qquad\qquad \mathsf{senc}(x^j, r_j^x)]] \\
&\qquad \text{for } k \leq j < 2k-2 \\
g_{x(2k-2)} &: \mathsf{senc}(x^{2k-2}, [t_{2k-2}^x, t_{2k-1}^x]) \Rightarrow [g_{y10} : \\
&\qquad [\mathsf{senc}(P_{k-2}(x), t_{2k-2}^x), \mathsf{senc}(P_{k-1}(x), t_{2k-1}^x)]]
\end{aligned}$$

After all the variables are chosen, the adversary has encrypted values of the truth of each predicate symbol at each of the values chosen. The protocol now moves to the phase where the 3CNF formulas $F_1(z)$, $F_2(y, x)$, and $F_3(y_1, y_2)$ are evaluated. The rules for this are very similar to the evaluation of QSAT formulas after the truth assignment is picked. We, therefore, skip describing these rules. Like in the QSAT case, it is easy to see that sec is revealed with probability 1 if and only if φ is satisfiable. The protocol keeps sec secret with probability at least $\frac{1}{2^{3k}}$ if φ is not satisfiable.

We conclude the proof by observing that all coin tosses in the above reduction are public. Thus, the **NEXPTIME**-hardness result holds even for checking the insecurity of randomized protocols where all coin tosses are public.