Maximizing Entropy over Markov Processes*

Fabrizio Biondi[1], Axel Legay[2], Bo Friis Nielsen[3], and Andrzej Wąsowski[1]

[1] IT University of Copenhagen, Denmark {fbio,wasowski}@itu.dk
[2] INRIA Rennes, France axel.legay@inria.fr
[3] Technical University of Denmark, Lyngby, Denmark bfn@imm.dtu.dk

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 an 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

Quantified Information Flow [7] is a quantitative approach to compute 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.

Leakage is defined as the difference between the attacker's information [17] about the confidential data before and after the attack. If we view the system as a channel through which the attacker gets information about the secret, its *capacity* can be computed as the maximum leakage over all prior infromations of attackers. This provides a security guarantee for the system, as no attack can leak an amount of information higher than the system's channel capacity [15]. For a deterministic system, the leakage is the entropy of the observable behavior of the system [13], and thus computing the channel capacity reduces to computing the behavior of the system that maximizes entropy.

Our goal is to develop theories and algorithms to 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. We use Markov chains (MCs) as process models and Interval Markov Chains (Interval

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

MCs) [11] as specification models. We use the continuity of the real-valued intervals to encode the infinite number of possible attackers and implementations we want to consider.

In the theoretical sense, in this paper we extend the well know Maximum Entropy Principle of Jaynes [10], 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.

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 [16, 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 [18]. 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 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 process is the channel capacity of all processes implementing the Interval MC.

Motivating Examples. Consider two examples of models of deterministic authentication processes. Figure 1a presents a specification of a two-step authentication protocol. A user is requested to input a username, and is rejected if the username is unknown. If it is correct the user is asked to input a password, and is accepted if the password corresponds to the username and rejected otherwise. 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. 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.

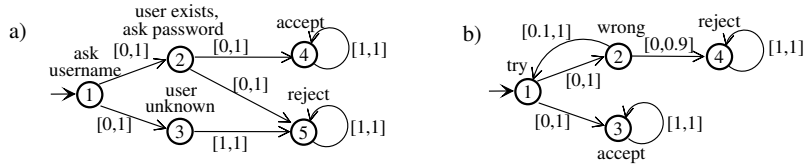

Fig. 1. a) Two-step Authentication b) Repeated Authentication

Consider another example. Figure 1b presents an Interval MC specification of the Repeated Authentication protocol. A 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 selection of passwords.

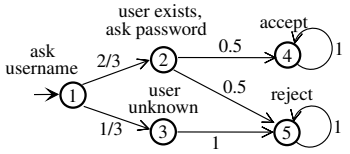

Fig. 2. Maximum Entropy implementation for the Two-step Authentication

Note that the transition probabilities from state 2 depend on a design choice left to the implementer of the system and on the attacker's knowledge about it, while the transition probabilities from state 1 depend on the length of the password and the attacker's knowledge about it. By abstracting these 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.

Implementations with higher entropy reveal more information about the system's secret. This is consistent with our intuition. For instance, in the example in Fig 1b decreasing the black list, which decreases probability of reject, increases the possible information leakage. 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. So, a larger black list increases the chance that the system will enter the absorbing state 4 and leak less information, and symmetrically a smaller black list increases the leakage.

In fact, it is not possible to give a Maximum Entropy implementation for such protocol. Whatever is the length n of the secret, it is possible to give an implementation that leaks all n bits of information. We show how these undesirable cases can be recognized in polynomial time and how the specification can be modified to avoid them.

Figure 3bc shows two implementations of the Repeated Authentication presented in Fig. 3a. None of them maximizes entropy. In fact, it is not possible to give a Maximum Entropy implementation for such protocol. We will discuss the significance of this in Sect. 6. The Maximum Entropy implementation for the Two-step Authentication is given in Fig. 2. We explain how it has been synthesized in Sect. 5.

Related Work. Channel capacity as a security guarantee [15] 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 [5], based on the probabilistic anonymity approach by Bhargava and Palamidessi [2]. Chen and Malacaria generalize this result to asymmetric protocols [6] and also study the channel capacity of deterministic systems under different observation models [14]. 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.

2 Background on Probabilistic Processes

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 abuse the 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 [8] 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 P^k. We call $\pi^{(k)}$ the probability distribution vector over S at time k and $\pi_s^{(k)}$ the probability of visiting the state s at time k; note that $\pi^{(k)} = \pi_0 P^k$, where $\pi_s^{(0)}$ is 1 if $s = s_0$ and 0 otherwise. 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. Let ξ_s denote the *residence time* in a state s: $\xi_s = \sum_{n=0}^{\infty} P_{s_0,s}^n$.

For the purpose of defining entropy, it is useful to consider the alternative, less automata-theoretical but more probabilistic, view of an MC. An MC can be seen as an infinite sequence of discrete random variables $(X_n, n \in \mathbb{N})$, where $\mathbf{P}(X_k = s) = \pi_s^{(k)}$ represents the probability that the chain will be visiting state $s \in S$ at time k. The processes must respect the *Markov property*: $P(X_n = s_n \mid X_{n-1} = s_{n-1}, \ldots, X_0 = s_0) = P(X_n = s_n \mid X_{n-1} = s_{n-1}), \forall s_0, s_1, ..., s_n \in S, n \in \mathbb{N}$.

A state s is *recurrent* iff $\xi_s = \infty$, *transient* otherwise. Residence time of each state of an MC can be calculated in polynomial time [16].

Usage of Markov chains to model generic secret-dependent processes has been previously introduced by the authors [3], 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 [3] for the full discussion.

Definition 2. [4] *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 $\check{P}_{s,t} \leq \hat{P}_{s,t}$.*

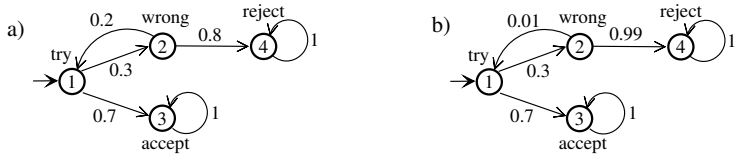

Fig. 3. a) Correct implementation of the Repeated Authentication. b) Incorrect implementation of the Repeated Authentication.

The following defines when an MC implements an Interval MC in the Uncertain Markov Chain (UMC) semantics [4]:

Definition 3. *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 of Fig. 1b. The MC in Fig. 3a uses a black list with 80% of the passwords and is thus an implementation of the Interval MC, while the Markov chain in Figure 3b uses a black list with 99% and it is not an implementation of the Interval MC.

We assume 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 [12]:
1) $\forall s, t \in S. \check{P}_{s,t} \geq (1 - \sum_{u \neq t} \hat{P}_{s,u})$ 2) $\forall s, t \in S. \hat{P}_{s,t} \leq (1 - \sum_{u \neq t} \check{P}_{s,u})$

Assuming coherence is not a limitation. If an Interval MC $\mathcal{I} = (S, s_0, \check{P}, \hat{P})$ is not coherent it can be made coherent in polynomial time [12]; we produce the 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 $\mathcal{I}$ [12], so in particular it has an implementation iff $\mathcal{I}$ has at least one implementation.

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.

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.

3 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. For a discrete set of n events $(x_1, ..., x_n)$ *entropy* is defined as $-\sum_{i=1}^{n} \mathbf{P}(x_i) \log_2(\mathbf{P}(x_i))$ and is maximal for the uniform distribution, in which case its value is $\log_2 n$ [8]. 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* [17, 8]. 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 $Y = t$, $X_1 = s_1, \ldots, X_n = s_n$.

Joint entropy is simply the entropy of several random variables computed jointly, i.e. the combined uncertainty due to the ignorance of n random variable. It turns out [8] 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 4. *We define the entropy of a Markov chain $\mathcal{C} = (X_n, n \in \mathbb{N})$ as the joint entropy over all X_n: $H(\mathcal{C}) = H(X_0, X_1, X_2, \ldots) = \sum_{i=0}^{\infty} H\,(X_i \,|X_{i-1} \ldots X_0\,)$.*

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 [13]. 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; we will give a characterization in Corollary 6 in the next Section showing 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 [17, 9] to compute the average entropy of each step of the MC and to call it the entropy of the MC, while it's technically an entropy rate [8]. Even though entropy rate is always finite, we want to compute the actual entropy since it represents the information leakage in a security scenario where the states of the MC are the observables of a deterministic program.

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 the value of entropy of an MC is in general not equal to the sum of the local entropy values for each state. Such sum will have to be weighted against the residence time of each state to characterize the "global" entropy.

Now consider the Interval MC specification of the Repeated Authentication 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 entropy of any other implementation (if such exists). The boundedness and synthesis of the Maximum Entropy implementation of an Interval MC will be treated in Sect. 5. 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. 6.

4 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 $R(\mathcal{C}) = \sum_{s \in S} R(s)\xi_s$ [16, 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 residence time for a state s is in PTIME, 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 it depends on the choice of probability distribution, 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 5. *For an MC $\mathcal{C} = (S, s_0, P)$ we have that $H(\mathcal{C}) = \sum_{s \in S} L(s)\xi_s$*

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 6. *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 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 6 we characterize the processes having finite entropy as terminating and the processes having infinite entropy as non-terminating.

5 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. 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 of $\log_2 3 \approx 1.58496$ 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 such an implementation is a challenging problem. In the first place, such an implementation may not exist, so we need an algorithm to verify its existence. Secondly, even if it exists finding it consist in solving a nonlinear optimization problem with constraints over an infinite domain.

In this section we present a new algorithm that given an Interval MC $\mathcal{I}$ finds its implementation, in the sense of Def. 3, that maximizes the entropy value. 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 reward function $R(s,t) = -\log_2(P_{s,t})$ is a Maximum Entropy implementation.

The expected reward of a non-negative 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 in some implementations 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—the logarithm is taken of the value that depends on the probability distribution of the implementation. 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.

5.1 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 [1] for Interval MCs. Then we show the algorithm for deciding the existence of a Maximum Entropy implementation.

We propose a definition of an end component for Interval MCs. 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 [1] for further discussion.

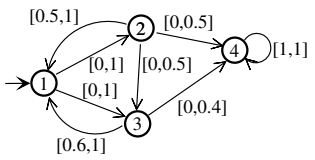

Fig. 4. Interval MC with multiple end components

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

Definition 7. *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*
1) R is strongly connected; 2) $\forall s \in R, t \in S \backslash R. \check{P}_{s,t} = 0$*; 3)* $\forall s \in R. \sum_{u \in R} \hat{P}_{s,u} \geq 1$.

An end component is maximal if no other end component contains it. In the Interval MC pictured in Fig. 4 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.

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.

Lemma 8. *Algorithm 1 runs in polynomial time, and upon termination precisely the states that are part of any maximal end component are tagged as* ENDCOMPONENTSTATE*, while the remaining states are tagged as* TRANSIENT.

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.

The algorithm to establish finiteness of maximum entropy across all implementations of an Interval MC follows these steps:

Make the Interval MC coherent (see Sect.2);
Find the maximal end components of the Interval MC and call their union S_ω;
If there is a stochastic state in S_ω, then no Maximum Entropy implementation exists.

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.

The following theorem states that the above approach to deciding existence of finite maximum entropy implementation is sound and complete:

Theorem 9. *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.*

5.2 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 [18]; 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 a MC. Since these constraints are linear, the domain is still a convex polytope. A point in the

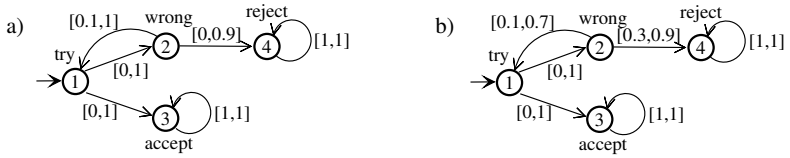

Fig. 5. a) Specification for the Repeated Authentication with unbounded entropy. b) Specification for the Repeated Authentication with bounded entropy.

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. 4.

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.

6 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 non-terminating implementations, while the unbounded case means that all implementations terminate but may leak the whole secret, and thus we cannot give security guarantees for their behavior.

Consider the Repeated Authentication in Fig. 5a; 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 5b 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.

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

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 all states in all end components of the coherent Interval MC are deterministic, then the original Interval MC does not allow infinite entropy implementations; else it does.

After step 2 the Interval MC will still have implementations, since by the definition of end components it's 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.

References

1. de Alfaro, L.: Formal Verification of Probabilistic Systems. Ph.D. thesis, Stanford (1997)
2. Bhargava, M., Palamidessi, C.: Probabilistic anonymity. In: Abadi, M., de Alfaro, L. (eds.) CONCUR. LNCS, vol. 3653, pp. 171–185. Springer (2005)
3. Biondi, F., Legay, A., Malacaria, P., Wąsowski, A.: Quantifying information leakage of randomized protocols. To appear in VMCAI 2013; available at www.itu.dk/people/fbio/vmcai2013.pdf (2013)
4. Chatterjee, K., Sen, K., Henzinger, T.: Model-checking ω-regular properties of interval markov chains. In: FoSSaCS (2008)
5. Chatzikokolakis, K., Palamidessi, C., Panangaden, P.: Anonymity protocols as noisy channels. In: Information and Computation. Springer (2006)
6. Chen, H., Malacaria, P.: Quantifying maximal loss of anonymity in protocols. In: Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R. (eds.) ASIACCS. ACM (2009)
7. Clark, D., Hunt, S., Malacaria, P.: A static analysis for quantifying information flow in a simple imperative language. Journal of Computer Security 15 (2007)
8. Cover, T., Thomas, J.: Elements of information theory. Wiley, New York (1991)
9. Girardin, V.: Entropy maximization for markov and semi-markov processes. Methodology and Computing in Applied Probability 6, 109–127 (2004)
10. Jaynes, E.T.: Information Theory and Statistical Mechanics. Physical Review Online Archive (Prola) 106(4), 620–630 (May 1957)
11. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS. pp. 266–277. IEEE Computer Society (1991)
12. Kozine, I., Utkin, L.V.: Interval-valued finite markov chains. Reliable Computing 8(2), 97–113 (2002)
13. Malacaria, P.: Algebraic foundations for information theoretical, probabilistic and guessability measures of information flow. CoRR abs/1101.3453 (2011)
14. Malacaria, P., Chen, H.: Lagrange multipliers and maximum information leakage in different observational models. pp. 135–146. PLAS '08, ACM, New York, USA (2008)
15. Millen, J.K.: Covert channel capacity. In: IEEE Symposium on Security and Privacy. pp. 60–66 (1987)
16. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley-Interscience (Apr 1994)
17. Shannon, C.E.: A mathematical theory of communication. The Bell system technical journal 27, 379–423 (Jul 1948)
18. Stoer, J., Bulirsch, R., Bartels, R., Gautschi, W., Witzgall, C.: Introduction to Numerical Analysis. Texts in Applied Mathematics, Springer (2010)