Proceedings Paper:
Dimitrova, R., Finkbeiner, B., Kovacs, M. et al. (2 more authors) (2012) Model checking information flow in reactive systems. In: Kuncak, V. and Rybalchenko, A., (eds.) Verification, Model Checking, and Abstract Interpretation (VMCAI 2012). Verification, Model Checking, and Abstract Interpretation - VMCAI 2012, 22-24 Jan 2012, Philadelphia, PA, USA. Lecture Notes in Computer Science (7148). Springer , pp. 169-185. ISBN 9783642279393

https://doi.org/10.1007/978-3-642-27940-9_12

Model Checking Information Flow in Reactive Systems

Rayna Dimitrova[1], Bernd Finkbeiner[1], Máté Kovács[2], Markus N. Rabe[1], and Helmut Seidl[2]

[1] Universität des Saarlandes, Germany
[2] Technische Universität München, Germany

Abstract. Most analysis methods for information flow properties do not consider temporal restrictions. In practice, however, such properties rarely occur statically, but have to consider constraints such as *when and under which conditions* a variable has to be kept secret. In this paper, we propose a natural integration of information flow properties into linear-time temporal logics (LTL). We add a new modal operator, the hide operator, expressing that the observable behavior of a system is independent of the valuations of a secret variable. We provide a complexity analysis for the model checking problem of the resulting logic SecLTL and we identify an expressive fragment for which this question is efficiently decidable. We also show that the path based nature of the hide operator allows for seamless integration into branching time logics.

1 Introduction

Temporal logics are well-suited for specifying classical requirements on the behavior of reactive systems. The key to the success of automated verification methods for temporal logics is the rich set of automata-theoretic techniques [1–3]. Based on these theoretical foundations, efficient model-checkers that are capable of verifying intricate properties have emerged over the last two decades. Reactive systems, however, often are not only safety-critical but also *security-critical*. Examples of reactive systems handling confidential information include communication protocols, cell phone apps, and document servers.

Information flow properties are of great importance in the realm of security-critical systems. Information flow summarizes properties that argue about the transfer of information from a secret source towards an observer or attacker. Notable examples of such properties are non-interference [4] and observational determinism [5], which require that no information is leaked in a strict sense. For many practical applications, however, requiring that information is kept secret forever is too strong: often secrets may (or even must) be released under certain conditions. The controlled release of information is called *declassification* [6, 7].

This work was partially supported by the German Research Foundation (DFG) under the project SpAGAT (grant no. FI 936/2-1) in the priority program "Reliably Secure Software Systems – RS3".

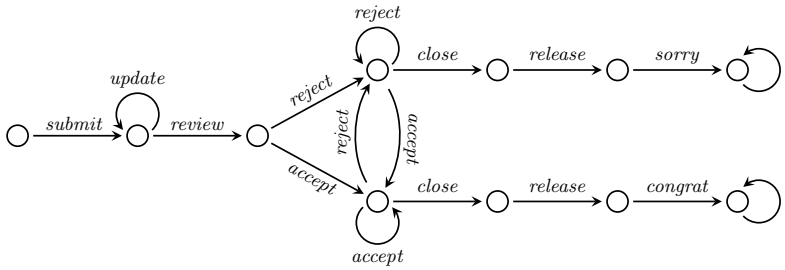

Fig. 1. Model of a conference management system. An author can submit a paper and later receive a notification of whether the paper was accepted or rejected via the output variables *congrat* and *sorry*, which he or she can observe.

For reactive systems it is typical that secrecy requirements vary over time, depending on the interaction of the system with its environment. For example access rights are seldom static, and secret data may be released under certain conditions. Therefore, it is imperative to consider information flow properties in their temporal context and integrate them in the theory of temporal logics.

A typical example for a security-critical reactive system is a conference management system. A minimalistic model of such a system is given in Fig. 1. Two properties of interest for this system are: (1) *"The final decision of the program committee remains secret until the notification"* and (2) *"All intermediate decisions of the program committee are never revealed to the author"*. These two information flow properties can be informally specified as follows:

(1) **last** *accept/reject* **before** *close* **remains secret until** *release* and
(2) **all** *accept/reject* **except the last before** *close* **remain secret forever**.

The above properties illustrate the two temporal aspects of information flow properties. Firstly, they specify *at which points in time a variable is considered secret*, e.g., "last before" or "all except last before". Secondly, they specify *for how long certain information should remain secret*, e.g., "forever" or "until release". Despite their obvious temporal nature, these properties cannot be expressed in classical temporal logics like LTL (Linear-time Temporal Logic), CTL (Computation Tree Logic), or even the μ-calculus [8].

The reason is that most information flow properties have structural differences to classical temporal properties. While the latter are interpreted on a single execution (in the linear-time case) or on the execution tree of a system (in the branching-time case), information flow properties require the comparison of multiple executions.

Contribution. In this paper we introduce a new modal operator $\mathcal{H}$ (*hide*), that expresses the requirement that the observable behavior of a system is independent of the choice of a secret. The novelty is the integration into the temporal context—the operator itself is evaluated over a single path, but tracks the alternative paths the system could take for different valuations of the secret.

Extending LTL with the operator $\mathcal{H}$ (Section 3) yields a powerful—yet decidable—logic called *SecLTL*. We provide an automata-theoretic verification

technique for SecLTL that extends the standard approach for LTL. We establish PSPACE-completeness of the model checking problem for SecLTL both in the size of the specification and in the size of the system under scrutiny.

We identify a fragment, *Restricted SecLTL*, for which the model checking problem is efficiently solvable: it is in NLOGSPACE with respect to the size of the system (Section 4). What makes Restricted SecLTL of practical relevance is the combination of efficiency and expressiveness. It is able to capture properties like non-interference and observational determinism.

The path-based semantics of the hide operator enables seamless integration in branching time logics (Section 5). We define the logics SecCTL and SecCTL* and determine the complexity of the corresponding model checking problems. Surprisingly, even for SecCTL the model checking problem is PSPACE-complete.

2 Preliminaries

In this section we introduce the system model we consider throughout the paper: transition systems whose edges are labeled with valuations of their input and output variables. The external behavior of such a system consists of the infinite sequences of labels during the possible executions. The temporal properties we specify are over such behavior paths and are consequently translated to automata over infinite words over the same alphabet.

For a finite set $\mathcal{V}$ of binary variables, we denote with $\mathsf{vals}(\mathcal{V})$ the set of all possible valuations of the variables in $\mathcal{V}$, i.e., all total functions from $\mathcal{V}$ to $\mathbb{B}$. For $a \in \mathsf{vals}(\mathcal{V})$ and $V \subseteq \mathcal{V}$ we use $a|_V$ to note the projection of a on the set V.

For a set A, A^* is the set of all finite sequences of elements of A and A^ω is the set of all infinite sequences of elements of A. For a finite or infinite sequence π of elements of A and $i \in \mathbb{N}$, $\pi[i]$ is the $(i+1)$-th element of π, $\pi[0, i)$ is the prefix of π of up to (excluding) position i, $\pi[0, i]$ is the prefix of π up to (including) position i and, if π is infinite, $\pi[i, \infty)$ is its infinite suffix starting at position i. For a finite sequence $\pi \in A^*$, we denote its length with $|\pi|$.

Definition 1 (Transition system). *A* transition system *(Mealy machine) $M = (\mathcal{V}_\mathcal{I}, \mathcal{V}_\mathcal{O}, S, s_0, \delta)$ consists of a finite set of states S, an initial state s_0, two disjoint finite sets of binary variables, the input variables $\mathcal{V}_\mathcal{I}$ and the output variables $\mathcal{V}_\mathcal{O}$, and a transition function that is a partial function $\delta : S \times \Sigma \to S$, where the alphabet $\Sigma = \mathsf{vals}(\mathcal{V}_\mathcal{I} \cup \mathcal{V}_\mathcal{O})$ is the set of valuations of the input and output variables. We define the size of a transitions system as $|M| = |S| + |\Sigma|$.*

We consider input enabled systems*, that is, we require for every $s \in S$ and $i \in \mathsf{vals}(\mathcal{V}_\mathcal{I})$ that there exists an $a \in \Sigma$ with $a|_{\mathcal{V}_\mathcal{I}} = i$ such that $\delta(s, a)$ is defined.*

Definition 2 (Transition function δ^*_M). *We extend the transition function of a transition system M to partial labels: $\delta^*_M : S \times \mathsf{vals}(V) \to 2^{\Sigma \times S}$, where $V \subseteq \mathcal{V}_\mathcal{I} \cup \mathcal{V}_\mathcal{O}$, $\delta^*_M(s, v) = \{(a, s') \in \Sigma \times S \mid a|_V = v \text{ and } \delta(s, a) = s'\}$.*

Definition 3 (Path, execution). Paths *of a transition system M are infinite sequences of labels: $\pi = a_0, a_1, \ldots$, with $a_i \in \Sigma$. Given a state $s \in S$, each path*

π is associated with a unique finite or infinite sequence of states, $s_0, \ldots, s_n$ or $s_0, s_1, \ldots$, called execution of M from s on π *and denoted $\mathsf{Exec}_M(s, \pi)$, such that $s_0 = s$ and $s_{i+1} = \delta(s_i, a_i)$ for all $i \geq 0$. The execution is unique, since the transition function is a function and might be finite since this function is partial.*

Given a state s, we denote the set of possible infinite paths in M by $\mathsf{Paths}_{s,M}$. Note that for every $\pi \in \mathsf{Paths}_{s,M}$, the execution $\mathsf{Exec}_M(s, \pi)$ is infinite.

Definition 4 (Observational equivalence). *Given a set of variables $V \subseteq \mathcal{V}_\mathcal{I} \cup \mathcal{V}_\mathcal{O}$, we define two valuations $a, a' \in \Sigma$, to be* observationally equivalent w.r.t. *V, noted $a =_V a'$, if the valuations' projections to the variables in V is the same: $a|_V = a'|_V$. Pairwise comparison immediately provides us with a notion of observational equivalence on paths.*

For a finite set Q, $\mathcal{B}^+(Q)$ is the set of positive boolean formulas over Q. These are formulas built from the formulas true, false and the elements of Q using $\wedge$ and $\vee$. For $\theta \in \mathcal{B}^+(Q)$ and a set $K \subseteq Q$ we write $K \models \theta$ if K satisfies θ.

A *tree* T is a subset of $\mathbb{N}_{>0}^*$ such that for every *node* $\tau \in \mathbb{N}_{>0}^*$ and every positive integer $n \in \mathbb{N}_{>0}$, if $\tau \cdot n \in T$ then the following hold:

- $\tau \in T$ (i.e., T is prefix-closed) and there is an edge from τ to $\tau \cdot n$, and
- for every $m \in \mathbb{N}_{>0}^*$ with $m < n$ it holds that $\tau \cdot m \in T$.

The root of T is the empty sequence ε and for a node $\tau \in T$, $|\tau|$ is the distance of τ from the root. A *Q-labeled tree* is a tuple (T, r), where T is a tree and the function $r : T \to Q$ labels every node with an element of Q.

Definition 5 (Alternating Büchi automaton). *An* alternating Büchi automaton *is a tuple $\mathcal{A} = (Q, q_0, \Sigma, \rho, F)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ is a finite alphabet, $\rho : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function that maps a state and a letter to a positive boolean combination of states, and $F \subseteq Q$ is a set of accepting states.*

A run *of $\mathcal{A}$ on an infinite word $\pi \in \Sigma^\omega$ is a Q-labeled tree (T, r) such that $r(\varepsilon) = q_0$ and for every node τ in T with children $\tau_1, \ldots, \tau_k$ it holds that $k \leq |Q|$ and $\{r(\tau_1), \ldots, r(\tau_k)\} \models \rho(q, \pi[i])$, where $q = r(\tau)$ and $i = |\tau|$.*

A run r of $\mathcal{A}$ on $\pi \in \Sigma^\omega$ is accepting *iff for every infinite path $\tau_0\tau_1 \ldots$ in T, $r(\tau_i) \in F$ for infinitely many $i \in \mathbb{N}$. We denote with $L_\omega(\mathcal{A})$ the set of infinite words in Σ^ω* accepted *by $\mathcal{A}$, i.e., for which there exists an accepting run of $\mathcal{A}$. For a state $q \in Q$, we note $L_\omega(\mathcal{A}, q) = L_\omega(\mathcal{A}_q)$, where $\mathcal{A}_q = (Q, q, \Sigma, \rho, F)$.*

Definition 6 (Nondeterministic Büchi automaton). *A* nondeterministic Büchi automaton *is an alternating Büchi automaton $\mathcal{N} = (Q, q_0, \Sigma, \rho, F)$ for which the transition formula $\rho(q, a)$ for each $q \in Q$ and $a \in \Sigma$ does not contain $\wedge$. Thus, for a nondeterministic Büchi automaton we can represent the transition function ρ as a function $\rho : Q \times \Sigma \to 2^Q$.*

A run of $\mathcal{N}$ on an infinite word $\pi \in \Sigma^\omega$ is an infinite sequence $\tau \in Q^\omega$ such that $\tau[0] = q_0$ and for every $i \in \mathbb{N}$, $\tau[i+1] \in \rho(\tau[i], \pi[i])$.

3 The Temporal Logic SecLTL

The logic SecLTL extends LTL with the *hide operator* $\mathcal{H}$. The SecLTL formulas over a set of variables $\mathcal{V} = \mathcal{V}_\mathcal{I} \dot\cup \mathcal{V}_\mathcal{O}$ are defined according to the following grammar, where $v \in \mathcal{V}$, φ and ψ are SecLTL formulas, $H \subseteq \mathcal{V}_\mathcal{I}$ and $O \subseteq \mathcal{V}_\mathcal{O}$,

$$\varphi \quad ::= \quad v \quad | \quad \neg\varphi \quad | \quad \varphi \vee \psi \quad | \quad \bigcirc\varphi \quad | \quad \varphi \,\mathcal{U}\, \psi \quad | \quad \mathcal{H}_{H,O}\varphi.$$

Additionally, we introduce the common abbreviations $\mathsf{true} = v \vee \neg v$, $\mathsf{false} = \neg\mathsf{true}$, $\diamondsuit\varphi = \mathsf{true}\,\mathcal{U}\,\varphi$, $\square\varphi = \neg\diamondsuit\neg\varphi$, and $\varphi\mathcal{W}\psi = \varphi\,\mathcal{U}\,\psi \vee \square\varphi$.

Intuitively, the operator $\mathcal{H}_{H,O}\varphi$ requires that the observable behavior of the system does not depend on the initial values of the *secret variables* H before the formula φ is satisfied. The operator also allows to specify the power of the observer, by choosing an appropriate set O of *observable variables* or *outputs*. That is, the hide operator specifies what is to be considered the secret, what we consider to be observable, and when the secret may be released.

What may seem a little odd initially, that we only consider the *first valuation* of the H-variables to be secret, is actually one of the strengths of SecLTL. It allows us, to precisely characterize the secret by using the hide operator within an appropriate temporal context. For example, we can express the temporal information flow properties from our motivating example in the introduction, as we demonstrate in Section 3.1.

Although SecLTL specifications are path properties, their semantics, more precisely the semantics of the hide operator, is defined using a *set of alternative paths* and involves comparison of each of these paths to the *main path*, i.e., the path over which the SecLTL formula is interpreted.

Definition 7 (Alternative paths). *The* set of alternative paths *for a given path $\pi \in \Sigma^\omega$ and a given state $s \in S$ with respect to a set of variables $H \subseteq \mathcal{V}$ is the set of paths starting in state s with a possibly different valuation of the secret variables H in the first position but otherwise adhering to the same input values.*

$$\mathsf{AltPaths}_M(s,\pi,H) = \{\, \pi' \in \mathsf{Paths}_{s,M} \mid \pi[0] =_{\mathcal{V}_\mathcal{I}\setminus H} \pi'[0], \text{ and } \pi[1,\infty) =_{\mathcal{V}_\mathcal{I}} \pi'[1,\infty) \,\}.$$

Definition 8 (Semantics of SecLTL). *Let $M = (\mathcal{V}_\mathcal{I}, \mathcal{V}_\mathcal{O}, S, s_0, \delta)$ be a transition system and $\Sigma = \mathsf{vals}(\mathcal{V}_\mathcal{I} \cup \mathcal{V}_\mathcal{O})$. An infinite path $\pi \in \mathsf{Paths}_{s,M}$ for some state $s \in S$ and the state s satisfy a SecLTL formula φ, denoted $M, s, \pi \models \varphi$ when the following conditions are satisfied:*

- *if $\varphi = v$ for some $v \in \mathcal{V}$, then $M, s, \pi \models \varphi$ iff $\pi[0]|_v$ is true;*
- *if $\varphi = \neg\varphi'$, then $M, s, \pi \models \varphi$ iff $M, s, \pi \not\models \varphi'$;*
- *if $\varphi = \varphi_1 \vee \varphi_2$, then $M, s, \pi \models \varphi$ iff $M, s, \pi \models \varphi_1$ or $M, s, \pi \models \varphi_2$,*
- *if $\varphi = \bigcirc\varphi'$, then $M, s, \pi \models \varphi$ iff $M, s', \pi[1,\infty) \models \varphi'$ where $s' = \delta(s, \pi[0])$,*
- *if $\varphi = \varphi_1 \mathcal{U} \varphi_2$, then $M, s, \pi \models \varphi$ iff for some $i \geq 0$, we have $M, \sigma[i], \pi[i,\infty) \models \varphi_2$ and for all j with $0 \leq j < i$ we have $M, \sigma[j], \pi[j,\infty) \models \varphi_1$, where $\sigma = \mathsf{Exec}_M(s,\pi)$.*

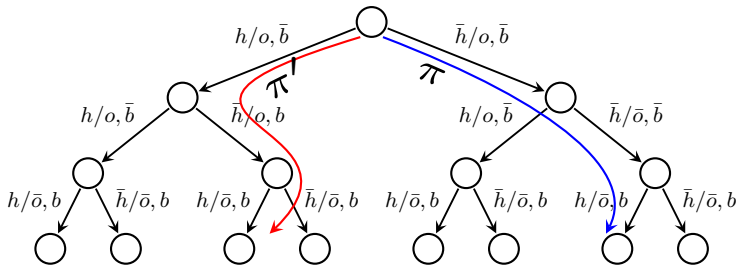

Fig. 2. Consider the formula $\varphi = \mathcal{H}_{\{h\},\{o\}} b$. This figure displays a computation tree of a system and a path π with its (in this case single) alternative path π'. The formula φ holds on π if π is observably equivalent (here, that is, equivalent with respect to variable o) to π' until b holds on π. Note that the occurrence of b in step 2 of path π' does not affect the evaluation of φ on path π—thus π violates property φ. The other path π', however, satisfies the property, as the comparison stops after the first step.

– *if* $\varphi = \mathcal{H}_{H,O}\psi$*, then* $M, s, \pi \models \varphi$ *iff for every* $\pi' \in \mathsf{AltPaths}_M(s, \pi, H)$ *it holds that* $\pi =_O \pi'$ *or there exists* $i \in \mathbb{N}$ *such that* $M, \sigma[i], \pi[i, \infty) \models \psi$ *and* $\pi[0, i) =_O \pi'[0, i)$*, where* $\sigma = \mathsf{Exec}_M(s, \pi)$*.*

We say that a transition system M *satisfies a SecLTL formula* φ*, denoted* $M \models \varphi$*, iff* $M, s_o, \pi \models \varphi$ *for every* $\pi \in \mathsf{Paths}_{s_0,M}$*.*

Using the operator $\mathcal{H}$ we can specify secrecy requirements within a temporal context. It *generates a secret at a single point* partitioning the input variables into *public* and *secret* variables. Thus we can use the standard LTL operators to capture the temporal aspect in that respect, i.e., *when are secrets introduced*. The hide operator is a temporal operator with a "weak until flavor" that captures requirements about for *how long the secret should be kept*.

The examples below demonstrate that these features enable the specification of a rich set of temporal information flow properties for reactive systems.

3.1 Examples

Example 1 (Non-interference and Observational determinism). Classical non-interference and related notions are fundamental security properties that any logic designed as a specification language for information flow requirements should be able to express. Classical non-interference as defined for (output) deterministic systems by Gougen and Meseguer [4] requires that the system's observable behavior may not change if the high user's actions would not have been issued. Modeling the high actions h_i and low actions l_i as input variables that are true iff the action is performed and defining O as the set of observable variables, we can translate [9] non-interference to our setting as follows:

$$NI(M) = \left\{\pi \in \mathsf{Paths}_{s_0,M} \mid \forall \pi' \in \mathsf{Paths}_{s_0,M} : \ \pi' =_H \vec{0} \ \wedge \ \pi =_L \pi' \Rightarrow \pi =_O \pi'\right\}$$

where $H = \bigcup_i h_i$ and $L = \bigcup_i l_i$. That is, we compare all paths with non-zero high inputs to their counterpart having only zero high inputs. By symmetry and transitivity this compares all paths having the same low inputs to each other.

This property can be expressed in SecLTL with the following formula:

$$\varphi_{NI(M)} = \square\, \mathcal{H}_{H,O}\, \mathsf{false}.$$

While a single hide operator only hides the first valuation of the secret variable, using it in combination with the globally operator ($\square$) has the effect that for all steps the valuations of the variables H are considered secret. In this case, the subformula of the hide operator is false, which means that the comparison will never stop—the secrets must be kept forever.

Zdancewic and Myers [5] generalize non-interference to systems that are *not* output deterministic. The resulting property, *observational determinism*, states that for all possible computations (paths) π and π' the observations must be indistinguishable: $\pi =_O \pi'$. Note that the model in [5] does not allow for low input, but we can easily extend it by such:

$$OD(M) = \left\{\pi \in \mathsf{Paths}_{s_0,M} \mid \forall \pi' \in \mathsf{Paths}_{s_0,M} :\ \pi =_L \pi' \implies \pi =_O \pi'\right\}$$

As it is easy to see, this property is also captured by the formula $\varphi_{NI(M)}$ above.

There are several other approaches, all with different semantics, that generalize non-interference to not output deterministic systems. We decided to follow the approach of Zdancewic and Myers as we consider it to be conservative.

Example 2 (Conference management system). Consider the model of a conference management system depicted on Fig. 1. The information flow properties informally specified there can be specified as follows:

$$(1)\quad \square\left((\bigcirc \mathit{close}) \Rightarrow \mathcal{H}_{H,O}\, \mathit{release}\right)$$

$$(2)\quad \square\left((\neg\bigcirc \mathit{close}) \Rightarrow \mathcal{H}_{H,O}\, \mathsf{false}\right).$$

where $H = \{\mathit{accept}, \mathit{reject}\}$ and $O = \{\mathit{congrat}, \mathit{sorry}\}$.

Here, the set H in the $\mathcal{H}$ subformulas specifies that the variables whose values in the corresponding point of time constitute the secret are *accept* and *reject*, and the set $O = \{\mathit{congrat}, \mathit{sorry}\}$ means that the observer, in this case the author, can observe all output variables of this system.

The subformula *release* of the $\mathcal{H}$ subformula in (1) specifies that the secret may be released as soon as *release* is satisfied, while in (2) the false in the $\mathcal{H}$ subformula requires that the secret is never released (as false is never satisfied).

Example 3 (Combination of path properties). Using combinations of path properties, we can rule out certain leaks in the analysis, which allows to analyze different sources of secrets in separation. To rule out security violations other than the obvious copy-operation from high to low variables, we can require that the hide operator is only evaluated on paths that do not show such behavior:

$$(\neg\mathsf{readhigh}\ \mathcal{W}\ \mathsf{writelow}) \Rightarrow \mathcal{H}_{H,O}\mathsf{false}.$$

As side conditions we need to require that $\mathsf{readhigh}$ and $\mathsf{writelow}$ are not part of the set of variables H, as in this case the hide operator would explore alternatives with different valuations for these variables.

Example 4 (Auction). We consider the bounded creation of secrets as one of the strengths of SecLTL. For example, we can express that all bids submitted before closing an auction are kept secret until the winner is announced:

$$(\mathcal{H}_{\mathsf{bids},O}\ \mathsf{winnerAnnounced})\ \mathcal{U}\ \mathsf{closingAuction}.$$

Example 5 (Key retrieval). SecLTL also enables specifications that argue about more than one different secrets. The following property specifies that on every path at most one of the two secrets can be compromised:

$$(\mathcal{H}_{\{k_1\},O}\ \mathsf{false})\ \vee\ (\mathcal{H}_{\{k_2\},O}\ \mathsf{false}).$$

This does not prevent the leakage of either secret for all paths but only prevents per path that both secrets are leaked.

Example 6 (Nesting). Nesting can be used to express that a secret (e.g. a key) may not be leaked until the generation of a second secret that is secure:

$$\mathcal{H}_{\{k_1\},O}(\mathcal{H}_{\{k_2\},O}\mathsf{false}).$$

3.2 Model Checking SecLTL

The *model checking problem* for SecLTL is, given a SecLTL formula φ and a transition system M to determine whether $M \models \varphi$.

We now describe an automata-theoretic technique for model checking SecLTL. To this end we show that given a SecLTL formula φ and a transition system M, we can construct a nondeterministic Büchi word automaton that accepts exactly those paths in $\mathsf{Paths}_{s_0,M}$ that satisfy φ. As an intermediate step of this translation we construct an alternating Büchi word automaton from M and φ with this property. This construction extends the standard translation for LTL and thus inherits its intuitiveness. An important advantage of the use of alternation is that it allows us to naturally follow the semantics of the operator $\mathcal{H}$ employing the universal branching in the automaton transitions.

One of the differences between the automaton we construct for a SecLTL formula and the one for an LTL formula obtained by the standard construction is that each state of the automaton for SecLTL carries a state of M as a component. This allows the automaton to keep track of the executions of M on the alternative paths for a given input word when necessary. Note that this construction could be slightly adapted in order to eliminate the need for a subsequent product construction of the resulting automaton with M. We decided, however, not to do that, in order to give a better intuition about the construction here and its relation to the corresponding construction for Restricted SecLTL in Section 4.

Definition 9. *The* closure operator $cl(\varphi)$ *maps a SecLTL formula φ to a set of SecLTL formulas that consists of all subformulas of φ and their negations. For LTL operators we use the standard definition of subformulas and the subformulas of a formula $\mathcal{H}_{H,O}\varphi$ contain the formula itself and the subformulas of φ.*

Proposition 1. *For a transition system $M = (\mathcal{V}_\mathcal{I}, \mathcal{V}_\mathcal{O}, S, s_0, \delta)$ and a SecLTL formula φ we can construct an alternating Büchi word automaton $\mathcal{A}_{M,\varphi} = (Q, q_0, \Sigma, \rho, F)$ with $\Sigma = \mathsf{vals}(\mathcal{V}_\mathcal{I} \cup \mathcal{V}_\mathcal{O})$ such that $|Q|$ is in $\mathcal{O}(|\varphi| \cdot |S|^2)$ and for every path $\pi \in \mathsf{Paths}_{s_0,M}$ it holds that $\pi \in L_\omega(\mathcal{A}_{M,\varphi})$ iff $M, s_0, \pi \models \varphi$.*

Proof. We define the set of states $Q = Q_\varphi \times S_\perp$, where $S_\perp = S \cup \{\perp\}$ and

$$Q_\varphi = cl(\varphi) \cup \{(O, \psi, m, s) \in 2^{\mathcal{V}_\mathcal{O}} \times \{\psi\} \times \{\forall, \exists\} \times S \mid \exists H \subseteq \mathcal{V}_\mathcal{I}. \mathcal{H}_{H,O}\psi \in cl(\varphi)\}.$$

The initial state of $\mathcal{A}_{M,\varphi}$ is $q_0 = (\varphi, s_0)$ and the set F of accepting states is defined as $F = \{(\neg(\psi \,\mathcal{U}\, \psi'), s) \in Q\} \cup \{((O, \psi, \forall, s'), s) \in Q\}$.

To define the transition function $\rho : Q \times \Sigma \to \mathcal{B}^+(Q)$, we extend the transition function δ of M to a total function $\delta_\perp : S_\perp \times \Sigma \to S_\perp$: for $s \in S$, $\delta_\perp(s, a) = \delta(s, a)$ if $\delta(s, a)$ is defined and $\delta_\perp(s, a) = \perp$ otherwise, and $\delta_\perp(\perp, a) = \perp$.

For convenience, we define the dual $\overline{q}$ of states in $q \in Q$: $\overline{(\psi, s)} = (\neg\psi, s)$, $\overline{((O, \psi, \forall, s'), s)} = ((O, \neg\psi, \exists, s'), s)$, and $\overline{((O, \psi, \exists, s'), s)} = ((O, \neg\psi, \forall, s'), s)$.

For $(\psi, \perp) \in Q$ where ψ is not an LTL formula and $a \in \Sigma$ we define $\rho((\psi, \perp), a) = \mathsf{false}$. For $((O, \psi, m, s'), \perp) \in Q$ and $a \in \Sigma$, we define $\rho(((O, \psi, m, s'), \perp), a) = \mathsf{false}$. For the remaining cases we define:

$$\begin{aligned}
\rho((v, s), a) &= \mathsf{true} \text{ if } a|_v = 1 \text{ and } \mathsf{false} \text{ if } a|_v = 0,\\
\rho((\neg\psi, s), a) &= \overline{\rho((\psi, s), a)},\\
\rho((\psi \vee \psi', s), a) &= \rho((\psi, s), a) \vee \rho((\psi', s), a),\\
\rho((\bigcirc\psi, s), a) &= (\psi, \delta_\perp(s, a)),\\
\rho((\psi \,\mathcal{U}\, \psi', s), a) &= \rho((\psi', s), a) \vee \rho((\psi, s), a) \wedge (\psi \,\mathcal{U}\, \psi', \delta_\perp(s, a)),\\
\rho((\mathcal{H}_{H,O}\psi, s), a) &= \rho((\psi, s), a) \vee \mathsf{check}(O, a, \delta^*_M(s, a|_{\mathcal{V}_\mathcal{I} \setminus H})) \wedge\\
&\quad \textstyle\bigwedge_{(a', s') \in \delta^*_M(s, a|_{\mathcal{V}_\mathcal{I} \setminus H})} ((O, \psi, \forall, s'), \delta_\perp(s, a)),\\
\rho(((O, \psi, \forall, s'), s), a) &= \rho((\psi, s), a) \vee \mathsf{check}(O, a, \delta^*_M(s', a|_{\mathcal{V}_\mathcal{I}})) \wedge\\
&\quad \textstyle\bigwedge_{(a', s'') \in \delta^*_M(s', a|_{\mathcal{V}_\mathcal{I}})} ((O, \psi, \forall, s''), \delta_\perp(s, a)),\\
\rho(((O, \psi, \exists, s'), s), a) &= \overline{\rho(((O, \neg\psi, \forall, s'), s), a)}\\
&= \rho((\psi, s), a) \wedge \big(\overline{\mathsf{check}(O, a, \delta^*_M(s', a|_{\mathcal{V}_\mathcal{I}}))} \vee\\
&\quad \textstyle\bigvee_{(a', s'') \in \delta^*_M(s', a|_{\mathcal{V}_\mathcal{I}})} ((O, \psi, \exists, s''), \delta_\perp(s, a))\big).
\end{aligned}$$

where check is defined as $\mathsf{check}(O, a, A) = \big(\forall (a', s') \in A : a' =_O a\big)$. Note that this function can be evaluated during the construction of $\mathcal{A}_{M,\varphi}$ in time $|M|$.

Applied to a state of the form $(\mathcal{H}_{H,O}\psi, s)$, the transition function follows the semantics of $\mathcal{H}$, which involves universal branching w.r.t. the alternative paths that start at state s. For states of the form $((O, \psi, \forall, s'), s)$, the transition relation is again defined according to the semantics of $\mathcal{H}$, which in its temporal aspect is similar to the LTL *weak until* operator. Here the transition relation verifies ψ on the main path or branches universally according to the alternative paths starting from s'. The function check has a similar effect to that of evaluating a variable but it instead compares for O-equivalence.

States of the form $((O, \psi, \forall, s'), s)$ are accepting, as branches for alternative paths that are forever equivalent to the main path should be accepting.

Let $\pi \in \Sigma^\omega$ and let τ be a path in some run tree of $\mathcal{A}_{M,\varphi}$ on π. Let us consider a state $\tau[i] \in Q$ for some $i \geq 0$. The set Q of states of $\mathcal{A}_{M,\varphi}$ contains two types of states. If $\tau[i]$ is of the form $(\psi, s) \in cl(\varphi) \times S_\perp$ and $s \neq \perp$, then $s = \sigma[i]$, where $\sigma = \mathsf{Exec}_M(s_0, \pi)$. That is, s is a state on the execution of M on π starting from s_0 that corresponds to the prefix of π read so far. Similarly, if $\tau[i]$ is of the form $((O, \psi, m, s'), s) \in (2^{\mathcal{V}_\mathcal{O}} \times cl(\varphi) \times \{\forall, \exists\} \times S) \times S_\perp$ and $s \neq \perp$, we have $s = \sigma[i]$. The state $s' \in S$ is a state on the execution $\mathsf{Exec}_M(\sigma[j], \pi')$, where $0 \leq j < i$ and $\pi' \in \mathsf{AltPaths}_M(\sigma[j], \pi[j, \infty), H)$ for some $H \subseteq \mathcal{V}_\mathcal{I}$. That is, the state s' is a state on the execution of M on some alternative path π' that branches off form π at some position prior to position i. We point out that:
Remark 1. For every $(\psi, s) \in Q$ where ψ is an LTL formula, it holds that $L_\omega(\mathcal{A}_{M,\varphi}, (\psi, s)) = L_\omega(\mathcal{A}_{M,\varphi}, (\psi, \perp))$. If the formula φ does not contain nested $\mathcal{H}$ operators, then for every $((O, \psi, m, s'), s)$ it holds that ψ is an LTL formula and, as a consequence of the above and the definition of ρ, we have that $L_\omega(\mathcal{A}_{M,\varphi}, ((O, \psi, m, s'), s)) = L_\omega(\mathcal{A}_{M,\varphi}, ((O, \psi, m, s'), \perp))$. □

Proposition 2. *[10] For every alternating Büchi word automaton $\mathcal{A}$ with n states there exists a nondeterministic Büchi word automaton $\mathcal{N}$ with $2^{\mathcal{O}(n)}$ states such that $L_\omega(\mathcal{N}) = L_\omega(\mathcal{A})$.*

Proof. Let $\mathcal{A} = (Q, q_0, \Sigma, \rho, F)$ be an alternating Büchi word automaton. We construct a nondeterministic Büchi word automaton $\mathcal{N} = (Q^{\mathsf{nd}}, q_0^{\mathsf{nd}}, \Sigma, \rho^{\mathsf{nd}}, F^{\mathsf{nd}})$ as follows: $Q^{\mathsf{nd}} = 2^Q \times 2^Q$, $q_0^{\mathsf{nd}} = (\{q_0\}, \emptyset)$, $F^{\mathsf{nd}} = \{(R, \emptyset) \mid R \subseteq Q\}$ and

$$\rho^{\mathsf{nd}}((R_1, R_2), a) = \begin{cases} \{(R_1', R_1' \setminus F) \mid R_1' \models \bigwedge_{q \in R_1} \rho(q, a)\} & \text{if } R_2 = \emptyset, \\ \{(R_1', R_2' \setminus F) \mid R_2' \subseteq R_1', R_1' \models \bigwedge_{q \in R_1} \rho(q, a), \\ \qquad R_2' \models \bigwedge_{q \in R_2} \rho(q, a)\} & \text{if } R_2 \neq \emptyset. \end{cases}$$

Theorem 1. *For a transition system $M = (\mathcal{V}_\mathcal{I}, \mathcal{V}_\mathcal{O}, S, s_0, \delta)$ and a SecLTL formula φ we can check in time $\mathcal{O}(|M| \cdot 2^{\mathcal{O}(|\varphi| \cdot |S|^2)})$ or in space $\mathcal{O}((\log |M| + |\varphi| \cdot |S|^2)^2)$ whether $M \models \varphi$ holds.*

Proof. We can view M as a nondeterministic Büchi word automaton and construct the product $\mathcal{B}_{M,\neg\varphi}$ of M and the nondeterministic automaton $\mathcal{N}_{M,\neg\varphi}$ for the negation of φ. Then $M \models \varphi$ iff $L_\omega(\mathcal{B}_{M,\neg\varphi}) = \emptyset$ and the claim of the theorem follows from the fact that the nonemptiness problem for nondeterministic Büchi automata of size n is decidable in time $\mathcal{O}(n)$ or in space $\mathcal{O}(\log^2 n)$ [1]. □

3.3 Complexity of the Model Checking Problem for SecLTL

A *concurrent program* is a parallel composition of a number of components using the interleaving semantics and synchronizing over shared actions. We reduce the model checking problem for concurrent programs (as defined in [2]) to the SecLTL model checking problem for a monolithic transition system.

Theorem 2. *[2] Model Checking CTL and CTL* for concurrent programs is* PSPACE*-complete both in the size of the formula [2, Thm. 6.1] and in the size of the transition systems [2, Thm. 6.2].*

Theorems 6.1 and 6.2 in [2] make use of a single CTL formula: $EF(a_1 \vee \cdots \vee a_n)$, for some atomic propositions a_i and a number of processes n. As the negation of that property can be expressed in LTL ($\square(\neg a_1 \wedge \cdots \wedge \neg a_n)$), we can immediately extend their result to LTL.

Lemma 1. *Model Checking LTL for concurrent programs is* PSPACE*-complete both in the size of the formula and in the size of the transition systems.*

Theorem 3. *The model checking problem for SecLTL is* PSPACE*-complete.*

Proof sketch. We reduce the LTL model checking problem for concurrent programs to model checking a SecLTL formula on a single transition system, which is the union of all individual programs together with a new initial state. For the initial state the transition function allows to select a program whose transition function is used subsequently. The program is then executed independently from the others. We give a SecLTL formula that ensures that the original specification is checked only on valid interleavings.

4 Restricted SecLTL

For some cases the nondeterministic automaton for a SecLTL formula does not have to track a set of executions on alternative paths, but it suffices to track one such execution. This raises hopes for more efficient fragments of SecLTL. In this section, we identify one such fragment, which we call *Restricted SecLTL*, that is characterized by a simple set of syntactic restrictions. We show that, indeed, the model checking problem for Restricted SecLTL has a lower computational complexity in terms of the size of the transition system.

Negation normal form (NNF). In order to elegantly state the restrictions, we introduce negation normal form for (not necessarily restricted) SecLTL formulas. As usual, the LTL operator $\mathcal{R}$ is the dual of $\mathcal{U}$, i.e., $\neg(\varphi \,\mathcal{U}\, \psi) \equiv \neg\varphi \,\mathcal{R}\, \neg\psi$. We define the SecLTL operator $\mathcal{L}$, the *leak* operator, as the dual of the SecLTL operator $\mathcal{H}$: For a given transition system $M = (\mathcal{V}_\mathcal{I}, \mathcal{V}_\mathcal{O}, S, s_0, \delta)$, infinite word $\pi \in \Sigma^\omega$, where $\Sigma = \mathsf{vals}(\mathcal{V}_\mathcal{I} \cup \mathcal{V}_\mathcal{O})$, and state $s \in S$, it holds that

$$M, s, \pi \models \mathcal{L}_{H,O}\varphi \text{ iff } M, s, \pi \models \neg(\mathcal{H}_{H,O}\neg\varphi).$$

For every SecLTL formula φ, we denote with $\mathsf{NNF}(\varphi)$ the NNF of φ. SecLTL formulas in NNF are defined according to the following grammar, where $v \in \mathcal{V}$, φ, ψ are SecLTL formulas in NNF, $H \subseteq \mathcal{V}_\mathcal{I}$ and $O \subseteq \mathcal{V}_\mathcal{O}$,

$$\begin{array}{rcccccccc} \varphi & ::= & v & \mid & \neg v & \mid & \varphi \vee \psi & \mid & \varphi \wedge \psi \mid \bigcirc\varphi \\ & & \varphi \,\mathcal{U}\, \psi & \mid & \varphi \,\mathcal{R}\, \psi & \mid & \mathcal{H}_{H,O}\, \varphi & \mid & \mathcal{L}_{H,O}\, \varphi. \end{array}$$

Restricted SecLTL. A *Restricted SecLTL formula* is a SecLTL formula φ in NNF that does not contain the operator$\mathcal{L}$, does not contain nested $\mathcal{H}$ operators, and:
($\mathcal{U}$) for every subformula $\varphi_1 \,\mathcal{U}\, \varphi_2$ of φ, the formula φ_2 is an LTL formula,
($\mathcal{R}$) for every subformula $\varphi_1 \,\mathcal{R}\, \varphi_2$ of φ, the formula φ_1 is an LTL formula.

Since we will build an alternating automaton for the negated formula, we also formulate the restrictions on the negated version for reference: For a Restricted SecLTL formula, the formula $\mathsf{NNF}(\neg\varphi)$ does not contain the operator $\mathcal{H}$, does not contain nested $\mathcal{L}$ operators, and satisfies the dual versions of $(\mathcal{U})$ and $(\mathcal{R})$:
$(\mathcal{U}\neg)$ for every subformula $\varphi_1 \,\mathcal{U}\, \varphi_2$ of $\mathsf{NNF}(\neg\varphi)$, φ_1 is an LTL formula,
$(\mathcal{R}\neg)$ for every subformula $\varphi_1 \,\mathcal{R}\, \varphi_2$ of $\mathsf{NNF}(\neg\varphi)$, φ_2 is an LTL formula.

Expressive power. The above restrictions do not have a significant effect on the expressive power. Examples 1 to 4 are still expressible in Restricted SecLTL. Thus, the main assets of SecLTL, that is the bounded secret generation and the use of hide operators in temporal contexts, are preserved.

Proposition 3. *The system complexity of the model checking problem for Restricted SecLTL is in* NLOGSPACE.

Proof. We adapt the construction from Proposition 1 to the special case of formulas of the form $\neg\varphi$ where φ is a Restricted SecLTL formula. As in the formula $\mathsf{NNF}(\neg\varphi)$ negation occurs only in front of variables, instead of $cl(\neg\varphi)$ we can use the set $sf(\mathsf{NNF}(\neg\varphi))$ that consists of all subformulas of $\mathsf{NNF}(\neg\varphi)$. Furthermore, since $\mathsf{NNF}(\neg\varphi)$ does not contain $\mathcal{H}$ operators, states of the form $((O,\psi,\forall,s'),s)$ are no longer needed. We define $Q = Q'_{\neg\varphi} \times S_\perp$, where $S_\perp = S \dot\cup \{\perp\}$ and

$$Q'_{\neg\varphi} = sf(\mathsf{NNF}(\neg\varphi)) \cup \{(O,\psi,\exists,s) \mid \exists H : \mathcal{L}_{H,O}\psi \in sf(\mathsf{NNF}(\neg\varphi))\}.$$

In the alternating Büchi word automaton $A_{M,\neg\varphi} = (Q, q_0, \Sigma, \rho, F)$ the initial state is $q_0 = (\mathsf{NNF}(\neg\varphi), s_0)$ if φ is not an LTL formula and $q_0 = (\mathsf{NNF}(\neg\varphi), \perp)$ otherwise, and the set of accepting states is $F = \{(\psi \,\mathcal{R}\, \psi', s) \in Q\}$.

According to Remark 1 we can replace in the definition of the transition function ρ the function $\delta_\perp$ by the function $\delta'_\perp : cl(\varphi) \times S_\perp \times \Sigma \to S_\perp$ where $\delta'_\perp(\psi,s,a) = \perp$ if ψ is an LTL formula and $\delta'_\perp(\psi,s,a) = \delta_\perp(s,a)$ otherwise.

As $\mathsf{NNF}(\neg\varphi)$ does not contain nested $\mathcal{L}$ operators we ensure by the definition of $\delta'_\perp$ that states of the form $((O,\psi,\exists,s'),s)$ where $s \neq \perp$ are not reachable.

The transition relation ρ is defined as follows:

$$\begin{aligned}
\rho((v,s),a) &= \mathsf{true} \text{ if } a|_v = 1 \text{ and } \mathsf{false} \text{ if } a|_v = 0,\\
\rho((\neg v,s),a) &= \mathsf{true} \text{ if } a|_v = 0 \text{ and } \mathsf{false} \text{ if } a|_v = 1,\\
\rho((\psi \vee \psi',s),a) &= \rho((\psi,s),a) \vee \rho((\psi',s),a),\\
\rho((\psi \wedge \psi',s),a) &= \rho((\psi,s),a) \wedge \rho((\psi',s),a),\\
\rho((\bigcirc\psi,s),a) &= (\psi, \delta'_\perp(\psi,s,a)),\\
\rho((\psi\,\mathcal{U}\,\psi',s),a) &= \rho((\psi',s),a) \vee \rho((\psi,s),a) \wedge (\psi\,\mathcal{U}\,\psi', \delta'_\perp(\psi\,\mathcal{U}\,\psi',s,a)),\\
\rho((\psi\mathcal{R}\psi',s),a) &= \rho((\psi',s),a) \wedge (\rho((\psi,s),a) \vee (\psi\mathcal{R}\psi', \delta'_\perp(\psi\mathcal{R}\psi',s,a))),\\
\rho((\mathcal{L}_{H,O}\psi,s),a) &= \rho((\psi,s),a) \wedge \big(\overline{\mathsf{check}(O,a,\delta^*_M(s,a|_{\mathcal{V}_\mathcal{I}\setminus H}))} \vee\\
&\quad \textstyle\bigvee_{(a',s')\in\delta^*_M(s,a|_{\mathcal{V}_\mathcal{I}\setminus H})}((O,\psi,\exists,s'),\delta'_\perp(\psi,s,a))\big),\\
\rho(((O,\psi,\exists,s'),s),a) &= \rho((\psi,s),a) \wedge \big(\overline{\mathsf{check}(O,a,\delta^*_M(s',a|_{\mathcal{V}_\mathcal{I}}))} \vee\\
&\quad \textstyle\bigvee_{(a',s'')\in\delta^*_M(s',a|_{\mathcal{V}_\mathcal{I}})}((O,\psi,\exists,s''),\delta'_\perp(\psi,s,a))\big).
\end{aligned}$$

We see already that the restrictions eliminated universal branching over successors. Disjunctive branching over successors does not lead to an exponential blow up in the size of the system during the construction of the nondeterministic Büchi automaton. In the following, we show that the number of executions that we have to track is bounded by the number of leak operators in the formula.

Let k be the number of leak operators in $\mathsf{NNF}(\neg\varphi)$. We construct a nondeterministic Büchi word automaton $\mathcal{N}'_{M,\neg\varphi} = (Q', q'_0, \Sigma, \rho', F')$ with $|Q'|$ in $\mathcal{O}(2^{\mathcal{O}(|\varphi|)} \cdot |S|^k)$ and such that $L_\omega(\mathcal{N}'_{M,\neg\varphi}) = L_\omega(\mathcal{N}_{M,\neg\varphi})$, where $\mathcal{N}_{M,\neg\varphi} = (Q^{\mathsf{nd}}, q_0^{\mathsf{nd}}, \Sigma, \rho^{\mathsf{nd}}, F^{\mathsf{nd}})$ is the nondeterministic Büchi automaton for $\mathcal{A}_{M,\neg\varphi}$ constructed using the construction from Proposition 2.

For a set $R \subseteq Q$, we denote with $\mathsf{ns}(R)$ the sum of the number of states in R of the form (ψ, s) with $s \neq \bot$ and the number of states in R of the form $((O, \psi, \exists, s'), s)$. We denote with $\mathsf{nl}(R)$ the sum of the number of occurrences of $\mathcal{L}$ in formulas in R and the number of states in R of the form $((O, \psi, \exists, s'), s)$. We define $Q' = \{(R_1, R_2) \in Q^{\mathsf{nd}} \mid \mathsf{ns}(R_1) \leq k, \mathsf{nl}(R_1) \leq k \text{ and } R_2 \subseteq R_1\}$.

Each state in $(R_1, R_2) \in Q'$ can be represented as a tuple $(A_1, A_2, \overline{s})$, where A_i is obtained from R_i by replacing each state of the form (ψ, s) where $s \neq \bot$ by $(\psi, ?)$ and each state of the form $((O, \psi, \exists, s'), s)$ by $(O, \psi, \exists, ?)$ and $\overline{s}$ is a vector of states in S of size k that assigns states to the ?-elements in A_1 and A_2 according to some fixed order on the formulas in $(\mathsf{NNF}(\neg\varphi))$. This is possible as the definition of Q' guarantees that A_1 contains at most k ?-elements and $A_2 \subseteq A_1$. Thus, the number of states of $\mathcal{N}'_{M,\neg\varphi}$ is in $\mathcal{O}(2^{\mathcal{O}(|\varphi|)} \cdot |S|^k)$.

The initial state of $\mathcal{N}'_{M,\neg\varphi}$ is $q'_0 = q_0^{\mathsf{nd}}$ and the accepting states and the transition relation are defined as in $\mathcal{N}_{M,\neg\varphi}$: $F' = \{(R_1, R_2) \in Q' \mid R_2 = \emptyset\}$ and

$$\rho'((R_1, R_2), a) = \begin{cases} \{(R'_1, R'_1 \setminus F) \in Q' \mid R'_1 \models \bigwedge_{q \in R_1} \rho(q, a)\} & \text{if } R_2 = \emptyset, \\ \{(R'_1, R'_2 \setminus F) \in Q' \mid R'_2 \subseteq R'_1, R'_1 \models \bigwedge_{q \in R_1} \rho(q, a), \\ \qquad R'_2 \models \bigwedge_{q \in R_2} \rho(q, a)\} & \text{if } R_2 \neq \emptyset. \end{cases}$$

For every $(R_1, R_2) \in Q'$ and $a \in \Sigma$, $\rho'((R_1, R_2), a) \subseteq \rho^{\mathsf{nd}}((R_1, R_2), a)$. Therefore, since $q'_0 = q_0^{\mathsf{nd}}$, it holds that $L_\omega(\mathcal{N}'_{M,\neg\varphi}) \subseteq L_\omega(\mathcal{N}_{M,\neg\varphi})$.

For $R_1, R_2, S_1, S_2 \in 2^Q$, $(R_1, R_2) \subseteq (S_1, S_2)$ iff $R_1 \subseteq S_1$ and $R_2 \subseteq S_2$.

We now show that for every $(S_1, S_2) \in Q^{\mathsf{nd}}$, $(S'_1, S'_2) \in \rho^{\mathsf{nd}}((S_1, S_2), a)$ and $(R_1, R_2) \subseteq (S_1, S_2)$ there exists $(R'_1, R'_2) \in Q'$ such that $(R'_1, R'_2) \in \rho'((R_1, R_2), a)$ and $(R'_1, R'_2) \subseteq (S'_1, S'_2)$. To this end, we prove by induction on the structure of Restricted SecLTL formulas and the definition of ρ that for every $i \in \{1, 2\}$, $q \in R_i$ there exists a set $R'_{i,q} \subseteq S'_i$ such that $R'_{i,q} \models \rho(q, a)$ and $\mathsf{nl}(R'_{i,q}) \leq \mathsf{nl}(\{q\})$. If $q = (\psi, s)$ where ψ is an LTL formula, we can clearly choose $R'_{i,q}$ such that $\mathsf{nl}(R'_{i,q}) = 0$. For $q = (\mathcal{L}_{H,O}\psi, s)$ or $q = ((O, \psi, \exists, s'), s)$ we have $\mathsf{nl}(\{q\}) = 1$ and, since ψ is an LTL formula, we can choose $R'_{i,q}$ with $\mathsf{nl}(R'_{i,q}) = 1$. For the other cases the property follows from the induction hypothesis and the fact that if $q = (\psi \ \mathcal{U} \ \psi')$ then ψ is an LTL formula and if $q = (\psi \ \mathcal{R} \ \psi')$ then ψ' is an LTL formula. Thus, we can choose $(S'_1, S'_2) \in Q^{\mathsf{nd}}$ such that $(R'_1, R'_2) \in \rho^{\mathsf{nd}}((R_1, R_2), a)$, $(R'_1, R'_2) \subseteq (S'_1, S'_2)$ and $\mathsf{nl}(R'_i) \leq k$ and hence also $\mathsf{ns}(R'_i) \leq k$ for $i \in \{1, 2\}$. Thus, $(R'_1, R'_2) \in \rho'((R_1, R_2), a)$.

The property above implies that for every run τ of $\mathcal{N}_{M,\neg\varphi}$ on a word $\pi \in \Sigma^\omega$ there exists a run τ' of $\mathcal{N}'_{M,\neg\varphi}$ on π such that $\tau'[i] \subseteq \tau[i]$ for every $i \geq 0$. If τ is accepting, then τ' is also accepting. This implies that $L_\omega(\mathcal{N}_{M,\neg\varphi}) \subseteq L_\omega(\mathcal{N}'_{M,\neg\varphi})$, which concludes the proof that $L_\omega(\mathcal{N}_{M,\neg\varphi}) = L_\omega(\mathcal{N}'_{M,\neg\varphi})$. □

Theorem 4. *Model Checking Restricted SecLTL is* PSPACE*-complete, and its system complexity is* NLOGSPACE*-complete.*

Proof. PSPACE-completeness follows from the fact that the model checking problem for LTL is PSPACE-hard and that we already showed that model checking full SecLTL is in PSPACE. By Proposition 3, model checking SecLTL can be done in space NLOGSPACE in the size of the system. Since the system complexity of LTL model checking is NLOGSPACE-hard, the theorem follows. □

5 Extension to Branching Time

To demonstrate that the hide operator allows for smooth extension of other temporal logics, we integrate it in the well known branching time logics CTL and CTL*. While for *SecCTL** the complexity is straight-forward to determine, the result for the extension of CTL might be surprising: it is PSPACE-complete.

We define the logic SecCTL* as a standard *branching-time extension of SecLTL*. SecCTL* state formulas are defined as follows, where $v \in V$, φ and φ' are SecCTL* state formulas and ψ and ψ' are SecCTL* path formulas:

$$\varphi \quad ::= \quad v \mid \neg\varphi \mid \varphi \vee \varphi' \mid A\psi \mid E\psi.$$

SecCTL* path formulas, defined below, can contain the temporal operator $\mathcal{H}$:

$$\psi \quad ::= \quad \varphi \mid \neg\psi \mid \psi \vee \psi' \mid \bigcirc\psi \mid \psi \,\mathcal{U}\, \psi' \mid \mathcal{H}_{H,O}\, \psi.$$

The path-based definition provides a simple and unique semantics for the hide operator in SecCTL*.

Since SecCTL* is a standard branching-time extension of SecLTL, we can employ the dynamic programming approach used for CTL* but use a SecLTL model checker instead of an LTL model checker. Formulas are, as usual, evaluated in a bottom-up manner. For a formula ψ, we evaluate all maximal proper state subformulas of ψ and label the edges of the transition system accordingly with values for fresh output variables. Then replace each maximal proper state subformula in ψ by the corresponding fresh output variable and proceed. For formulas of the form $E\psi$ after the substitution in ψ we have that ψ is a SecLTL formula and thus, we can compute the set of states that satisfy $E\psi$ using a SecLTL model checker.

Thus, for a SecCTL* formula φ and a transition system $M = (\mathcal{V}_\mathcal{I}, \mathcal{V}_\mathcal{O}, S, s_0, \delta)$ we can check $M \models \varphi$ by using $\mathcal{O}(|S| \cdot |\varphi|)$ calls of a SecLTL model checker.

Theorem 5. *The model checking problem for SecCTL* is* PSPACE*-complete.*

Proof. Membership in $\mathsf{P}^{\mathsf{PSPACE}}$, that is, in PSPACE, is implied by the algorithm described above. PSPACE-hardness follows from Theorem 3. □

SecCTL. The subset SecCTL of SecCTL* is defined in the standard way by restricting the path formulas to be of the form $\bigcirc\varphi$, $\varphi\,\mathcal{U}\,\varphi'$ or $\mathcal{H}_{H,O}\,\varphi$, where φ and φ' are SecCTL *state* formulas.

Theorem 6. *The model checking problem for SecCTL is* PSPACE*-complete.*

Proof sketch. Similarly to the hardness proof for SecLTL, we provide a reduction from the CTL model checking problem for concurrent systems to model checking a SecCTL formula on a monolithic system of polynomial size.

6 Related Work

Recent works [9] provide a uniform framework for classification of properties that refer to multiple paths at once, called *hyperproperties*. These works, however, do not provide means to specify and verify hyperproperties. Such a formalism is, of course, not even possible for the set of hyperproperties in its full generality. Of particular interest is the class of k-safety hyperproperties which consists of those hyperproperties that can be refuted by considering at most k finite paths. The verification problem for such properties can be reduced to checking a safety property on a system obtained by k-fold *self-composition*. Huisman et al. [11] specify observational determinism in CTL* and in the polyadic modal μ-calculus interpreted over the 2-fold self-composition of the system.

SecLTL, in contrast, can express properties that go beyond k-safety hyperproperties; a counterexample for a SecLTL specification (e.g. for $\mathcal{L}_{H,O}\ \mathsf{true}$) might require an infinite number of paths, and are therefore out of the scope of self-composition based approaches.

A different approach to analyze information flow properties in combination with temporal properties, is that of *epistemic logics* [12–14]. Epistemic logics introduce *knowledge operators* to temporal logics and allow for properties that refer to the knowledge of an agent at a certain point in the system run—thus they are able to express information flow properties like non-interference [15, 16].

The fundamental difference between epistemic logics and SecLTL is, that a knowledge operator expresses the knowledge of an agent, whereas the hide operator specifies the secret. This allows us to argue in a forward-manner starting at the point at which the secret is introduced.

Alur et al. [17] extended CTL and μ-calculus by two modal operators, the first of which, similarly to the knowledge operator in epistemic logics allows for quantifying over a set of equivalent states, and the second allows for referring to non-equivalent states. The formulas in the resulting logics are interpreted over computation trees augmented with edges representing observational equivalences between path prefixes.

7 Conclusion

We proposed a new modal operator that allows for natural path-based integration of information flow properties in temporal logics. The rich set of examples

we considered demonstrates that the resulting linear time logic is expressive enough to precisely specify many interesting information flow and secrecy properties. The operator allows for simple characterizations of sufficiently expressive fragments with better computational complexity, like Restricted SecLTL, and seamless integration into branching time logics like the presented SecCTL and SecCTL*. Future work includes identifying fragments of the branching time logics with reduced complexity and extensions to the alternating-time setting.

References

1. Vardi, M.Y., Wolper, P.: Reasoning about infinite computations. Inf. Comput. **115** (1994) 1–37
2. Kupferman, O., Vardi, M.Y., Wolper, P.: An automata-theoretic approach to branching-time model checking. J. ACM **47** (2000) 312–360
3. Vardi, M.Y.: Alternating automata and program verification. In: In Computer Science Today. LNCS 1000, Springer-Verlag (1995) 471–485
4. Goguen, J.A., Meseguer, J.: Security policies and security models. In: IEEE Symposium on Security and Privacy. (1982) 11–20
5. Zdancewic, S., Myers, A.C.: Observational determinism for concurrent program security. In: Proc. 16th IEEE Computer Security Foundations Workshop. (2003)
6. Broberg, N., Sands, D.: Paralocks – role-based information flow control and beyond. In: Proc. of POPL'10. (2010)
7. Askarov, A., Myers, A.: A semantic framework for declassification and endorsement. In: Proc. of ESOP'10. Volume 6012 of LNCS. Springer-Verlag (2010) 64–84
8. Alur, R., Zdancewic, S.: Preserving secrecy under refinement. In: Proc. of ICALP'06, LNCS 4052, Springer-Verlag (2006) 107–118
9. Clarkson, M.R., Schneider, F.B.: Hyperproperties. Journal of Computer Security **18** (2010) 1157–1210
10. Miyano, S., Hayashi, T.: Alternating finite automata on omega-words. Theor. Comput. Sci. **32** (1984) 321–330
11. Huisman, M., Worah, P., Sunesen, K.: A temporal logic characterisation of observational determinism. In: CSFW, IEEE Computer Society (2006) 3
12. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press (1995)
13. van der Meyden, R., Shilov, N.V.: Model checking knowledge and time in systems with perfect recall. In: FSTTCS. Volume 1738 of LNCS., Springer (1999) 432–445
14. Shilov, N.V., Garanina, N.O.: Model checking knowledge and fixpoints. In: FICS. (2002) 25–39
15. Engelhardt, K., Gammie, P., Meyden, R.: Model checking knowledge and linear time: PSPACE cases. In: Proc. of LFCS'07. (2007) 195–211
16. Balliu, M., Dam, M., Guernic, G.L.: Epistemic temporal logic for information flow security. In: Proc. PLAS'11. (2011)
17. Alur, R., Cerný, P., Chaudhuri, S.: Model checking on trees with path equivalences. In: TACAS. (2007) 664–678