Maximizing Entropy over Markov Processes

Fabrizio Biondi, Axel Legay, Bo Friis Nielsen, Andrzej Wasowski

Maximizing Entropy over Markov Processes

Fabrizio Biondi[a], Axel Legay[b], Bo Friis Nielsen[c], Andrzej Wąsowski[a]

[a] *IT University of Copenhagen, 7 Rued Langgaards Vej, 2300 Copenhagen-S, Denmark*
[b] *IRISA/INRIA Rennes, Campus de Beaulieu, 263 Avenue du Général Leclerc, 35042 Rennes, France*
[c] *Technical University of Denmark, Kongens Lyngby, Denmark*

Abstract

The channel capacity of a deterministic system with confidential data is an upper bound on the amount of bits of data an attacker can learn from the system. We encode all possible attacks to a system using a probabilistic specification, an Interval Markov Chain. Then the channel capacity computation reduces to finding a model of a specification with highest entropy.

Entropy maximization for probabilistic process specifications has not been studied before, even though it is well known in Bayesian inference for discrete distributions. We give a characterization of global entropy of a process as a reward function, a polynomial algorithm to verify the existence of a system maximizing entropy among those respecting a specification, a procedure for the maximization of reward functions over Interval Markov Chains and its application to synthesize an implementation maximizing entropy.

We show how to use Interval Markov Chains to model abstractions of deterministic systems with confidential data, and use the above results to compute their channel capacity. These results are a foundation for ongoing work on computing channel capacity for abstractions of programs derived from code.

1. Introduction

Context. Consider a common authentication protocol: a user is asked to enter a password. She is granted access to the system if the password corresponds to the one stored in the system. The goal of the protocol is to refuse access to users that do not know the password (the secret). The protocol should also prevent an attacker from guessing or learning the secret, by interacting with the system. We want to decide whether the protocol is secure or not against an attacker that wants to access it, but does not know the password. But what does it mean for a system to be secure?

☆ Earlier versions of this paper appeared in the 24th Nordic Workshop on Programming Theory (2012), and in the 7th International Conference on Language and Automata Theory and Applications (2013).

☆☆ The research presented in this paper has been partially supported by MT-LAB, a VKR Centre of Excellence for the Modelling of Information Technology.

Email addresses: fbio@itu.dk (Fabrizio Biondi), axel.legay@inria.fr (Axel Legay), bfn@imm.dtu.dk (Bo Friis Nielsen), wasowski@itu.dk (Andrzej Wąsowski)

One possible answer is that a system is secure if the attacker cannot authenticate in the system. This, however, cannot be guaranteed. There is a non-zero probability that an attacker is lucky and simply guesses the password. We could require that the attacker cannot learn the password by interacting with the system. However, a password is not an atomic object: would we consider the system secure if the attacker could learn 90% of the password? Learning 90% of the password significantly increases the attacker's chance of guessing the whole password. Finally, we could require that in a safe authentication protocol no attacker can learn anything about the password through interaction, the so called *non-interference* definition [1]. Unfortunately, this definition usually breaks in practice: if the attacker inserts a random password and gets rejected he will learn that the password is not the one he tried, breaking non-interference [2].

We believe that it is not possible to give a qualitative, yes-no definition of security that is not overly strict (rejects any authentication protocol) or overly permissive (accepts protocols compromising a significant amount of the password). The core problem is that even a secure protocol will allow the attacker to learn a small amount of information about the password [3].

Quantitative Information Flow techniques can be employed to precisely quantify the number of bits of information an attacker would gain about the confidential data of a system by interacting with the system and observing its behavior [4], leaving to the analyst to decide whether this amount is acceptable for the protocol analyzed.

The difference between the information, in Shannon's information-theoretical sense [5], that a given attacker possesses about the secret before and after a single attack is called *leakage* [4]. Different attackers would infer different amount of information. A possible quantitative definition of security of a system is the maximum leakage of information (so leakage towards the attacker that can infer the most information). Maximum leakage is known in security theory as *channel capacity*, ranging over all attackers with the same observational power but different prior information on the secret. As said, no single attack can leak an amount of information higher than the system's channel capacity [6]. For a deterministic system, the leakage is the entropy of the observable behavior of the system conditioned by the attacker's behavior [7], and thus computing the channel capacity reduces to computing the behavior of the system that maximizes entropy. This security guarantee is only valid for one process; any perturbation of the process' behavior would invalidate it.

Problem. The quantification of information leakage evaluates only the security of a completely defined system, since it needs to evaluate the likelihood and estimate the loss of confidential data of each possible user behavior and system's reaction to it. Alas, when a security protocol is designed the designer needs the freedom to leave some parts underspecified, for instance the size of a password or the number of usernames in the system. An underspecified system can be seen as the infinite union of all its implementations. Providing a security guarantee for it is challenging, as it requires computing the maximum leakage over all attacks by all possible attackers on all possible implementations.

Contribution. We will present a method to obtain a safety guarantee for a protocol specification when it is possible and signal that the protocol is unsafe otherwise. To the

best of our knowledge, no other approach to channel capacity computation for infinite classes of processes exists.

To execute a formal analysis of a process specification we need models for the processes and specifications and procedures to analyze them. Following the approach we introduced in [8] we use Markov chains (MCs) as process models, with the states of the MC representing the observable components of the system and the transition probabilities raising from the attacker's uncertainty about the secret. A single MC represents an attack scenario, in which a given attacker interacts with and observes a given deterministic system. The entropy of the MC for the scenario is the leakage from the system to the attacker.

In order to encode the infinite number of possible attackers and implementations, we need the continuity of real-valued transition probability intervals, so we use Interval Markov Chains (Interval MCs) [9] as specification models. We also show how to employ Markov Decision Processes (MDPs) [10] as specification models. We present algorithms that synthesize the process with maximum entropy among all those respecting a given probabilistic specification, allowing us to give a security guarantee valid for all the infinite processes respecting the specification, encoded as an Interval MC or as an MDP.

Specification models, and Interval MCs in particular, allow us to generalize the system-attacker scenario to all possible attackers interacting with all possible implementations of a specification. The Maximum Entropy resolution of the nondeterminism in the Interval MC is thus the scenario with the greatest leakage, and its entropy is the channel capacity of the specification and the maximum amount of information that can be leaked by any single attack to any given implementation of the specification.

Given a deterministic protocol specified as an Interval MC, we show how to:

1. *Compute the entropy of a given implementation.* We provide a polynomial-time procedure to compute entropy for a Markov chain, by reduction to computation of the Expected Total Reward [10, Chpt. 5] of a local non-negative reward function associated with states over the infinite horizon.

2. *Check if a protocol allows for insecure implementations.* We provide a polynomial-time procedure for deciding finiteness and boundedness of the entropy of all implementations of an Interval MC. In general a Maximum Entropy implementation might not exist. Implementations might be non-terminating and accumulate infinite entropy, or they may have arbitrarily high, i.e. unbounded, finite entropy, so standard optimization techniques would diverge. In such case it is not possible to give a security guarantee for the implementations. We provide a polynomial-time algorithm to distinguish the two cases. If a protocol allows implementations with infinite or unbounded entropy then no matter the size n of the secret, it will have an implementation leaking all n bits of it. We detect this so that the designer can strengthen the protocol design appropriately.

3. *Compute channel capacity of a protocol.* This is a multidimensional nonlinear maximization problem on convex sets [11]. We use a numerical procedure for synthesizing with arbitrary approximation an implementation maximizing a reward function over Interval MCs. An Interval MC can be considered as an infinite set of processes, and since entropy is a nonlinear function of all possible behaviors of a

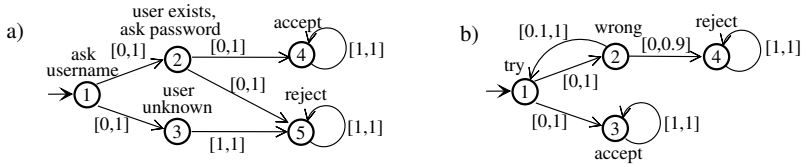

Figure 1: a) Two-step Authentication b) Repeated Authentication

system, finding the one with highest entropy is not trivial. We apply this procedure to synthesize a Maximum Entropy process implementing an Interval MC; the entropy of such a process is the channel capacity of all processes implementing the Interval MC.

On the theoretical level, this paper extends the well know Maximum Entropy Principle of Jaynes [12], from constraints on probability distributions to interval constraints on discrete probabilistic processes. We consider Interval MCs as constraints over probabilistic processes and resolve them for maximum entropy. As a result we validate the intuition that channel capacity computation as known in security research corresponds to obtaining least biased solutions in Bayesian inference for processes.

Examples. Consider two examples of models of deterministic authentication processes. Figure 1a presents an Interval MC for a two-step authentication protocol with username and password. The actual transition probabilities will depend on how many usernames exist in the system, on the respective passwords, on their length and on the attacker's knowledge about all of these; but staying at the specification level allows us to consider the worst case of all these possible combinations, and thus gives an upper bound on the leakage. The Maximum Entropy implementation for the Two-step Authentication is given in Fig. 2; its entropy is the channel capacity of the system over all possible prior informations and behaviors of the attacker and design choices of the implementer.

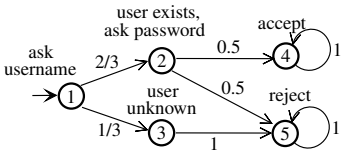

Figure 2: Maximum Entropy implementation for the Two-step Authentication

Consider another example. Figure 1b presents an Interval MC specification of the Repeated Authentication protocol, in which the user inserts a password to authenticate, and is allowed access if the password is correct. If not, the system verifies if the password entered is in a known black list of common passwords, in which case it rejects the user, considering it a malicious attacker. If the password is wrong but not black listed the user is allowed to try again. The black list cannot cover more than 90% of the possible passwords, and we assume that the attacker has no information about it.

Note that the transition probabilities from state 2 is a design choice left to the implementer of the system, while the transition probabilities from state 1 depend on a given attacker's knowledge about the password, and thus cannot be controlled even by the implementer. By abstracting these two different sources of nondeterminism at the same time we maximize entropy over all possible combinations of design choices and attackers, effectively finding the channel capacity of the specification.

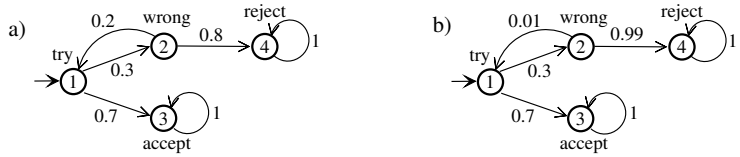

Figure 3: a) Correct implementation of the Repeated Authentication. b) Incorrect implementation of the Repeated Authentication.

The implementation maximizing entropy is the one revealing most information about the system's secret. This is consistent with our intuition. For instance, if the black list is empty the user can continue guessing the password indefinitely: the probability of eventually reaching state 3 is 1. In this implementation sooner or later the attacker will discover the password, and thus the system's secret will be completely revealed.

Figure 3 shows two implementations of the Repeated Authentication presented in Fig. 1a. None of them maximizes entropy. In fact, it is not possible to give a Maximum Entropy implementation for such protocol; this means that whatever the length n of the secret, it is possible to give an implementation that leaks all n bits of information. We will discuss the significance of this in Sect. 7. The Maximum Entropy implementation for the Two-step Authentication is given in Fig. 2. We explain how it has been synthesized in Sect. 6.

The paper is structured as follows. We introduce terminology and notation in Sect. 3. Section 4 recalls the definition of entropy for MCs, and relates it to abstract specifications. In Sect. 5 we introduce algorithms computing entropy of a given MC. In Sect. 6 we find the implementation of an Interval MC maximizing the entropy. In Sect. 7 we deal with Interval MCs that allow for implementations with infinite or unbounded entropy. In Sect. 8 we discuss the use of MDPs as specification models. We conclude in Sect. 9.

2. State of the Art and Related Work

This paper appeared previously as an extended abstract in the 24th Nordic Workshop on Programming Theory (2012) and subsequently in the conference version in the 7th International Conference on Language and Automata Theory and Applications (2013). This considerably extended version features extended introduction and related work sections, a section on the application of the technique described to Markov Decision Processes and a conclusion section. We also added detailed proof to all lemmata and theorems, some definition and many explainations and intuitions to clarify the ideas behind the paper.

The information-theoretical approach to system security that we consider here follows Clark et al. [4, 13] in using the difference between the Shannon entropy of the probability distribution over the secret before and after the attack as a measure for insecurity. Among the other measures vulnerability has been shown by Smith to have interesting qualities when considering a single attack by an attacker with no prior

information about the secret [14], and Köpf et al. show how to use probabilistic measures to quantify the security of side channels [15, 16]. Providentially, for deterministic protocols the orderings induced by all measures of insecurity coincide [7, 17], showing the robustness of the approach. Recently, Alvim et al. proposed to generalize the approach to generic gain functions depending on properties of the protocol, and also introduced an ordering based on channel factorization that is conjectured to extend the ordering to the probabilistic case [18].

The ordering of the attackers according to their potential for inferring information about the secret forms a lattice, known as the Lattice of Information [19]. The states of the Markov chains we use to model protocols represent observables, i.e. classes of equivalence of the possible models according to the attacker's observational power; we refer to [8] for the details. This use of equivalence relation is closely related to the application of abstract interpretation to security [20, 21, 22], even though they have been applied to obtain or approximate non-interference. The union of abstract interpretation and information theory would be a natural and promising development in the field of quantitative information flow.

Channel capacity as a security guarantee [6] has been studied for many different models of computation. Chatzikokolakis, Palamidessi and Panangaden use it to give a formula for anonymity analysis of protocols described by weakly symmetric matrices [23], based on the probabilistic anonymity approach by Bhargava and Palamidessi [24]. Chen and Malacaria generalize this result to asymmetric protocols [25] and also study the channel capacity of deterministic systems under different observation models [26]. Our method, unlike theirs, handles infinite classes of models instead of single models, and uses different states of a Markov chain to represent the different logical states of the system instead of considering the system as a function from inputs to outputs.

3. Background on Probabilistic Processes

We often refer to deterministic, stochastic and nondeterministic behavior. We use the adjective *deterministic* for a completely predictable behavior, *stochastic* for a behavior that follows a probability distribution over some possible choices, and *nondeterministic* for a choice where no probability distribution is given.

We define common concepts in probability theory and information theory that are used through the paper. We refer to basic books on the subject [27] for the definitions of sample space S, probability of event $P(E)$ and so on. We call X a *discrete random variable* and $\mathcal{X}$ a *discrete stochastic process*, i.e. an indexed infinite sequence of discrete random variables $(X_0.X_1, X_2, ...)$ ranging over the same sample space S. The index of the random variables in a stochastic process can be understood as modeling a concept of discrete time, so X_k is the random variable representing the system at time unit k.

3.1. Markov Chains

A discrete stochastic process is a *Markov chain* $\mathcal{C} = (C_0, C_1, C_2, ...)$ iff $\forall k \in \mathbb{N}$. $P(C_k|C_{k-1}, ..., C_0) = P(C_k|C_{k-1})$. A Markov chain on a sample space S can also be defined as follows:

Definition 1. A triple $\mathcal{C} = (S, s_0, P)$ is a Markov Chain (MC), if S is a finite set of states containing the initial state s_0 and P is an $|S| \times |S|$ probability transition matrix, so $\forall s, t \in S.\ P_{s,t} \geq 0$ and $\forall s \in S.\ \sum_{t \in S} P_{s,t} = 1$.

We slightly misuse notation, interpreting states as natural numbers and indexing matrices with state names. This is reflected in figures by labeling of states both with textual descriptions and numbers.

A state is *deterministic* if it has exactly one outgoing transition with probability 1, *stochastic* otherwise. It is known [27] that the probability of transitioning from any state s to a state t in k steps can be found as the entry of index (s, t) in matrix P^k.

We call $\pi^{(k)}$ the probability distribution vector over S at time k and $\pi_s^{(k)}$ the probability $\pi^k(s)$ of visiting the state s at time k. This means that, considering a Markov chain $\mathcal{C}$ as a time-indexed discrete stochastic process $(C_0, C_1, ...)$, we write $\pi^{(k)}$ for the probability distribution over the random variable C_k. Since we assume that the chain starts in state s_0, then $\pi_s^{(0)}$ is 1 if $s = s_0$ and 0 otherwise. Note that $\pi^{(k)} = \pi_0 P^k$, where P^k is matrix P elevated to power k, and P^0 is the identity matrix of size $|S| \times |S|$.

A state t is *reachable* from a state s if $\exists k. P_{s,t}^k > 0$. We assume that all states are reachable from s_0 in the MCs considered. A subset $R \subseteq S$ is *strongly connected* if for each pair of states $s, t \in R$, t is reachable from s.

For a state s of a Markov chain, the *expected residence time* denotes the expected amount of time units that the Markov chain will spend in state s, while the *first hitting probability* of state t after state s denotes the probability that state t will ever be visited under the condition that state s is being visited.

Definition 2. The *expected residence time* in a state s is defined as

$$\xi_s = \sum_{n=0}^{\infty} P_{s_0,s}^n$$

Definition 3. The *first hitting probability* of state t after state s is defined as

$$\rho(s,t) = \sum_{n=1}^{\infty} \mathbf{P}(X_n = t, X_{n-1} \neq t, X_{n-2} \neq t, ..., X_1 \neq t \mid X_0 = s)$$

We will sometime write ρ_s for $\rho(s_0, s)$. Note that $\xi_s = \rho_s \sum_{n=0}^{\infty} P_{s,s}^n$. A state s is *recurrent* iff $\xi_s = \infty$, *transient* otherwise.

We show how the expected residence time of all states in a Markov chain can be computed in polynomial time by reducing the problem to solving a system of linear equations. By definition expected residence time is infinite for each recurrent state, so we first check which states are recurrent, and then we calculate the expected residence time for the remaining (transient) states. We say that two states $s, t \in S$ *communicate*, written $s \leftrightarrow t$, if s is reachable from t and vice versa. Note that $s \leftrightarrow s$. Communication is an equivalence relation and as such it induces *communicating classes*:

Definition 4. A *communicating class* $C \subseteq S$ of a Markov chain is a maximal subset of the states of the Markov chain such that each pair of states in C communicates, i.e.

$$\forall s, t \in C. \mid s \leftrightarrow t$$

A Markov chain can be divided in its communicating classes by finding its Strongly Connected Components with e.g. Tarjan's algorithm [28]. The Strongly Connected Components correspond exactly to the communicating classes.

A communicating class C is said to be closed if, once it is visited, the probability that a state outside it will ever be visited is zero:

Definition 5. A *closed communicating class* (CCC) of a Markov chain is a communicating class C such that $\mathbf{P}(X_{k+1} \in C \mid X_k \in C) = 1$

Whether a communicating class is closed can be decided in polynomial time. It is known [27] that all states in a closed communicating class of a finite Markov Chain are recurrent, while all other states are transient, so we can divide the states in recurrent and transient states in polynomial time. Let R be the set of the recurrent states; then $\forall s \in R.\, \xi_s = \infty$.

The expected residence time of the transient states can be found by solving the system of equations $\xi_t = \pi_t^{(0)} + \sum_{u \in S \setminus R} P_{u,t} \xi_u$ for all $t \in S \setminus R$ (in polynomial time).

Use of Markov chains to model generic secret-dependent processes has been previously introduced in [8], including ways to automatically generate them from imperative program code. Each state of the MC represents a reachable combination of values of the public variables of the system and levels of knowledge about the private variables. We refer to [8] for the full discussion.

3.2. Interval Markov Chains

Definition 6. *[29]* A closed-interval *Interval Markov Chain* (Interval MC) is a tuple $\mathcal{I} = (S, s_0, \check{P}, \hat{P})$ where S is a finite set of states containing the initial state s_0, $\check{P}$ is an $|S| \times |S|$ bottom transition probability matrix, $\hat{P}$ is a $|S| \times |S|$ top transition probability matrix, such that for each pair of states $s, t \in S$ we have $0 \leq \check{P}_{s,t} \leq \hat{P}_{s,t} \leq 1$.

Note that for the bottom and top probability matrices it is not necessarily true that the values for a given state s will sum up to 1, as is the case for the probability transition matrix in Definition 1. The top and bottom matrices just state maximums and minimums for the transition probabilities between each pair of states.

The following defines when an MC implements an Interval MC in the Uncertain Markov Chain (UMC) semantics [29]:

Definition 7. A Markov chain $\mathcal{C} = (S, s_0, P)$ *implements* an Interval Markov Chain $\mathcal{I} = (S, s_0, \check{P}, \hat{P})$, written $\mathcal{C} \vDash \mathcal{I}$, if $\forall s, t \in S.\, \check{P}_{s,t} \leq P_{s,t} \leq \hat{P}_{s,t}$.

An example of an Interval MC is the Repeated Authentication dpicted in Fig. 1b. The MC in Fig. 3a uses a black list covering 80% of the passwords and is thus an implementation of the Interval MC in Fig. 1b, while the Markov chain in Figure 3b uses a black list covering 99% of the passwords and it is not an implementation of the Interval MC in Fig. 1b.

We assume without loss of generality that our Interval MCs are *coherent*, meaning that every value for each transition interval is attained by some implementation. Coherence can be established by checking that both following conditions hold [30]:

1. $\forall s, t \in S. \check{P}_{s,t} \geq (1 - \sum_{u \neq t} \hat{P}_{s,u})$
2. $\forall s, t \in S. \hat{P}_{s,t} \leq (1 - \sum_{u \neq t} \check{P}_{s,u})$

Any Interval MC $\mathcal{I} = (S, s_0, \check{P}, \hat{P})$ can be transformed into a coherent Interval MC $\mathcal{I}' = (S, s_0, \check{P}', \hat{P}')$ by changing the top and bottom transition probability matrices to the following:

1. $\check{P}'_{s,t} = \max(\check{P}_{s,t}, 1 - \sum_{u \neq t} \hat{P}_{s,u})$
2. $\hat{P}'_{s,t} = \min(\hat{P}_{s,t}, 1 - \sum_{u \neq t} \check{P}_{s,u})$

The resulting coherent Interval MC $\mathcal{I}'$ is unique and has the same implementations as the original incoherent Interval MC $\mathcal{I}$ [30], so in particular it has an implementation iff $\mathcal{I}$ has at least one implementation. It is also interesting to note that if an Interval MC is coherent then it is possible to give an implementation in which a positive probability is assigned to all transitions for which this is allowed by the Interval MC, as stated by the following lemma.

Lemma 1. *Let $\mathcal{I} = (S, s_0, \check{P}, \hat{P})$ be a coherent Interval MC. Then there exists an implementation $\mathcal{C} = (S, s_0, P)$ of $\mathcal{I}$ in which $\forall s, t \in S. \hat{P}_{s,t} > 0 \Rightarrow P_{s,t} > 0$.*

Proof. [1] We recall the coherence assumptions:

1. $\forall s, t \in S. \check{P}_{s,t} \geq (1 - \sum_{u \neq t} \hat{P}_{s,u})$
2. $\forall s, t \in S. \hat{P}_{s,t} \leq (1 - \sum_{u \neq t} \check{P}_{s,u})$

Assume with no loss of generality that $\forall s, t \in S. \hat{P}_{s,t} > 0 \Rightarrow \check{P}_{s,t} < \hat{P}_{s,t}$.

We will proceed ad absurdum. Let's consider a state $s \in S$ and assume there is no implementation of $\mathcal{I}$ in which $\forall t \in S. \hat{P}_{s,t} > 0 \Rightarrow P_{s,t} > 0$, meaning in all implementations there exists at least one state t' such that $\hat{P}_{s,t'} > 0 \wedge P_{s,t'} = 0$. Let $\mathcal{C} = (S, s_0, P)$ be one such implementation. Clearly it must be $\check{P}_{s,t'} = 0$. Since the implementation is a Markov chain it holds that $\sum_{t \in S} P_{s,t} = 1$ thus $\sum_{\substack{t \in S \\ t \neq t'}} P_{s,t} = 1$. Let's assume that $\forall u \neq t'. P_{s,u} = \check{P}_{s,u}$; we have that $1 - \sum_{u \neq t} \check{P}_{s,u} = 1 - \sum_{u \neq t} P_{s,u} = 0$, but this is not possible since $\hat{P}_{s,t'} > 0$ by assumption and $\hat{P}_{s,t'} \leq (1 - \sum_{u \neq t'} \check{P}_{s,u})$ by coherence. Thus there must be a state $t'' \in S$ such that $t'' \neq t'$ and $P_{s,t''} > \check{P}_{s,t''}$.

But then consider a positive constant ε such that $\varepsilon < \hat{P}_{s,t'}$ and $\varepsilon < P_{s,t''} - \check{P}_{s,t''}$; the implementation with transition probabilities P' defined as

$$P'_{s,t} = \begin{cases} \varepsilon & \text{if } t = t' \\ P_{s,t''} - \varepsilon & \text{if } t = t'' \\ P'_{s,t} & \text{otherwise} \end{cases}$$

[1] We extend our thanks to Lei Song for the proof of this lemma.

is an implementation of $\mathcal{I}$ in which $\forall t \in S.\hat{P}_{s,t} > 0 \Rightarrow P_{s,t} > 0$, contradicting the assumption that no such implementation exists. □

We will use this result later in the proof of Theorem 2.

A state s of an Interval MC is *deterministic* if $\exists t.\check{P}_{s,t} = 1$, *stochastic* otherwise. We say that a state t is *reachable* from a state s if $\exists s_1, s_2, ..., s_n \in S.s_1 = s \wedge s_n = t \wedge \hat{P}_{s_i,s_{i+1}} > 0$ for $1 \leq i < n$. We say that a subset $R \subseteq S$ is *strongly connected* if $\forall s, t \in R.$ t is reachable from s.

Note that if there is an implementation in which a subset of states $R \subseteq S$ is strongly connected, then R must be strongly connected in the Interval MC.

3.3. Markov Decision Processes

Definition 8. A *Markov Decision Process* (MDP) [10] is a tuple $\mathcal{P} = (S, s_0, P, \Lambda)$ where S is a finite set of states containing the initial state s_0, Λ_s is the finite set of available actions in a state $s \in S$ and $\Lambda = \bigcup_{s \in S} \Lambda_s$, and $P : S \times \Lambda \times S \to [0, 1]$ is a transition probability function such that $\forall s, t \in S.\forall a \in \Lambda_s.$ $P(s, a, t) \geq 0$ and $\forall s \in S.\forall a \in \Lambda_s.$ $\sum_{t \in S} P(s, a, t) = 1$. We write $P_{(s,a),t} = x$ for $P(s, a, t) = x$.

An MDP makes a nondeterministic choice of an action in each state. A selection of an action determines a probability distribution over the successor states. A *probabilistic strategy* is a function $\sigma : S \to \delta(\Lambda_s)$ assigning to a state s a probability distribution over the actions in Λ_s. A probabilistic strategy is *positional* if its probability distributions assign probability 1 to some action. The (infinite) set of strategies of the process $\mathcal{P}$ is denoted $\Sigma_{\mathcal{P}}$.

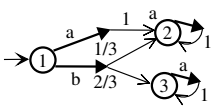

Figure 4: A simple MDP

In the MDP depicted in Fig. 4 there are two possible positional strategies, namely the one choosing action *a* in state **1** and transitioning to state **2** with probability 1, and the one choosing action *b* and transitioning to state **2** with probability $\frac{1}{3}$ and to state **3** with probability $\frac{2}{3}$. There is also one probabilistic strategy for each probability distribution over the two actions.

A strategy resolves an MDP into an MC, and in that sense MDPs, similarly to Interval MCs, can also be used as specifications for MCs; every strategy defines a different MC.

4. Entropy of Processes and Specifications

The entropy of a discrete probability distribution quantifies lack of information about the events involved. This idea can be extended to quantify nondeterminism, understood as degree of unpredictability of an MC.

Definition 9. Let X be a random variable on the discrete domain $(x_1, ..., x_n)$ and $\mathbf{P}(X = x_k)$ its probability distribution. Then the *entropy* of the random variable X is defined as

$$H(X) = -\sum_{i=1}^{n} \mathbf{P}(x_i) \log_2(\mathbf{P}(x_i))$$

Entropy is always non-negative. It is maximal for the uniform distribution, in which case its value is $\log_2 n$ [27], representing complete ignorance about the value assumed by the random variable. On the other extreme, $H(X)$ is 0 if and only if the probability distribution on X assigns a given value to it with probability 1. For MCs, entropy is maximum for the process in which all possible paths in the chain have the same probability.

To define the entropy of a Markov chain $\mathcal{C}$ we need to introduce the concepts of *conditional entropy* and *joint entropy* [27, 5]. Conditional entropy quantifies the remaining entropy of a variable Y given that the value of other random variables (X_i here) is known.

$$H(Y \mid X_1, \ldots, X_n) = -\sum_{t \in S} \sum_{s_1 \in S} \cdots \sum_{s_n \in S} \mathbf{P}(Y = t, X_1 = s_1, \ldots, X_n = s_n) \cdot \\ \cdot \log_2 \mathbf{P}(Y = t \mid X_1 = s_1, \ldots, X_n = s_n) \ ,$$

where $\mathbf{P}(Y = t, X_1 = s_1, \ldots, X_n = s_n)$ denotes the joint probability of the events $Y = t$, $X_1 = s_1$, ..., $X_n = s_n$.

Joint entropy is simply the entropy of several random variables computed jointly, i.e. the combined uncertainty due to the ignorance of n random variables. It turns out [27] that joint entropy can be calculated in the following way, using conditional entropy, which will be instrumental in our developments for MCs.

$$H(X_0, X_1, \ldots, X_n) = -\sum_{s_0 \in S} \sum_{s_1 \in S} \cdots \sum_{s_n \in S} \mathbf{P}(X_0 = s_0, X_1 = s_1, \ldots, X_n = s_n) \cdot \\ \cdot \log_2(\mathbf{P}(X_0 = s_0, X_1 = s_1, \ldots, X_n = s_n)) = \\ = H(X_0) + H(X_1 \mid X_0) + \cdots + H(X_n \mid X_0, X_1, \ldots, X_{n-1})$$

Now, the definition of entropy of an MC is unsurprising, if we take the view of the processes as a series of random variables; it is the joint entropy of these variables (recall that due to the Markov property, the automata-view, and the probabilistic view of MCs are interchangeable):

Definition 10. We define the entropy of a Markov chain $\mathcal{C} = (X_n, n \in \mathbb{N})$ as the joint entropy over all X_n: $H(\mathcal{C}) = H(X_0, X_1, X_2, \ldots) = \sum_{i=0}^{\infty} H\left(X_i \,|X_{i-1} \ldots X_0\right)$.

Note that since we have assumed a single starting state in each MC, it is always the case that $H(X_0) = 0$. Also the above series always converges to a real number, or to infinity, since it is a sum of non-negative real numbers.

In leakage analysis, entropy corresponds to the information leakage of the system only when the system is deterministic and the attacker cannot interact with it [7]. Using probability intervals we can lift the latter restriction, as different distributions on the attacker's input would only lead to different transition probabilities, and the intervals already consider all possible transition probabilities.

The entropy of an MC is in general infinite. In the next Section we will show that the entropy of an MC is finite if and only if the chain is absorbing. Considering only absorbing MCs avoids the problem of the entropy of an MC being in general infinite.

We always consider terminating protocols, and they can be encoded as absorbing MCs where the absorbing states represent the termination of the protocol; consequently the entropy of a Markov chain encoding a terminating process is always finite.

We stress that it is common [31, 5] to compute the average entropy of each step of the MC and to call it the entropy of the MC. In our opinion, technically speaking this notion should be more precisely named an entropy rate [27]. Here we want to compute the actual entropy since the entropy, not its rate, represents the information leakage in a security scenario where the states of the MC are the observables of a deterministic protocol.

An alternative characterization of entropy of a process depends explicitly on which states get visited during the lifetime of the process. Since every state s has a probability distribution over the next state we can compute the entropy of that distribution, which we will call *local entropy* $L(s)$ of s: $L(s) = H(X_{k+1} \mid X_k = s) = -\sum_{t \in S} P_{s,t} \log_2 P_{s,t}$. Note that $L(s) \leq \log_2(|S|)$. Also, in general, the value of entropy of an MC is not equal to the sum of the local entropy values for each state. Such sum will have to be weighted against the expected residence time of each state to characterize the "global" entropy.

Now consider the Interval MC specification of the Repeated Authentication protocol in Fig. 1b. Different implementations of it, like the ones in Fig. 3ab, will have different entropy values. We define a *Maximum Entropy implementation* for an Interval MC, as an implementation MC, which has entropy not smaller than the entropy of any other implementation (if such exists):

Definition 11. Let $\mathcal{I}$ be an Interval Markov Chain. Then a Markov chain $\mathcal{C}$ is a *Maximum Entropy implementation* of $\mathcal{I}$ if the following holds:

1. $\mathcal{C} \models \mathcal{I}$
2. $H(\mathcal{C})$ is finite
3. $\forall \mathcal{C}'.\mathcal{C}' \models \mathcal{I} \Rightarrow H(\mathcal{C}') \leq H(\mathcal{C})$

The boundedness and synthesis of a Maximum Entropy implementation of an Interval MC will be treated in Sect. 6. It may be that the maximum entropy is actually infinite, or that the set of attainable entropies is unbounded; we discuss these cases in Sect. 7.

5. Computing Entropy of Markov Chains

We now provide an algorithm for computing entropy of a given MC. We cast entropy as a non-negative reward function on an MC, and then apply standard techniques to compute it. We also provide a simple decision procedure for deciding whether entropy of an MC is finite.

A *non-negative reward function* over the transitions of an MC is a function $R : S \times S \to \mathbb{R}^+$ assigning a non-negative real value, called reward, to each transition. Given a reward function R we can compute the value of the reward for a concrete execution of an MC by summing reward values for the transitions exercised in the execution. More interestingly, we can compute the expected reward of each state $s \in S$

as $R(s) = \sum_{t \in S} P_{s,t} R_{s,t}$, and then the expected reward over the infinite behavior of an MC $\mathcal{C}$ is $R(\mathcal{C}) = \sum_{s \in S} R(s)\xi_s$ [10, Chpt. 5].

Each $R(s)$ can be computed in time linear in the number of states, so calculation of expected rewards for all states can be done in quadratic time. Since computing the expected residence time for a state s is in PTIME as shown in Sect. 3, we can also compute $R(\mathcal{C})$ in polynomial time.

Let the reward function be $R(s,t) = -\log_2 P_{s,t}$. Then the expected reward for each state is its local entropy, or $R(s) = -\sum_{t \in S} P_{s,t} \log_2 P_{s,t} = L(s)$. Note that this is an unorthodox non-negative reward function, since the reward for choosing a transition in a given state is not a constant but depends on the probability distribution over the successors of the state, as a function of the form $R : S \times S \times P \to \mathbb{R}^+$. It turns out that the (global) entropy of the MC is the expected reward with this reward:

Theorem 1. *For an MC $\mathcal{C} = (S, s_0, P)$ we have that $H(\mathcal{C}) = \sum_{s \in S} L(s)\xi_s$.*

Proof. Consider the chain rule for conditional entropy:

$$
\begin{aligned}
&H(X_n|X_1, X_2, ..., X_{n-1}) = \\
&= -\sum_{t \in S}^{N} \sum_{s_0 \in S}^{N} \cdots \sum_{s_{n-1} \in S}^{N} P(X_n = t, X_0 = s_0, ..., X_{n-1} = s_{n-1}) \cdot \\
&\cdot \log(P(X_n = t | X_1 = s_1, ..., X_{n-1} = s_{n-1})) \\
&= \sum_{s \in S} \pi_s^{(n-1)} H(X_n | X_{n-1} = s) = \sum_{s \in S} \pi_s^{(n-1)} L(s)
\end{aligned}
$$

Where $\pi_s^{(0)}$ is 1 if $s = s_0$ and 0 otherwise. We can see that the entropy at time n is the sum on each state of the local entropy of the state multiplied by the probability to be in that state at time $n-1$. Applying this to the chain rule for joint entropy over an infinite time we have that

$$
\begin{aligned}
H(X_0, X_1, X_2, ...) &= H(X_0) + \sum_{n=1}^{\infty} H(X_n | X_0, X_1, ..., X_{n-1}) \\
&= H(X_0) + \sum_{n=1}^{\infty} \sum_{s \in S} \pi_s^{(n-1)} L(s) \\
&= H(X_0) + \sum_{s \in S} L(s) \sum_{n=1}^{\infty} \pi_s^{(n)} \\
&= H(X_0) + \sum_{s \in S} L(s) \sum_{n=1}^{\infty} P_{s_0,s}^n \\
&= \sum_{s \in S} L(s) \sum_{n=0}^{\infty} P_{s_0,s}^n = \sum_{s \in S} L(s)\xi_s
\end{aligned}
$$

□

As any other reward of this kind, the entropy of an MC can be infinite. Intuitively, the entropy is finite if it almost surely stops increasing. This happens if the execution is eventually confined to a set of states with zero local entropy (deterministic). Since the recurrent states of a chain are exactly the ones that are visited infinitely often, we obtain the following characterization:

Corollary 1. *The entropy $H(\mathcal{C})$ of a chain $\mathcal{C}$ is finite iff the local entropy of all its recurrent states is zero.*

The above observation gives us an algorithmic characterization of finiteness of entropy for MCs: the entropy of a chain is finite if and only if the chain has one or more absorbing states or absorbs into closed deterministic cycles. Entropy can only be infinite for infinite behaviors; for the first n execution steps the entropy is always bounded by $n \log_2 |S|$.

We can classify the processes in two categories: those which eventually terminate the stochastic behavior, and those which do not. Many processes become deterministic (or even terminate) after some time. This is the case for a terminating algorithm like a randomized primality test, or for randomized IP negotiation protocols like Zeroconf, which stops behaving randomly as soon as an IP number is assigned. Such processes have finite entropy. On the other hand, the processes that take probabilistic choices forever and never become deterministic have infinite entropy. Using Corollary 1 we characterize the processes having finite entropy as terminating and the processes having infinite entropy as non-terminating. In this sense we consider a process as terminated when its behavior will be forever deterministic, as it will not reveal any more information about the secret.

6. Maximum Entropy Implementation of an Interval MC

Interval MCs describe infinite sets of MCs. We now show how to find an implementation that maximizes entropy, in the sense of Definition 11. Since our Markov chains represent the behavior of deterministic processes, the Maximum Entropy implementation we synthesize is also the one with maximum leakage, and its leakage is thus the channel capacity of all implementations.

In Fig. 2 we show the Maximum Entropy implementation of the Two-step Authentication specification in Fig. 1a. Its entropy is $\log_2 3 \approx 1.58496$ bits. This allows us to guarantee that none of the infinite possible implementations of the Two-step Authentication will leak more than $\log_2 3$ bits of information to any possible attacker.

Obtaining a Maximum Entropy implementation is a challenging problem. In the first place, such an implementation may not exist, as the set of attainable entropies may be unbounded, so we need an algorithm to verify its existence. Secondly, even if it exists finding it consists of solving a nonlinear optimization problem with constraints over an infinite domain.

We propose a numerical approach to the general problem of solving Interval MCs for non-negative reward functions, and apply it to finding a Maximum Entropy implementation. We first check that such an implementation exists, and then proceed to synthesize it. Remember that an implementation maximizing the expected total reward for the reward function $R(s,t) = -\log_2(P_{s,t})$ is a Maximum Entropy implementation.

The expected reward of a reward function may be infinite. An Interval MC admits implementations with infinite entropy if it has a state that can be recurrent and stochastic in the same implementation. We call this the *infinite* case.

If an Interval MC has a state that is recurrent and deterministic in some implementations and transient and stochastic in some others, but never both recurrent and stochastic in the same implementation, the set of entropies of its implementations is unbounded, despite all the individual implementations having finite entropy; an example is the Repeated Authentication in Fig. 1b. We call this the *unbounded* case. This happens because the reward assigned to a transition is not a constant, but a logarithmic function of the actual transition probability. With such reward it is possible that the total reward value can be unbounded across possible implementations (not just finite or infinite as for classical non-negative rewards). Note that this does not happen with constant rewards, and is specific to our problem.

6.1. Existence of a Maximum Entropy Implementation

We now show an algorithm for determining whether an Interval MC has a Maximum Entropy implementation with finite entropy. To do this we first give a definition of end components [32, 33] for Interval MCs. Then we show the algorithm for deciding the existence of a Maximum Entropy implementation.

An end component is a set of states of the Interval MC for which there exists an implementation such that once the behavior enters the end component it will stay inside it forever and choose all transitions inside it an infinite number of times with probability 1. We refer to [32] for further discussion.

Definition 12. Given an Interval MC $\mathcal{I} = (S, s_0, \check{P}, \hat{P})$, a set of states $R \subseteq S$ is an *end component of* $\mathcal{I}$ if the following holds:

1. R is strongly connected;
2. $\forall s \in R, t \in S \backslash R. \check{P}_{s,t} = 0$;
3. $\forall s \in R. \sum_{u \in R} \hat{P}_{s,u} \geq 1$.

Corollary 2. *For an Interval MC $\mathcal{I} = (S, s_0, \check{P}, \hat{P})$, if $R \subseteq S$ is an end component of $\mathcal{I}$ then there is an implementation of $\mathcal{I}$ in which $\boldsymbol{P}(X_{n+1} \notin R \mid X_n \in R) = 0$.*

An end component is maximal if no other end component contains it. In the Interval MC pictured in Fig. 5 we have that $\{1, 2\}$ is an end component, $\{1, 3\}$ is an end component, and $\{1, 2, 3\}$ and $\{4\}$ are maximal end components.

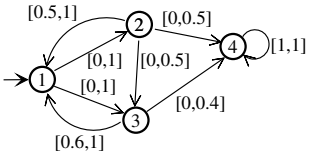

Figure 5: Interval MC with multiple end components

Algorithm 1 finds all maximal end components of an Interval MC. It first identifies all candidate end-components and their complement—the obviously transient states; then it propagates transient states backwards to their predecessors who cannot avoid reaching them. The predecessors are pruned from the candidate end-components and the procedure is iterated until a fixed point is reached.

Tag all states of S as UNCHECKED;
Find the strongly connected components (SCCs) of the Interval MC (e.g. with Tarjan's algorithm) and tag any state not in any SCC as TRANSIENT;
repeat
  foreach *SCC* C **do**
    Select a state $s \in C$ tagged UNCHECKED;
    Check that $\forall t \in S \backslash C . \check{P}_{s,t} = 0$. If not, remove s from C, tag it TRANSIENT, tag UNCHECKED all states in C with a transition to s and select another state;
    Check that $\sum_{u \in C} . \hat{P}_{s,u} \geq 1$. If not, remove s from C, tag it TRANSIENT, tag UNCHECKED all states in C with a transition to s and select another state;
    Tag s as ENDCOMPONENTSTATE;
  end
until *all states in any non-empty SCC are tagged as* ENDCOMPONENTSTATE;

Algorithm 1: Find all maximal end components of an Interval MC.

Lemma 2. *Algorithm 1 runs in time $O(|S|^3)$, and upon termination the states of the Interval MC are tagged as* ENDCOMPONENTSTATE *if they are part of any maximal end component, and tagged as* TRANSIENT *otherwise.*

Proof. Clearly the algorithm always terminates, and when it does no state is tagged UNCHECKED.

Remember from Section 3 that the communicating classes of a Markov chain correspond to the Strongly Connected Components (SCCs) in its graph representation. In Algorithm 1 we similarly use the SCCs of the Interval MC graph to identify the potential end components.

We show that a state t is tagged as TRANSIENT iff there is no maximal end component containing it. If t is not in any strongly connected component, it is obvious. Note that if C is a SCC then there is no end component that contains both states inside C and outside C. If t is in a SCC C at the beginning of the loop, but not after the end of the algorithm, it was removed in some iteration of the loop; remember that the two checks that are performed for each state check that there is at least one implementation such that $\mathbf{P}(X_{n+1} \notin R | X_n = t)$. Let's consider the iteration in which t was removed. If it was the first state to be removed from C, it means that for any implementation there is a positive probability of transitioning from t to a state outside C, so t must be transient in any implementation. Thus we can assume by inductive hypothesis that any state removed from C at a certain iteration k is necessarily transient; then if t gets removed after iteration k it means that it has a positive probability to go to a state $u \notin C$ or to a state that has been removed from C in a previous step; in both cases t has necessarily positive probability to go to a transitive state, and is thus transitive.

Since we proved that no states are tagged UNCHECKED after running the algorithm and that states are tagged TRANSITIVE iff they are not in any maximal end component, necessarily states are tagged ENDCOMPONENTSTATE iff they are in some maximal

Make the Interval MC coherent (see Sect.3);
Find the maximal end components of the Interval MC with Algorithm 1 and call their union S_ω;
if *there is a stochastic state in* S_ω **then**
| signal that no Maximum Entropy implementation exists
else
| signal that a Maximum Entropy implementation exists
end

Algorithm 2: Verify whether an IMC has a Maximum Entropy implementation

end component.

COMPLEXITY: Let $n = |S|$. Tarjan's algorithm runs in time quadratic in n. Each time a state is selected in the loop it is tested in time linear in n, and each state gets visited at most $O(n^2)$ times, so the total running time of the algorithm is $O(|S|^3)$. □

Algorithm 2 establishes finiteness of Maximum Entropy across all implementations of an Interval MC. After finding the maximal end components we check whether each end component state in $\mathcal{I}$ is deterministic. Because the Interval MC is coherent, this check simply amounts to verifying that for each state in each end component there is a successor state with lower bound on transition probability being 1. If this is the case, then there exists a Maximum Entropy implementation for $\mathcal{I}$ with a finite entropy value.

Algorithm 2 for deciding existence of finite maximum entropy implementation is sound and complete:

Theorem 2. *Let $\mathcal{I}$ be an Interval MC and S_ω the union of all its end components. Then $\mathcal{I}$ has no Maximum Entropy implementation iff a state $s \in S_\omega$ is stochastic.*

Before proving Theorem 2 we need an additional lemma showing a connection between recurrent states and end components:

Lemma 3. *Let $\mathcal{I} = (S, s_0, \check{P}, \hat{P})$ be an Interval MC and S_ω the union of all states of all its end components. Then $s \in S_\omega$ iff there exists an implementation of $\mathcal{I}$ in which s is recurrent.*

Proof. Let R be an end component of $\mathcal{I}$ and $s \in R$.

IF: We want to show that there exists an implementation in which s is recurrent. Consider an Interval MC $\mathcal{I}' = (S, s_0, \check{P}, \hat{P}')$ identical to $\mathcal{I}$ except that $\hat{P}'$ is defined as follows:

$$\forall s, t \in S. \hat{P}'_{s,t} = \begin{cases} 0 & \text{if } s \in R \wedge t \notin R \\ \hat{P}_{s,t} & \text{otherwise} \end{cases}$$

Note the following:

- $\mathcal{I}'$ is coherent: since $\mathcal{I}$ is coherent we need to check $\mathcal{I}'$ only on the intervals that differ from $\mathcal{I}$, meaning the $\hat{P}_{s,t}$ where $s \in R \wedge t \notin R$; we need to show that $\hat{P}_{s,t} = 0 \leq 1 - \sum_{u \neq t} \check{P}_{s,u}$, but this is obvious since $\forall s. \sum_{t \in S} \check{P}_{s,t} \leq 1$ by consistency assumption of $\mathcal{I}$.

- R is an end component in $\mathcal{I}'$: this is obvious since $\check{P}$ is not modified and $\hat{P}$ for transitions to state inside R is also unmodified.

- Any implementation of $\mathcal{I}'$ is also an implementation of $\mathcal{I}$, since $\mathcal{I}'$ refines $\mathcal{I}$.

We know by Definition 12 that R is a strongly connected component in $\mathcal{I}'$, and by Lemma 1 this means that there is an implementation in which R is strongly connected. By the definition of $\mathcal{I}'$ clearly for any implementation of $\mathcal{I}'$, $\forall n \in \mathbb{N}.\mathbf{P}(X_{n+1} \notin R | X_n \in R) = 0$. This means that there is an implementation of $\mathcal{I}'$ in which R is a closed communicating class, meaning all states in it are recurrent. Since every implementation of $\mathcal{I}'$ is also an implementation of $\mathcal{I}$, this is also true for $\mathcal{I}$.

ONLY IF: We want to show that if there exists an implementation $\mathcal{C}$ of $\mathcal{I}$ in which s is recurrent, then there is an end component R of $\mathcal{I}$ such that $s \in R$. By definition of recurrent states we know that in the implementation $\mathcal{C}$ we have $\rho(s,s) = 1$. Let's consider the set of the successors of s in any number of steps: $Succ^*(s) = \{t \in S | \exists k \in \mathbb{N}.P^k_{s,t} > 0\} \subseteq S$.

Clearly $s \in Succ^*(s)$. Let's consider a state $t \in Succ^*(s)$: since t is a successor of s obviously $\rho(s,t) > 0$, but then since $\rho(s,s) = 1$ it must be $\rho(t,s) = 1$, meaning that $s \in Succ^*(t)$. Since $\forall t, u \in Succ^*(s).\exists k, k'.P^k_{t,s} > 0 \wedge P^{k'}_{s,u} > 0$ then $\forall t, u \in Succ^*(s).P^{k+k'}_{t,u} > 0$, meaning $Succ^*(s)$ is strongly connected in $\mathcal{C}$ and consequently in $\mathcal{I}$.

Let's again consider a state $t \in Succ^*(s)$: we proved that all states in $Succ^*(s)$ are reachable from it, now we prove ad absurdum that no state outside $Succ^*(s)$ is reachable from t. Consider $u \in S \backslash Succ^*(s)$ such that $P_{t,u} > 0$. Then since t is reachable from s and u is reachable from t then u is reachable from s, contradicting $u \in S \backslash Succ^*(s)$. Thus in $\mathcal{C}$ there is no transition with probability greater than 0 from a state in $Succ^*(s)$ to a state outside $Succ^*(s)$; since $\mathcal{C}$ implements $\mathcal{I}$ it must be $\forall t \in Succ^*(s), u \in S \backslash Succ^*(s).\check{P}_{t,u} = 0$ and since $\mathcal{C}$ is a Markov chain it must be that $\forall t \in Succ^*(s), \sum_{u \in Succ^*(s)}.\hat{P}_{t,u} \geq 1$. We have shown that $Succ^*(s)$ respects all conditions in Definition 12, and thus it is an end component. □

Now we can proceed with the proof of Theorem 2:

Proof of Theorem 2. Let R be an end component of $\mathcal{I}$ and $s \in R$.

IF: Let $\mathtt{N} = \{\mathcal{C} \vDash \mathcal{I} | s \text{ is stochastic in } \mathcal{C}\}$ and $\mathtt{R} = \{\mathcal{C} \vDash \mathcal{I} | s \text{ is recurrent in } \mathcal{C}\}$. Since $s \in S_\omega$ then by Lemma 3 $\mathtt{R} \neq \emptyset$, while $\mathtt{N} \neq \emptyset$ by assumption. We can distinguish two cases:

- $\mathtt{N} \cap \mathtt{R} \neq \emptyset$: let $\mathcal{C} \in \mathtt{N} \cap \mathtt{R}$; then in the implementation $\mathcal{C}$ state s is both recurrent and stochastic, and consequently $H(\mathcal{C}) = \infty$ by Theorem 1, thus no Maximum Entropy implementation exists. In this case we say that the entropy of $\mathcal{I}$ is *infinite*.

- $\mathtt{N} \cap \mathtt{R} = \emptyset$: let $\mathcal{C} \in \mathtt{R}$ be an implementation in which R is strongly connected; it exists by Lemma 1. We argue that $\exists s \in R.\exists t \notin R.\hat{P}_{s,t} > 0$ ad absurdum: let's assume no state in R has an allowed transition outside R, then since $\mathtt{N} \neq \emptyset$ there is an implementation in which a state $s \in R$ is stochastic and recurrent, contradicting $\mathtt{N} \cap \mathtt{R} = \emptyset$. Then $\exists s \in R.\exists t \notin R.\hat{P}_{s,t} > 0$. Consider $\mathcal{C}' \in \mathtt{N}$ which is identical to

$\mathcal{C}$ except that we assign an arbitrarily small positive probability ε to the transition from s to t and the remaining $1-\varepsilon$ to states inside R. Note that $\mathcal{C}' \notin \mathtt{R}$. Since we did not modify any other state in R it is still true that $\forall u \in R \backslash \{s\}.\rho(u,s) = 1$ while $\rho(s,s) = 1-\varepsilon$. Consider an arbitrarily small $1 > \delta \geq \varepsilon$. If there exists a path of finite length l from s to itself with probability $1-\varepsilon$, let $\delta = \varepsilon$; if such a finite path does not exist it means that $\sum_{n=0}^{\infty} P^n_{s,s} = 1-\varepsilon$, in which case take $\delta = 1 - \sum_{n=0}^{l} P^n_{s,s}$ for an arbitrarily large fixed $l \in \mathbb{N}$. In any case we will have $\sum_{n=0}^{l} P^l_{s,s} = 1-\delta$. Note that δ gets arbitrarily close to ε as $l \to \infty$.

Let's study ξ_s. Consider the set $\mathtt{K}=\{n \mid P^n_{s_0,s} > 0\}$; it is nonempty by reachability of s. Let $k = \inf(\mathtt{K})$ and $\rho = P^k_{s_0,s}$. Since $\rho \leq \rho_s$ we can write

$$\begin{aligned}
\xi_s &= \sum_{n=0}^{\infty} P^n_{s_0,s} \geq \rho \sum_{n=0}^{\infty} P^n_{s,s} \\
&= \rho \left(\sum_{n=0}^{l-1} P^n_{s,s} + \sum_{n=l}^{2l-1} P^n_{s,s} + \sum_{n=2l}^{3l-1} P^n_{s,s} + ... \right) \\
&\geq \rho \left((1-\delta) + (1-\delta) \sum_{n=1}^{l} P^n_{s,s} + ... \right) \\
&= \rho \left((1-\delta) + (1-\delta)^2 + (1-\delta)^3 + ... \right) \\
&= \rho \sum_{m=1}^{\infty} (1-\delta)^m = \frac{\rho}{\delta} - 1 \approx \frac{\rho}{\varepsilon}
\end{aligned}$$

since δ can get arbitrarily close to ε. Note that ρ does not depend on ε. We proved that ξ_s is bounded from below by $\frac{\rho}{\varepsilon}$, which can grow as much as we want since ρ is a positive constant and ε can be chosen arbitrarily small.

Now let's study the local entropy of s. By the properties of entropy we have that the smallest case[2] is when it has two outgoing transitions, one with probability ε to a state outside R and the other with probability $1-\varepsilon$ to a state inside R. Thus we say that $L(s) \geq -((\varepsilon \log_2 \varepsilon) + ((1-\varepsilon)\log_2(1-\varepsilon)))$, and consequently

$$L(s)\xi_s \geq -\frac{\rho((\varepsilon \log_2 \varepsilon) + ((1-\varepsilon)\log_2(1-\varepsilon)))}{\varepsilon}$$

Note that

$$\lim_{\varepsilon \to 0} -\frac{\rho((\varepsilon \log_2 \varepsilon) + ((1-\varepsilon)\log_2(1-\varepsilon)))}{\varepsilon} = \infty$$

meaning $L(s)\xi_s$ is an increasing function as ε tends to zero, and since $H(\mathcal{C}) = \sum_{t \in S} L(t)\xi_t$ clearly for each implementation it is possible to give another with a smaller ε and finite but greater entropy, meaning no maximum entropy implementation exists. In this case we say that the entropy of $\mathcal{I}$ is *unbounded*.

[2]Since ε can be arbitrarily small, the cases in which the transitions have probability ε' and $1-\varepsilon'$ with $\varepsilon' < \varepsilon$ are included

ONLY IF: We will assume we can give implementations of the Interval MC with any entropy value and show that such a state s must exist. We will proceed ad absurdum: let's assume for any implementation it is possible to give another implementation with higher entropy for the Interval MC but for any implementation any state $s \in S_\omega$ is deterministic. Since deterministic states contribute no entropy, only states not in the end component have positive entropy; let $t \notin S_\omega$ be any one of them. From Lemma 3 we know that there is no implementation in which t is recurrent, meaning that in each implementation ξ_t is finite and $\rho(t,t) < 1$. This means that there cannot be a cycle in any implementation from t to itself with probability 1, and since we allow only closed intervals this also means that there exists a small positive δ such that in no implementation t has a cycle to itself with probability greater than $1-\delta$. ξ_t can be written as the expected residence time in t from itself multiplied by the first hitting probability ρ_t of t:

$$\xi_t = \rho_t \sum_{n=0}^{\infty} P_{t,t}^n$$

Note that:

- since ρ_t is a probability, then it is bounded from above by 1;
- since t has no path to itself with probability greater than $1-\delta$, then $\forall l \in \mathbb{N}. P_{t,t}^l \leq 1-\delta$;

thus we can write

$$\xi_t = \rho_t \sum_{n=0}^{\infty} P_{t,t}^n \leq \sum_{n=0}^{\infty} (1-\delta)^n = \frac{1}{\delta}$$

Since local entropy is bounded from above by $\log_2 |S|$ then for each state t it holds that $L(t)\xi_t \leq \frac{\log_2 |S|}{\delta}$; then $H(\mathcal{C}) = \sum_{t \in S} L(t)\xi_t \leq \frac{|S| \log_2 |S|}{\delta}$, contradicting the assumption that for any implementation it is possible to give one with higher entropy. □

Theorem 2 proves that Alg. 2 is sound and complete to verify the existence of a Maximum Entropy implementation as defined in Def. 11 for a given Interval MC $\mathcal{I}$. If such an implementation exists, its entropy is the channel capacity for all implementations of the system and can be used as an upper bound on the information leakage of all such implementations, providing a security guarantee for the protocol represented by $\mathcal{I}$. We will now show a numerical method to synthesize a Maximum Entropy implementation with arbitrary precision. In Sect. 7 we discuss the case in which such an implementation does not exist.

6.2. Synthesis of a Maximum Entropy Implementation

We have been characterizing the existence of a Maximum Entropy implementation with finite entropy, now we propose a numerical technique to synthesize it with an arbitrary precision [11]; the Maximum Entropy implementation of the Two-step Authentication in Fig. 2 has been obtained this way. We reduce the problem to solving a

multidimensional maximization on convex sets by considering each of the $|S|^2$ transition probabilities $P_{s,t}$ in the chain as different dimensions, each of which can take values in the interval $[\check{P}_{s,t}, \hat{P}_{s,t}]$, generating a convex polytope.

Due to coherence of the Interval MC there exists at least one Markov chain implementing it, so the polytope will be nonempty. We need to add to the system the constraints $\forall s \in S. \sum_{t \in S} P_{s,t} = 1$ to ensure every solution can be interpreted as an MC. Since these constraints are linear, the domain is still a convex polytope. A point in the polytope thus defines a Markov chain. The objective function to maximize is the entropy of such Markov chain, which can be calculated in PTIME as shown in Sect. 5.

This optimization problem for an everywhere differentiable function can be solved using numerical methods. Once the global maximum is found with a numerical algorithm, the parameters $P_{s,t}$ interpreted as a MC give a Maximum Entropy implementation.

Example. Consider the Two-step Authentication in Fig. 1a. The entropy of the system is $H = (-(P_{1,2} \log_2(P_{1,2})) - ((1-P_{1,2}) \log_2(1-P_{1,2})))+P_{1,2}(-(P_{2,4} \log_2(P_{2,4})) - ((1-P_{2,4}) \log_2(1-P_{2,4})))$ under constraints $0 \leq P_{1,2} \leq 1 \wedge 0 \leq P_{2,4} \leq 1$. It is maximal for $P_{1,2} = 2/3$, $P_{2,4} = 0.5$. The Maximum Entropy implementation is shown in Fig. 2.

7. Infinite vs Unbounded Entropy for Interval MCs

We now give insight about the difference between unbounded and infinite entropy for an Interval MC and give a decision procedure to distinguish the two cases.

The infinite case means that it is possible to give implementations with infinite entropy, and by Corollary 1 an implementation with infinite entropy must be non terminating. Thus this case means that the specification allows for non-terminating implementations, during which an arbitrary amount of information will be leaked.

The unbounded case means that all implementations terminate, but whatever the size of the secret, it is possible to give an implementation that leaks all of it, and thus we cannot give security guarantees for their behavior.

Consider the Repeated Authentication in Fig. 6a; since state 1 can be both recurrent and stochastic but never both, we are in the unbounded case, and in fact it is possible to give implementations with arbitrary entropy. Since the Repeated Authentication is a security scenario, this means that it is possible to give implementations that leak any amount of information about the confidential data, and thus this should be considered an insecure authentication protocol, as it is not possible to give any security guarantee for it.

In this particular case this depends on the fact that we allow the black list to be empty; in this implementation the attacker can try all possible passwords, and thus will eventually leak all of the confidential data. In Figure 6b we show a modified version in which the black list covers at least 30% of the passwords; for this case the Interval MC has a Maximum Entropy implementation, and is thus possible to give a security guarantee.

Algorithm 3 discriminates these two cases. The idea is to build an implementation that maximizes the end components (in which all states that can be stochastic are stochastic). If this implementation has stochastic states in a strongly connected component,

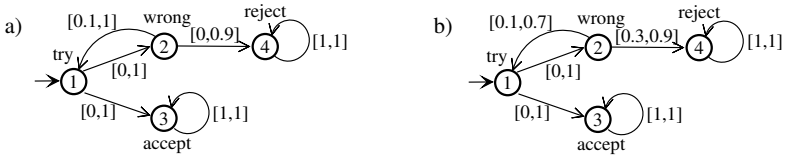

Figure 6: a) Specification for the Repeated Authentication with unbounded entropy. b) Specification for the Repeated Authentication with bounded entropy.

Find all maximal end components of the Interval MC;
Modify the transition probabilities so that all end components are closed: for each end component $\mathcal{R}$, set $\hat{P}_{s,t} = 0$ for all $s \in \mathcal{R}$, $t \notin \mathcal{R}$;
Make the Interval MC coherent again with the coherence algorithm;
if *the states of all end components are deterministic* **then**
| signal that no infinite entropy implementation exists
else
| signal that an infinite entropy implementation exists
end

Algorithm 3: Verify whether an Interval MC allows for infinite implementations.

then it will be possible to generate an infinite amount of entropy, otherwise the entropy of any implementation is always finite.

After step 2 the Interval MC will still have implementations, since by the definition of end components it is possible to give an implementation that has probability 0 of leaving the end component; we are just forcing it to happen for all our end components and checking if this makes them necessarily deterministic or not.

8. Discussion about Maximum Entropy Strategy of an MDP

We briefly discuss the usage of Markov Decision Processes as specification models, to show that the approach can be generalized to different specification theories.

As for Interval MCs, reward functions over transitions can be defined for MDPs. In an MDP $\mathcal{P} = (S, s_0, P, \Lambda)$ a non-negative reward function is a function $R : S \times \Lambda \times S \to \mathbb{R}^+$ assigning a non-negative reward value to each triple (s, a, t) such that $s, t \in S$, $a \in \Lambda_s$. The expected reward for a state-action pair (s, a) is a linear combination $R(s, a) = \sum_{t \in S} P_{(s,a),t} R_{(s,a),t}$. Finding a strategy maximizing a non-negative reward function is known as the Positive-bounded Expected Total Reward problem in Puterman's book [10].

As for Interval MCs, we want to cast entropy as a non-negative reward function over transitions of MDPs and we want to find the strategy that maximizes entropy. Alas, the reward function for entropy would have to sum up all the probabilities of going from a state s to a state t through all the actions in Λ_s, and thus cannot be cast in this form. Consider Fig. 4; the probability of transitioning from state **1** to state **2** depends on both $P_{(1,a),2}$ and $P_{(1,b),2}$ and thus we cannot compute it by assigning rewards to a transition without considering every other transition from the

same state. The reward function would have to be defined on states, becoming $R(s) = -\sum_{t \in S} \left(\left(\sum_{a \in \Lambda_s} P_{(s,a),t} \right) \log_2 \left(\sum_{a \in \Lambda_s} P_{(s,a),t} \right) \right)$, which contains a non-linear dependency on the choice of strategy ($P_{(s,a),t}$), so standard reward optimization methods do not apply.

More importantly, the Maximum Entropy strategy would not be positional. Consider the MDP model of the Two-step Authentication protocol in Fig. 7. Each strategy for assigning probabilities to actions a and b in states **1** and **2** resolves the MDP in a Markov chain, allowing us to use the MDP as a specification model. The Maximum Entropy strategy is the strategy assigning probability $\frac{2}{3}$ to *a* and $\frac{1}{3}$ to *b* in state **1** and probability $\frac{1}{2}$ to *a* and $\frac{1}{2}$ to *b* in state **2**, so this is an example of a case in which the Maximum Entropy strategy is not positional. This prevents us from using common algorithms like value iteration, policy iteration and reduction to linear programming [10, Chpt. 7] to find the optimal solution. A numerical approximation approach needs to be applied, showing the inherent hardness of the problem and validating our approximation approach in the case of Interval MCs.

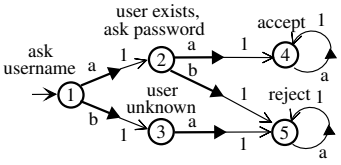

Figure 7: MDP model for the Two-step Authentication

9. Conclusion and Future Work

It can be possible to give an upper bound to the leakage of a protocol even if some of its parameters are unspecified, computing the entropy of the Maximum Entropy implementation of the specification when it exists. If such implementation does not exist, the protocol is inherently insecure and should be modified. Existence of the Maximum Entropy implementation can be determined in polynomial time in the size of the model, while its synthesis requires the solution of a nonlinear multidimensional maximization problem.

The intention is to allow implementation of intelligent tools that can automatically concretize underspecified models and give security guarantees to specifications, and eventually optimize the parameters of such models to make them more resistant to attacks. It would be interesting to extend our results (and the tool) to specifications with open intervals, and to other abstractions such as non-trivial subclasses of Constraint MCs [34] or Abstract Probabilistic Automata [35].

[1] Goguen, J.A., Meseguer, J.: Security policies and security models. In: IEEE Symposium on Security and Privacy. (1982) 11–20

[2] Ryan, P., McLean, J., Millen, J., Gligor, V.: Non-interference, who needs it? In: Computer Security Foundations Workshop, 2001. Proceedings. 14th IEEE. (2001) 237 –238

[3] Malacaria, P.: Assessing security threats of looping constructs. In: Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. POPL '07, New York, NY, USA, ACM (2007) 225–235

[4] Clark, D., Hunt, S., Malacaria, P.: A static analysis for quantifying information flow in a simple imperative language. Journal of Computer Security **15**(3) (2007) 321–371

[5] Shannon, C.E.: A mathematical theory of communication. The Bell system technical journal **27** (July 1948) 379–423

[6] Millen, J.K.: Covert channel capacity. In: IEEE Symposium on Security and Privacy. (1987) 60–66

[7] Malacaria, P.: Algebraic foundations for information theoretical, probabilistic and guessability measures of information flow. CoRR **abs/1101.3453** (2011)

[8] Biondi, F., Legay, A., Malacaria, P., Wasowski, A.: Quantifying information leakage of randomized protocols. In Giacobazzi, R., Berdine, J., Mastroeni, I., eds.: VMCAI. Volume 7737 of Lecture Notes in Computer Science., Springer (2013) 68–87

[9] Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS, IEEE Computer Society (1991) 266–277

[10] Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley-Interscience (April 1994)

[11] Stoer, J., Bulirsch, R., Bartels, R., Gautschi, W., Witzgall, C.: Introduction to Numerical Analysis. Texts in Applied Mathematics. Springer (2010)

[12] Jaynes, E.T.: Information Theory and Statistical Mechanics. Physical Review Online Archive (Prola) **106**(4) (May 1957) 620–630

[13] D. Clark, S.H..P.M.: Quantitative information flow, relations and polymorphic types. Journal of Logic and Computation, Special Issue on Lambda-calculus, type theory and natural language **18**(2) (2005) 181–199

[14] Smith, G.: On the foundations of quantitative information flow. In de Alfaro, L., ed.: FOSSACS. Volume 5504 of Lecture Notes in Computer Science., Springer (2009) 288–302

[15] Köpf, B., Basin, D.A.: An information-theoretic model for adaptive side-channel attacks. In Ning, P., di Vimercati, S.D.C., Syverson, P.F., eds.: ACM Conference on Computer and Communications Security, ACM (2007) 286–296

[16] Köpf, B., Mauborgne, L., Ochoa, M.: Automatic quantification of cache side-channels. In Madhusudan, P., Seshia, S.A., eds.: CAV. Volume 7358 of Lecture Notes in Computer Science., Springer (2012) 564–580

[17] Yasuoka, H., Terauchi, T.: Quantitative information flow - verification hardness and possibilities. In: CSF, IEEE Computer Society (2010) 15–27

[18] Alvim, M.S., Chatzikokolakis, K., Palamidessi, C., Smith, G.: Measuring information leakage using generalized gain functions. In Chong, S., ed.: CSF, IEEE (2012) 265–279

[19] Landauer, J., Redmond, T.: A lattice of information. In: CSFW. (1993) 65–70

[20] Aldini, A., Pierro, A.D.: Estimating the maximum information leakage. Int. J. Inf. Sec. **7**(3) (2008) 219–242

[21] Giacobazzi, R., Mastroeni, I.: Abstract non-interference: parameterizing non-interference by abstract interpretation. In Jones, N.D., Leroy, X., eds.: POPL, ACM (2004) 186–197

[22] Mastroeni, I., Giacobazzi, R.: An abstract interpretation-based model for safety semantics. Int. J. Comput. Math. **88**(4) (2011) 665–694

[23] Chatzikokolakis, K., Palamidessi, C., Panangaden, P.: Anonymity protocols as noisy channels. Inf. Comput. **206**(2-4) (2008) 378–401

[24] Bhargava, M., Palamidessi, C.: Probabilistic anonymity. In Abadi, M., de Alfaro, L., eds.: CONCUR. Volume 3653 of LNCS., Springer (2005) 171–185

[25] Chen, H., Malacaria, P.: Quantifying maximal loss of anonymity in protocols. In Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V., eds.: ASIACCS, ACM (2009) 206–217

[26] Malacaria, P., Chen, H.: Lagrange multipliers and maximum information leakage in different observational models. PLAS '08, New York, USA, ACM (2008) 135–146

[27] Cover, T.M., Thomas, J.A.: Elements of information theory (2. ed.). Wiley (2006)

[28] Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2) (1972) 146–160

[29] Chatterjee, K., Sen, K., Henzinger, T.A.: Model-checking omega-regular properties of interval Markov chains. In Amadio, R.M., ed.: FoSSaCS. Volume 4962 of Lecture Notes in Computer Science., Springer (2008) 302–317

[30] Kozine, I., Utkin, L.V.: Interval-valued finite Markov chains. Reliable Computing **8**(2) (2002) 97–113

[31] Girardin, V.: Entropy maximization for Markov and semi-Markov processes. Methodology and Computing in Applied Probability **6** (2004) 109–127

[32] de Alfaro, L.: Formal Verification of Probabilistic Systems. PhD thesis, Stanford (1997)

[33] de Alfaro, L.: Temporal logics for the specification of performance and reliability. In Reischuk, R., Morvan, M., eds.: STACS. Volume 1200 of Lecture Notes in Computer Science., Springer (1997) 165–176

[34] Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wasowski, A.: Constraint Markov chains. Theor. Comput. Sci. **412**(34) (2011) 4373–4404

[35] Delahaye, B., Katoen, J.P., Larsen, K.G., Legay, A., Pedersen, M.L., Sher, F., Wasowski, A.: Abstract probabilistic automata. In Jhala, R., Schmidt, D.A., eds.: VMCAI. Volume 6538 of Lecture Notes in Computer Science., Springer (2011) 324–339