Model checking indistinguishability of randomized security protocols

Matthew S. Bauer[1,4⋆], Rohit Chadha[2⋆⋆], A. Prasad Sistla[3⋆⋆⋆], and Mahesh Viswanathan[1⋆]

[1] University of Illinois at Urbana-Champaign
[2] University of Missouri
[3] University of Illinois at Chicago
[4] Galois Inc.

Abstract. The design of security protocols is extremely subtle and vulnerable to potentially devastating flaws. As a result, many tools and techniques for the automated verification of protocol designs have been developed. Unfortunately, these tools don't have the ability to model and reason about protocols with randomization, which are becoming increasingly prevalent in systems providing privacy and anonymity guarantees. The security guarantees of these systems are often formulated by means of the indistinguishability of two protocols. In this paper, we give the first practical algorithms for model checking indistinguishability properties of randomized security protocols against the powerful threat model of a bounded Dolev-Yao adversary. Our techniques are implemented in the Stochastic Protocol ANalayzer (SPAN) and evaluated on several examples. As part of our evaluation, we conduct the first automated analysis of an electronic voting protocol based on the 3-ballot design.

1 Introduction

Security protocols are highly intricate and vulnerable to design flaws. This has led to a significant effort in the construction of tools for the automated verification of protocol designs. In order to make automation feasible [12, 34, 48, 23, 55, 15, 8], the analysis is often carried out in the *Dolev-Yao* threat model [30], where the assumption of perfect cryptography is made. In the Dolev-Yao model, the omnipotent adversary has the ability to read, intercept, modify and replay all messages on public channels, remember the communication history as well as non-deterministically inject its own messages into the network while remaining anonymous. In this model, messages are symbolic terms modulo an equational theory (as opposed to bit-strings) and cryptographic operations are modeled via equations in the theory.

A growing number of security protocols employ randomization to achieve privacy and anonymity guarantees. Randomization is essential in protocols/systems

⋆ Partially supported by grant NSF CNS 1314485
⋆⋆ Partially supported by grants NSF CNS 1314338 and NSF CNS 1553548
⋆⋆⋆ Partially supported by grants NSF CNS 1314485 and CCF 1564296

for anonymous communication and web browsing such as Crowds [49], mix-networks [21], onion routers [37] and Tor [29]. It is also used in fair exchange [11, 35], vote privacy in electronic voting [52, 20, 6, 54] and denial of service prevention [40]. In the example below, we demonstrate how randomization is used to achieve privacy in electronic voting systems.

Example 1. Consider a simple electronic voting protocol for 2 voters Alice and Bob, two candidates and an election authority. The protocol is as follows. Initially, the election authority will generate two private tokens t_A and t_B and send them to Alice and Bob encrypted under their respective public keys. These tokens will be used by the voters as proofs of their eligibility. After receiving a token, each voter sends his/her choice to the election authority along with the proof of eligibility encrypted under the public key of the election authority. Once all votes have been collected, the election authority tosses a fair private coin. The order in which Alice and Bob's votes are published depends on the result of this coin toss. *Vote privacy* demands that an adversary not be able to deduce how each voter voted.

All the existing Dolev-Yao analysis tools are fundamentally limited to protocols that are purely non-deterministic, where non-determinism models concurrency as well as the interaction between protocol participants and their environment. There are currently no analysis tools that can faithfully reason about protocols like those in Example 1, a limitation that has long been identified by the verification community. In the context of electronic voting protocols, [28] identifies three main classes of techniques for achieving vote privacy; blind signature schemes, homomorphic encryption and randomization. There the authors concede that protocols based on the latter technique are "hard to address with our methods that are purely non-deterministic." Catherine Meadows, in her summary of the over 30 year history of formal techniques in cryptographic protocol analysis [46, 47], identified the development of formal analysis techniques for anonymous communication systems, almost exclusively built using primitives with randomization, as a fundamental and still largely unsolved challenge. She writes, "it turned out to be difficult to develop formal models and analyses of large-scale anonymous communication. The main stumbling block is the threat model".

In this work, we take a major step towards overcoming this long-standing challenge and introduce the first techniques for automated Dolev-Yao analysis of randomized security protocols. In particular, we propose two algorithms for determining indistinguishability of randomized security protocols and implemented them in the Stochastic Protocol ANalyzer (SPAN). Several works [28, 7, 9, 32, 41] have identified indistinguishability as the natural mechanism to model security guarantees such as anonymity, unlinkability, and privacy. Consider the protocol from Example 1, designed to preserve vote privacy. Such a property holds if the executions of the protocol in which Alice votes for candidate 1 and Bob votes for candidate 2 cannot be distinguished from the executions of the protocol in which Alice votes for candidate 2 and Bob votes for candidate 1.

Observe that in Example 1, it is crucial that the result of the election authority's coin toss is not visible to the adversary. Indeed if the adversary is allowed to "observe" the results of private coin tosses, then the analysis may reveal "security flaws" in correct security protocols (see examples in [22, 13, 36, 19, 17]). Thus, many authors [26, 22, 13, 36, 19, 17, 10, 18] have proposed that randomized protocols be analyzed with respect to adversaries that are forced to schedule the same action in any two protocol executions that are indistinguishable to them.

For randomized security protocols, [10, 18, 53] have proposed that trace equivalence from the applied π-calculus [5] serve as the indistinguishability relation on traces. In this framework, the protocol semantics are described by partially observable Markov decision processes (POMDPs) where the adversary's actions are modeled non-deterministically. The adversary is required to choose its next action based on the partial information that is can observe about the execution thus far. This allows us to model the privacy of coin tosses. Two security protocols are said to be indistinguishable [18, 53] if their semantic descriptions as POMDPs are indistinguishable. Two POMDPs $\mathcal{M}$ and $\mathcal{M}'$ are said to indistinguishable if for any adversary $\mathcal{A}$ and trace $\overline{o}$, the probability of the executions that generate the trace $\overline{o}$ with respect to $\mathcal{A}$ are the same for both $\mathcal{M}$ and $\mathcal{M}'$.

Our algorithms for indistinguishability in randomized security protocols are built on top of techniques for solving indistinguishability in finite POMDPs. Our first result shows that indistinguishability of finite POMDPs is **P**-complete. Membership in **P** is established by a reduction of POMDP indistinguishability to equivalence in probabilistic finite automata (PFAs), which is known to be **P**-complete [57, 31, 45]. Further, we show that the hardness result continues to hold for acyclic POMDPs. An acyclic POMDP is a POMDP that has a set of "final" absorbing states and the only cycles in the underlying graph are self-loops on these states.

For acyclic finite POMDPs, we present another algorithm for checking indistinguishability based on the technique of translating a POMDP $\mathcal{M}$ into a fully observable Markov decision process (MDP), known as the belief MDP $\mathcal{B}(\mathcal{M})$ of $\mathcal{M}$. It was shown in [14] that two POMDPs are indistinguishable if and only if the belief MDPs they induce are bisimilar as labeled Markov decision processes. When $\mathcal{M}$ is acylic and finite then its belief MDP $\mathcal{B}(\mathcal{M})$ is finite and acyclic and its bisimulation relation can be checked recursively.

Protocols in SPAN are described by a finite set of roles (agents) that interact asynchronously by passing messages. Each role models an agent in a protocol session and hence we only consider bounded number of sessions. An action in a role performs either a message input, or a message output or a test on messages. The adversary schedules the order in which these actions are executed and generates input recipes comprised of public information and messages previously output by the agents. In general, there are an unbounded number of input recipes available at each input step, resulting in POMDPs that are infinitely branching. SPAN, however, searches for bounded attacks by bounding the size of attacker messages. Under this assumption, protocols give rise to finite acyclic POMDPs. Even with this assumption, protocols specified in SPAN describe POMDPs that

are exponentially larger than their description. Nevertheless, we show that when considering protocols defined over subterm convergent equational theories, indistinguishability of randomized security protocols is in **PSPACE** for bounded Dolev-Yao adversaries. We further show that the problem is harder than $\#\mathsf{SAT}_\mathsf{D}$ and hence it is both **NP**-hard and **coNP**-hard.

The main engine of Span translates a randomized security protocol into an acyclic finite POMDP by recursively unrolling all protocol executions and grouping states according to those that are indistinguishable. We implemented two algorithms for checking indistinguishability in Span. The first algorithm, called the PFA algorithm, checks indistinguishability of P and P' by converting them to corresponding PFAs A and A' as in the proof of decidability of indistinguishability of finite POMDPs. PFA equivalence can then be solved through a reduction to linear programming [31]. The second algorithm, called the on-the-fly (OTF) algorithm, is based on the technique of checking bisimulation of belief MDPs. Although asymptotically less efficient than the PFA algorithm, the recursive procedure for checking bisimulation in belief MDPs can be embedded into the main engine of Span with little overhead, allowing one to analyze indistinguishability on-the-fly as the POMDP models are constructed.

In our evaluation of the indistinguishability algorithms in Span, we conduct the first automated Dolev-Yao analysis for several new classes of security protocols including dinning cryptographers networks [38], mix networks [21] and a 3-ballot electronic voting protocol [54]. The analysis of the 3-ballot protocol, in particular, demonstrates that our techniques can push symbolic protocol verification to new frontiers. The protocol is a full scale, real world example, which to the best of our knowledge, hasn't been analyzed using any existing probabilistic model checker or protocol analysis tool.

Summary of Contributions. We showed that the problem of checking indistinguishability of POMDPs is **P**-complete. The indistinguishability problem for bounded instances of randomized security protocols over subterm convergent equational theories (bounded number of sessions and bounded adversarial non-determinism) is shown to be in **PSPACE** and $\#\mathsf{SAT}_\mathsf{D}$-hard. We proposed and implemented two algorithms in the Span protocol analysis tool for deciding indistinguishability in bounded instances of randomized security protocols and compare their performance on several examples. Using Span, we conducted the first automated verification of a 3-ballot electronic voting protocol.

Related work. As alluded to above, techniques for analyzing security protocols have remained largely disjoint from techniques for analyzing systems with randomization. Using probabilistic model checkers such as PRISM [44], STORM [27] and APEX [42] some have attempted to verify protocols that explicitly employ randomization [56]. These ad-hoc techniques fail to capture powerful threat models, such as a Dolev-Yao adversary, and don't provide a general verification framework. Other works in the Dolev-Yao framework [28, 43] simply abstract away essential protocol components that utilize randomization, such as anonymous channels. The first formal framework combining Dolev-Yao analysis with

randomization appeared in [10], where the authors studied the conditions under which security properties of randomized protocols are preserved by protocol composition. In [53], the results were extended to indistinguishability.

Complexity-theoretic results on verifying secrecy and indistinguishability properties of bounded sessions of randomized security protocols against unbounded Dolev-Yao adverasries were studied in [18]. There the authors considered protocols with a fixed equational theory[5] and no negative tests (else branches). Both secrecy and indistinguishability were shown to be in **coNEXPTIME**, with secrecy being **coNEXPTIME**-hard. The analogous problems for purely non-deterministic protocols are known to be **coNP**-complete [25, 33, 51]. When one fixes, a priori, the number of coin tosses, secrecy and indistinguishability in randomized protocols again become **coNP**-complete. In our asymptotic complexity results and in the SPAN tool, we consider a general class of equational theories and protocols that allow negative tests.

2 Preliminaries

We assume that the reader is familiar with probability distributions. For a set X, $\mathsf{Dist}(X)$ shall denote the set of all discrete distributions μ on X such that $\mu(x)$ is a rational number for each $x \in X$. For $x \in X$, δ_x will denote the Dirac distribution, i.e., the measure μ such that $\mu(x) = 1$. The *support* of a discrete distribution μ, denoted $\mathsf{support}(\mu)$, is the set of all elements x such that $\mu(x) \neq 0$.

Markov decision processes (MDPs). MDPs are used to model processes that exhibit both probabilistic and non-deterministic behavior. An MDP $\mathcal{M}$ is a tuple $(Z, z_s, \mathsf{Act}, \Delta)$ where Z is a countable set of states, $z_s \in Z$ is the initial state, Act is a countable set of actions and $\Delta : Z \times \mathsf{Act} \to \mathsf{Dist}(Z)$ is the probabilistic transition function. $\mathcal{M}$ is said to be finite if the sets Z and Act are finite. An execution of an MDP is a sequence $\rho = z_0 \xrightarrow{\alpha_1} z_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_m} z_m$ such that $z_0 = z_s$ and $z_{i+1} \in \mathsf{support}(\Delta(z_i, \alpha_{i+1}))$ for all $i \in \{0, \ldots, m-1\}$. The *measure* of ρ, denoted $\mathsf{prob}_{\mathcal{M}}(\rho)$, is $\prod_{i=0}^{m-1} \Delta(z_i, \alpha_{i+1})(z_{i+1})$. For the execution ρ, we write $\mathsf{last}(\rho) = z_m$ and say that the length of ρ, denoted $|\rho|$, is m. The set of all executions of $\mathcal{M}$ is denoted as $\mathsf{Exec}(\mathcal{M})$.

Partially observable Markov decision processes (POMDPs). A POMDP $\mathcal{M}$ is a tuple $(Z, z_s, \mathsf{Act}, \Delta, \mathcal{O}, \mathsf{obs})$ where $\mathcal{M}_0 = (Z, z_s, \mathsf{Act}, \Delta)$ is an MDP, $\mathcal{O}$ is a countable set of observations and $\mathsf{obs} : Z \to \mathcal{O}$ is a labeling of states with observations. $\mathcal{M}$ is said to be finite if $\mathcal{M}_0$ is finite. The set of executions of $\mathcal{M}_0$ is taken to be the set of executions of $\mathcal{M}$, i.e., we define $\mathsf{Exec}(\mathcal{M})$ as the set $\mathsf{Exec}(\mathcal{M}_0)$. Given an execution $\rho = z_0 \xrightarrow{\alpha_1} z_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_m} z_m$ of $\mathcal{M}$, the trace of ρ is $\mathsf{tr}(\rho) = \mathsf{obs}(z_0)\alpha_1\mathsf{obs}(z_1)\alpha_2 \cdots \alpha_m\mathsf{obs}(z_m)$. For a POMDP $\mathcal{M}$ and a sequence $\overline{o} \in \mathcal{O} \cdot (\mathsf{Act} \cdot \mathcal{O})^*$, the probability of $\overline{o}$ in $\mathcal{M}$, written $\mathsf{prob}_{\mathcal{M}}(\overline{o})$, is the sum of the measures of executions in $\mathsf{Exec}(\mathcal{M})$ with trace $\overline{o}$. Given two

[5] The operations considered are pairing, hashing, encryption and decryption.

POMDPs $\mathcal{M}_0$ and $\mathcal{M}_1$ with the same set of actions Act and the same set of observations $\mathcal{O}$, we say that $\mathcal{M}_0$ and $\mathcal{M}_1$ are *distinguishable* if there exists $\overline{o} \in \mathcal{O} \cdot (\mathsf{Act} \cdot \mathcal{O})^*$ such that $\mathsf{prob}_{\mathcal{M}_0}(\overline{o}) \neq \mathsf{prob}_{\mathcal{M}_1}(\overline{o})$. If $\mathcal{M}_0$ and $\mathcal{M}_1$ cannot be distinguished, they are said to be *indistinguishable*. We write $\mathcal{M}_0 \approx \mathcal{M}_1$ if $\mathcal{M}_0$ and $\mathcal{M}_1$ are indistinguishable. As is the case in [53, 18], indistinguishability can also be defined through a notion of an adversary. Our formulation is equivalent, even when the adversary is allowed to toss coins [18].

Probabilistic finite automata (PFAs). A PFA is like a finite-state deterministic automaton except that the transition function from a state on a given input is described as a probability distribution. Formally, a PFA A is a tuple $(Q, \Sigma, q_s, \Delta, F)$ where Q is a finite set of states, Σ is a finite input alphabet, $q_s \in Q$ is the initial state, $\Delta : Q \times \Sigma \to \mathsf{Dist}(Q)$ is the transition relation and $F \subseteq Q$ is the set of accepting states. A run ρ of A on an input word $u \in \Sigma^* = a_1 a_2 \cdots a_m$ is a sequence $q_0 q_1 \cdots q_m \in Q^*$ such that $q_0 = q_s$ and $q_i \in \mathsf{support}(\Delta(q_{i-1}, a_i))$ for each $1 \leq i \leq m$. For the run ρ on word u, its measure, denoted $\mathsf{prob}_{\mathsf{A},u}(\rho)$, is $\prod_{i=1}^{m} \Delta(q_{i-1}, a_i)(q_i)$. The run ρ is called *accepting* if $q_m \in F$. The probability of accepting a word $u \in \Sigma$, written $\mathsf{prob}_{\mathsf{A}}(u)$, is the sum of the measures of the accepting runs on u. Two PFAs A_0 and A_1 with the same input alphabet Σ are said to be equivalent, denoted $\mathsf{A}_0 \equiv \mathsf{A}_1$, if $\mathsf{prob}_{\mathsf{A}_0}(u) = \mathsf{prob}_{\mathsf{A}_1}(u)$ for all input words $u \in \Sigma^*$.

3 POMDP indistinguishability

In this section, we study the underlying semantic objects of randomized security protocols, POMDPs. The techniques we develop for analyzing POMDPs provide the foundation for the indistinguishability algorithms we implement in the SPAN protocol analysis tool. Our first result shows that indistinguishability of finite POMDPs is decidable in polynomial time by a reduction to PFA equivalence, which is known to be decidable in polynomial time [57, 31].

Proposition 1. *Indistinguishability of finite POMDPs is in* **P**.

Proof (sketch). Consider two POMDPs $\mathcal{M}_i = (Z_i, z_s^i, \mathsf{Act}, \Delta_i, \mathcal{O}, \mathsf{obs}_i)$ for $i \in \{0, 1\}$ with the same set of actions Act and the set of observations $\mathcal{O}$. We shall construct PFAs A_0 and A_1 such that $\mathcal{M}_0 \approx \mathcal{M}_1$ iff $\mathsf{A}_0 \equiv \mathsf{A}_1$ as follows. For $i \in \{0, 1\}$, let "bad_i" be a new state and define the PFA $\mathsf{A}_i = (Q_i, \Sigma, q_s^i, \Delta_i', F_i)$ where $Q_i = Z_i \cup \{\mathsf{bad}_i\}$, $\Sigma = \mathsf{Act} \times \mathcal{O}$, $q_s^i = z_s^i$, $F_i = Z_i$ and Δ_i' is defined as follows.

$$\Delta_i'(q, (\alpha, o))(q') = \begin{cases} \Delta_i(q, \alpha)(q') & \text{if } q, q' \in Z_i \text{ and } \mathsf{obs}(q) = o \\ 1 & \text{if } q \in Z_i,\ \mathsf{obs}(q) \neq o \text{ and } q' = \mathsf{bad}_i \\ 1 & \text{if } q, q' = \mathsf{bad}_i \\ 0 & \text{otherwise} \end{cases}.$$

Let $u = (\alpha_1, o_0)\dots(\alpha_k, o_{k-1})$ be a non-empty word on Σ. For the word u, let $\overline{o}_u$ be the trace $o_0\alpha_1 o_1\alpha_2\cdots\alpha_{k-1}o_{k-1}$. The proposition follows immediately from the observation that $\mathsf{prob}_{\mathsf{A}_i}(u) = \mathsf{prob}_{\mathcal{M}_i}(\overline{o}_u)$. □

An MDP $\mathcal{M} = (Z, z_s, \mathsf{Act}, \Delta)$ is said to be acyclic if there is a set of absorbing states $Z_{\mathsf{abs}} \subseteq Z$ such that for all $\alpha \in \mathsf{Act}$ and $z \in Z_{\mathsf{abs}}$, $\Delta(z,\alpha)(z) = 1$ and for all $\rho = z_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_m} z_m \in \mathsf{Exec}(\mathcal{M})$ if $z_i = z_j$ for $i \neq j$ then $z_i \in Z_{\mathsf{abs}}$. Intuitively, acyclic MDPs are MDPs that have a set of "final" absorbing states and the only cycles in the underlying graph are self-loops on these states. A POMDP $\mathcal{M} = (Z, z_s, \mathsf{Act}, \Delta, \mathcal{O}, \mathsf{obs})$ is acyclic if the MDP $\mathcal{M}_0 = (Z, z_s, \mathsf{Act}, \Delta)$ is acyclic. We have the following result, which can be shown from the **P**-hardness of the PFA equivalence problem [45].

Proposition 2. *Indistinguishability of finite acyclic POMDPs is* **P***-hard. Hence Indistinguishability of finite POMDPs is* **P***-complete.*

Thanks to Proposition 1, we can check indistinguishability for finite POMDPs by reducing it to PFA equivalence. We now present a new algorithm for indistinguishability of finite acyclic POMDPs. A well-known POMDP analysis technique is to translate a POMDP $\mathcal{M}$ into a fully observable belief MDP $\mathcal{B}(\mathcal{M})$ that emulates it. One can then analyze $\mathcal{B}(\mathcal{M})$ to infer properties of $\mathcal{M}$. The states of $\mathcal{B}(\mathcal{M})$ are probability distributions over the states of $\mathcal{M}$. Further, given a state $b \in \mathcal{B}(\mathcal{M})$, if states z_1, z_2 of $\mathcal{M}$ are such that $b(z_1), b(z_2)$ are non-zero then z_1 and z_2 must have the same observation. Hence, by abuse of notation, we can define $\mathsf{obs}(b)$ to be $\mathsf{obs}(z)$ if $b(z) \neq 0$. Intuitively, an execution $\rho = b_0 \xrightarrow{\alpha_1} b_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_m} b_m$ of $\mathcal{B}(\mathcal{M})$ corresponds to the set of all executions ρ' of $\mathcal{M}$ such that $\mathsf{tr}(\rho') = \mathsf{obs}(b_0)\alpha_1\mathsf{obs}(b_1)\alpha_2\cdots\alpha_m\mathsf{obs}(b_m)$. The measure of execution ρ in $\mathcal{B}(\mathcal{M})$ is exactly $\mathsf{prob}_{\mathcal{M}}(\mathsf{obs}(b_0)\alpha_1\mathsf{obs}(b_1)\alpha_2\cdots\alpha_m\mathsf{obs}(b_m))$.

The initial state of $\mathcal{B}(\mathcal{M})$ is the distribution that assigns 1 to the initial state of $\mathcal{M}$. Intuitively, on a given state $b \in \mathsf{Dist}(\mathcal{M})$ and an action α, there is at most one successor state $b^{\alpha,o}$ for each observation o. The probability of transitioning from b to $b^{\alpha,o}$ is the probability that o is observed given that the distribution on the states of $\mathcal{M}$ is b and action α is performed; $b^{\alpha,o}(z)$ is the conditional probability that the actual state of the POMDP is z. The formal definition follows.

Definition 1. *Let $\mathcal{M} = (Z, z_s, \mathsf{Act}, \Delta, \mathcal{O}, \mathsf{obs})$ be a POMDP. The belief MDP of $\mathcal{M}$, denoted $\mathcal{B}(\mathcal{M})$, is the tuple $(\mathsf{Dist}(Z), \delta_{z_s}, \mathsf{Act}, \Delta^{\mathcal{B}})$ where $\Delta^{\mathcal{B}}$ is defined as follows. For $b \in \mathsf{Dist}(Z)$, action $\alpha \in \mathsf{Act}$ and $o \in \mathcal{O}$, let*

$$p_{b,\alpha,o} = \sum_{z \in Z} b(z) \cdot \left(\sum_{z' \in Z \wedge \mathsf{obs}(z') = o} \Delta(z,\alpha)(z') \right).$$

$\Delta^{\mathcal{B}}(b,\alpha)$ is the unique distribution such that for each $o \in \mathcal{O}$, if $p_{b,\alpha,o} \neq 0$ then $\Delta^{\mathcal{B}}(b,\alpha)(b^{\alpha,o}) = p_{b,\alpha,o}$ where for all $z' \in Z$,

$$b^{\alpha,o}(z') = \begin{cases} \frac{\sum_{z \in Z} b(z) \cdot \Delta(z,\alpha)(z')}{p_{b,\alpha,o}} & \text{if } \mathsf{obs}(z') = o \\ 0 & \text{otherwise} \end{cases}.$$

Let $\mathcal{M}_i = (Z_i, z_s^i, \mathsf{Act}, \Delta_i, \mathcal{O}, \mathsf{obs}_i)$ for $i \in \{0,1\}$ be POMDPs with the same set of actions and observations. In [14] the authors show that $\mathcal{M}_0$ and $\mathcal{M}_1$ are indistinguishable if and only if the beliefs $\delta_{z_s^0}$ and $\delta_{z_s^1}$ are *strongly belief bisimilar*. Strong belief bisimilarity coincides with the notion of bisimilarity of labeled MDPs: a pair of states $(b_0, b_1) \in \mathsf{Dist}(Z_0) \times \mathsf{Dist}(Z_1)$ is said to be strongly belief bisimilar if (i) $\mathsf{obs}(b_0) = \mathsf{obs}(b_1)$, (ii) for all $\alpha \in \mathsf{Act}, o \in \mathcal{O}$, $p_{b_0,\alpha,o} = p_{b_1,\alpha,o}$ and (iii) the pair $(b_0^{\alpha,o}, b_1^{\alpha,o})$ is strongly belief bisimilar if $p_{b_0,\alpha,o} = p_{b_1,\alpha,o} > 0$. Observe that, in general, belief MDPs are defined over an infinite state space. It is easy to see that, for a finite acyclic POMDP $\mathcal{M}$, $\mathcal{B}(\mathcal{M})$ is acyclic and has a finite number of reachable belief states. Let $\mathcal{M}_0$ and $\mathcal{M}_1$ be as above and assume further that $\mathcal{M}_0, \mathcal{M}_1$ are finite and acyclic with absorbing states $Z_{\mathsf{abs}} \subseteq Z_0 \cup Z_1$. As a consequence of the result from [14] and the observations above, we can determine if two states $(b_0, b_1) \in \mathsf{Dist}(Z_0) \times \mathsf{Dist}(Z_1)$ are strongly belief bisimilar using the on-the-fly procedure from Algorithm 1.

Algorithm 1 On-the-fly bisimulation for finite acyclic POMDPs

```
1: function BISIMILAR(beliefState b_0, beliefState b_1)
2:     if obs(b_0) ≠ obs(b_1) then return false
3:     if support(b_0) ∪ support(b_1) ⊆ Z_abs then return true
4:     for α ∈ Act do
5:         for o ∈ O do
6:             if p_{b_0,α,o} ≠ p_{b_1,α,o} then return false
7:             if p_{b_0,α,o} > 0 and !BISIMILAR(b_0^{α,o}, b_1^{α,o}) then return false
8:     return true
```

4 Randomized security protocols

We now present our core process calculus for modeling security protocols with coin tosses. The calculus closely resembles the ones from [10, 53]. First proposed in [39], it extends the applied π-calculus [5] by the inclusion of a new operator for probabilistic choice. As in the applied π-calculus, the calculus assumes that messages are terms in a first-order signature identified up-to an equational theory.

4.1 Terms, equational theories and frames

A signature $\mathcal{F}$ contains a *finite* set of function symbols, each with an associated arity. We assume $\mathcal{F}$ contains two special disjoint sets, $\mathcal{N}_{\mathsf{pub}}$ and $\mathcal{N}_{\mathsf{priv}}$, of 0-ary symbols.[6] The elements of $\mathcal{N}_{\mathsf{pub}}$ are called *public names* and represent public nonces that can be used by the Dolev-Yao adversary. The elements of $\mathcal{N}_{\mathsf{priv}}$ are

[6] As we assume $\mathcal{F}$ is finite, we allow only a fixed number of public nonces are available to the adversary.

called *names* and represent secret nonces and secret keys. We also assume a set of variables that are partitioned into two disjoint sets $\mathcal{X}$ and $\mathcal{X}_w$. The variables in $\mathcal{X}$ are called *protocol variables* and are used as placeholders for messages input by protocol participants. The variables in $\mathcal{X}_w$ are called *frame variables* and are used to point to messages received by the Dolev-Yao adversary. Terms are built by the application of function symbols to variables and terms in the standard way. Given a signature $\mathcal{F}$ and $\mathcal{Y} \subseteq \mathcal{X} \cup \mathcal{X}_w$, we use $\mathcal{T}(\mathcal{F}, \mathcal{Y})$ to denote the set of terms built over $\mathcal{F}$ and $\mathcal{Y}$. The set of variables occurring in a term u is denoted by $\mathsf{vars}(u)$. A ground term is a term that contains no free variables.

A substitution σ is a partial function with a finite domain that maps variables to terms. $\mathsf{dom}(\sigma)$ will denote the domain and $\mathsf{ran}(\sigma)$ will denote the range. For a substitution σ with $\mathsf{dom}(\sigma) = \{x_1, \ldots, x_k\}$, we denote σ as $\{x_1 \mapsto \sigma(x_1), \ldots, x_k \mapsto \sigma(x_k)\}$. A substitution σ is said to be ground if every term in $\mathsf{ran}(\sigma)$ is ground and a substitution with an empty domain will be denoted as $\emptyset$. Substitutions can be aplied to terms in the usual way and we write $u\sigma$ for the term obtained by applying the substitution σ to the term u.

Our process algebra is parameterized by an equational theory $(\mathcal{F}, E)$, where E is a set of $\mathcal{F}$-Equations. By an $\mathcal{F}$-Equation, we mean a pair $u = v$ where $u, v \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}_{\mathsf{priv}}, \mathcal{X})$ are terms that do not contain private names. We will assume that the equations of $(\mathcal{F}, E)$ can be oriented to produce a convergent rewrite system. Two terms u and v are said to be equal with respect to an equational theory $(\mathcal{F}, E)$, denoted $u =_E v$, if $E \vdash u = v$ in the first order theory of equality. We often identify an equational theory $(\mathcal{F}, E)$ by E when the signature is clear from the context.

In the calculus, all communication is mediated through an adversary: all outputs first go to an adversary and all inputs are provided by the adversary. Hence, processes are executed in an environment that consists of a frame $\varphi : \mathcal{X}_w \to \mathcal{T}(\mathcal{F}, \emptyset)$ and a ground substitution $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \emptyset)$. Intuitively, φ represents the sequence of messages an adversary has received from protocol participants and σ records the binding of the protocol variables to actual input messages. An adversary is limited to sending only those messages that it can deduce from the messages it has received thus far. Formally, a term $u \in \mathcal{T}(\mathcal{F}, \emptyset)$ is *deducible* from a frame φ with recipe $r \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}_{\mathsf{priv}}, \mathsf{dom}(\varphi))$ in equational theory E, denoted $\varphi \vdash^r_E u$, if $r\varphi =_E u$. We will often omit r and E and write $\varphi \vdash u$ if they are clear from the context.

We now recall an equivalence on frames, called *static equivalence* [5]. Intuitively, two frames are statically equivalent if the adversary cannot distinguish them by performing tests. The tests consists of checking whether two recipes deduce the same term. Formally, two frames φ_1 and φ_2 are said to be statically equivalent in equational theory E, denoted $\varphi_1 \equiv_E \varphi_2$, if $\mathsf{dom}(\varphi_1) = \mathsf{dom}(\varphi_2)$ and for all $r_1, r_2 \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}_{\mathsf{priv}}, \mathcal{X}_w)$ we have $r_1\varphi_1 =_E r_2\varphi_1$ iff $r_1\varphi_2 =_E r_2\varphi_2$.

4.2 Process syntax

Processes in our calculus are the parallel composition of roles. Intuitively, a role models a single actor in a single session of the protocol. Syntactically, a role is derived from the grammar:

$$R \ ::= \ 0 \ \big| \ \mathtt{in}(x)^\ell \ \big| \ \mathtt{out}(u_0 \cdot R +_p u_1 \cdot R)^\ell \ \big| \ \mathsf{ite}([c_1 \wedge \ldots \wedge c_k], R, R)^\ell \ \big| \ (R \cdot R)$$

where p is a rational number in the unit interval $[0,1]$, $\ell \in \mathbb{N}$, $x \in \mathcal{X}$, $u_0, u_1 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and c_i is $u_i = v_i$ with $u_i, v_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ for all $i \in \{1, \ldots, k\}$. The constructs $\mathtt{in}(x)^\ell$, $\mathtt{out}(u_0 \cdot R +_p u_1 \cdot R)^\ell$ and $\mathsf{ite}([c_1 \wedge \ldots \wedge c_k], R, R)^\ell$ are said to be labeled operations and $\ell \in \mathbb{N}$ is said to be their label. The role 0 does nothing. The role $\mathtt{in}(x)^\ell$ reads a term u from the public channel and binds it to x. The role $\mathtt{out}(u_0 \cdot R +_p u_1 \cdot R')^\ell$ outputs the term u_0 on the public channel and becomes R with probability p and it outputs the term u_1 and becomes R' with probability $1-p$. A test $[c_1 \wedge \ldots \wedge c_k]$ is said to pass if for all $1 \leq i \leq k$, the equality c_i holds. The conditional role $\mathsf{ite}([c_1 \wedge \ldots \wedge c_k], R, R')^\ell$ becomes R if $[c_1 \wedge \ldots \wedge c_k]$ passes and otherwise it becomes R'. The role $R \cdot R'$ is the sequential composition of role R followed by role R'. The set of variables of a role R is the set of variables occurring in R. The construct $\mathtt{in}(x)^\ell \cdot R$ binds variable x in R. The set of free and bound variables in a role can be defined in the standard way. We will assume that the set of free variables and bound variables of a role are disjoint and that a bound variable is bound only once in a role. A role R is said to be *well-formed* if every labeled operation occurring in R has the same label ℓ; the label ℓ is said to be the label of the well-formed role R.

A process is the parallel composition of a finite set of roles $R_1, \ldots, R_n$, denoted $R_1 \mid \ldots \mid R_n$. We will use P and Q to denote processes. A process $R_1 \mid \ldots \mid R_n$ is said to be well-formed if each role is well-formed, the sets of variables of R_i and R_j are disjoint for $i \neq j$, and the label of role R_i and label of role R_j are different for $i \neq j$. For the remainder of this paper, processes are assumed to be well-formed. The set of free (resp. bound) variables of P is the union of the sets of free (resp. bound) variables of its roles. P is said to be ground if the set of its free variables is empty. We shall omit labels when they are not relevant in a particular context.

Example 2. We model the electronic voting protocol from Example 1 in our formalism. The protocol is built over the equational theory with signature $\mathcal{F} = \{\mathsf{sk}/1,\ \mathsf{pk}/1,\ \mathsf{aenc}/3,\ \mathsf{adec}/2,\ \mathsf{pair}/2,\ \mathsf{fst}/1,\ \mathsf{snd}/1\}$ and the equations

$$E = \{\ \mathsf{adec}(\mathsf{aenc}(m, r, \mathsf{pk}(k)), \mathsf{sk}(k)) = m,\\ \mathsf{fst}(\mathsf{pair}(m_1, m_2)) = m_1,\ \mathsf{snd}(\mathsf{pair}(m_1, m_2)) = m_2\}.$$

The function sk (resp. pk) is used to generate a secret (resp. public) key from a nonce. For generation of their pubic key pairs, Alice, Bob and the election authority hold private names k_A, k_B and k_{EA}, respectively. The candidates will be modeled using public names c_0 and c_1 and the tokens will be modeled using private names t_A and t_B. Additionally, we will write y_i and r_i for $i \in \mathbb{N}$ to denote fresh input variables and private names, respectively. The roles of Alice, Bob and the election authority are as follows.

$$
\begin{aligned}
A(c_A) &:= \mathtt{in}(y_0) \cdot \mathtt{out}(\mathsf{aenc}(\mathsf{pair}(\mathsf{adec}(y_0, \mathsf{sk}(k_A)), c_A), r_0, \mathsf{pk}(k_{EA}))) \\
B(c_B) &:= \mathtt{in}(y_1) \cdot \mathtt{out}(\mathsf{aenc}(\mathsf{pair}(\mathsf{adec}(y_1, \mathsf{sk}(k_B)), c_B), r_1, \mathsf{pk}(k_{EA}))) \\
EA &:= \mathtt{out}(\mathsf{aenc}(t_A, r_2, \mathsf{pk}(k_A))) \cdot \mathtt{out}(\mathsf{aenc}(t_B, r_3, \mathsf{pk}(k_B))) \cdot \mathtt{in}(y_3) \cdot \mathtt{in}(y_4) \cdot \\
&\quad \mathsf{ite}([\mathsf{fst}(\mathsf{adec}(y_3, sk(k_{EA}))) = t_A \wedge \mathsf{fst}(\mathsf{adec}(y_4, sk(k_{EA}))) = t_B], \\
&\quad \mathtt{out}(\mathsf{pair}(\mathsf{snd}(\mathsf{adec}(y_3, sk(k_{EA}))), \mathsf{snd}(\mathsf{adec}(y_4, sk(k_{EA}))))+_{\frac{1}{2}} \\
&\quad\quad \mathsf{pair}(\mathsf{snd}(\mathsf{adec}(y_4, sk(k_{EA}))), \mathsf{snd}(\mathsf{adec}(y_3, sk(k_{EA}))))), 0)
\end{aligned}
$$

In roles above, we write $\mathtt{out}(u_0)$ as shorthand for $\mathtt{out}(u_0 \cdot 0 +_1 u_0 \cdot 0)$. The entire protocol is $\mathsf{evote}(c_A, c_B) = A(c_A) \mid B(c_B) \mid EA$.

4.3 Process semantics

An extended process is a 3-tuple (P, φ, σ) where P is a process, φ is a frame and σ is a ground substitution whose domain contains the free variables of P. We will write $\mathcal{E}$ to denote the set of all extended processes. Semantically, a ground process P with n roles is a POMDP $[\![P]\!] = (Z, z_s, \mathsf{Act}, \Delta, \mathcal{O}, \mathsf{obs})$, where $Z = \mathcal{E} \cup \{\mathsf{error}\}$, z_s is $(P, \emptyset, \emptyset)$, $\mathsf{Act} = (\mathcal{T}(\mathcal{F} \setminus \mathcal{N}_{\mathsf{priv}}, \mathcal{X}_w) \cup \{\tau, \} \times \{1, \ldots, n\})$, Δ is a function that maps an extended process and an action to a distribution on $\mathcal{E}$, $\mathcal{O}$ is the set of equivalence classes on frames over the static equivalence relation $\equiv_E$ and obs is as follows. Let $[\varphi]$ denote the equivalence class of φ with respect to $\equiv_E$. Define obs to be the function such that for any extended process $\eta = (P, \varphi, \sigma)$, $\mathsf{obs}(\eta) = [\varphi]$. We now give some additional notation needed for the definition of Δ. Given a measure μ on $\mathcal{E}$ and role R we define $\mu \cdot R$ to be the distribution μ_1 on $\mathcal{E}$ such that $\mu_1(P', \varphi, \sigma) = \mu(P, \varphi, \sigma)$ if $\mu(P, \varphi, \sigma) > 0$ and P' is $P \cdot R$ and 0 otherwise. Given a measure μ on $\mathcal{E}$ and a process Q, we define $\mu \mid Q$ to be the distribution μ_1 on $\mathcal{E}$ such that $\mu_1(P', \varphi, \sigma) = \mu(P, \varphi, \sigma)$ if $\mu(P, \varphi, \sigma) > 0$ and P' is $P \mid Q$ and 0 otherwise. The distribution $Q \mid \mu$ is defined analogously. For distributions μ_1, μ_2 over $\mathcal{E}$ and a rational number $p \in [0, 1]$, the convex combination $\mu_1 +_p^{\mathcal{E}} \mu_2$ is the distribution μ on $\mathcal{E}$ such that $\mu(\eta) = p \cdot \mu_1(\eta) + (1 - p) \cdot \mu_2(\eta)$ for all $\eta \in \mathcal{E}$. The definition of Δ is given in Figure 1, where we write $(P, \varphi, \sigma) \xrightarrow{\alpha} \mu$ if $\Delta((P, \varphi, \sigma), \alpha) = \mu$. If $\Delta((P, \varphi, \sigma), \alpha)$ is undefined in Figure 1 then $\Delta((P, \varphi, \sigma), \alpha) = \delta_{\mathsf{error}}$. Note that Δ is well-defined, as roles are deterministic.

4.4 Indistinguishability in randomized cryptographic protocols

Protocols P and P' are said to indistinguishable if $[\![P]\!] \approx [\![P']\!]$. Many interesting properties of randomized security protocols can be specified using indistinguishability. For example, consider the simple electronic voting protocol from Example 2. We say that the protocol satisfies the vote privacy property if $\mathsf{evote}(c_0, c_1)$ and $\mathsf{evote}(c_1, c_0)$ are indistinguishable.

In the remainder of this section, we study the problem of deciding when two protocols are indistinguishable by a bounded Dolev-Yao adversary. We restrict our attention to indistinguishability of protocols over subterm convergent equational theories [4]. Before presenting our results, we give some relevant definitions. $(\mathcal{F}, E)$ is said to be *subterm convergent* if for every equation $u = v \in E$

$$\frac{r \in \mathcal{T}(\mathcal{F} \setminus \mathcal{N}_{\mathsf{priv}}, \mathcal{X}_w) \quad \varphi \vdash^r u \quad x \notin \mathsf{dom}(\sigma)}{(\mathtt{in}(x)^\ell, \varphi, \sigma) \xrightarrow{(r,\ell)} \delta_{(0,\varphi,\sigma\cup\{x\mapsto u\})}} \text{ IN}$$

$$\frac{i = |\mathsf{dom}(\varphi)| + 1 \quad \varphi_j = \varphi \cup \{w_{(i,\ell)} \mapsto u_j\sigma\} \text{ for } j \in \{0,1\}}{(\mathtt{out}(u_0 \cdot R_0 +_p u_1 \cdot R_1)^\ell, \varphi, \sigma) \xrightarrow{(\tau,\ell)} \delta_{(R_0,\varphi_0,\sigma)} +_p^{\mathcal{E}} \delta_{(R_1,\varphi_1,\sigma)}} \text{ OUT}$$

$$\frac{\forall i \in \{1,\ldots,k\},\ c_i \text{ is } u_i = v_i \text{ and } u_i\sigma =_E v_i\sigma}{(\mathsf{ite}([c_1 \wedge \ldots \wedge c_k], R, R')^\ell, \varphi, \sigma) \xrightarrow{(\tau,\ell)} \delta_{(R,\varphi,\sigma)}} \text{ COND}_{\mathsf{IF}}$$

$$\frac{\exists i \in \{1,\ldots,k\},\ c_i \text{ is } u_i = v_i \text{ and } u_i\sigma \neq_E v_i\sigma}{(\mathsf{ite}([c_1 \wedge \ldots \wedge c_k], R, R')^\ell, \varphi, \sigma) \xrightarrow{(\tau,\ell)} \delta_{(R',\varphi,\sigma)}} \text{ COND}_{\mathsf{ELSE}}$$

$$\frac{R \neq 0 \quad (R,\varphi,\sigma) \xrightarrow{\alpha} \mu}{(R \cdot R', \varphi, \sigma) \xrightarrow{\alpha} \mu \cdot R'} \text{ SEQ} \qquad \frac{(R,\varphi,\sigma) \xrightarrow{\alpha} \mu}{(0 \cdot R, \varphi, \sigma) \xrightarrow{\alpha} \mu} \text{ NULL}$$

$$\frac{(Q,\varphi,\sigma) \xrightarrow{\alpha} \mu}{(Q \mid Q', \varphi, \sigma) \xrightarrow{\alpha} \mu \mid Q'} \text{ PAR}_{\text{L}} \qquad \frac{(Q',\varphi,\sigma) \xrightarrow{\alpha} \mu}{(Q \mid Q', \varphi, \sigma) \xrightarrow{\alpha} Q \mid \mu} \text{ PAR}_{\text{R}}$$

Fig. 1: Process semantics

oriented as a rewrite rule $u \to v$, either v is a proper subterm of u or v is a public name. A term u can be represented as a directed acyclic graph (dag), denoted $\mathsf{dag}(u)$ [4, 51]. Every node in $\mathsf{dag}(u)$ is a function symbol, name or a variable. Nodes labeled by names and variables have out-degree 0. A node labeled with a function symbol f has out-degree equal to the arity of f where outgoing edges of the node are labeled from 1 to the arity of f. Every node of $\mathsf{dag}(u)$ represents a unique sub-term of u. The depth of a term u, denoted $\mathsf{depth}(u)$, is the length of the longest simple path from the root in $\mathsf{dag}(u)$. Given an action α, $\mathsf{depth}(\alpha) = 0$ if $\alpha = (\tau, j)$ and $\mathsf{depth}(\alpha) = m$ if $\alpha = (r, j)$ and $\mathsf{depth}(r) = m$.

Let P be a protocol such that $[\![P]\!] = (Z, z_s, \mathsf{Act}, \Delta, \mathcal{O}, \mathsf{obs})$. Define $[\![P]\!]_d$ to be the POMDP $(Z, z_s, \mathsf{Act}_d, \Delta, \mathcal{O}, \mathsf{obs})$ where $\mathsf{Act}_d \subseteq \mathsf{Act}$ is such that every $\alpha \in \mathsf{Act}$ has $\mathsf{depth}(\alpha) \leq d$. For a constant $d \in \mathbb{N}$, we define $\mathsf{InDist}(d)$ to be the decision problem that, given a subterm convergent theory $(\mathcal{F}, E)$ and protocols P and P' over $(\mathcal{F}, E)$, determines if $[\![P]\!]_d$ and $[\![P']\!]_d$ are indistinguishable. We assume that the arity of the function symbols in $\mathcal{F}$ is given in unary. We have the following.

Theorem 1. *For any constant $d \in \mathbb{N}$, $\mathsf{InDist}(d)$ is in* **PSPACE**.

We now show $\mathsf{InDist}(d)$ is both **NP**-hard and **coNP**-hard by showing a reduction from $\#\mathsf{SAT}_{\mathsf{D}}$ to $\mathsf{InDist}(d)$. $\#\mathsf{SAT}_{\mathsf{D}}$ is the decision problem that, given a 3CNF formula ϕ and a constant $k \in \mathbb{N}$, checks if the number of satisfying assignments of ϕ is equal to k.

Theorem 2. *There is a* $d_0 \in \mathbb{N}$ *such that* $\#\mathsf{SAT}_\mathsf{D}$ *reduces to* $\mathsf{InDist}(d)$ *in logspace for every* $d > d_0$*. Thus,* $\mathsf{InDist}(d)$ *is* ***NP****-hard and* ***coNP****-hard for every* $d > d_0$*.*

5 Implementation and Evaluation

Using (the proof of) Proposition 1, we can solve the indistinguishability problem for randomized security protocols as follows. For protocols P, P', translate $[\![P]\!], [\![P']\!]$ into PFAs A, A' and determine if $\mathsf{A} \equiv \mathsf{A}'$ using the linear programming algorithm from [31]. We will henceforth refer to this approach as the PFA algorithm and the approach from Algorithm 1 as the OTF algorithm. We have implemented both the PFA and OTF algorithms as part of Stochastic Protocol ANalayzer (Span), which is a Java based tool for analyzing randomized security protocols. The tool is avaliable for download at [1]. The main engine of Span translates a protocol into a POMDP, belief MDP or PFA by exploring all protocol executions and grouping equivalent states using an engine, Kiss [4] or Akiss [16], for static equivalence. Kiss is guaranteed to terminate for subterm convergent theories and Akiss provides support for XOR while considering a slighly larger class of equational theories called *optimally reducing*. Operations from rewriting logic are provided by queries to Maude [24] and support for arbitrary precision numbers is given by Apfloat [2]. Our experiments were conducted on an Intel core i7 dual quad core processor at 2.67GHz with 12Gb of RAM. The host operating system was 64 bit Ubuntu 16.04.3 LTS.

Our comparison of the PFA and OTF algorithms began by examining how each approach scaled on a variety of examples (detailed at the end of this section). The results of the analysis are given in Figure 2. For each example, we consider a fixed recipe depth and report the running times for 2 parties as well as the maximum number of parties for which one of the algorithms terminates within the timeout bound of 60 minutes. On small examples for which the protocols were indistinguishable, we found that the OTF and PFA algorithms were roughly equivalent. On large examples where the protocols were indistinguishable, such as the 3 ballot protocol, the PFA algorithm did not scale as well as the OTF algorithm. In particular, an out-of-memory exception often occurred during construction of the automata or the linear programming constraints. On examples for which the protocols were distinguishable, the OTF algorithm demonstrated a significant advantage. This was a result of the fact that the OTF approach analyzed the model as it was constructed. If at any point during model construction the bisimulation relation was determined not to hold, model construction was halted. By contrast, the PFA algorithm required the entire model to be constructed and stored before any analysis could take place.

In addition to stress-testing the tool, we also examined how each algorithm performed under various parameters of the mix-network example. The results are given in Figure 3, where we examine how running times are affected by scaling the number of protocol participants and the recipe depth. Our results coincided with the observations from above. One interesting observation is that the number of beliefs explored on the 5 party example was identical for recipe

1	2	3	4	5	6	7	8	9	10
Protocol	Parties	Depth	Equiv	Time (s)				States	Beliefs
				PFA		OTF			
				Kiss	AKiSs	Kiss	Akiss		
DC-net	2	10	true	n/s	5.5	n/s	4	58	24
DC-net	3	10	true	n/s	OOM	n/s	3013	n/a	286
mix-net	2	10	false	TO	TO	.3	.4	n/a	7
mix-net	5	10	false	OOM	OOM	582	1586	n/a	79654
Evote	2	10	true	1	1	.5	1	34	33
Evote	8	10	true	105	105	131	124	94	93
3 Ballot	2	10	true	n/s	OOM	n/s	1444	n/a	408

Fig. 2. Experimental Results: Columns 1 and 2 describe the example being analyzed. Column 3 gives the maximum recipe depth and column 4 indicates when the example protocols were indistinguishable. Columns 5-8 give the running time (in seconds) for the respective algorithms and static equivalence engines. We report OOM for an out of memory exception and TO for a timeout - which occurs if no solution is generated in 60 minutes. Column 9 gives the number of states in the protocol's POMDP and Column 10 gives the number of belief states explored in the OTF algorithm. When information could not be determined due to a failure of the tool to terminate, we report n/a. For protocols using equational theories that were not subterm convergent, we write n/s (not supported) for the Kiss engine.

depth 4 and recipe depth 10. The reason is that, for a given protocol input step, Span generates a minimal set of recipes. This is in the sense that if recipes r_0, r_1 are generated at an input step with frame φ, then $r_0\varphi \neq_E r_1\varphi$. For the given number of public names available to the protocol, changing the recipe depth from 4 to 10 did not alter the number of unique terms that could be constructed by the attacker. We conclude this section by describing our benchmark examples, which are available at [3]. Evote is the simple electronic voting protocol derived from Example 2 and the the DC-net, mix-net and 3 ballot protocols are described below.

Dinning cryptographers networks. In a simple DC-net protocol [38], two parties Alice and Bob want to anonymously publish two confidential bits m_A and m_B, respectively. To achieve this, Alice and Bob agree on three private random bits b_0, b_1 and b_2 and output a pair of messages according to the following scheme. In our modeling the protocol, the private bits are generated by a trusted third party who communicates them with Alice and Bob using symmetric encryption.

$$
\begin{array}{ll}
\text{If } b_0 = 0 & \text{Alice: } M_{A,0} = b_1 \oplus m_A,\ M_{A,1} = b_2 \\
& \text{Bob: } M_{B,0} = b_1,\ M_{B,1} = b_2 \oplus m_B \\
\text{If } b_0 = 1 & \text{Alice: } M_{A,0} = b_1,\ M_{A,1} = b_2 \oplus m_A \\
& \text{Bob: } M_{B,0} = b_1 \oplus m_B,\ M_{B,1} = b_2
\end{array}
$$

From the protocol output, the messages m_A and m_B can be retrieved as $M_{A,0} \oplus M_{B,0}$ and $M_{A,1} \oplus M_{B,1}$. The party to which the messages belong, however, remains unconditionally private, provided the exchanged secrets are not revealed.

1	2	3	4	5	6	7	8	9
Parties	Depth	Equiv	Time (s)				States	Beliefs
			PFA		OTF			
			Kiss	Akiss	Kiss	Akiss		
2	1	true	.3	.3	.2	.3	15	12
3	1	true	1	1.2	.4	.9	81	50
4	1	true	47	47	2	6	2075	656
5	1	true	OOM	OOM	34	79	n/a	4032
5	2	false	OOM	OOM	13	33	n/a	1382
5	3	false	OOM	OOM	124	354	n/a	6934
5	4	false	OOM	OOM	580	1578	n/a	79654

Fig. 3. Detailed Experimental Results for Mix Networks: The columns have an identical meaning to the ones from Figure 2. We report OOM for an out of memory exception and when information could not be determined due to a failure of the tool to terminate, we report n/a.

Mix networks. A mix-network [21], is a routing protocol used to break the link between a message's sender and the message. This is achieved by routing messages through a series of proxy servers, called mixes. Each mix collects a batch of encrypted messages, privately decrypts each message and forwards the resulting messages in random order. More formally, consider a sender Alice (A) who wishes to send a message m to Bob (B) through Mix (M). Alice prepares a cipher-text of the form $\mathsf{aenc}(\mathsf{aenc}(m, n_1, \mathsf{pk}(B)), n_0, \mathsf{pk}(M))$ where aenc is asymmetric encryption, n_0, n_1 are nonces and $\mathsf{pk}(M)$, $\mathsf{pk}(B)$ are the public keys of the Mix and Bob, respectively. Upon receiving a batch of N such cipher-texts, the Mix unwraps the outer layer of encryption on each message using its secret key, randomly permutes and forwards the messages. A passive attacker, who observes all the traffic but does not otherwise modify the network, cannot (with high probability) correlate messages entering and exiting the Mix. Unfortunately,

this simple design, known as a *threshold mix*, is vulnerable to a very simple active attack. To expose Alice as the sender of the message $\mathsf{aenc}(m, n_1, \mathsf{pk}(B))$, an attacker simply forwards Alice's message along with $N-1$ dummy messages to the Mix. In this way, the attacker can distinguish which of the Mix's N output messages is not a dummy message and hence must have originated from Alice.

3-Ballot electronic voting. We have modeled and analyzed the 3-ballot voting system from [54]. To simplify the presentation of this model, we first describe the major concepts behind 3-ballot voting schemes, as originally introduced by [50]. At the polling station, each voter is given 3 ballots at random. A ballot is comprised of a list of candidates and a ballot ID. When casting a vote, a voter begins by placing exactly one mark next to each candidate on one of the three ballots chosen a random. An additional mark is then placed next to the desired candidate on one of the ballots, again chosen at random. At the completion of the procedure, at least one mark should have been placed on each ballot and two ballots should have marks corresponding to the desired candidate. Once all of the votes have been cast, ballots are collected and released to a public bulletin board. Each voter retains a copy of one of the three ballots as a receipt, which can be used to verify his/her vote was counted.

In the full protocol, a registration agent is responsible for authenticating voters and receiving ballots and ballot ids generated by a vote manager. Once a voter marks his/her set of three ballots, they are returned to the vote manager who forwards them to one of three vote repositories. The vote repositories store the ballots they receive in a random position. After all votes have been collected in the repositories, they are released to a bulletin board by a vote collector. Communication between the registration agent, vote manager, vote repositories and vote collector is encrypted using asymmetric encryption and authenticated using digital signatures. In our modeling, we assume all parties behave honestly.

6 Conclusion

In this paper, we have considered the problem of model checking indistinguishability in randomized security protocols that are executed with respect to a Dolev-Yao adversary. We have presented two different algorithms for the indistinguishability problem assuming bounded recipe sizes. The algorithms have been implemented in the SPAN protocol analysis tool, which has been used to verify some well known randomized security protocols. We propose the following as part of future work: (i) extension of the current algorithms as well the tool to the case of unbounded recipe sizes; (ii) application of the tool for checking other randomized protocols; (iii) giving tight upper and lower bounds for the indistinguishability problem for the randomized protocols.

References

1. `https://github.com/bauer-matthews/SPAN`.
2. `http://www.apfloat.org/`.
3. `https://github.com/bauer-matthews/SPAN/tree/master/src/test/resources/examples/indistinguishability`.
4. Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1-2):2–32, 2006.
5. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *ACM Sigplan Notices*, volume 36, pages 104–115. ACM, 2001.
6. Ben Adida. Helios: Web-based open-audit voting. In *USENIX Security Symposium*, volume 17, pages 335–348, 2008.
7. Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Computer Security Foundations*, pages 107–121, 2010.
8. Alessandro Armando and Luca Compagna. SAT-based model-checking for security protocols analysis. *International Journal of Information Security*, 7(1):3–32, Jan 2008.
9. David Basin, Jannik Dreier, and Ralf Sasse. Automated symbolic proofs of observational equivalence. In *Computer and Communications Security*, pages 1144–1155, 2015.
10. Matthew S. Bauer, Rohit Chadha, and Mahesh Viswanathan. Composing protocols with randomized actions. In *Principles of Security and Trust*, pages 189–210, 2016.
11. Michael Ben-Or, Oded Goldreich, Silvio Micali, and Ronald L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.
12. Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *The Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.
13. Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Task-structured probabilistic I/O automata. In *Discrete Event Systems*, 2006.
14. Pablo Samuel Castro, Prakash Panangaden, and Doina Precup. Equivalence relations in fully and partially observable Markov decision processes. In *International Joint Conference on Artificial Intelligence*, volume 9, pages 1653–1658, 2009.
15. Rohit Chadha, Vincent Cheval, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocol. *ACM Transactions on Computational Logic*, 17(4), 2016.
16. Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. In *European Symposium on Programming*, pages 108–127. Springer, 2012.
17. Rohit Chadha, A Prasad Sistla, and Mahesh Viswanathan. Model checking concurrent programs with nondeterminism and randomization. In *Foundations of Software Technology and Theoretical Computer Science*, pages 364–375, 2010.
18. Rohit Chadha, A Prasad Sistla, and Mahesh Viswanathan. Verification of randomized security protocols. In *Logic in Computer Science*, pages 1–12. IEEE, 2017.
19. Konstantinos Chatzikokolakis and Catuscia Palamidessi. Making random choices invisible to the scheduler. In *International Conference on Concurrency Theory*, pages 42–58. Springer, 2007.

20. David Chaum, Peter YA Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, pages 118–139. Springer, 2005.
21. David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
22. Ling Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. PhD thesis, Radboud University of Nijmegen, 2006.
23. Vincent Cheval. Apte: an algorithm for proving trace equivalence. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 587–592. Springer, 2014.
24. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martı-Oliet, José Meseguer, and José F Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002.
25. Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *Computer Security Foundations*, pages 266–276. IEEE, 2009.
26. Luca de Alfaro. The verification of probabilistic systems under memoryless partial-information policies is hard. Technical report, 1999.
27. Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *Computer Aided Verification*, pages 592–600. Springer, 2017.
28. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
29. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.
30. Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
31. Laurent Doyen, Thomas A Henzinger, and Jean-François Raskin. Equivalence of labeled Markov chains. *Foundations of Computer Science*, 19(03):549–563, 2008.
32. Jannik Dreier, Charles Duménil, Steve Kremer, and Ralf Sasse. Beyond subterm-convergent equational theories in automated verification of stateful protocols. In *Principles of Security and Trust*, 2017.
33. Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Computer Security*, 12(2):247–311, 2004.
34. Santiago Escobar, Catherine Meadows, and José Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design*, pages 1–50. Springer, 2009.
35. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
36. Flavio D Garcia, Peter Van Rossum, and Ana Sokolova. Probabilistic anonymity and admissible schedulers. *arXiv preprint arXiv:0706.1019*, 2007.
37. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In *Workshop on Information Hiding*, pages 137–150, 1996.
38. Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Theory and Applications of Cryptographic Techniques*, pages 456–473. Springer, 2004.
39. Jean Goubault-Larrecq, Catuscia Palamidessi, and Angelo Troina. A probabilistic applied pi–calculus. In *Asian Symposium on Programming Languages and Systems*, pages 175–190, 2007.

40. Carl A. Gunter, Sanjeev Khanna, Kaijun Tan, and Santosh S. Venkatesh. DoS protection for reliably authenticated broadcast. In *Network and Distributed System Security*, 2004.
41. Lucca Hirschi, David Baelde, and Stéphanie Delaune. A method for verifying privacy-type properties: The unbounded case. In *Security and Privacy*, pages 564–581, 2016.
42. Stefan Kiefer, Andrzej S Murawski, Joël Ouaknine, Björn Wachter, and James Worrell. Apex: An analyzer for open probabilistic programs. In *Computer Aided Verification*, pages 693–698. Springer, 2012.
43. Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *European Symposium on Programming*, pages 186–200. Springer, 2005.
44. Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification*, pages 585–591. Springer, 2011.
45. Rastislav Lenhardt. Probabilistic automata with parameters. Master's thesis, 2009.
46. Catherine Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *IEEE Journal on Selected Areas in Communications*, 21(1):44–54, 2003.
47. Catherine Meadows. Emerging issues and trends in formal methods in cryptographic protocol analysis: Twelve years later. In *Logic, Rewriting, and Concurrency*, pages 475–492. Springer, 2015.
48. Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 696–701. Springer, 2013.
49. Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
50. Ronald L Rivest. The threeballot voting system. 2006.
51. Michaël Rusinowitch and Mathieu Turuani. *Protocol insecurity with finite number of sessions is NP-complete*. PhD thesis, INRIA, 2001.
52. Peter YA Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. Prêt à voter: A voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, 2009.
53. Matthew S. Bauer, Rohit Chadha, and Mahesh Viswanathan. Modular verification of protocol equivalence in the presence of randomness. In *European Symposium on Research in Computer Security*, pages 187–205, 01 2017.
54. Altair O Santin, Regivaldo G Costa, and Carlos A Maziero. A three-ballot-based secure electronic voting system. *Security and Privacy*, 6(3):14–21, 2008.
55. Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *Computer Security Foundations*, pages 78–94, 2012.
56. Vitaly Shmatikov. Probabilistic analysis of anonymity. In *Computer Security Foundations*, pages 119–128. IEEE, 2002.
57. Wen-Guey Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.